Table of Contents

Part I  What's New  1

Part II  General Information  17
  1 Overview ..................................................................................................................... 17
  2 Features ...................................................................................................................... 21
  3 Requirements .............................................................................................................. 27
  4 Compatibility .............................................................................................................. 28
  5 Using Several DAC Products in One IDE .............................................................. 32
  6 Component List ......................................................................................................... 33
  7 Hierarchy Chart ......................................................................................................... 35
  8 Editions ...................................................................................................................... 37
  9 Licensing .................................................................................................................... 40
 10 Getting Support ........................................................................................................ 43
 11 Frequently Asked Questions .................................................................................... 44

Part III  Getting Started  52
  1 Installation ................................................................................................................ 52
  2 Migrating from BDE and DOA ................................................................................ 57
  3 Connecting to Oracle ............................................................................................... 58
  4 Creating Database Objects ...................................................................................... 62
  5 Retrieving and Modifying Data .............................................................................. 64
  6 Inserting Data Into Tables ...................................................................................... 66
  7 Working With Oracle Stored Procedures .......................................................... 71
    Stored Procedures - General Information ............................................................. 71
    Using Stored Procedures via the TOraStoredProc Class .................................. 73
    Using Package Procedures .................................................................................. 75
  8 Working With PL/SQL ............................................................................................. 76
    PL/SQL - General Information ........................................................................... 77
    Using PL/SQL via the TOraSQL Class ............................................................... 79
  9 Using Transactions .................................................................................................. 80
 10 Demo Projects ......................................................................................................... 82
 11 Deployment .............................................................................................................. 91

Part IV  Using ODAC  92
  1 Updating Data with ODAC Dataset Components .............................. 93
  2 Connecting in Direct Mode ................................................................. 94
  3 Master/Detail Relationships ............................................................... 98
  4 Data Type Mapping ......................................................................................... 100
  5 Data Encryption ............................................................................................... 106

© 2021 Devart
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Working in an Unstable Network</td>
<td>108</td>
</tr>
<tr>
<td>7</td>
<td>Secure Connections</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>Connecting via SSL</td>
<td>109</td>
</tr>
<tr>
<td></td>
<td>Connecting via SSH</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>Network Tunneling</td>
<td>117</td>
</tr>
<tr>
<td>8</td>
<td>Disconnected Mode</td>
<td>122</td>
</tr>
<tr>
<td>9</td>
<td>Batch Operations</td>
<td>124</td>
</tr>
<tr>
<td>10</td>
<td>Increasing Performance</td>
<td>128</td>
</tr>
<tr>
<td>11</td>
<td>Macros</td>
<td>130</td>
</tr>
<tr>
<td>12</td>
<td>DataSet Manager</td>
<td>131</td>
</tr>
<tr>
<td>13</td>
<td>Using Connection Pooling</td>
<td>137</td>
</tr>
<tr>
<td>14</td>
<td>Automatic Key Field Value Generation</td>
<td>139</td>
</tr>
<tr>
<td>15</td>
<td>TOraLoader Component</td>
<td>140</td>
</tr>
<tr>
<td>16</td>
<td>TOraTransaction Component</td>
<td>141</td>
</tr>
<tr>
<td>17</td>
<td>TOraQueue, TOraQueueAdmin and TOraQueueTable Components</td>
<td>143</td>
</tr>
<tr>
<td>18</td>
<td>TOraChangeNotification Component</td>
<td>146</td>
</tr>
<tr>
<td>19</td>
<td>Working With Data</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>BLOB and CLOB Data Types</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>Unicode Character Data</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>Objects</td>
<td>154</td>
</tr>
<tr>
<td></td>
<td>XMLTYPE Data Type</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>VARRAY Data Type</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>DML Array</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>Cursors</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>PL/SQL Tables</td>
<td>164</td>
</tr>
<tr>
<td>20</td>
<td>Writing Oracle External Procedures with ODAC</td>
<td>165</td>
</tr>
<tr>
<td>21</td>
<td>Transparent Application Failover Support</td>
<td>168</td>
</tr>
<tr>
<td>22</td>
<td>DBMonitor</td>
<td>171</td>
</tr>
<tr>
<td>23</td>
<td>Writing GUI Applications with ODAC</td>
<td>172</td>
</tr>
<tr>
<td>24</td>
<td>Compatibility with Previous Versions</td>
<td>172</td>
</tr>
<tr>
<td>25</td>
<td>Oracle Package Wizard</td>
<td>173</td>
</tr>
<tr>
<td>26</td>
<td>64-bit Development with Embarcadero RAD Studio XE2</td>
<td>175</td>
</tr>
<tr>
<td>27</td>
<td>Database Specific Aspects of 64-bit Development</td>
<td>181</td>
</tr>
</tbody>
</table>

**Part V Reference**

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CRAccess</td>
<td>184</td>
</tr>
<tr>
<td></td>
<td>Classes</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>TCRCursor Class</td>
<td>185</td>
</tr>
<tr>
<td></td>
<td>Members</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>Types</td>
<td>186</td>
</tr>
<tr>
<td></td>
<td>TBeforeFetchProc Procedure Reference</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>Enumerations</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>TCRIsolationLevel Enumeration</td>
<td>188</td>
</tr>
<tr>
<td></td>
<td>TCRTxnActionEnumeration</td>
<td>189</td>
</tr>
<tr>
<td></td>
<td>TCurState Enumeration</td>
<td>189</td>
</tr>
</tbody>
</table>

© 2021 Devart
Oracle Data Access Components

OptionsEx Property .................................................................................................................. 222
OptionsMenu Property .............................................................................................................. 223
Methods ................................................................................................................................... 223
ActivateFilterEdit Method ........................................................................................................... 225
ActivateSearchEdit Method ........................................................................................................ 225
AdjustColumns Method ............................................................................................................. 226
ApplyFilter Method .................................................................................................................... 226
CalcTitleLevel Method .............................................................................................................. 227
ClearFilters Method .................................................................................................................... 227
ClearSorting Method ................................................................................................................... 228
DataChanged Method .................................................................................................................. 228
GetGridSize Method ................................................................................................................... 228
GetTitleLevel Method ................................................................................................................ 229
Events ....................................................................................................................................... 229
OnGetCellParams Event ............................................................................................................. 230

5 CRVio ........................................................................................................................................ 230

Classes ....................................................................................................................................... 231
THttpOptions Class ..................................................................................................................... 231
Members .................................................................................................................................... 232
Properties .................................................................................................................................. 232
Enabled Property ........................................................................................................................ 233
Password Property ..................................................................................................................... 233
ProxyOptions Property .............................................................................................................. 234
TrustServerCertificate Property ............................................................................................... 234
Url Property .............................................................................................................................. 235
Username Property ..................................................................................................................... 235
TProxyOptions Class .................................................................................................................. 236
Members .................................................................................................................................... 236
Properties .................................................................................................................................. 237
Hostname Property ..................................................................................................................... 237
Password Property ..................................................................................................................... 238
Port Property ............................................................................................................................. 238
Username Property ..................................................................................................................... 238
Enumerations .............................................................................................................................. 239
TIPVersion Enumeration ............................................................................................................. 239

6 DAAlerter .................................................................................................................................. 240

Classes ....................................................................................................................................... 240
TDAAlerter Class ....................................................................................................................... 241
Members .................................................................................................................................... 241
Properties .................................................................................................................................. 242
Active Property .......................................................................................................................... 243
AutoRegister Property .............................................................................................................. 243
Connection Property .................................................................................................................. 244
Methods ..................................................................................................................................... 244
SendEvent Method ..................................................................................................................... 245
Start Method .............................................................................................................................. 245
Stop Method ............................................................................................................................... 246
Events ....................................................................................................................................... 246
OnError Event .............................................................................................................................. 247
Types ......................................................................................................................................... 247
TAlerterErrorEvent Procedure Reference ................................................................................ 247

7 DADump .................................................................................................................................... 248

Classes ....................................................................................................................................... 249
# DALoader

8 **DALoader** ................................................................. 268

<table>
<thead>
<tr>
<th>Class</th>
<th>Members</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDADump Class</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Members</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TableNames Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backup Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackupQuery Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackupToFile Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackupToStream Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restore Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RestoreFromFile Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RestoreFromStream Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnBackupProgress Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnError Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnRestoreProgress Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TDADumpOptions Class</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Members</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddDrop Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompleteInsert Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenerateHeader Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QuoteNames Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Types</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TDABackupProgressEvent Procedure Reference</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TDARestoreProgressEvent Procedure Reference</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## 8 DALoader

<table>
<thead>
<tr>
<th>Class</th>
<th>Members</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDADump Class</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Members</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQL Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TableNames Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Backup Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackupQuery Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackupToFile Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BackupToStream Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Restore Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RestoreFromFile Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RestoreFromStream Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Events</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnBackupProgress Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnError Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnRestoreProgress Event</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TDADumpOptions Class</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Members</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AddDrop Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CompleteInsert Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GenerateHeader Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QuoteNames Property</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Types</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TDABackupProgressEvent Procedure Reference</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>TDARestoreProgressEvent Procedure Reference</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2021 Devart
VI  Oracle Data Access Components

9  DAScript .......................................................................................................................... 289

Classes ....................................................................................................................................... 290
TDAScript Class ......................................................................................................................... 290
  Members ......................................................................................................................................... 291
  Properties ........................................................................................................................................ 293
  Connection Property .................................................................................................................. 294
  DataSet Property ....................................................................................................................... 295
  Debug Property .......................................................................................................................... 295
  Delimiter Property .................................................................................................................... 296
  EndLine Property ......................................................................................................................... 296
  EndOffset Property .................................................................................................................... 297
  EndPos Property ........................................................................................................................ 297
  Macros Property ......................................................................................................................... 297
  SQL Property ............................................................................................................................... 298
  StartLine Property ..................................................................................................................... 298
  StartOffset Property .................................................................................................................. 299
  StartPos Property ....................................................................................................................... 299
  Statements Property ................................................................................................................... 300
  Methods ......................................................................................................................................... 301
  BreakExec Method ...................................................................................................................... 302
  ErrorOffset Method .................................................................................................................... 302
  Execute Method ........................................................................................................................... 303
  ExecuteFile Method ................................................................................................................... 303
  ExecuteNext Method .................................................................................................................. 304
  ExecuteStream Method .............................................................................................................. 304
  FindMacro Method ...................................................................................................................... 305
  MacroByName Method ............................................................................................................... 306
  Events .......................................................................................................................................... 306
  AfterExecute Event .................................................................................................................... 307
  BeforeExecute Event ................................................................................................................... 308
  OnError Event .............................................................................................................................. 308

TDALoaderOptions Class ......................................................................................................... 309
  Members ......................................................................................................................................... 309
  Properties ........................................................................................................................................ 310
  EndLine Property .......................................................................................................................... 311
  EndOffset Property ....................................................................................................................... 312
  EndPos Property .......................................................................................................................... 312
  Omit Property ............................................................................................................................... 312
  Params Property ........................................................................................................................... 313
  Script Property ............................................................................................................................ 313
  SQL Property ............................................................................................................................... 314
  StartLine Property ....................................................................................................................... 314

Types .............................................................................................................................................. 287
TDAPutDataEvent Procedure Reference .................................................................................. 287
TGetColumnDataEvent Procedure Reference ........................................................................... 287
TLoaderProgressEvent Procedure Reference .......................................................................... 288

Properties ....................................................................................................................................... 285
UseBlankValues Property .......................................................................................................... 286

Events .......................................................................................................................................... 282
OnGetColumnData Event .............................................................................................................. 283
OnProgress Event ......................................................................................................................... 284
OnPutData Event ........................................................................................................................... 284

Members ......................................................................................................................................... 285
Properties ........................................................................................................................................ 286

Methods ......................................................................................................................................... 283

Events .......................................................................................................................................... 284
OnPutData Event ........................................................................................................................... 284

Members ......................................................................................................................................... 285
Properties ........................................................................................................................................ 286

Properties ....................................................................................................................................... 285
UseBlankValues Property .......................................................................................................... 286

© 2021 Devart
Classes

TCRDataSource Class

Properties

Methods

Enumerations

TMonitorOption Enumeration

TDATraceFlag Enumeration

TDASubStatement Class

Properties

Methods

TBeforeStatementExecuteEvent Procedure Reference

TAfterStatementExecuteEvent Procedure Reference

TDAStatements Class

Members

Properties

Events

Types

TDAStatements继承

TDAStatements子类

TDAStatements成员

TDAStatements方法

TDAStatements事件

TDAStatements类型

Enumerations

TErrorAction Enumeration

Members

Properties

Methods

Events

Types

TDAStatements继承

TDAStatements子类

TDAStatements成员

TDAStatements方法

TDAStatements事件

TDAStatements类型

10 DASQLMonitor

Classes

TCustomDASQLMonitor Class

Properties

Members

Events

Types

TDAStatements继承

TDAStatements子类

TDAStatements成员

TDAStatements方法

TDAStatements事件

TDAStatements类型

Enumerations

TErrorAction Enumeration

Members

Properties

Methods

Events

Types

TDAStatements继承

TDAStatements子类

TDAStatements成员

TDAStatements方法

TDAStatements事件

TDAStatements类型

11 DBAccess

Classes

EDAError Class

Properties

Members

Component Property

Error Code Property

TCRDataSource Class

Members

Properties

CancelButton Property

Caption Property

© 2021 Devar
TCustomDADataSet Class

Methods
- OnConnectionLost Event
- StartTransaction Method
- RemoveFromPool Method
- Ping Method
- MonitorMessage Method
- GetTableNames Method
- GetKeyFieldNames Method
- GetDatabaseNames Method
- ExecProcEx Method
- ExecProc Method
- CreateSQL Method
- ApplyUpdates Method
- ApplyUpdates Method
- Server Property
- PoolingOptions Property
- Pooling Property
- Options Property
- LoginPrompt Property
- InTransaction Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConnectButton Property

Properties
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConvertEOL Property
- InTransaction Property
- Server Property
- LoginPrompt Property
- ConnectButton Property
- ConnectDialog Property
- ConvertEOL Property
- ConnectString Property
- ConnectDialog Property
- ConnectButton Property
- ConnectDialog Property
<table>
<thead>
<tr>
<th>Property/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacroByName Method</td>
<td>438</td>
</tr>
<tr>
<td>ParamByName Method</td>
<td>439</td>
</tr>
<tr>
<td>Prepare Method</td>
<td>439</td>
</tr>
<tr>
<td>RefreshRecord Method</td>
<td>440</td>
</tr>
<tr>
<td>RestoreSQL Method</td>
<td>441</td>
</tr>
<tr>
<td>SaveSQL Method</td>
<td>441</td>
</tr>
<tr>
<td>SetOrderBy Method</td>
<td>442</td>
</tr>
<tr>
<td>SQLSaved Method</td>
<td>443</td>
</tr>
<tr>
<td>UnLock Method</td>
<td>443</td>
</tr>
<tr>
<td>Events</td>
<td>444</td>
</tr>
<tr>
<td>AfterExecute Event</td>
<td>445</td>
</tr>
<tr>
<td>AfterFetch Event</td>
<td>445</td>
</tr>
<tr>
<td>AfterUpdateExecute Event</td>
<td>446</td>
</tr>
<tr>
<td>BeforeFetch Event</td>
<td>446</td>
</tr>
<tr>
<td>BeforeUpdateExecute Event</td>
<td>447</td>
</tr>
<tr>
<td>TCustomDASQL Class</td>
<td>447</td>
</tr>
<tr>
<td>Members</td>
<td>448</td>
</tr>
<tr>
<td>Properties</td>
<td>450</td>
</tr>
<tr>
<td>ChangeCursor Property</td>
<td>451</td>
</tr>
<tr>
<td>Connection Property</td>
<td>452</td>
</tr>
<tr>
<td>Debug Property</td>
<td>452</td>
</tr>
<tr>
<td>FinalSQL Property</td>
<td>453</td>
</tr>
<tr>
<td>MacroCount Property</td>
<td>454</td>
</tr>
<tr>
<td>Macros Property</td>
<td>454</td>
</tr>
<tr>
<td>ParamCheck Property</td>
<td>455</td>
</tr>
<tr>
<td>ParamCount Property</td>
<td>455</td>
</tr>
<tr>
<td>Params Property</td>
<td>456</td>
</tr>
<tr>
<td>ParamValues Property (Indexer)</td>
<td>457</td>
</tr>
<tr>
<td>Prepared Property</td>
<td>458</td>
</tr>
<tr>
<td>RowsAffected Property</td>
<td>458</td>
</tr>
<tr>
<td>SQL Property</td>
<td>459</td>
</tr>
<tr>
<td>Methods</td>
<td>459</td>
</tr>
<tr>
<td>BreakExec Method</td>
<td>460</td>
</tr>
<tr>
<td>Execute Method</td>
<td>461</td>
</tr>
<tr>
<td>Execute Method</td>
<td>461</td>
</tr>
<tr>
<td>Execute Method</td>
<td>462</td>
</tr>
<tr>
<td>Executing Method</td>
<td>462</td>
</tr>
<tr>
<td>FindMacro Method</td>
<td>463</td>
</tr>
<tr>
<td>FindParam Method</td>
<td>464</td>
</tr>
<tr>
<td>MacroByName Method</td>
<td>464</td>
</tr>
<tr>
<td>ParamByName Method</td>
<td>465</td>
</tr>
<tr>
<td>Prepare Method</td>
<td>466</td>
</tr>
<tr>
<td>UnPrepare Method</td>
<td>466</td>
</tr>
<tr>
<td>WaitForExecuting Method</td>
<td>467</td>
</tr>
<tr>
<td>Events</td>
<td>468</td>
</tr>
<tr>
<td>AfterExecute Event</td>
<td>468</td>
</tr>
<tr>
<td>TCustomDAUpdateSQL Class</td>
<td>469</td>
</tr>
<tr>
<td>Members</td>
<td>469</td>
</tr>
<tr>
<td>Properties</td>
<td>471</td>
</tr>
<tr>
<td>DataSet Property</td>
<td>472</td>
</tr>
<tr>
<td>DeleteObject Property</td>
<td>472</td>
</tr>
<tr>
<td>DeleteSQL Property</td>
<td>473</td>
</tr>
<tr>
<td>InsertObject Property</td>
<td>473</td>
</tr>
<tr>
<td>InsertSQL Property</td>
<td>474</td>
</tr>
</tbody>
</table>
Members .................................................................................................................. 502
Properties ................................................................................................................ 505
  AutoPrepare Property.............................................................................................. 508
  CacheCalcFields Property...................................................................................... 508
  CompressBlobMode Property................................................................................ 509
  DefaultValues Property........................................................................................ 509
 _DetailDelay Property.......................................................................................... 509
  FieldsOrigin Property............................................................................................ 510
  FlatBuffers Property.............................................................................................. 510
  InsertAllSetFields Property.................................................................................. 511
  LocalMasterDetail Property................................................................................... 511
  LongStrings Property............................................................................................. 512
  MasterFieldsNullable Property............................................................................. 512
  NumberRange Property.......................................................................................... 513
  QueryRecCount Property....................................................................................... 513
  QuoteNames Property............................................................................................. 514
  RemoveOnRefresh Property..................................................................................... 514
  RequiredFields Property....................................................................................... 515
  ReturnParams Property.......................................................................................... 515
  SetFieldsReadOnly Property.................................................................................. 516
  StrictUpdate Property............................................................................................ 516
  TrimFixedChar Property......................................................................................... 517
  UpdateAllFields Property....................................................................................... 517
  UpdateBatchSize Property..................................................................................... 518
TDAEncryption Class ................................................................................................. 518
Members .................................................................................................................. 519
Properties ................................................................................................................ 519
  Encryptor Property................................................................................................ 520
  Fields Property...................................................................................................... 520
TDAMapRule Class .................................................................................................... 521
Members .................................................................................................................. 521
Properties ................................................................................................................ 522
  DBLengthMax Property........................................................................................ 523
  DBLengthMin Property......................................................................................... 524
  DBScaleMax Property............................................................................................ 524
  DBScaleMin Property............................................................................................ 524
  DBType Property.................................................................................................. 525
  FieldLength Property............................................................................................ 525
  FieldName Property............................................................................................... 526
  FieldScale Property.............................................................................................. 526
  FieldType Property................................................................................................ 527
  IgnoreErrors Property........................................................................................... 527
TDAMapRules Class ................................................................................................... 527
Members .................................................................................................................. 528
Properties ................................................................................................................ 528
  IgnoreInvalidRules Property................................................................................ 529
TDAIDataClass ........................................................................................................... 529
Members .................................................................................................................. 530
Properties ................................................................................................................ 534
  Connection Property............................................................................................. 535
  MetaDataKind Property......................................................................................... 536
  Restrictions Property............................................................................................. 537
Methods .................................................................................................................... 537
  GetMetaDataKinds Method..................................................................................... 539
© 2021 Devart
<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDAParam Class</td>
<td>541</td>
</tr>
<tr>
<td>Members</td>
<td>542</td>
</tr>
<tr>
<td>Properties</td>
<td>543</td>
</tr>
<tr>
<td>AsBlob Property</td>
<td>545</td>
</tr>
<tr>
<td>AsBlobRef Property</td>
<td>545</td>
</tr>
<tr>
<td>AsFloat Property</td>
<td>546</td>
</tr>
<tr>
<td>AsInteger Property</td>
<td>546</td>
</tr>
<tr>
<td>AsLargeInt Property</td>
<td>547</td>
</tr>
<tr>
<td>AsMemo Property</td>
<td>547</td>
</tr>
<tr>
<td>AsMemoRef Property</td>
<td>548</td>
</tr>
<tr>
<td>AsSQLTimeStamp Property</td>
<td>548</td>
</tr>
<tr>
<td>AsString Property</td>
<td>548</td>
</tr>
<tr>
<td>AsWideString Property</td>
<td>549</td>
</tr>
<tr>
<td>DataType Property</td>
<td>550</td>
</tr>
<tr>
<td>IsNull Property</td>
<td>550</td>
</tr>
<tr>
<td>ParamType Property</td>
<td>551</td>
</tr>
<tr>
<td>Size Property</td>
<td>551</td>
</tr>
<tr>
<td>Value Property</td>
<td>552</td>
</tr>
<tr>
<td>Methods</td>
<td>552</td>
</tr>
<tr>
<td>AssignField Method</td>
<td>553</td>
</tr>
<tr>
<td>AssignFieldValue Method</td>
<td>553</td>
</tr>
<tr>
<td>LoadFromFile Method</td>
<td>554</td>
</tr>
<tr>
<td>LoadFromStream Method</td>
<td>555</td>
</tr>
<tr>
<td>SetBlobData Method</td>
<td>555</td>
</tr>
<tr>
<td>SetBlobData Method</td>
<td>556</td>
</tr>
<tr>
<td>TDAParams Class</td>
<td>557</td>
</tr>
<tr>
<td>Members</td>
<td>557</td>
</tr>
<tr>
<td>Properties</td>
<td>558</td>
</tr>
<tr>
<td>Items Property (Indexer)</td>
<td>558</td>
</tr>
<tr>
<td>Methods</td>
<td>559</td>
</tr>
<tr>
<td>FindParam Method</td>
<td>559</td>
</tr>
<tr>
<td>ParamByName Method</td>
<td>560</td>
</tr>
<tr>
<td>TDATransaction Class</td>
<td>561</td>
</tr>
<tr>
<td>Members</td>
<td>561</td>
</tr>
<tr>
<td>Properties</td>
<td>562</td>
</tr>
<tr>
<td>Active Property</td>
<td>563</td>
</tr>
<tr>
<td>DefaultCloseAction Property</td>
<td>563</td>
</tr>
<tr>
<td>Methods</td>
<td>564</td>
</tr>
<tr>
<td>Commit Method</td>
<td>564</td>
</tr>
<tr>
<td>Rollback Method</td>
<td>565</td>
</tr>
<tr>
<td>StartTransaction Method</td>
<td>566</td>
</tr>
<tr>
<td>Events</td>
<td>566</td>
</tr>
<tr>
<td>OnCommit Event</td>
<td>567</td>
</tr>
<tr>
<td>OnCommitRetaining Event</td>
<td>568</td>
</tr>
<tr>
<td>OnError Event</td>
<td>568</td>
</tr>
<tr>
<td>OnRollback Event</td>
<td>569</td>
</tr>
<tr>
<td>OnRollbackRetaining Event</td>
<td>570</td>
</tr>
<tr>
<td>TMacro Class</td>
<td>570</td>
</tr>
<tr>
<td>Members</td>
<td>571</td>
</tr>
<tr>
<td>Properties</td>
<td>571</td>
</tr>
<tr>
<td>Active Property</td>
<td>572</td>
</tr>
<tr>
<td>AsDateTime Property</td>
<td>573</td>
</tr>
</tbody>
</table>

© 2021 Devart
### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAttribute Class</td>
<td>574</td>
</tr>
<tr>
<td>SQLGenerator Compatibility Variable</td>
<td>575</td>
</tr>
<tr>
<td>ChangeCursor Variable</td>
<td>575</td>
</tr>
<tr>
<td>BaseSQLOldBehavior Variable</td>
<td>575</td>
</tr>
<tr>
<td>TRefreshOption Enumeration</td>
<td>584</td>
</tr>
<tr>
<td>TLockMode Enumeration</td>
<td>585</td>
</tr>
<tr>
<td>TLabelSet Enumeration</td>
<td>585</td>
</tr>
<tr>
<td>TCheckMode Enumeration</td>
<td>585</td>
</tr>
<tr>
<td>TUpdateExecuteEvent Procedure Reference</td>
<td>586</td>
</tr>
<tr>
<td>TRefreshOptions Set</td>
<td>587</td>
</tr>
<tr>
<td>Scan Method</td>
<td>587</td>
</tr>
<tr>
<td>TSmartFetchOptions Class</td>
<td>588</td>
</tr>
<tr>
<td>Items Property(Indexer)</td>
<td>577</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLGetKeyValues Property</td>
<td>588</td>
</tr>
<tr>
<td>PrefetchedFields Property</td>
<td>588</td>
</tr>
<tr>
<td>LiveBlock Property</td>
<td>587</td>
</tr>
<tr>
<td>Validate Property</td>
<td>587</td>
</tr>
<tr>
<td>MinPoolSize Property</td>
<td>586</td>
</tr>
<tr>
<td>MaxPoolSize Property</td>
<td>586</td>
</tr>
<tr>
<td>ConnectionLifetime Property</td>
<td>584</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignValues Method</td>
<td>578</td>
</tr>
<tr>
<td>Expand Method</td>
<td>579</td>
</tr>
<tr>
<td>FindMacro Method</td>
<td>579</td>
</tr>
<tr>
<td>IsEqual Method</td>
<td>580</td>
</tr>
<tr>
<td>MacroByName Method</td>
<td>581</td>
</tr>
<tr>
<td>Scan Method</td>
<td>581</td>
</tr>
</tbody>
</table>

### Enums

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCheckMode</td>
<td>594</td>
</tr>
<tr>
<td>TLabelSet</td>
<td>595</td>
</tr>
<tr>
<td>TLockMode</td>
<td>595</td>
</tr>
<tr>
<td>TRefreshOption</td>
<td>596</td>
</tr>
<tr>
<td>TRetryMode</td>
<td>597</td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQLOldBehavior Variable</td>
<td>598</td>
</tr>
<tr>
<td>ChangeCursor Variable</td>
<td>598</td>
</tr>
<tr>
<td>SQLGeneratorCompatibility Variable</td>
<td>599</td>
</tr>
</tbody>
</table>

### Section 12: MemData

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemData</td>
<td>601</td>
</tr>
<tr>
<td>TAttribute Class</td>
<td>602</td>
</tr>
<tr>
<td>Members</td>
<td>602</td>
</tr>
<tr>
<td>Properties</td>
<td>603</td>
</tr>
<tr>
<td>Class</td>
<td>Members</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>TBlob Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TCompressedBlob Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TDBObject Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TMemData Class</td>
<td></td>
</tr>
<tr>
<td>TObjectType Class</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TAppDelegate Class</td>
<td></td>
</tr>
<tr>
<td>TExceptOptions Class</td>
<td></td>
</tr>
<tr>
<td>TFindOptions Class</td>
<td></td>
</tr>
<tr>
<td>TGetObjectOptions Class</td>
<td></td>
</tr>
<tr>
<td>TLookupOptions Class</td>
<td></td>
</tr>
<tr>
<td>TNativeOptions Class</td>
<td></td>
</tr>
<tr>
<td>TNullOptions Class</td>
<td></td>
</tr>
<tr>
<td>TReportOptions Class</td>
<td></td>
</tr>
<tr>
<td>TSharedObject Class</td>
<td></td>
</tr>
<tr>
<td>TUpdateRecKinds Set</td>
<td></td>
</tr>
</tbody>
</table>
Environments

TCompressBlobMode Enumeration: 637
TConnLostCause Enumeration: 638
TDANumericType Enumeration: 639
TLocateExOption Enumeration: 639
TSortType Enumeration: 640
TUpdateRecKind Enumeration: 641

Classes

TMemDataSet Class: 642
Members: 643
Properties: 646
CachedUpdates Property: 647
IndexFieldNames Property: 648
KeyExclusive Property: 649
LocalConstraints Property: 650
LocalUpdate Property: 650
Prepared Property: 651
Ranged Property: 651
UpdateRecordTypes Property: 652
UpdatesPending Property: 653
Methods: 653
ApplyRange Method: 655
ApplyUpdates Method: 656
ApplyUpdates Method: 657
ApplyUpdates Method: 658
CancelRange Method: 659
CancelUpdates Method: 659
CommitUpdates Method: 660
DeferredPost Method: 661
EditRangeEnd Method: 661
EditRangeStart Method: 662
GetBlob Method: 663
GetBlob Method: 663
GetBlob Method: 664
Locate Method: 664
Locate Method: 665
Locate Method: 665
LocateEx Method: 667
LocateEx Method: 667
LocateEx Method: 668
Prepare Method: 669
RestoreUpdates Method: 670
RevertRecord Method: 670
SaveToXML Method: 671
SaveToXML Method: 671
SaveToXML Method: 672
SetRange Method: 673
SetRangeEnd Method: 674
SetRangeStart Method: 675
UnPrepare Method: 675
UpdateResult Method: 676
UpdateStatus Method: 677
Events: 677
Properties
----------------------------------------------- 734
ChangeNotification Property.......................... 739
CheckMode Property..................................... 740
CommandTimeout Property.............................. 740
Cursor Property.......................................... 741
DMLRefresh Property..................................... 742
Encryption Property...................................... 742
FetchAll Property........................................ 743
IsPLSQL Property......................................... 743
IsQuery Property......................................... 744
KeyFields Property...................................... 744
KeySequence Property.................................. 745
LockMode Property....................................... 746
NonBlocking Property.................................. 747
Options Property........................................ 747
OptionsDS Property...................................... 749
Params Property.......................................... 750
RefreshMode Property................................... 751
ReturnParams Property.................................. 752
RowsProcessed Property................................ 752
SequenceMode Property................................ 753
Session Property........................................ 753
SmartFetch Property..................................... 754
SQLType Property........................................ 754
StrictUpdate Property.................................. 755
UpdateObject Property.................................. 755
Methods
----------------------------------------------- 756
CreateProcCall Method.................................. 761
ErrorOffset Method...................................... 761
FindParam Method....................................... 762
GetArray Method......................................... 763
GetErrorPos Method..................................... 763
GetFile Method.......................................... 764
GetInterval Method...................................... 765
GetKeyList Method....................................... 765
GetLob Method........................................... 766
GetLobLocator Method.................................. 767
GetObject Method........................................ 768
GetRef Method........................................... 768
GetTable Method......................................... 769
GetTimeStamp Method................................... 770
OpenNext Method........................................ 771
ParamByName Method.................................... 772
TOraDataSetField Class................................ 773
Members................................................................ 773
Properties.................................................. 773
Modified Property........................................ 774
TOraDataSetOptions Class.............................. 774
Members................................................................ 775
Properties.................................................. 779
AutoClose Property...................................... 783
CacheLobs Property...................................... 784
DefaultValues Property................................. 785
DeferredLobRead Property.............................. 785
EnableBCD Property ................................................................. 786
EnableFMTBCD Property ............................................................ 786
ExtendedFieldsInfo Property ....................................................... 786
FieldsAsString Property ............................................................ 787
FullRefresh Property ................................................................. 787
PrefetchLobSize Property .......................................................... 788
PrefetchRows Property ............................................................... 788
PrepareUpdateSQL Property ........................................................ 789
ProcNamedParams Property ......................................................... 789
RawAsString Property ............................................................... 790
ReflectChangeNotify Property ..................................................... 790
ScrollableCursor Property .......................................................... 791
StatementCache Property ........................................................... 791
TemporaryLobUpdate Property ..................................................... 792
ToraDataSetOptionsDS Class ...................................................... 792
Members ................................................................................. 793
Properties ................................................................................ 796
AutoClose Property ................................................................... 799
CacheLobs Property ................................................................... 800
DefaultValue Property ................................................................ 800
DeferredLobRead Property .......................................................... 801
FieldsAsString Property ........................................................... 801
HideRowId Property ................................................................... 802
KeepPrepared Property ............................................................. 802
RawAsString Property ................................................................ 803
ScrollableCursor Property .......................................................... 803
ToraDataSource Class ................................................................. 804
Members ................................................................................. 804
ToraEncryptor Class ................................................................. 805
Members ................................................................................. 805
ToraIntervalField Class ............................................................. 806
Members ................................................................................. 806
Properties ................................................................................ 807
AsInterval Property ................................................................... 808
FracPrecision Property .............................................................. 808
LeadPrecision Property .............................................................. 809
ToraMetaData Class ................................................................. 809
Members ................................................................................. 810
ToraNestedTable Class ............................................................... 814
Members ................................................................................. 814
Properties ................................................................................ 817
Ref Property .............................................................................. 819
Table Property ........................................................................... 819
ToraNumberField Class ............................................................ 820
Members ................................................................................. 820
Properties ................................................................................ 821
AsNumber Property ................................................................... 821
ToraParam Class ....................................................................... 822
Members ................................................................................. 823
Properties ................................................................................ 826
AsArray Property ....................................................................... 830
AsBFile Property ........................................................................ 830
AsBLOBLocator Property ............................................................ 831
AsCLOBLocator Property ............................................................ 831
## TOraSessionOptions Class

### Properties

- `EnableIntegers` Property
- `EnableFMTBCD` Property
- `EnableBCD` Property
- `ConvertEOL` Property
- `ConnectionTimeout` Property
- `ClientIdentifier` Property
- `Charset` Property
- `OnFailover` Event
- `OnConnectChange` Event
- `StartTransaction` Method
- `StartTransaction` Method
- `Savepoint` Method
- `RollbackToSavepoint` Method
- `ChangePassword` Method
- `AssignSvcCtx` Method
- `AssignLDA` Method
- `AssignConnect` Method
- `Username` Property
- `ThreadSafety` Property
- `OracleVersion` Property
- `Passw ord` Property
- `PoolingOptions` Property
- `ProxySession` Property
- `Schema` Property
- `Server` Property
- `SQL` Property
- `SSLOptions` Property
- `ThreadSafety` Property
- `Username` Property

### Methods

- `AssignConnect` Method
- `AssignLDA` Method
- `AssignSvcCtx` Method
- `AssignSvcCtx` Method
- `AssignSvcCtx` Method
- `ChangePassw ord` Method
- `GetSequenceNames` Method
- `ParamByName` Method
- `RollbackToSavepoint` Method
- `Savepoint` Method
- `StartTransaction` Method
- `StartTransaction` Method
- `StartTransaction` Method

### Events

- `OnConnectChange` Event
- `OnFailover` Event
- `OnInfoMessage` Event

---

### TOraSessionOptions Class

### Members

- `EnableB C D` Property
- `EnableFMTBCD` Property
- `EnableIntegers` Property
- `EnableLargeint` Property
- `EnableNumbers` Property
- `EnableO raTimestamp` Property
- `IPv er sion` Property
## TOraSQL Class

**Members**

- GetSessionPID Method
- PreparedStatement Method
- ResultSet Method
- Statement Method
- StatementSet Method
- SQLSet Method

**Properties**

- Properties
  - MaxTraceFileSize Property
  - Enabled Property
  - Format Property

**Methods**

- Properties
  - PlSqlTraceLimit Method
  - GetSessionPID Method
  - SqlTraceMode Property
  - MaxTraceFileSize Property
  - Enabled Property
  - Format Property
  - StoredProcName Property
  - CommandTimeout Property
  - ArrayLength Property
  - params Property
  - rowsProcessed Property
  - Session Property
  - SQLType Property
  - TemporaryLobUpdate Property

## TOraStoredProc Class

**Members**

- CreateProcCall Method
- ErrorOffset Method
- FindParam Method
- ParamByName Method

**Properties**

- Properties
  - LockMode Property
  - Overload Property
  - StoredProcName Property

**Methods**

- Properties
  - ExecProc Method
  - PrepareSQL Method

## TOraTimeStampField Class

**Members**

- AsTimeStamp Property
- Format Property

**Properties**

- Properties
  - AsTimeStamp Property
  - Format Property

## TOraTrace Class

**Members**

- GetSessionPID Method
- GetTraceFileName Method
- PlSqlTraceComment Method
- PlSqlTraceLimit Method
- PlSqlTracePause Method
<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OraAlerter Class</td>
<td>993</td>
</tr>
<tr>
<td>Members</td>
<td>994</td>
</tr>
<tr>
<td>Properties</td>
<td>995</td>
</tr>
<tr>
<td>AutoCommit Property</td>
<td>996</td>
</tr>
<tr>
<td>Events Property</td>
<td>997</td>
</tr>
<tr>
<td>EventType Property</td>
<td>999</td>
</tr>
<tr>
<td>Interval Property</td>
<td>1000</td>
</tr>
<tr>
<td>Session Property</td>
<td>1000</td>
</tr>
<tr>
<td>TimeOut Property</td>
<td>1001</td>
</tr>
<tr>
<td>Methods</td>
<td>1001</td>
</tr>
<tr>
<td>GetMessage Method</td>
<td>1002</td>
</tr>
<tr>
<td>GetMessage Method</td>
<td>1003</td>
</tr>
<tr>
<td>GetMessage Method</td>
<td>1003</td>
</tr>
<tr>
<td>NextItemType Method</td>
<td>1004</td>
</tr>
<tr>
<td>NextMessageType Method</td>
<td>1004</td>
</tr>
<tr>
<td>PackMessage Method</td>
<td>1005</td>
</tr>
<tr>
<td>PurgePipe Method</td>
<td>1006</td>
</tr>
<tr>
<td>PutMessage Method</td>
<td>1006</td>
</tr>
<tr>
<td>SendMessage Method</td>
<td>1007</td>
</tr>
<tr>
<td>SendPipeMessage Method</td>
<td>1007</td>
</tr>
<tr>
<td>UnpackMessage Method</td>
<td>1008</td>
</tr>
<tr>
<td>UnpackMessage Method</td>
<td>1008</td>
</tr>
<tr>
<td>UnpackMessage Method</td>
<td>1009</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Types</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TConnectChangeEvent Procedure Reference</td>
<td>985</td>
</tr>
<tr>
<td>TRalfoverEvent Procedure Reference</td>
<td>986</td>
</tr>
<tr>
<td>TOraChangeNotificationEvent Procedure</td>
<td>986</td>
</tr>
<tr>
<td>TRSqlTraceMode Set</td>
<td>987</td>
</tr>
<tr>
<td>TSqlTraceMode Set</td>
<td>987</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFailoverState Enumeration</td>
<td>988</td>
</tr>
<tr>
<td>TFailoverType Enumeration</td>
<td>989</td>
</tr>
<tr>
<td>TOraIsolationLevel Enumeration</td>
<td>989</td>
</tr>
<tr>
<td>TRefreshMode Enumeration</td>
<td>990</td>
</tr>
<tr>
<td>TSequenceMode Enumeration</td>
<td>991</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefSession Variable</td>
<td>992</td>
</tr>
<tr>
<td>OraQueryCompatibilityMode Variable</td>
<td>992</td>
</tr>
<tr>
<td>Sessions Variable</td>
<td>993</td>
</tr>
<tr>
<td>UseDefSession Variable</td>
<td>993</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16 OraAlerter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>994</td>
</tr>
<tr>
<td>Members</td>
<td>995</td>
</tr>
<tr>
<td>Properties</td>
<td>997</td>
</tr>
<tr>
<td>AutoCommit Property</td>
<td>998</td>
</tr>
<tr>
<td>Events Property</td>
<td>999</td>
</tr>
<tr>
<td>EventType Property</td>
<td>999</td>
</tr>
<tr>
<td>Interval Property</td>
<td>1000</td>
</tr>
<tr>
<td>Session Property</td>
<td>1000</td>
</tr>
<tr>
<td>TimeOut Property</td>
<td>1001</td>
</tr>
<tr>
<td>Methods</td>
<td>1001</td>
</tr>
<tr>
<td>GetMessage Method</td>
<td>1002</td>
</tr>
<tr>
<td>GetMessage Method</td>
<td>1003</td>
</tr>
<tr>
<td>GetMessage Method</td>
<td>1003</td>
</tr>
<tr>
<td>NextItemType Method</td>
<td>1004</td>
</tr>
<tr>
<td>NextMessageType Method</td>
<td>1004</td>
</tr>
<tr>
<td>PackMessage Method</td>
<td>1005</td>
</tr>
<tr>
<td>PurgePipe Method</td>
<td>1006</td>
</tr>
<tr>
<td>PutMessage Method</td>
<td>1006</td>
</tr>
<tr>
<td>SendMessage Method</td>
<td>1007</td>
</tr>
<tr>
<td>SendPipeMessage Method</td>
<td>1007</td>
</tr>
<tr>
<td>UnpackMessage Method</td>
<td>1008</td>
</tr>
<tr>
<td>UnpackMessage Method</td>
<td>1008</td>
</tr>
<tr>
<td>UnpackMessage Method</td>
<td>1009</td>
</tr>
</tbody>
</table>
Oracle Data Access Components

17 OraAQ .......................................................................................................................... 1013

17.1 OraQueue Class .......................................................................................................... 1015

Properties ......................................................................................................................... 1018

- ConsumerName Property .......................................................................................... 1019
- Correlation Property ................................................................................................. 1019
- DequeueCondition Property ..................................................................................... 1020
- DequeueMode Property .............................................................................................. 1021
- MessageId Property .................................................................................................. 1021
- Navigation Property ................................................................................................ 1022
- Transformation Property .......................................................................................... 1022
- Visibility Property ..................................................................................................... 1023
- WaitTimeout Property ............................................................................................... 1023

Members .......................................................................................................................... 1024

- DeliveryMode Property ............................................................................................ 1025
- RelativeMessageId Property ...................................................................................... 1026
- SequenceDeviation Property ..................................................................................... 1027
- Transformation Property .......................................................................................... 1028
- Visibility Property ..................................................................................................... 1028

Methods ............................................................................................................................ 1036

- Dequeue Method ........................................................................................................ 1036
- Enqueue Method ....................................................................................................... 1037
- Dequeue Method ........................................................................................................ 1038
- Dequeue Method ........................................................................................................ 1038
- Dequeue Method ........................................................................................................ 1039
- Dequeue Array Method .............................................................................................. 1040
- Enqueue Method ....................................................................................................... 1041
- Enqueue Method ....................................................................................................... 1042
- Enqueue Method ....................................................................................................... 1042
- Enqueue Method ....................................................................................................... 1043
- Enqueue Method ....................................................................................................... 1043

© 2021 Devart
### TораQueueAdmin Class

<table>
<thead>
<tr>
<th>Members</th>
<th>1048</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>1052</td>
</tr>
<tr>
<td>Methods</td>
<td>1057</td>
</tr>
</tbody>
</table>

#### Properties

- Comment Property
- MaxRetries Property
- MultipleConsumers Property
- QueueName Property
- QueueTableName Property
- QueueType Property
- RetentionPolicy Property
- RetryDelay Property
- Session Property

#### Methods

- AddSubscriber Method
- AlterComment Method
- AlterMaxRetries Method
- AlterPropagationSchedule Method
- AlterQueue Method
- AlterRetentionTime Method
- AlterRetryDelay Method
- AlterSubscriber Method
- CreateQueue Method
- DisablePropagationSchedule Method
- DropQueue Method
- EnablePropagationSchedule Method
- GetSubscribers Method
- GrantQueuePrivilege Method
- ReadQueueProperties Method
- RemoveSubscriber Method
- RevokeQueuePrivilege Method
- SchedulePropagation Method
- StartDequeue Method
- StartEnqueue Method
- StartQueue Method
- StopDequeue Method
- StopEnqueue Method
- StopQueue Method
- UnschedulePropagation Method
- VerifyQueueTypes Method

### TораQueueTable Class

<table>
<thead>
<tr>
<th>Members</th>
<th>1078</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>1080</td>
</tr>
</tbody>
</table>

- Comment Property
- Compatible Property
- MessageGrouping Property
- MultipleConsumers Property
- PayloadTableName Property
- PrimaryInstance Property

© 2021 Devart
<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQueueAgents Class</td>
<td>1097</td>
</tr>
<tr>
<td>TQueueMessage Class</td>
<td>1104</td>
</tr>
<tr>
<td>TQueueMessageProperties Class</td>
<td>1107</td>
</tr>
<tr>
<td>QueueTableName Property</td>
<td>1085</td>
</tr>
<tr>
<td>SecondaryInstance Property</td>
<td>1085</td>
</tr>
<tr>
<td>Secure Property</td>
<td>1086</td>
</tr>
<tr>
<td>Session Property</td>
<td>1087</td>
</tr>
<tr>
<td>SortList Property</td>
<td>1087</td>
</tr>
<tr>
<td>StorageClause Property</td>
<td>1088</td>
</tr>
<tr>
<td>Methods</td>
<td>1088</td>
</tr>
<tr>
<td>AlterComment Method</td>
<td>1089</td>
</tr>
<tr>
<td>AlterPrimaryInstance Method</td>
<td>1090</td>
</tr>
<tr>
<td>AlterQueueTable Method</td>
<td>1090</td>
</tr>
<tr>
<td>AlterSecondaryInstance Method</td>
<td>1091</td>
</tr>
<tr>
<td>CreateQueueTable Method</td>
<td>1092</td>
</tr>
<tr>
<td>DropQueueTable Method</td>
<td>1093</td>
</tr>
<tr>
<td>GrantSystemPrivilege Method</td>
<td>1093</td>
</tr>
<tr>
<td>MigrateQueueTable Method</td>
<td>1094</td>
</tr>
<tr>
<td>PurgeQueueTable Method</td>
<td>1095</td>
</tr>
<tr>
<td>ReadQueueTableProperties Method</td>
<td>1095</td>
</tr>
<tr>
<td>RevokeSystemPrivilege Method</td>
<td>1096</td>
</tr>
<tr>
<td>Members</td>
<td>1098</td>
</tr>
<tr>
<td>Properties</td>
<td>1098</td>
</tr>
<tr>
<td>Address Property</td>
<td>1099</td>
</tr>
<tr>
<td>Name Property</td>
<td>1099</td>
</tr>
<tr>
<td>Protocol Property</td>
<td>1099</td>
</tr>
<tr>
<td>Members</td>
<td>1101</td>
</tr>
<tr>
<td>Properties</td>
<td>1101</td>
</tr>
<tr>
<td>Items Property(Indexer)</td>
<td>1102</td>
</tr>
<tr>
<td>Methods</td>
<td>1102</td>
</tr>
<tr>
<td>Add Method</td>
<td>1103</td>
</tr>
<tr>
<td>Insert Method</td>
<td>1103</td>
</tr>
<tr>
<td>Members</td>
<td>1104</td>
</tr>
<tr>
<td>Properties</td>
<td>1105</td>
</tr>
<tr>
<td>MessageId Property</td>
<td>1106</td>
</tr>
<tr>
<td>MessageProperties Property</td>
<td>1106</td>
</tr>
<tr>
<td>Raw Payload Property</td>
<td>1107</td>
</tr>
<tr>
<td>StringPayload Property</td>
<td>1107</td>
</tr>
<tr>
<td>Members</td>
<td>1108</td>
</tr>
<tr>
<td>Properties</td>
<td>1109</td>
</tr>
<tr>
<td>Attempts Property</td>
<td>1111</td>
</tr>
<tr>
<td>Correlation Property</td>
<td>1111</td>
</tr>
<tr>
<td>Delay Property</td>
<td>1111</td>
</tr>
<tr>
<td>DeliveryMode Property</td>
<td>1112</td>
</tr>
<tr>
<td>EnqueueTime Property</td>
<td>1113</td>
</tr>
<tr>
<td>ExceptionQueue Property</td>
<td>1113</td>
</tr>
<tr>
<td>Expiration Property</td>
<td>1113</td>
</tr>
<tr>
<td>OriginalMessageId Property</td>
<td>1114</td>
</tr>
<tr>
<td>Priority Property</td>
<td>1114</td>
</tr>
<tr>
<td>RecipientList Property</td>
<td>1115</td>
</tr>
<tr>
<td>SenderId Property</td>
<td>1115</td>
</tr>
<tr>
<td>State Property</td>
<td>1116</td>
</tr>
<tr>
<td>TransactionGroup Property</td>
<td>1116</td>
</tr>
</tbody>
</table>
18 OraCall ................................................................. 1124

Classes ........................................................................ 1124
TOraHome Class .......................................................... 1124
  Members ..................................................................... 1125
  Properties .................................................................... 1126
    Name Property ............................................................ 1127
    OCIConnection Property ............................................ 1127
    OCEncoding Property ................................................ 1127
    OCIDLL Property ....................................................... 1128
    OCVision Property .................................................... 1128
    OCVersionSt Property ............................................... 1128
    Path Property ........................................................... 1129
    PossibleOICallStyles Property .................................. 1129
    TNSPath Property ...................................................... 1129

19 OraClasses ........................................................... 1130

Classes ......................................................................... 1131
TNotifyTableChanges Class ............................................ 1132
  Members ..................................................................... 1132
  Properties .................................................................... 1133
    Changes Property(indexer) .......................................... 1133
TOraCursor Class .......................................................... 1134
  Members ..................................................................... 1134
  Properties .................................................................... 1135
    CDA Property ............................................................ 1136
    OCIConnection Property ............................................ 1136
    OCIDLL Property ....................................................... 1137
    OCIStmt Property ..................................................... 1137
    State Property .......................................................... 1137
  Methods ..................................................................... 1138
    AllocCursor Method .................................................. 1138
    CanFetch Method ...................................................... 1139
    FreeCursor Method ................................................... 1139
TOraFile Class .............................................................. 1140
  Members ..................................................................... 1141
  Properties .................................................................... 1143
    FileDir Property ........................................................ 1145
    FileName Property ..................................................... 1145
  Methods ..................................................................... 1146
    Close Method ............................................................ 1147
    Exists Method ........................................................... 1148
    IsOpen Method .......................................................... 1149
    Open Method ............................................................. 1149
### TOraInterval Class

#### Members
- AsString Property
- DescriptorType Property
- FracPrecision Property
- IsNull Property
- LeadPrecision Property
- OCIInterval Property
- OCIIntervalPtr Property

#### Methods
- AllocInterval Method
- Compare Method
- FreeInterval Method
- GetDaySecond Method
- GetYearMonth Method
- SetDaySecond Method
- SetYearMonth Method

#### Properties
- Cached Property
- OCLobLocator Property
- OCLobLocatorPtr Property
- OCSvcCtx Property

### TOraNumber Class

#### Members
- AsFloat Property
- AsInteger Property
- AsLargeInt Property
- AsString Property
- IsNull Property
- OCINumber Property
- OCINumberPtr Property

#### Methods
- AssignTo Method
- Compare Method

### TOraTimeStamp Class

#### Members
Properties ................................................................. 1186
AsDateTime Property.................................................. 1187
AsString Property..................................................... 1188
DescriptorType Property........................................... 1188
Format Property....................................................... 1189
IsNull Property....................................................... 1189
OCIDateTime Property.............................................. 1190
OCIDateTimePtr Property......................................... 1190
Precision Property................................................... 1191
TimeZone Property................................................... 1191
Methods ................................................................. 1192
AllocDateTime Method.............................................. 1193
Compare Method...................................................... 1193
Construct Method.................................................... 1194
GetDate Method....................................................... 1195
GetTime Method....................................................... 1196
GetTimeZoneOffset Method....................................... 1197
SetDate Method....................................................... 1197
SetTime Method....................................................... 1198
SetTimeZoneOffset Method....................................... 1199
Enumerations ........................................................... 1199
TChangeNotifyEventType Enumeration...................... 1200
TConnectMode Enumeration...................................... 1201
TMessageType Enumeration..................................... 1201
TOptimizerMode Enumeration.................................... 1202
Variables ............................................................... 1203
FloatPrecision Variable............................................ 1204
IntegerPrecision Variable....................................... 1205
LargeIntPrecision Variable..................................... 1205

20 OraConnectionPool .................................................. 1205
Enumerations .......................................................... 1206
TOraPoolingType Enumeration.................................. 1206

21 OraErrHand .......................................................... 1206
Classes .................................................................... 1207
TOraErrorHandler Class.......................................... 1207
Members .................................................................. 1208
Properties ................................................................ 1209
Active Property...................................................... 1209
Debug Property....................................................... 1210
Session Property.................................................... 1211
TableName Property................................................ 1211
Methods .................................................................. 1212
Close Method......................................................... 1212
Open Method......................................................... 1213
Events .................................................................. 1213
OnError Event........................................................ 1214
Types ..................................................................... 1214
TOpraErrorEvent Procedure Reference........................ 1214

22 OraErr .................................................................. 1215
Classes .................................................................... 1215
EOraError Class........................................................ 1216
Members .................................................................. 1216
Properties ................................................................ 1217
### 23 OraLoader

**Classes**

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraLoader Class</td>
<td>1223</td>
</tr>
<tr>
<td>Members</td>
<td>1225</td>
</tr>
<tr>
<td>Properties</td>
<td>1226</td>
</tr>
<tr>
<td>LoadMode Property</td>
<td>1227</td>
</tr>
<tr>
<td>Session Property</td>
<td>1227</td>
</tr>
<tr>
<td>Events</td>
<td>1228</td>
</tr>
<tr>
<td>OnError Event</td>
<td>1229</td>
</tr>
<tr>
<td>OnGetColumnData Event</td>
<td>1229</td>
</tr>
<tr>
<td>OnPutData Event</td>
<td>1230</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>1219</td>
</tr>
<tr>
<td><strong>Members</strong></td>
<td>1220</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1221</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1222</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1223</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1224</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1225</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1226</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1227</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1228</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1229</td>
</tr>
<tr>
<td><strong>Enumeration</strong></td>
<td>1230</td>
</tr>
</tbody>
</table>

**Types**

- TDErrorEvent Procedure Reference: 1231
- TDErrorAction Enumeration: 1233
- TLoadMode Enumeration: 1234

**Enumerations**

- TSecurityLevel Enumeration: 1235

**Variables**

- DataIntegrityLevel Variable: 1237
- EncryptionLevel Variable: 1237
- PacketSize Variable: 1238

### 24 OraNet

**Enumerations**

- TSecurityLevel Enumeration: 1235

**Variables**

- DataIntegrityLevel Variable: 1237
- EncryptionLevel Variable: 1237
- PacketSize Variable: 1238

### 25 OraObjects

**Classes**

<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraArray Class</td>
<td>1239</td>
</tr>
<tr>
<td>Members</td>
<td>1240</td>
</tr>
<tr>
<td>Properties</td>
<td>1245</td>
</tr>
<tr>
<td>ItemAsAnsiString Property(Indexer)</td>
<td>1248</td>
</tr>
<tr>
<td>ItemAsDateTime Property(Indexer)</td>
<td>1249</td>
</tr>
<tr>
<td>ItemAsFloat Property(Indexer)</td>
<td>1249</td>
</tr>
<tr>
<td>ItemAsInteger Property(Indexer)</td>
<td>1250</td>
</tr>
<tr>
<td>ItemAsInterval Property(Indexer)</td>
<td>1250</td>
</tr>
<tr>
<td>ItemAsLargeInt Property(Indexer)</td>
<td>1251</td>
</tr>
<tr>
<td>ItemAsLob Property(Indexer)</td>
<td>1251</td>
</tr>
<tr>
<td>ItemAsObject Property(Indexer)</td>
<td>1252</td>
</tr>
<tr>
<td>ItemAsOCString Property(Indexer)</td>
<td>1253</td>
</tr>
<tr>
<td>ItemAsRef Property(Indexer)</td>
<td>1253</td>
</tr>
<tr>
<td>ItemAsString Property(Indexer)</td>
<td>1254</td>
</tr>
<tr>
<td>ItemAsTimeStamp Property(Indexer)</td>
<td>1254</td>
</tr>
<tr>
<td>ItemAsWideString Property(Indexer)</td>
<td>1255</td>
</tr>
<tr>
<td>ItemExists Property(Indexer)</td>
<td>1256</td>
</tr>
<tr>
<td>ItemIsNull Property(Indexer)</td>
<td>1256</td>
</tr>
</tbody>
</table>
itemSubType Property ................................................................. 1257
itemType Property ................................................................. 1257
maxSize Property .................................................................... 1258
size Property ........................................................................... 1258
methods .................................................................................. 1259
appendItem Method .................................................................. 1260
assign Method .......................................................................... 1261
clear Method ............................................................................ 1261
insertItem Method .................................................................... 1262
TOraNestTable Class ............................................................... 1262
members .................................................................................. 1263
methods .................................................................................. 1267
deleteItem Method .................................................................. 1269
TOraObject Class ................................................................. 1269
members .................................................................................. 1270
properties .............................................................................. 1273
attrAsArray Property (Indexer) .............................................. 1275
attrAsDateTime Property (Indexer) ........................................ 1275
attrAsFloat Property (Indexer) ............................................... 1276
attrAsInteger Property (Indexer) ............................................ 1276
attrAsLob Property (Indexer) .................................................. 1277
attrAsObject Property (Indexer) ............................................ 1277
attrAsOCIDate Property (Indexer) ......................................... 1278
attrAsOCINumber Property (Indexer) .................................... 1279
attrAsOCIString Property (Indexer) ....................................... 1279
attrAsString Property (Indexer) .............................................. 1280
attrIsNull Property (Indexer) .................................................. 1280
indicator Property ................................................................. 1281
instance Property ................................................................... 1281
isNull Property ...................................................................... 1282
objectType Property .............................................................. 1282
OCISvcCtx Property ............................................................. 1283
methods .................................................................................. 1283
allocObject Method ............................................................... 1284
allocObject Method ............................................................... 1285
allocObject Method ............................................................... 1285
allocObject Method ............................................................... 1286
allocObject Method ............................................................... 1286
assign Method ....................................................................... 1287
createObject Method ............................................................ 1287
exists Method ........................................................................ 1288
flush Method ......................................................................... 1288
freeObject Method ............................................................... 1289
isDirty Method ..................................................................... 1289
isLocked Method ................................................................... 1290
lock Method .......................................................................... 1290
markDelete Method .............................................................. 1291
markUpdate Method ............................................................. 1291
refresh Method ....................................................................... 1292
unmark Method ...................................................................... 1292
TOraRef Class ........................................................................ 1293
members .................................................................................. 1293
properties .............................................................................. 1296
asHex Property ...................................................................... 1298
<table>
<thead>
<tr>
<th>Class</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraScript Class</td>
<td></td>
</tr>
<tr>
<td>TOraProvider Class</td>
<td></td>
</tr>
<tr>
<td>TCUSTOMORAPACKAGE CLASS</td>
<td></td>
</tr>
<tr>
<td>OraPackage</td>
<td>1326</td>
</tr>
<tr>
<td></td>
<td>1327</td>
</tr>
<tr>
<td>Classes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1337</td>
</tr>
<tr>
<td>OraScript</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1338</td>
</tr>
<tr>
<td>Classes</td>
<td></td>
</tr>
<tr>
<td>29 OraSmart</td>
<td>1350</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Classes</td>
<td>1351</td>
</tr>
<tr>
<td>TCustomSmartQuery Class</td>
<td>1352</td>
</tr>
<tr>
<td>Members</td>
<td>1353</td>
</tr>
<tr>
<td>Properties</td>
<td>1364</td>
</tr>
<tr>
<td>DependEvents Property</td>
<td>1370</td>
</tr>
<tr>
<td>Expand Property</td>
<td>1370</td>
</tr>
<tr>
<td>Options Property</td>
<td>1371</td>
</tr>
<tr>
<td>RefreshEvent Property</td>
<td>1372</td>
</tr>
<tr>
<td>SmartRefresh Property</td>
<td>1372</td>
</tr>
<tr>
<td>SmartState Property</td>
<td>1373</td>
</tr>
<tr>
<td>Methods</td>
<td>1373</td>
</tr>
<tr>
<td>View Method</td>
<td>1378</td>
</tr>
<tr>
<td>Events</td>
<td>1379</td>
</tr>
<tr>
<td>AfterSmartRefresh Event</td>
<td>1380</td>
</tr>
<tr>
<td>RefreshFields Event</td>
<td>1380</td>
</tr>
<tr>
<td>TOraTable Class</td>
<td>1381</td>
</tr>
<tr>
<td>Members</td>
<td>1382</td>
</tr>
<tr>
<td>Properties</td>
<td>1393</td>
</tr>
<tr>
<td>FetchAll Property</td>
<td>1399</td>
</tr>
<tr>
<td>LockMode Property</td>
<td>1400</td>
</tr>
<tr>
<td>OrderFields Property</td>
<td>1400</td>
</tr>
<tr>
<td>TableName Property</td>
<td>1401</td>
</tr>
<tr>
<td>Methods</td>
<td>1401</td>
</tr>
<tr>
<td>EmptyTable Method</td>
<td>1406</td>
</tr>
<tr>
<td>PrepareSQL Method</td>
<td>1407</td>
</tr>
<tr>
<td>TSmartQuery Class</td>
<td>1407</td>
</tr>
<tr>
<td>Members</td>
<td>1408</td>
</tr>
<tr>
<td>TSmartQueryOptions Class</td>
<td>1419</td>
</tr>
<tr>
<td>Members</td>
<td>1420</td>
</tr>
</tbody>
</table>

| 30 OraSQLMonitor | 1425 |
| Classes | 1425 |
| TOraSQLMonitor Class | 1425 |
| Members | 1426 |

| 31 OraTransaction | 1427 |
| Classes | 1427 |
| TOraTransaction Class | 1428 |
| Members | 1428 |
### VirtualDataSet

#### Classes

- TCustomVirtualDataSet Class
- TVirtualDataSet Class

#### Properties

- Active Property
- BranchQualifiers Property
- DefaultSession Property
- GlobalCoordinator Property
- InactiveTimeOut Property
- IsolationLevel Property
- ResumeTimeOut Property
- Sessions Property
- SessionsCount Property
- TransactionId Property
- TransactionName Property

#### Methods

- AddSession Method
- ClearSessions Method
- Detach Method
- RemoveSession Method
- Resume Method
- RollbackToSavepoint Method
- Savepoint Method
- StartTransaction Method
- StartTransaction Method
- StartTransaction Method

#### Events

- OnError Event

#### Enumerations

- TGlobalCoordinator Enumeration

### VirtualTable

#### Classes

- TVirtualTable Class

#### Properties

- DefaultSortType Property

#### Methods

- Assign Method
- LoadFromFile Method
- LoadFromStream Method
1 What's New

02-Mar-21 New Features in ODAC 11.4:
- Oracle 21c is supported
- RAD Studio 10.4.2 Sydney is supported
- macOS 11 Big Sur is supported
- iOS 14 is supported
- Android 11 is supported
- Performance of batch operations is improved
- Performance of the FindFirst, FindNext, FindLast, and FindPrior methods is improved
- The PrefetchRows option in the Direct mode is supported
- Data fetch performance in the Direct mode is improved
- LOB read/write performance is improved

26-Aug-20 New Features in ODAC 11.3:
- Oracle 20c is supported
- Connection via SSL protocol is supported
- Connection via SSH protocol is supported
- Connection via HTTP tunnel is supported
- Lazarus 2.0.10 and FPC 3.2.0 are supported
- Performance of Batch Insert, Update, and Delete operations is improved

01-Jun-20 New Features in ODAC 11.2:
- RAD Studio 10.4 Sydney is supported
- Lazarus 2.0.8 is supported
- macOS 64-bit in Lazarus is supported
- Mapping the FLOAT Oracle data type to the ftNumber field is added

26-Nov-19 New Features in ODAC 11.1:
- Android 64-bit is supported
- Lazarus 2.0.6 is supported
- Oracle 19c is supported
- Now Trial edition for macOS and Linux is fully functional
- Long database object names is supported
22-Jul-19 New Features in ODAC 11.0:
- macOS 64-bit is supported
- Release 2 for RAD Studio 10.3 Rio, Delphi 10.3 Rio, and C++Builder 10.3 Rio is now required

24-Jun-19 New Features in ODAC 10.4:
- Lazarus 2.0.2 is supported
- CommandTimeout is supported
- Support for ChangePassword in the Direct mode is improved
- The DefaultSortType property for TVirtualTable is added
- Performance of the SaveToFile/LoadFromFile methods of TVirtualTable is significantly increased

26-Nov-18 New Features in ODAC 10.3:
- RAD Studio 10.3 Rio is supported
- Oracle 18c is supported
- Implicit result sets in Oracle 12 are supported
- The OpenNext method for opening a next cursor or implicit result set is added
- Support of UPPER and LOWER functions for Unified SQL is added

09-Jul-18 New Features in ODAC 10.2:
- Lazarus 1.8.4 is supported
- Performance of data fetching in the Direct mode is improved
- Performance of describing stored procedures in the Direct mode is improved
- Performance of batch operations is improved
- Demo projects for IntraWeb 14 are added
- Now the TOraTimeStamp.AsDateTime method returns TDateTime value with milliseconds
- Now non-compiled stored procedures can be described in the Direct mode
- Performance of describing stored procedures in the Direct mode is improved
- Support for TIMESTAMP WITH TIMEZONE in the Direct mode is improved

19-Sep-17 New Features in ODAC 10.1:
- Oracle 12c connection modes (SYSBACKUP, SYSDG, SYSKM) in the Direct mode are supported
- OS authentication in the Direct mode is supported
• NChar literal replacement is supported

05-Apr-17 New Features in ODAC 10.0:
• RAD Studio 10.2 Tokyo is supported
• Linux in RAD Studio 10.2 Tokyo is supported
• Lazarus 1.6.4 and Free Pascal 3.0.2 is supported
• Oracle Encryption in the Direct mode is supported
• Oracle Data Integrity in the Direct mode is supported
• Oracle Cloud (DBaaS) in the Direct mode is supported
• Oracle 12c authentication in the Direct mode is supported
• SECUREFILE in the Direct mode is supported
• Prefetch LOBs for Oracle 11 and higher is supported
• EDITIONABLE and NONEDITIONABLE clause is supported
• The PrefetchLobSize option is added
• ANYDATA is supported
• Field size detecting for servers with multi-byte charset when UseUnicode=False is improved
• Now NUMBER data type without fixed scale has precision=39 and scale=39 instead of 38

25-Apr-16 New Features in ODAC 9.7:
• RAD Studio 10.1 Berlin is supported
• Lazarus 1.6 and FPC 3.0.0 is supported
• Support for the BETWEEN statement in TDADataSet.Filter is added
• Data Type Mapping performance is improved
• Performance of TDALoader on loading data from TDataSet is improved

09-Sep-15 New Features in ODAC 9.6:
• RAD Studio 10 Seattle is supported
• Now NULL and empty strings are different values for ftOraLob and ftOraClob parameters
• Now Trial for Win64 is a fully functional Professional Edition
• Support for Offset is added for DML arrays
• Support for OraNet.PacketSize is added to improve performance in VPN and Wireless networks
• Support for Object References in the Direct mode is added
• Support for Object attributes with the XML data type is added
14-Apr-15 New Features in ODAC 9.5:
- RAD Studio XE8 is supported
- AppMethod is supported
- Direct mode in Lazarus is supported
- Now the Direct mode is supplied as source code
- Support for Objects in the Direct mode is added
- Support for XML in the Direct mode is added
- Support for EZCONNECT in the Direct mode is added
- Support for fields with Cursor data type in the Direct mode is added
- Now statements with RETURN INTO clauses can return RowsAffected in the Direct mode

15-Sep-14 New Features in ODAC 9.4:
- RAD Studio XE7 is supported
- Lazarus 1.2.4 is supported
- RAC server support is improved
- Support for WITH FUNCTION clause for Oracle 12c is added
- GetServerList doesn't cut the WORLD postfix anymore
- The HideRowId option is added
- Demo projects for FastReport 5 are added
- The TCustomDADataSet.GetKeyFieldNames method is added
- The ConstraintColumns metadata kind for the TDAMetadata component is added
- Workaround for the bug with calling Halt in the OnCreate event is added

29-Apr-14 New Features in ODAC 9.3:
- RAD Studio XE6 is supported
- Android in C++Builder XE6 is supported
- Lazarus 1.2.2 and FPC 2.6.4 is supported
- SmartFetch mode for TDataSet descendants is added
- Now update queries inside TDataSet descendants have correct owner
- Possibility to assign external SvcCtx to connection is added
- DataTypeMapping conversion from XMLType to ftString is added
- DataTypeMapping conversion from Interval to ftString is added
- The TOraDataSetOptions.MasterFieldsNullable property is added

25-Dec-13 New Features in ODAC 9.2:
iOS in C++Builder XE5 is supported
RAD Studio XE5 Update 2 is now required
Now .obj and .o files are supplied for C++Builder
An ability to establish OCI and Direct connections in the same application is supported
New Oracle 12c connection modes are added (SYSBACKUP, SYSDG, SYSKM)
The AsTimeStamp property is added to the TOraTimeStamp class
Compatibility of migrating floating-point fields from other components is improved

18-Sep-13 New Features in ODAC 9.1:
- RAD Studio XE5 is supported
- Application development for Android is supported
- Lazarus 1.0.12 is supported
- Performance is improved
- Automatic checking for new versions is added
- Flexible management of conditions in the WHERE clause is added
- The possibility to use conditions is added
- IPv6 protocol support is added
- The possibility to use ranges is added
- Support of the IN keyword in the TDataSet.Filter property is added
- Like operator behaviour when used in the Filter property is now similar to TClientDataSet
- The AllowImplicitConnect option for the TOraSession component is added
- The SQLRecCount property for the TOraQuery and TOraStoredProc components is added
- The ScanParams property for the TOraScript component is added
- The RowsAffected property for the TOraScript component is added
- The UROWID data type is supported in the Direct mode

25-Apr-13 New Features in ODAC 9.0:
- Rad Studio XE4 is supported
- NEXTGEN compiler is supported
- Application development for iOS is supported
- FPC 2.6.2 and Lazarus 1.0.8 are supported
- BINARY_DOUBLE & BINARY_FLOAT data types support in the Direct mode is added
- Connection string support is improved
- Possibility to encrypt entire tables and datasets is added
- Possibility to determine if data in a field is encrypted is added
• Support for TimeStamp, Single and Extended fields in VirtualTable is added

12-Dec-12 New Features in ODAC 8.6:
• Rad Studio XE3 Update 1 is now required
• C++Builder 64-bit for Windows is supported

05-Sep-12 New Features in ODAC 8.5:
• Rad Studio XE3 is supported
• Windows 8 is supported

24-Jul-12 New Features in ODAC 8.2:
• Update 4 Hotfix 1 for RAD Studio XE2, Delphi XE2, and C++Builder XE2 is now required
• Data Type Mapping support is added
• Data Encryption in a client application is added
• The TOraEncryptor component for data encryption is added
• Integration with dbForge Studio for Oracle is added
• Calling of the TCustomDASQL.BeforeExecute event is added
• FieldType ftOraTimeStamp is added

23-Nov-11 New Features in ODAC 8.1:
• Update 2 for RAD Studio XE2, Delphi XE2, and C++Builder XE2 is now required
• Mac OS X and iOS in RAD Studio XE2 is supported
• FireMonkey support is improved
• Lazarus 0.9.30.2 and FPC 2.4.4 are supported
• Mac OS X in Lazarus is supported
• Linux x64 in Lazarus is supported
• FreeBSD in Lazarus is supported
• Oracle 11 Express Edition is supported
• Support for the NonBlocking option is added
• The QueryResultOnly option is added to TOraChangeNotification

15-Sep-11 New Features in Oracle Data Access Components 8.00:
• Embarcadero RAD Studio XE2 is supported
• Application development for 64-bit Windows is supported
• FireMonkey application development platform is supported
• Support of master/detail relationship for TVirtualTable is added
• OnProgress event in TVirtualTable is added
• TDADatasetOptions.SetEmptyStrToNull property that allows inserting NULL value instead of empty string is added

28-Apr-11 New Features in Oracle Data Access Components 7.20:
• Lazarus 0.9.30 and FPC 2.4.2 is supported
• Oracle 9, Oracle 10, and Oracle 11 authentication in the Direct mode is supported
• Case sensitive login and password in the Direct mode is supported
• Unicode login and password in the Direct mode is supported
• Client Identifier in the Direct mode is supported
• Support of BLOB, CLOB, and NCLOB data types in ToraLoader is improved
• Support of “table of blob/clob” data type is improved

26-Jan-11 New Features in Oracle Data Access Components 7.10:
• Support for connection with using Service Name in the Direct mode
• Support for the ChangePassword functionality in the Direct mode
• Improved returning cursors with different fields list from TOraStoredProc
• Search for the TNS_ADMIN variable in the Oracle root location in the registry
• Checking that dataset is open on calling the TDataSet.Locate method

13-Sep-10 New Features in Oracle Data Access Components 7.00:
• Embarcadero RAD Studio XE supported
• Support of National parameter to Package Wizard
• Ability to lock records in the CachedUpdate mode
• Ability to send call stack information to the dbMonitor component
• Added ability to lock records in the CachedUpdate mode
• Added OnStart, OnCommit, OnRollback events to TDATransaction
• Added OnInfoMessage event
• Added support for dbMonitor 3
• Added support for using user-defined Updatable when ODAC cannot detect a table list for a query
• Updated Oracle client version detection for Linux by OCI API
• Changed the LocateEx method behavior: now LocateEx centers records equal to Locate
• Now Required flag is set for UpdatingTable fields only
• Now the AssignConnect method copies transaction state

10-Sep-09 New Features in Oracle Data Access Components 6.90:
• Embarcadero RAD Studio 2010 supported
• Support for BINARY_FLOAT and BINARY_DOUBLE data types in TOraArray
• Support for reading a comma separated list of aliases from TNSNAMES.ORA
• Support for ALTER .. COMPILe in GetCompilationError
• Unicode support in CLOB attributes of OBJECT type
• Added EnableOraTimeStamP option of TOraSession
• Added distinction between empty string and null value when saving/loading string fields in TVirtualTable
• Added support for Free Pascal under Linux
• Added NoPreconnect property to TOraScript for executing CONNECT and CREATE DATABASE commands
• The Disconnected property to TCustomDADataset
• Now the value from the master dataset has priority over the DefaultExpression value

02-Apr-09 New Features in Oracle Data Access Components 6.80:
• Free Pascal under Linux supported
• Added NoPreconnect property to TOraScript for executing CONNECT and CREATE DATABASE commands

23-Oct-08 New Features in Oracle Data Access Components 6.70:
• Delphi 2009 and C++Builder 2009 supported
• Extended Unicode support for Delphi 2007 added (special Unicode build)
• Free Pascal 2.2 supported
• Powerful design-time editors implemented in Lazarus
• Optimized LOB processing in Direct mode
• Completed with more comprehensive structured Help
23-May-08 New Features in Oracle Data Access Components 6.50:
- Improved support of default field values
- The new component for metadata receiving added
- Added support of TWideMemoField
- The BCD types supported

14-Nov-07 New Features in Oracle Data Access Components 6.25:
- Oracle 11g supported

27-Sep-07 New Features in Oracle Data Access Components 6.20:
- CodeGear RAD Studio 2007 supported
- Added ability to customize whether update SQL statements should be prepared
- Added ability to set number of rows to be prefetched by OCI
- Added the OnProgress event in TOraLoader

12-Jun-07 New Features in Oracle Data Access Components 6.10:
- C++Builder 2007 supported

16-May-07 New Features in Oracle Data Access Components 6.05:
- Added Oracle Package Wizard that simplifies working with PL/SQL Packages

22-Mar-07 New Features in Oracle Data Access Components 6.00:

New functionality:
- Delphi 2007 for Win32 supported
- Implemented Disconnected Model for working offline and automatically connecting and disconnecting
- Implemented Local Failover for detecting connection loss and implicitly re-executing some operations
• **LargeInt fields** supported
• WideMemo field type in Delphi 2006 supported
• Added **DataSet Manager** to control project datasets
• Integration with **OraDeveloper Tools** 2.00 added
• New **TCRBatchMove** component for transferring data between all types of TDataSet descendants added
• Data **export** and M:Devart.Dac.TVirtualTable.LoadFromFile(System.String,System.Boolean) to/from XML supported
• Support for **sending messages** to DBMonitor from any point in your program added

Support for more Oracle server functionality:
• **Distributed transactions** supported
• **Advanced Queuing** support added
• **Change notifications** functionality of Oracle 10g supported
• **DBMS_TRACE package and SQLTrace functionality** supported
• **OCI Connection Pooling, Statement Caching, and Proxy Session Pooling** added
• **External Procedures** support added

Extensions and improvements to existing functionality:
• General performance improved
• **Master/detail** functionality extensions:
  o **Local master/detail** relationship support added
  o Support for master/detail relationships in **CachedUpdates** mode added

• **Connection pool** functionality improvements:
  o Efficiency significantly improved
  o **API for draining the connection pool** added

• **TOraScript** component improvements:
  o Support for executing **individual statements** in scripts added
  o Support for **executing huge scripts stored in files** with dynamic loading added
  o Ability to use standard SQL*Plus tool syntax added

• Greatly increased **performance of applying updates** in **CachedUpdates** mode
• Working with **calculated and lookup fields** improvements:
  o **Local sorting** and filtering added
  o Record **location** speed increased
  o Improved working with lookup fields
What's New

- Ability to customize update commands by attaching external components to TораUpdateSQL objects added
- Support for using BeforeFetch and AfterFetch events in NonBlocking mode added
- Temporary LOBs for updating LOB fields supported
- Support for setting connection timeout values for Direct mode added
- Ability to include all fields in automatically generated update SQLs added
- Support for default field value expressions added

Usability improvements:
- Syntax highlighting in design-time editors added
- Completely restructured and clearer demo projects

26-Jan-06 New Features in Oracle Data Access Components 5.70:
- Delphi 2006 supported

31-May-05 New Features in Oracle Data Access Components 5.55:
- Added ability to automatically prepare queries (TCustomDADataset.Options.AutoPrepare)
- Added ability to synchronize positions in different DataSets (TCustomDADataset.GotoCurrent)

24-Jan-05 New Features in Oracle Data Access Components 5.50:
- Delphi 2005 supported
- Performance of Net-option improved
- Added Schema property for TOraSession component
- Added ProxySession property for TOraSession component

30-Jun-04 New Features in Oracle Data Access Components 5.10:
- Local sort ability using IndexFieldNames property added
- T:Devart.Odac.OraDataAdapter component for Delphi 8 added

09-Apr-04 New Features in Oracle Data Access Components 5.00:
• Support for Delphi 8 added
• Oracle 10g support added
• Connection pooling supported
• Character conversion supported in Oracle 9i in Direct mode
• Unicode character data supported in Direct mode
• Support TIMESTAMP, INTERVAL data types in Direct mode
• Support for Oracle internal NUMBER datatype added in Direct mode
• Performance improved
• TCRGrid sources in Standard and Net editions
• .NET Windows Forms demo project added
• ASP.NET demo project added

08-Oct-03 New Features in Oracle Data Access Components 4.50:
• XMLTYPE datatype support added
• WideString support added to work with Unicode character data
• Transparent Application Failover support added

05-Jun-03 New Features in Oracle Data Access Components 4.15:
• Support for Oracle internal NUMBER datatype added. Allows to work with high precision numbers without accuracy losses

30-Jan-03 New Features in Oracle Data Access Components 4.10:
• Support for Oracle 9i NOT FINAL objects added
• TIMESTAMP and INTERVAL support for Oracle objects added

25-Dec-02 Oracle Data Access Components 4.05 new features:
• Transaction control schema changed. Now TOraSession.InTransaction shows actual user transaction state on server (implicit commit and rollback are considered).
• DBMonitor client implementation moved to COM server. Now ODAC is incompatible with DBMonitor 2.02 or lower.
• LOB attributes support for object fields added
• Temporary LOBs support added
• Constants ftTimeStampTZ and ftTimeStampLTZ added. Used in TOraTimeStampField.
• UROWID support for index organized tables added
• Option ConvertEOL added

30-Sep-02 New Features in Oracle Data Access Components 4.00:
• Delphi 7 support
• Kylix 3 for C++ support
• Oracle 9 scrollable cursors support
• New memory management model for ftString and ftVarBytes types. Allows significantly decrease memory usage on large tables fetch. Controlled by FlatBuffers dataset option;
• RAW datatype support (as ftVarBytes)
• Support for complex fields (blobs, objects etc.) in CachedUpdates mode
• New 'Prepare' schema. Now if user does not explicitly call Prepare method before opening dataset there is no additional roundtrip to server for select-list describe (OCIStmtExecute(DESCRIPT_ONLY) call). I.e. Open (Execute for SELECT) without Prepare is performed in a more optimal way.

30-Aug-02 New Features in Oracle Data Access Components 3.90:
• Kylix 3 support

09-Aug-02 New Features in Oracle Data Access Components 3.85:
• DBMonitor support
• New version of OraTools support (v. 2.50)

18-Jul-02 New Features in Oracle Data Access Components 3.80:
• Oracle9 timestamp and interval datatypes support
• Performance optimization for queries with many fields, especially for TSmartQuery and TOraTable
• Runtime packages division for Delphi6, C++Builder6, Kylix, Kylix2, see manual
• Auto generation RETURNING clause for LOBs added to design-time component editor
21-Mar-02 New Features in Oracle Data Access Components 3.60:
• supports C++Builder 6

14-Dec-01 New Features in Oracle Data Access Components 3.50:
• supports Kylix 2
• multibytes charsets support
• direct lob access support
• using OraTools Add-in

12-Oct-01 New Features in Oracle Data Access Components 3.30:
• supports Oracle 9i
• Net edition for Kylix

08-Aug-01 New Features in Oracle Data Access Components 3.20:
• supports Delphi 6
• new version of OraDesigner
• OraExplorer
• printed documentation
• BDE Migration Wizard

20-Feb-01 New Features in Oracle Data Access Components 3.00:
• using standard TParam object
• separate run- and design-time packages
• get original name of fields
• retrieve field's default value
• in Direct mode support
• Kylix ready

11-Jul-00 New Features in Oracle Data Access
Components 2.50:
- supports multiple Oracle Homes
- supports Borland SQL Monitor
- \textit{TOraSQLMonitor} component
- default session
- customizable connect dialog
- ConnectDialog and Threads demos added

30-Mar-00 New Features in Oracle Data Access Components 2.20:
- supports C++Builder 5
- macros in update SQL

10-Jan-00 New Features in Oracle Data Access Components 2.10:
- customized TSmartQuery data updating
- supports DML array operations
- macros in TOraSQL and TOraScript
- TOraLoader component
- supports Oracle 8 Lite
- easy installation

14-Oct-99 New Features in Oracle Data Access Components 2.00:
- supports Oracle8 Objects
- supports Oracle8 REFs
- supports Oracle8 Arrays
- supports Oracle8 Nested tables
- supports Oracle8 BFiles
- using RETURNING with Oracle8
- smart refreshing
- TOraNestedTable component
- TOraScript component
- TOraAlerter component
• TBField component
• TOraFile class
• TOraLob class
• TOraType class
• TOraObject class
• TOraRef class
• TOraArray class
• TOraNestTable class
• Alerter, Arrays, BFile, BLOBPics, DLL, DMLArray, FetchCursors, NestedTables, Objects, Refs demos added

26-May-99 New Features in Oracle Data Access Components 1.85:
• TOraProvider component
• TBDESSession component
• Supports Oracle 8i
• C++Builder 4 package

12-Mar-99 New Features in Oracle Data Access Components 1.75:
• TOraTable component
• TStoredProc component

01-Mar-99 New Features in Oracle Data Access Components 1.70:
• Supports BLOB and CLOB data types Oracle 8
• Supports nested tables
• TVirtualTable component
• Embedded SQL Designer with PL/SQL debugger
• C++Builder version

22-Oct-98 New Features in Oracle Data Access Components 1.50:
• Supports native interface Oracle 8.0
• Supports PL/SQL tables
2 General Information

This section contains general information about Oracle Data Access Components

- Overview
- Features
- Requirements
- Compatibility
- Using Several DAC Products in One IDE
- Component List
- Hierarchy Chart
- Editions
- Getting Support
- Frequently Asked Questions

2.1 Overview

Oracle Data Access Components (ODAC) is a library of components that provides native connectivity to Oracle from Delphi, C++Builder, Lazarus (and Free Pascal) for 32-bit and 64-bit Windows, macOS, and Linux platforms. The ODAC library is designed to help programmers develop faster, cleaner and more native Oracle database applications. ODAC, a high-performance and feature-rich Oracle connectivity solution, is an efficient native alternative to the Borland Database Engine (BDE) and standard dbExpress driver. It provides both possibility of connection to Oracle by means of native Oracle data access and direct access to Oracle without Oracle Client.

The ODAC library is actively developed and supported by the Devart Team. If you have questions about ODAC, email the developers at odac@devart.com or visit ODAC online at https://www.devart.com/odac/

Table of Contents
- Advantages of ODAC Technology
1. **Wide Coverage of Oracle Features**
2. **Native Connection Options**
3. **Oracle Advanced Features Support**
4. **Cross-Platform Solution for Delphi, C++Builder, and Lazarus**
5. **Optimized Code**
6. **Compatibility with Other Connectivity Methods**
7. **Development and Support**

- **Key Features**
- **How does ODAC work?**

### Advantages of ODAC Technology

ODAC is a direct database connectivity wrapper built specifically for the Oracle server. ODAC offers wide coverage of the Oracle feature set, supports both Client and Direct connection modes, and emphasizes optimized data access strategies.

#### Wide Coverage of Oracle Features

By providing access to the most advanced database functionality, ODAC allows developers to harness the full capabilities of the Oracle and optimize their database applications. ODAC stands out as the set of components with the widest support of Oracle functionality. It is the only component to support Oracle distributed transactions and implements support for controlling *statement caching*, *OCI pooling*, and *Oracle Advanced Queuing*. View the full list of supported Oracle features in [Features](#).

#### Native Connection Options

ODAC offers two connection modes to the Oracle server: connection through the Oracle Call Interface and direct connection over TCP/IP. ODAC-based database applications are easy to deploy, do not require installation of other data provider layers (such as BDE), and tend to be faster than those that use standard data connectivity solutions. See the [How does ODAC Work](#) section.

#### Oracle Advanced Features Support

ODAC has extra components designed to simplify some tasks and support Oracle-specific technologies. Particularly, OraScript serves to execute series of SQL statements, OraLoader serves to load external data into Oracle databases, OraeAlerter and OraPipe transfer messages and data between connections or client applications, and so on. ODAC includes set of classes designed to work with Oracle Advanced Queuing technology.
Cross-Platform Solution for Delphi, C++Builder, and Lazarus

ODAC is a cross-platform solution for developing applications using various IDEs: RAD Studio, Delphi, C++Builder, Lazarus (and FPC) on Windows, macOS, and Linux, for both x86 and x64 platforms. ODAC also provides support for the FireMonkey platform, which allows you to develop visually spectacular high-performance native applications for Windows and macOS.

Optimized Code

The goal of ODAC is to enable developers to write efficient and flexible database applications. The ODAC library is implemented using optimized code and advanced data access algorithms. Component interfaces undergo comprehensive performance tests and are designed to help you write thin and efficient product data access layers. Find out more about using ODAC to optimize your database applications in Increasing Performance. To see the results of ODAC performance tests, consult the performance comparison section on the ODAC website.

Compatibility with Other Connectivity Methods

The ODAC interface retains compatibility with standard VCL data access components like BDE. Existing BDE-based applications can be easily migrated to ODAC and enhanced to take advantage of Oracle-specific features like alerts, pipes, and the direct-path interface. Project migration can be automated with the BDE Migration Wizard. Find out more about Migration Wizard in Using Migration Wizard.

Development and Support

ODAC is an Oracle connectivity solution that has been actively developed and supported since 1998. ODAC comes with full documentation, demo projects, and fast (usually within one business day) technical support by the ODAC development team. Find out more about getting help or submit feedback and suggestions to the ODAC development team in Getting Support.

The description of the ODAC components is provided in the Component List.

Key Features

- Direct access to server data. Does not require installation of other data provider layers (such as BDE and ODBC)
- In Direct mode does not require Oracle client software and works directly through TCP/IP
- VCL, LCL and FMX versions of library available

© 2021 Devart
• Support of Windows, macOS, and Linux, for both x86 and x64 platforms.
• Full support of the latest versions of Oracle
• Support for all Oracle data types
• **Disconnected Model** with automatic connection control for working with data offline
• **Local Failover** for detecting connection loss and implicitly re-executing certain operations
• **Oracle Transparent Application Failover** support
• All types of local **sorting** and **filtering**, including by calculated and lookup fields
• Automatic **data updating** with **TOraQuery**, **TSmartQuery**, and **TOraTable** components
• **Unicode and national charsets** support
• **Distributed transaction** support
• **Oracle Advanced Queuing** support
• Support for many Oracle-specific features, such as **alerts**, **pipes** and **direct path interface**
• Advanced script execution functionality with **TOraScript** component
• Support for using **Macros** in SQL
• Easy migration from BDE with **Migration Wizard**
• Lets you **use Professional Edition of Delphi and C++Builder** to develop client/server applications
• Included annual **ODAC Subscription** with **Priority Support**
• Full Oracle Objects support
• Comprehensive REF CURSOR support
• PL/SQL tables and PL/SQL records support
• Operates in both connected and disconnected models
• Typed OraPackage component for wrapping PL/SQL packages
• Auxiliary components for SQL scripts and bulk data transfer
• Ability of monitoring query execution **TOraSQLMonitor**
• Licensed royalty-free per developer, per team, or per site

The full list of ODAC features can be found in **Features**.

**How does ODAC work?**

ODAC allows you to connect to Oracle in two ways: in the Direct mode over TCP/IP or in the Client mode using Oracle Client. The Direct mode can be enabled or disabled using the **Direct** property.

In the Direct mode, ODAC connects to Oracle directly via TCP/IP without using Oracle client software and enables you to build really thin database applications.
In Client mode, ODAC connects to Oracle through the Oracle Call Interface (OCI). OCI is an application programming interface that allows an application developer to use a third-generation language's native procedure or function calls to access the Oracle database server and control all phases of SQL statement execution. OCI is a library of standard database access and retrieval functions in the form of a dynamic-link library.

In contrast, the Borland Database Engine (BDE) uses several layers to access Oracle and requires additional data access software to be installed on client machines.

### 2.2 Features

**Supported target platforms:**
- Windows, 32-bit and 64-bit
• macOS, 32-bit and 64-bit
• iOS, 32-bit and 64-bit
• Android, 32-bit and 64-bit
• Linux, 32-bit and 64-bit

**General usability:**
• Direct access to server data. Does not require installation of other data provider layers (such as BDE and ODBC)
• Interface compatible with standard data access methods, such as BDE and ADO
• VCL, LCL, and FMX versions of library available
• [Separated run-time and GUI specific parts](#) allow you to create pure console applications such as CGI
• [Unicode and national charset support](#)
• Highly usable design time support
• Easy to deploy

**Network and connectivity:**
• In [Direct mode](#) does not require Oracle client software and works directly through TCP/IP
• [Disconnected Model](#) with automatic connection control for working with data offline
• [Local Failover](#) for detecting connection loss and implicitly reexecuting certain operations
• Support for setting [connection timeout](#) values for Direct mode
• Support for OraNet.PacketSize to improve performance in VPN and Wireless networks
• Secure connections with SSL, SSH, and HTTP/HTTPS tunneling (using [SecureBridge](#))
• Support for OS authentication
• Support for Proxy Authentication
• DBA privileges to open a session with
• Support for Oracle 19c connection modes
• Support for the change expired password
• Connection using Service Name or SID in the Direct mode
• Support for RAC Server
• Support for both IPv6 and IPv4 protocol
• Support for IFILE in tnsnames.ora
• Support for EZCONNECT connection string

**Compatibility:**
• [Full support of the latest versions of Oracle](#)
• Support for all versions of Oracle Clients, including Instant Client
• Support for all Oracle data types in both OCI and Direct modes
• Includes provider for UniDAC Express Edition
• Wide reporting component support, including support for InfoPower, ReportBuilder, FastReport
• Support for all standard visual data-aware controls
• Support for Lazarus 2.0.8 and FPC 3.0.4 for Windows, macOS, and Linux (32-bit and 64-bit)
• Allows you to use Professional Edition of Delphi and C++Builder to develop client/server applications.

Oracle technology support:
• Oracle Advanced Queuing support
• Distributed transactions support with TOraTransaction component
• Oracle package support
• Support for Oracle alerts and pipes with TOraAlerter component
• Support for Direct Path interface with TOraLoader component
• Support for DBMS_TRACE package and SQLTrace functionality with TOraTrace
• Support for Oracle Change notifications functionality of Oracle 10g with TOraChangeNotification component
• Oracle Transparent Application Failover support
• Oracle 9i scrollable cursor support
• Multiple Oracle Homes support
• Oracle sequence support
• DML array operations support
• Direct lob access support
• Temporary LOB management routines
• Temporary LOBs for updating LOB fields
• OCI Connection Pooling, Proxy Session Pooling, and Statement Caching
• Oracle optimizer control
• ProxySession support
• External Procedure support
• CLIENT_IDENTIFIER support
• Statement Caching
• ROWID values retrieval
• Overloaded stored procedures support
• Support for WITH FUNCTION clause

Oracle data types:
• All Oracle data types support
• Oracle Object (including NOT FINAL objects) types support
• Nested table support
• PL/SQL table support
• PL/SQL records support
• Support for REF CURSORs
• XMLTYPE datatype support
• Oracle 9i TIMESTAMP and INTERVAL data types support

Performance:
• High overall performance
• Fast controlled fetch of large data blocks
• Optimized string data storing
• Advanced connection pooling
• High performance applying of cached updates with batches
• Caching of calculated and lookup fields
• Expanded fields in TSmartQuery
• Fast Locate in a sorted DataSet
• Preparation of user-defined update statements
• High performance batch processing
• Intelligent fetch block size control
• Advanced connection pooling
• SmartFetch Mode enabling fast bi-directional navigation through large datasets

Local data storage operations:
• Database-independent data storage with TVirtualTable component
• CachedUpdates operation mode
• Local sorting and filtering, with included calculated and lookup fields
• Local master/detail relationship
• Master/detail relationship in CachedUpdates mode
Data access and data management automation:
- Automatic data updating with TOraQuery, TSmartQuery and TOraTable components
- Automatic record refreshing and locking
- Automatic query preparing
- SmartRefresh option allows you to synchronize two or more datasets automatically
- Support for ftWideMemo field type in Delphi 2006 and higher
- Data Type Mapping
- Support for Data Encryption in a client application

Extended data access functionality:
- Separate component for executing SQL and PL/SQL blocks
- Simplified access to table data with TOraTable component
- BLOB compression support
- Support for using macros in SQL
- NonBlocking mode allows background executing and fetching data in separate threads
- Ability to customize update commands by attaching external components to TOraUpdateSQL objects
- Deferred detail DataSet refresh in master/detail relationships
- LargeInt fields support
- MIDAS technology support
- Ability to customize Oracle error messages with TOraErrorHandler
- Structural representation and editing of Oracle objects
- Fill DataSet with several REF CURSOR
- Fill DataSet with object, array and nested table data
- Object-oriented building of SELECT statements

Data exchange:
- Transferring data between all types of TDataSet descendants with TCRBatchMove component
- Data export and import to/from XML (ADO format)
- Ability to synchronize positions in different DataSets

Script execution:
- Advanced script execution features with TOraScript component
- Support for executing individual statements in scripts
• Support for **executing huge scripts stored in files** with dynamic loading
• Ability to use standard SQL*Plus tool syntax in scripts

**SQL execution monitoring:**
• Extended SQL tracing capabilities provided by **TOraSQLMonitor** component and **DBMonitor**
• Borland SQL Monitor support
• Ability to **send messages to DBMonitor** from any point in your program
• Display executing statement, all its parameters’ values, and the type of parameters.

**Visual extensions:**
• Includes source code of enhanced TCRDBGrid data-aware grid control
• Customizable **connection dialog**
• **Cursor changes** during non-blocking execution

**Design-time enhancements:**
• **DataSet Manager tool** to control DataSet instances in the project
• **Oracle Package Wizard** that simplifies working with PL/SQL Packages
• Advanced design-time component and property editors
• Automatic design-time component linking
• Easy migration from BDE with **Migration Wizard**
• More convenient data source setup with the **TOraDataSource** component
• **Syntax highlighting** in design-time editors

**Resources:**
• Code documentation and guides in the CHM, PDF, and HXS formats
• Many helpful **demo** projects

**Licensing and support:**
• Included annual **ODAC Subscription** with **Priority Support**
• Licensed royalty-free per developer, per team, or per site

© 1997-2021
Devar. All Rights Reserved.  Request Support  DAC Forum  Provide Feedback
2.3 Requirements

Two versions of ODAC cannot be installed in parallel for the same IDE, and, since the Devart Data Access Components products have some shared bpl files, newer versions of ODAC may be incompatible with older versions of other Devart Data Access Components products.

So, before installing a new version of ODAC, uninstall any previous version of ODAC you may have, and check if your new install is compatible with other Devart Data Access Components products you have installed. For more information please see Using several products in one IDE. If you run into problems or have any compatibility questions, please email odac@devart.com

Note: You can avoid performing ODAC uninstallation manually when upgrading to a new version by directing the ODAC installation program to overwrite previous versions. To do this, execute the installation program from the command line with a /force parameter (Start | Run and type odacXX.exe /force, specifying the full path to the appropriate version of the installation program).

When installing ODAC from the sources to Windows Vista or Windows 7, it is necessary to have full access to the ODAC folder.

To use full set of Oracle features you have to have Oracle client software installed. If you do not want to install it, you can use Direct mode, in which ODAC communicates with Oracle server without intermediate libraries. In order to use the Direct mode, the operating system must have TCP/IP protocol support installed. For more information about using Direct mode, refer to Connecting in Direct mode article.

ODAC supports both 32-bit and 64-bit Oracle client versions in OCI mode. Developing applications for 64-bit client is possible only in RadStudio XE2 and Lazarus (FPC).

Note: RadStudio XE2 is a 32-bit application, therefore, for connecting to Oracle in OCI mode, even on a 64-bit platform, the 32-bit Oracle client is needed to be installed.

ODAC supports work with Oracle Instant Client in OCI mode. For correct work with Instant Client, the data about the client must be recorded to the registry or the Path environment variable, or Instant Client files(including tnsnames.ora) must be located in the application directory.
2.4 Compatibility

Oracle Compatibility

ODAC supports Oracle servers 21c, 19c, 18c, 12c, 11g, 10g, 9i, 8i, 8.0, including Oracle Express Edition 11g and 10g.

ODAC supports both x86 and x64 versions of the following Oracle Clients: 21c, 19c, 18c, 12c, 11g, 10g, 9i, 8i, 8.0. 

Note that support for x64 versions of Oracle Clients is available in Delphi XE2 for 64-bit Windows and is not available in C++Builder and older versions of Delphi.

IDE Compatibility

ODAC is compatible with the following IDEs:

- Embarcadero RAD Studio 10.4 Sydney (Requires Release 1)
  - Embarcadero Delphi 10.4 Sydney for Windows
  - Embarcadero Delphi 10.4 Sydney for macOS
  - Embarcadero Delphi 10.4 Sydney for Linux
  - Embarcadero Delphi 10.4 Sydney for iOS
  - Embarcadero Delphi 10.4 Sydney for Android
  - Embarcadero C++Builder 10.4 Sydney for Windows
  - Embarcadero C++Builder 10.4 Sydney for iOS
  - Embarcadero C++Builder 10.4 Sydney for Android
- Embarcadero RAD Studio 10.3 Rio (Requires Release 2 or Release 3)
  - Embarcadero Delphi 10.3 Rio for Windows
  - Embarcadero Delphi 10.3 Rio for macOS
  - Embarcadero Delphi 10.3 Rio for Linux
  - Embarcadero Delphi 10.3 Rio for iOS
  - Embarcadero Delphi 10.3 Rio for Android
  - Embarcadero C++Builder 10.3 Rio for Windows
  - Embarcadero C++Builder 10.3 Rio for macOS
  - Embarcadero C++Builder 10.3 Rio for iOS
  - Embarcadero C++Builder 10.3 Rio for Android
- Embarcadero RAD Studio 10.2 Tokyo
  - Embarcadero Delphi 10.2 Tokyo for Windows
  - Embarcadero Delphi 10.2 Tokyo for macOS
• Embarcadero Delphi 10.2 Tokyo for Linux
• Embarcadero Delphi 10.2 Tokyo for iOS
• Embarcadero Delphi 10.2 Tokyo for Android
• Embarcadero C++Builder 10.2 Tokyo for Windows
• Embarcadero C++Builder 10.2 Tokyo for macOS
• Embarcadero C++Builder 10.2 Tokyo for iOS
• Embarcadero C++Builder 10.2 Tokyo for Android

• Embarcadero RAD Studio 10.1 Berlin
  • Embarcadero Delphi 10.1 Berlin for Windows
  • Embarcadero Delphi 10.1 Berlin for macOS
  • Embarcadero Delphi 10.1 Berlin for iOS
  • Embarcadero Delphi 10.1 Berlin for Android
  • Embarcadero C++Builder 10.1 Berlin for Windows
  • Embarcadero C++Builder 10.1 Berlin for macOS
  • Embarcadero C++Builder 10.1 Berlin for iOS
  • Embarcadero C++Builder 10.1 Berlin for Android

• Embarcadero RAD Studio 10 Seattle
  • Embarcadero Delphi 10 Seattle for Windows
  • Embarcadero Delphi 10 Seattle for macOS
  • Embarcadero Delphi 10 Seattle for iOS
  • Embarcadero Delphi 10 Seattle for Android
  • Embarcadero C++Builder 10 Seattle for Windows
  • Embarcadero C++Builder 10 Seattle for macOS
  • Embarcadero C++Builder 10 Seattle for iOS
  • Embarcadero C++Builder 10 Seattle for Android

• Embarcadero RAD Studio XE8
  • Embarcadero Delphi XE8 for Windows
  • Embarcadero Delphi XE8 for macOS
  • Embarcadero Delphi XE8 for iOS
  • Embarcadero Delphi XE8 for Android
  • Embarcadero C++Builder XE8 for Windows
  • Embarcadero C++Builder XE8 for macOS
  • Embarcadero C++Builder XE8 for iOS
  • Embarcadero C++Builder XE8 for Android

• Embarcadero RAD Studio XE7
  • Embarcadero Delphi XE7 for Windows
  • Embarcadero Delphi XE7 for macOS
- Embarcadero Delphi XE7 for iOS
- Embarcadero Delphi XE7 for Android
- Embarcadero C++Builder XE7 for Windows
- Embarcadero C++Builder XE7 for macOS
- Embarcadero C++Builder XE7 for iOS
- Embarcadero C++Builder XE7 for Android
- Embarcadero RAD Studio XE6
  - Embarcadero Delphi XE6 for Windows
  - Embarcadero Delphi XE6 for macOS
  - Embarcadero Delphi XE6 for iOS
  - Embarcadero Delphi XE6 for Android
  - Embarcadero C++Builder XE6 for Windows
  - Embarcadero C++Builder XE6 for macOS
  - Embarcadero C++Builder XE6 for iOS
  - Embarcadero C++Builder XE6 for Android
- Embarcadero RAD Studio XE5 (Requires Update 2)
  - Embarcadero Delphi XE5 for Windows
  - Embarcadero Delphi XE5 for macOS
  - Embarcadero Delphi XE5 for iOS
  - Embarcadero Delphi XE5 for Android
  - Embarcadero C++Builder XE5 for Windows
  - Embarcadero C++Builder XE5 for macOS
  - Embarcadero C++Builder XE5 for iOS
- Embarcadero RAD Studio XE4
  - Embarcadero Delphi XE4 for Windows
  - Embarcadero Delphi XE4 for macOS
  - Embarcadero Delphi XE4 for iOS
  - Embarcadero C++Builder XE4 for Windows
  - Embarcadero C++Builder XE4 for macOS
- Embarcadero RAD Studio XE3 (Requires Update 2)
  - Embarcadero Delphi XE3 for Windows
  - Embarcadero Delphi XE3 for macOS
  - Embarcadero C++Builder XE3 for Windows
  - Embarcadero C++Builder XE3 for macOS
- Embarcadero RAD Studio XE2 (Requires Update 4 Hotfix 1)
  - Embarcadero Delphi XE2 for Windows
  - Embarcadero Delphi XE2 for macOS
- Embarcadero C++Builder XE for Windows
- Embarcadero C++Builder XE for macOS
- Embarcadero RAD Studio XE
- Embarcadero Delphi XE
- Embarcadero C++Builder XE
- Embarcadero RAD Studio 2010
- Embarcadero Delphi 2010
- Embarcadero C++Builder 2010
- CodeGear RAD Studio 2009 (Requires Update 3)
  - CodeGear Delphi 2009
  - CodeGear C++Builder 2009
- CodeGear RAD Studio 2007
  - CodeGear Delphi 2007
  - CodeGear C++Builder 2007
- Borland Developer Studio 2006
  - Borland Delphi 2006
  - Borland C++Builder 2006
- Borland Delphi 7
- Borland Delphi 6 (Requires Update Pack 2 – Delphi 6 Build 6.240)
- Borland C++Builder 6 (Requires Update Pack 4 – C++Builder 6 Build 10.166)
- Lazarus 2.0.10 and Free Pascal 3.2.0 for Windows, macOS, and Linux.

All the existing Delphi and C++Builder editions are supported: Architect, Enterprise, Professional, Community, and Starter.

Lazarus and Free Pascal are supported only in Trial Edition and Professional Edition with source code.

Supported Target Platforms
- Windows, 32-bit and 64-bit
- macOS, 32-bit and 64-bit
- Linux, 32-bit (only in Lazarus and Free Pascal) and 64-bit
- iOS, 32-bit and 64-bit
- Android, 32-bit and 64-bit

Note that support for 64-bit Windows and macOS was introduced in RAD Studio XE2, and is not available in older versions of RAD Studio. Support for iOS is available since RAD Studio
XE4, but support for iOS 64-bit is available since RAD Studio XE8. Support for Android is available since RAD Studio XE5. Support for Linux 64-bit is available since RAD Studio 10.2 Tokyo. Support for macOS 64-bit is available since RAD Studio 10.3 Rio. Support for Android 64-bit is available since RAD Studio 10.3.3 Rio. Support for macOS 32-bit and iOS 32-bit was removed in RAD Studio 10.4.

Supported GUI Frameworks
- FireMonkey (FMX)
- Visual Component Library (VCL)
- Lazarus Component Library (LCL)

Devart Data Access Components Compatibility
All DAC products are compatible with each other.

But, to install several DAC products to the same IDE, it is necessary to make sure that all DAC products have the same common engine (BPL files) version. The latest versions of DAC products or versions with the same release date always have the same version of the common engine and can be installed to the same IDE.

2.5 Using Several DAC Products in One IDE

UniDAC, ODAC, SDAC, MyDAC, IBDAC, PgDAC, LiteDAC and VirtualDAC components use common base packages listed below:

Packages:
- dacXX.bpl
- dacvclXX.bpl
- dcldacXX.bpl

Note that product compatibility is provided for the current build only. In other words, if you upgrade one of the installed products, it may conflict with older builds of other products. In order to continue using the products simultaneously, you should upgrade all of them at the same time.
2.6 Component List

This topic presents a brief description of the components included in the Oracle Data Access Components library. Click on the name of each component for more information. These components are added to the ODAC page of the Component palette except for TCRBatchMove and TVirtualTable components. TCRBatchMove and TVirtualTable components are added to the Data Access page of the Component palette. Basic ODAC components are included in all ODAC editions. ODAC Professional and Developer Edition components are not included in ODAC Standard Edition.

## Basic ODAC components

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraSession</td>
<td>Lets you set up and control connections to Oracle.</td>
</tr>
<tr>
<td>TOraQuery</td>
<td>Uses SQL statements to retrieve data from Oracle tables and pass it to one or more data-aware components through a TDataSource object. This component provides a mechanism for updating data in Oracle.</td>
</tr>
<tr>
<td>TSmartQuery</td>
<td>A more efficient query component. This component is aware of all the fields that belong to a table being updated and performs on-demand full row retrieval with Expand Fields. A traffic-efficient alternative to TOraQuery for working with large tables with lots of fields. Includes Smart Refresh functionality in Professional and Developer Editions.</td>
</tr>
<tr>
<td>TOraSQL</td>
<td>Executes SQL statements, PL/SQL blocks, and stored procedures, which do not return rowsets.</td>
</tr>
<tr>
<td>TOraTable</td>
<td>Lets you retrieve and update data in a single table without writing SQL statements.</td>
</tr>
<tr>
<td>TOraStoredProc</td>
<td>Executes stored procedures and functions. Lets you edit cursor data returned as parameter.</td>
</tr>
<tr>
<td>TOraNestedTable</td>
<td>Component for controlling nested table data.</td>
</tr>
<tr>
<td>TOraUpdateSQL</td>
<td>Lets you tune update operations for a DataSet component.</td>
</tr>
<tr>
<td>TOraDataSource</td>
<td>Provides an interface for connecting data-aware controls on a form and ODAC dataset components.</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>TOraScript</code></td>
<td>Executes sequences of SQL and PL/SQL statements.</td>
</tr>
<tr>
<td><code>TOraSQLMonitor</code></td>
<td>Interface for monitoring dynamic SQL execution in ODAC-based applications.</td>
</tr>
<tr>
<td><code>TConnectDialog</code></td>
<td>Allows you to build custom prompts for usernames, passwords, and server names.</td>
</tr>
<tr>
<td><code>TVirtualTable</code></td>
<td>Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.</td>
</tr>
<tr>
<td><code>TVirtualDataSet</code></td>
<td>Dataset that processes arbitrary non-tabular data.</td>
</tr>
<tr>
<td><strong>ODAC Professional and Developer Edition components</strong></td>
<td></td>
</tr>
<tr>
<td><code>TOraEncryptor</code></td>
<td>Represents data encryption and decryption in client application.</td>
</tr>
<tr>
<td><code>TOraPackage</code></td>
<td>Provides a straightforward way to access Oracle packages.</td>
</tr>
<tr>
<td><code>TOraAlerter</code></td>
<td>Allows you to transfer messages between sessions.</td>
</tr>
<tr>
<td><code>TOraLoader</code></td>
<td>Allows you to quickly load data into Oracle databases.</td>
</tr>
<tr>
<td><code>TOraTransaction</code></td>
<td>Provides discrete transaction control over sessions. Can be used to manipulate both simple and distributed transactions.</td>
</tr>
<tr>
<td><code>TOraQueue</code></td>
<td>Lets you monitor the message queue. Provides an interface for enqueuing and dequeuing messages.</td>
</tr>
<tr>
<td><code>TOraQueueTable</code></td>
<td>Component for managing queue tables.</td>
</tr>
<tr>
<td><code>TOraQueueAdmin</code></td>
<td>Component for managing queues.</td>
</tr>
<tr>
<td><code>TOraChangeNotification</code></td>
<td>Allows you to use Oracle Database Change Notifications.</td>
</tr>
<tr>
<td><code>TOraTrace</code></td>
<td>Allows you to start and stop the SQL trace for a specified session. This component provides access to the DBMS_TRACE package.</td>
</tr>
</tbody>
</table>
Many ODAC classes are inherited from standard VCL/LCL classes. The inheritance hierarchy chart for ODAC is shown below. The ODAC classes are represented by hyperlinks that point to their description in this documentation. The description of the standard classes can be found in the documentation of your IDE.

```
TObject
  |-TPersistent
    |-TComponent
      | |-TCustomConnection
      |   | |-TCustomDAConnection
      |   | |-TOraSession
      | |-TDataSet
      |   | |-TMemDataSet
      |   |   | |-TCustomDADataSet
      |   |   | ` |-TOraDataSet
      |   |   |   | |-TCustomOraQuery
      |   |   |   |   | |-TCustomSmartQuery
      |   |   |   |   | |-TOraTable
      |   |   |   |   | |-TSmartQuery
      |   |   |   | |-TOraQuery
      |   |   | |-TOraStoredProc
      |   | |-TOraNestedTable
      | | |-TDAMetaData
```
2.8 Editions

Oracle Data Access Components comes in two editions: Standard and Professional.

The **Standard** edition includes the ODAC basic connectivity components and ODAC Migration Wizard. ODAC Standard Edition is a cost-effective solution for database application developers who are looking for high-performance connectivity to Oracle for secure, reliable, and high-speed data transmission.

The **Professional** edition shows off the full power of ODAC, enhancing ODAC Standard Edition with support for Oracle-specific functionality, access to the Direct mode for connecting to the Oracle server directly via TCP/IP, and some advanced dataset management features.

You can get **Source Access** to the Client mode implementation of all the component classes in ODAC by purchasing a special ODAC Professional Edition with Source Code. The source code of DataSet Manager and Migration Wizard is not distributed. The source code of the Direct mode for Oracle is distributed obfuscated.

The matrix below compares the features of ODAC editions. See [Features](#) for the detailed list of ODAC features.

### ODAC Edition Matrix

<table>
<thead>
<tr>
<th>Feature</th>
<th>Standard</th>
<th>Professional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Connectivity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connection without Oracle client</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Desktop Application Development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>macOS</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Linux</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Mobile Application Development**

| iOS   | × | ✓ |
| Android | × | ✓ |

**Data Access Components**

| TOraSession | ✓ | ✓ |
| TOraQuery   | ✓ | ✓ |
| TOraTable   | ✓ | ✓ |
| TOraStoredProc | ✓ | ✓ |
| TOraUpdateSQL | ✓ | ✓ |
| TOraSQL     | ✓ | ✓ |
| TOraDataSource | ✓ | ✓ |

- Script executing
  - TOraScript ✓ ✓
- Transactions managing
  - TOraTransaction × ✓
- Fast data loading into the server
  - TOraLoader × ✓

**Advanced Query Components**

- Expanded field representation
  - TSmartQuery ✓ ✓
- Smart refresh in TSmartQuery component × ✓

**Oracle Specific Components**
<table>
<thead>
<tr>
<th>Feature</th>
<th>TOraclePackage</th>
<th>TOracleNestedTable</th>
<th>TOracleAlerter</th>
<th>TOracleChangeNotification</th>
<th>TOracleQueue</th>
<th>TOracleQueueAdmin</th>
<th>TOracleQueueTable</th>
<th>TOracleTrace</th>
<th>TOracleMetaData</th>
<th>TOracleErrorHandler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle packages</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td></td>
<td>√</td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Oracle nested tables</td>
<td></td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
</tr>
<tr>
<td>Messaging between sessions and applications</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction on server side changes on-the-fly</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle advanced queing</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PL/SQL tracing</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obtaining metadata about database objects</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle errors handling</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataBase Activity Monitoring</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitoring of per-component SQL execution</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additional Components</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced connection dialog</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data encryption and decryption</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced DataSet provider</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data storing in memory table</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dataset that wraps arbitrary non-tabular data</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advanced DBGrid with extended functionality</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Records transferring between datasets</td>
<td>×</td>
<td></td>
<td>×</td>
<td>√</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design-Time Features</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enhanced component and property editors</td>
<td></td>
<td></td>
<td>×</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Licensing

Please read this license agreement carefully. By installing or using this software, you indicate acceptance of and agree to become bound by the terms and conditions of this license. If you do not agree to the terms of this license, do not install or use this software and promptly return it to Devart.

### Introduction

This Devart end-user license agreement ("Agreement") is a legal agreement between you (either an individual person or a single legal entity) and Devart, for the use of ODAC software application, source code, demos, intermediate files, printed materials, and online or electronic documentation contained in this installation file. For the purpose of this Agreement, the software program(s) and supporting documentation will be referred to as the "Software".

### License

1. **Grant of License**

   The enclosed Software is licensed, not sold. You have the following rights and privileges, subject to all limitations, restrictions, and policies specified in this Agreement.

   1.1. If you are a legally licensed user, depending on the license type specified in the registration letter you have received from Devart upon purchase of the Software, you are
entitled to either:
- install and use the Software on one or more computers, provided it is used by 1 (one) for the sole purposes of developing, testing, and deploying applications in accordance with this Agreement (the "Single Developer License"); or
- install and use the Software on one or more computers, provided it is used by up to 4 (four) developers within a single company at one physical address for the sole purposes of developing, testing, and deploying applications in accordance with this Agreement (the "Team Developer License"); or
- install and use the Software on one or more computers, provided it is used by developers in a single company at one physical address for the sole purposes of developing, testing, and deploying applications in accordance with this Agreement (the "Site License").

1.2. If you are a legally licensed user of the Software, you are also entitled to:
- make one copy of the Software for archival purposes only, or copy the Software onto the hard disk of your computer and retain the original for archival purposes;
- develop and test applications with the Software, subject to the Limitations below;
- create libraries, components, and frameworks derived from the Software for personal use only;
- deploy and register run-time libraries and packages of the Software, subject to the Redistribution policy defined below.

1.3. You are allowed to use evaluation versions of the Software as specified in the Evaluation section.

No other rights or privileges are granted in this Agreement.

2. LIMITATIONS

Only legally registered users are licensed to use the Software, subject to all of the conditions of this Agreement. Usage of the Software is subject to the following restrictions.

2.1. You may not reverse engineer, decompile, or disassemble the Software.

2.2. You may not build any other components through inheritance for public distribution or commercial sale.

2.3. You may not use any part of the source code of the Software (original or modified) to build any other components for public distribution or commercial sale.

2.4. You may not reproduce or distribute any Software documentation without express written permission from Devart.
2.5. You may not distribute and sell any portion of the Software without integrating it into your Applications as Executable Code, except a Trial version that can be distributed for free as original Devart's ODAC Trial package.

2.6. You may not transfer, assign, or modify the Software in whole or in part. In particular, the Software license is non-transferable, and you may not transfer the Software installation package.

2.7. You may not remove or alter any Devart's copyright, trademark, or other proprietary rights notice contained in any portion of Devart units, source code, or other files that bear such a notice.

3. REDISTRIBUTION

The license grants you a non-exclusive right to compile, reproduce, and distribute any new software programs created using ODAC. You can distribute ODAC only in compiled Executable Programs or Dynamic-Link Libraries with required run-time libraries and packages.

All Devart's units, source code, and other files remain Devart's exclusive property.

4. TRANSFER

You may not transfer the Software to any individual or entity without express written permission from Devart. In particular, you may not share copies of the Software under “Single Developer License” and “Team License” with other co-developers without obtaining proper license of these copies for each individual.

5. TERMINATION

Devart may immediately terminate this Agreement without notice or judicial resolution in the event of any failure to comply with any provision of this Agreement. Upon such termination you must destroy the Software, all accompanying written materials, and all copies.

6. EVALUATION

Devart may provide evaluation ("Trial") versions of the Software. You may transfer or distribute Trial versions of the Software as an original installation package only. If the Software you have obtained is marked as a "Trial" version, you may install and use the Software for a period of up to 60 calendar days from the date of installation (the "Trial Period"), subject to the additional restriction that it is used solely for evaluation of the Software and not in conjunction with the development or deployment of any application in production. You may not use applications developed using Trial versions of the Software for any commercial purposes.
Upon expiration of the Trial Period, the Software must be uninstalled, all its copies and all accompanying written materials must be destroyed.

7. WARRANTY

The Software and documentation are provided "AS IS" without warranty of any kind. Devart makes no warranties, expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose or use.

8. SUBSCRIPTION AND SUPPORT

The Software is sold on a subscription basis. The Software subscription entitles you to download improvements and enhancement from Devart's web site as they become available, during the active subscription period. The initial subscription period is one year from the date of purchase of the license. The subscription is automatically activated upon purchase, and may be subsequently renewed by Devart, subject to receipt applicable fees. Licensed users of the Software with an active subscription may request technical assistance with using the Software over email from the Software development. Devart shall use its reasonable endeavours to answer queries raised, but does not guarantee that your queries or problems will be fixed or solved.

Devart reserves the right to cease offering and providing support for legacy IDE versions.

9. COPYRIGHT

The Software is confidential and proprietary copyrighted work of Devart and is protected by international copyright laws and treaty provisions. You may not remove the copyright notice from any copy of the Software or any copy of the written materials, accompanying the Software.

This Agreement contains the total agreement between the two parties and supersedes any other agreements, written, oral, expressed, or implied.

© 1997-2021 Devart. All Rights Reserved.

2.10 Getting Support

This page lists several ways you can find help with using ODAC and describes the ODAC Priority Support program.

Support Options
There are a number of resources for finding help on installing and using ODAC.

- You can find out more about ODAC installation or licensing by consulting the Licensing and FAQ sections.
- You can get community assistance and technical support on the ODAC Community Forum.
- You can get advanced technical assistance by ODAC developers through the ODAC Priority Support program.

If you have a question about ordering ODAC or any other Devart product, please contact sales@devart.com.

**ODAC Priority Support**

ODAC Priority Support is an advanced product support service for getting expedited individual assistance with ODAC-related questions from the ODAC developers themselves. Priority Support is carried out over email and has two business days response policy. Priority Support is available for users with an active ODAC Subscription.

To get help through the ODAC Priority Support program, please send an email to support@devart.com describing the problem you are having. Make sure to include the following information in your message:

- The version of Delphi, C++Builder you are using.
- Your ODAC Registration number.
- Full ODAC edition name and version number. You can find both of these from the ODAC | ODAC About menu in the IDE.
- Versions of the Oracle server and client you are using.
- A detailed problem description.
- If possible, a small test project that reproduces the problem. It is recommended to use Scott or SYS schema objects only. Please include definitions for all and avoid using third-party components.

© 1997-2021
Devart. All Rights Reserved.

### 2.11 Frequently Asked Questions

This page contains a list of Frequently Asked Questions for Oracle Data Access Components.

If you have encounter a question with using ODAC, please browse through this list first. If this page does not answer your question, refer to the Getting Support topic in ODAC help.
Installation and Deployment

1. I'm having a problem with installing ODAC or compiling ODAC-based projects...

You may be having a compatibility issue that shows up in one or more of the following forms:
- Get a "Setup has detected already installed DAC packages which are incompatible with current version" message during ODAC installation.
- Get a "Procedure entry point ... not found in ... " message when starting IDE.
- Get a "Unit ... was compiled with a different version of ..." message on compilation.

You can have such problems if you installed incompatible ODAC, SDAC, MyDAC or IBDAC versions. All these products use common base packages. The easiest way to avoid the problem is to uninstall all installed DAC products and then download from our site and install the last builds.

2. What software should be installed on a client computer so that my applications that use ODAC can run?

To use the full set of Oracle features, the client computer has to have Oracle client software (OCI) installed. If you do not want to install OCI, you can use Direct mode, in which ODAC communicates with Oracle server without intermediate libraries. In order to use the Direct mode, the operating system on the client computer must have TCP/IP protocol support installed.

3. How can I quickly convert a project from BDE to ODAC?

To quickly migrate your project from BDE you can use the BDE Migration Wizard. To start it, open your project and choose BDE Migration Wizard from the ODAC menu of your IDE.

Licensing and Subscriptions

1. Am I entitled to distribute applications written with ODAC?

If you have purchased a full version of ODAC, you are entitled to distribute pre-compiled programs created with its use. You are not entitled to propagate any components inherited from ODAC or using ODAC source code. For more information see the License.rtf file in your ODAC installation directory.

2. Can I create components using ODAC?

You can create your own components that are inherited from ODAC or that use the ODAC source code. You are entitled to sell and distribute compiled application executables that use such components, but not their source code and not the components themselves.
3. What licensing changes can I expect with ODAC 6.00?

The basic ODAC license agreement will remain the same. With ODAC 6.00, the ODAC Edition Matrix will be reorganized and a new ODAC Subscription Program will be introduced.

4. What do the ODAC 6.00 Edition Levels correspond to?

ODAC 6.00 will come in six editions: Trial, Standard, Professional, Professional with Sources, Developer, and Developer with Sources.

When you upgrade to the new version, your edition level will be automatically updated using the following Edition Correspondence Table.

<table>
<thead>
<tr>
<th>Old Edition Level</th>
<th>New Edition Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>- No Correspondence -</td>
<td>ODAC Standard Edition</td>
</tr>
<tr>
<td>ODAC Standard Edition</td>
<td>ODAC Professional Edition</td>
</tr>
<tr>
<td>ODAC Net Edition</td>
<td>ODAC Professional Edition</td>
</tr>
<tr>
<td>ODAC Professional Edition</td>
<td>ODAC Professional Edition with Sources</td>
</tr>
<tr>
<td>- No Correspondence -</td>
<td>ODAC Professional Edition</td>
</tr>
<tr>
<td>ODAC Trial Edition</td>
<td>ODAC Trial Edition</td>
</tr>
<tr>
<td>- No Correspondence -</td>
<td>ODAC Developer Edition</td>
</tr>
<tr>
<td>ODAC Developer Edition</td>
<td>ODAC Developer Edition with Sources</td>
</tr>
</tbody>
</table>

The feature list for each edition can be found in the ODAC documentation and the ODAC website.

5. I have a registered version of ODAC. Will I need to pay to upgrade to future versions?

After ODAC 6.00, all upgrades to future versions are free to users with an active ODAC Subscription.

Users that have a registration for versions of ODAC prior to ODAC 6.00 will have to first
upgrade to ODAC 6.00 to jump in on the Subscription Program.

6. What are the benefits of the ODAC Subscription Program?

The ODAC ODAC Subscription Program is an annual maintenance and support service for ODAC users.

Users with a valid ODAC Subscription get the following benefits:
- Access to new versions of ODAC when they are released
- Access to all ODAC updates and bug fixes
- Product support through the ODAC Priority Support program
- Notification of new product versions

Priority Support is an advanced product support program which offers you expedited individual assistance with ODAC-related questions from the ODAC developers themselves. Priority Support is carried out over email and has a guaranteed two business day response policy.

The ODAC Subscription Program is available for registered users of ODAC 6.00 and higher.

7. Can I use my version of ODAC after my Subscription expires?

Yes, you can. ODAC version licenses are perpetual.

8. I want a ODAC Subscription! How can I get one?

You can renew your ODAC Subscription on the ODAC Ordering Page. For more information, please contact sales@devart.com.

You will be able to renew your ODAC Subscription by email or on the ODAC website. For more information, please contact sales@devart.com.

9. Does this mean that if I upgrade to ODAC 6 from ODAC 5, I'll get an annual ODAC Subscription for free?

Yes.

10. How do I upgrade to ODAC 6.00?

To upgrade to ODAC 6.00, you can get a Version Update from the ODAC Ordering Page. For more information, please contact sales@devart.com.

Performance
1. How productive is ODAC?
From time to time we compare ODAC with other products, and ODAC always takes first place. For more information, please refer to the ODAC performance comparison results posted on the [ODAC website](http://example.com).

2. Why does the Locate function work so slowly the first time I use it?

Locate is performed on the client. So if you had set FetchAll to False when opening your dataset, cached only some of the rows on the client, and then invoked Locate, ODAC will have to fetch all the remaining rows from the server before performing the operation. On subsequent calls, Locate should work much faster.

If the Locate method keeps working slowly on subsequent calls or if you are working with FetchAll=True, try the following. Perform local sorting by a field that is used in the Locate method. Just assign corresponding field name to the IndexFieldNames property.

How To

1. How can I find out which version of ODAC I am using?

You can determine your ODAC version number in several ways:

- During installation of ODAC, consult the ODAC Installer screen.
- After installation, see the `history.html` file in your ODAC installation directory.
- At design-time, select Oracle | About ODAC from the main menu of your IDE.
- At run-time, check the value of the OdacVersion and DACVersion constants.

2. How can I stop the cursor from changing to an hour glass during query execution?

Just set the DBAccess.ChangeCursor variable to False anywhere in your program. The cursor will stop changing after this command is executed.

3. How can I execute a query saved in the SQLInsert, SQLUpdate, SQLDelete, or SQLRefresh properties of a ODAC dataset?

The values of these properties are templates for query statements, and they cannot be manually executed. Usually there is no need to fill these properties because the text of the query is generated automatically.

In special cases, you can set these properties to perform more complicated processing during a query. These properties are automatically processed by ODAC during the execution of the Post, Delete, or RefreshRecord methods, and are used to construct the query to the server. Their values can contain parameters with names of fields in the underlying data source, which will be later replaced by appropriate data values.
For example, you can use the SQLInsert template to insert a row into a query instance as follows.
  - Fill the SQLInsert property with the parametrized query template you want to use.
  - Call Insert.
  - Initialize field values of the row to insert.
  - Call Post.

The value of the SQLInsert property will then be used by ODAC to perform the last step.

Setting these properties is optional and allows you to automatically execute additional SQL statements, add calls to stored procedures and functions, check input parameters, and/or store comments during query execution. If these properties are not set, the ODAC dataset object will generate the query itself using the appropriate insert, update, delete, or refresh record syntax.

4. **My program allows users to edit records directly in a DBGrid instance. How can I disable record deletion?**

   If TOraQuery acts as TDataSet, it is very simple to prohibit deleting, inserting and/or updating of records. Simply clear the relevant property (SQLDelete, SQLInsert, SQLUpdate). The action with empty SQL statement will not be allowed.

5. **How do I allow users to delete, insert, and edit records (e.g. in DBGrid), but ensure deletions are not represented in the database?**

   Assign the following PL/SQL block to TOraQuery.SQLDelete:

   ```sql
   begin
     Null;
   end;
   ```

6. **How can I tune the NUMBER fields definition in ODAC?**

   There are three typed constants in the OraClasses.pas module: IntegerPrecision, LargeIntPrecision and FloatPrecision. Using the values of these constants and the EnableIntegers and EnableNumbers options, the Oracle NUMBER type is mapped to ODAC field classes as follows.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Field class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision &lt;= IntegerPrecision, Scale = 0,</td>
<td>TIntegerField</td>
</tr>
<tr>
<td>EnableIntegers = True</td>
<td></td>
</tr>
<tr>
<td>IntegerPrecision &lt; Precision &lt;=</td>
<td>TLargeInt</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The default values of these constants are:

| IntegerPrecision = 9, LargeIntPrecision = 0, FloatPrecision = 15 |

7. **How to enable support of TLargeIntField?**
   Set the value of the global constant LargeIntPrecision to 18.

**General Questions**

1. **What are the advantages of ODAC compared to BDE?**

   BDE provides a more or less uniform way for accessing different servers (SQL Server, MySQL, Oracle and so on)

   ODAC is a set of components optimized for working specifically with Oracle, and has a server-specific component interface and advanced design-time support.

   As a result, there are a number of reasons why ODAC may be better than BDE for your Oracle-based application. Some of them are enumerated here. For more information refer to the ODAC feature list.
   - Incomparably higher speed of data access, fetching, and processing
   - Possibility to individually adjust and optimize execution parameters for every query
   - Complete support of PL/SQL
   - Return of cursor from a stored procedure or anonymous PL/SQL block (by parameter)
   - Return of PL/SQL result sets from a stored procedure or anonymous PL/SQL block
   - Support for asynchronous execution of queries
   - Possibility to break the execution of long-duration queries
   - Built-in system for debugging in run-time
   - Convenient and simple-to-use standardized error processing mechanism
   - No need to install and configure BDE on client machines during deployment

2. **What happened to the ODAC Net option and the TOraSession.Options.Net property?**
As of ODAC 6.00, the ODAC Net Option has been renamed to **ODAC Direct mode**, and TOraSession.Options.Net has been replaced with TOraSession.Options.Direct. You can configure ODAC to connect directly to Oracle over TCP/IP by setting TOraSession.Options.Direct to True.

**Note** TOraSession.Options.Net has been retained for backwards-compatibility. This property is depreciated as of ODAC 6.00. Use TOraSession.Options.Direct instead.

3. **Are the ODAC connection components thread-safe?**
   In Client mode, ODAC is thread-safe. In Direct mode, we do not guarantee complete thread safety and recommend setting up a separate Connection set for each thread that uses ODAC.

4. **Behaviour of my application has changed when I upgraded ODAC. How can I restore the old behaviour with the new version?**
   We always try to keep ODAC compatible with previous versions, but sometimes we have to change behaviour of ODAC in order to enhance its functionality, or avoid bugs. If either of changes is undesirable for your application, and you want to save the old behaviour, please refer to the "Compatibility with previous versions" topic in ODAC help. This topic describes such changes, and how to revert to the old ODAC behaviour.

5. **When editing a DataSet, I get an exception with the message 'Update failed. Found %d records.' or 'Refresh failed. Found %d records.'**
   This error occurs when the database server is unable to determine which record to modify or delete. In other words, there are either more than one record or no records that suit the UPDATE criteria. Such situation can happen when you omit the unique field in a SELECT statement (TCustomDADataset.SQL) or when another user modifies the table simultaneously. This exception can be suppressed. Refer to TCustomDADataset.Options topic in ODAC help for more information.

6. **I would like to use MIDAS technology. Does ODAC support the IProvider interface?**
   Yes. Check out the Provider property of the TOraProvider component.

7. **What's the difference between TOraQuery, TSmartQuery, and TOraTable?**
   All these components are inherited from TDataSet and have all its capabilities. However, each component has a number of differences.
   - TOraQuery represents the most general way of executing queries and editing data. It is the most universal component, while TSmartQuery and TOraTable are designed for convenience only.
   - TSmartQuery includes all the functionality of TOraQuery component, and provides
additional features: expand fields, that lets all data controls be aware of all the fields belonging to updating table an not only those requested in SELECT clause, and smart refresh (in Professional and Developer editions only).

- TOraTable makes it unnecessary to write SQL statements to perform both data updates and data selection and emulates working with a local table. With TOraTable, you do not have to use SQL at all, and only have to specify the name of the appropriate table or view.

8. Can ODAC and BDE functions be used side-by-side in a single application?
Yes. There is no problem with using both ODAC and BDE functions in the same application.

© 1997-2021 Devart. All Rights Reserved.

3 Getting Started

This section introduces Oracle Data Access Components. It contains the information on how to install Oracle Data Access Components, quick walkthroughs to get started developing applications with it, information on technical licensing and deployment, and brief description of ODAC documentation and samples.

- Installation
- Migration Wizard
- Connecting to Oracle
- Creating Database Objects
- Retrieving and Modifying Data
- Inserting Data Into Tables
- Working With Oracle Stored Procedures
- Using Transactions

© 1997-2021 Devart. All Rights Reserved.

3.1 Installation

This topic contains the environment changes made by the ODAC installer. If you are having problems with using ODAC or compiling ODAC-based products, check this list to make sure
that your system is properly configured.

Compiled versions of ODAC are installed automatically by the ODAC Installer for all supported IDEs except for Lazarus. Versions of ODAC with Source Code must be installed manually. Installation of ODAC from sources is described in the supplied ReadmeSrc.html file.

Table of Contents
1. Before installing ODAC
2. Installed packages
   - Delphi/C++Builder Win32 project packages
   - Additional packages for using ODAC managers and wizards
3. Environment Changes
   - Delphi
   - C++Builder
   - Lazarus
4. Installation of Additional Components and Add-ins
   - TOraProvider
   - DBMonitor
5. Uninstalling
   - Delphi and C++Builder
   - Lazarus

Before installing ODAC ...

Please read our Compatibility page and make sure that your system satisfies the requirements.

Two versions of ODAC cannot be simultaneously installed for the same IDE, and, since the Devart Data Access Components products have some shared bpl files, newer versions of ODAC may be incompatible with older versions of MyDAC, IBDAC, and SDAC.

So before installing a new version of ODAC, uninstall any previous version of ODAC you may have, and check if your new install is compatible with other Devart Data Access Components products you have installed. For more information please see Using several products in one IDE. If you run into problems or have any compatibility questions, please email odac@devart.com
**Note:** You can avoid performing ODAC uninstallation manually when upgrading to a new version by directing the ODAC installation program to overwrite previous versions. To do this, execute the installation program from the command line with a `/force` parameter (Start | Run and type `odacxx.exe /force`, specifying the full path to the appropriate version of the installation program).

### Installed packages

**Note:** `%ODAC%` denotes the path to your ODAC installation directory.

#### Delphi/C++Builder Win32 project packages

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>dacXX.bpl</td>
<td>DAC run-time package</td>
<td>Windows\System32</td>
</tr>
<tr>
<td>dcdacXX.bpl</td>
<td>DAC design-time package</td>
<td>Delphi\Bin</td>
</tr>
<tr>
<td>dacvclXX.bpl*</td>
<td>DAC VCL support package</td>
<td>Delphi\Bin</td>
</tr>
<tr>
<td>odacXX.bpl</td>
<td>ODAC run-time package</td>
<td>Windows\System32</td>
</tr>
<tr>
<td>dclodacXX.bpl</td>
<td>ODAC design-time package</td>
<td>Delphi\Bin</td>
</tr>
<tr>
<td>odacvclXX.bpl*</td>
<td>VCL support package</td>
<td>Delphi\Bin</td>
</tr>
<tr>
<td>oraprovXX.bpl</td>
<td>TOraProvider component</td>
<td>Delphi\Bin</td>
</tr>
<tr>
<td>crcontrolsXX.bpl</td>
<td>TCRDBGrid component</td>
<td>Delphi\Bin</td>
</tr>
</tbody>
</table>

#### Additional packages for using ODAC managers and wizards

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasetmanagerXX.bpl</td>
<td>DataSet Manager package</td>
<td>Delphi\Bin</td>
</tr>
<tr>
<td>oramigwizardXX.dll</td>
<td>ODAC BDE Migration wizard</td>
<td>%ODAC%\Bin</td>
</tr>
</tbody>
</table>

### Environment Changes

To compile ODAC-based applications, your environment must be configured to have access to the ODAC libraries. Environment changes are IDE-dependent.

For all instructions, replace `%ODAC%` with the path to your ODAC installation directory.
Delphi
- `%ODAC%\Lib` should be included in the Library Path accessible from Tools | Environment options | Library.

The ODAC Installer performs Delphi environment changes automatically for compiled versions of ODAC.

C++Builder

C++Builder 6:
- `$\{BCB\}\ODAC\Lib` should be included in the Library Path of the Default Project Options accessible from Project | Options | Directories/Conditionals.
- `$\{BCB\}\ODAC\Include` should be included in the Include Path of the Default Project Options accessible from Project | Options | Directories/Conditionals.

C++Builder 2006 and higher:
- `$\{BCB\}\ODAC\Lib` should be included in the Library search path of the Default Project Options accessible from Project | Default Options | C++Builder | Linker | Paths and Defines.
- `$\{BCB\}\ODAC\Include` should be included in the Include search path of the Default Project Options accessible from Project | Default Options | C++Builder | C++ Compiler | Paths and Defines.

The ODAC Installer performs C++Builder environment changes for compiled versions of ODAC automatically.

Lazarus

The ODAC installation program only copies ODAC files. You need to install ODAC packages to Lazarus IDE manually. Open `%ODAC%\Source\Lazarus1\dclodac10.lpk` (for Trial version `%ODAC%\Packages\dclodac10.lpk`) file in Lazarus and press the Install button. After that Lazarus IDE will be rebuilded with ODAC packages.

Do not press the Compile button for the package. Compiling will fail because there are no ODAC sources.

To check that your environment has been properly configured, try to compile one of the demo projects included with ODAC. The ODAC demo projects are located in `%ODAC%\Demos`.

Installation of Additional Components and Add-ins

TOraProvider

DBMonitor

DBMonitor is a an easy-to-use tool to provide visual monitoring of your database applications. It is provided as an alternative to Borland SQL Monitor which is also supported by ODAC. DBMonitor is intended to hamper an application that is being monitored as little as possible. For more information, visit the [DBMonitor page online](#).

Uninstalling

To uninstall the Compiled versions of ODAC, select **Settings -> Control Panel -> Add or Remove Programs** from the Start menu (Windows XP and earlier) or select Control Panel from the Start menu and click **Uninstall a program** in the Control Panel window (Windows Vista and higher). Then select ODAC from the installed software list and click **Uninstall** (Windows XP and earlier) or **Uninstall/Change** (Windows Vista and higher).

Delphi and C++Builder

To uninstall Source versions of ODAC, select **Install Packages...** from the **Components** menu and remove the following packages:
- Devart Controls (CrControlsXXX.bpl)
- Devart Data Access Components (dcldavXXX.bpl)
- Devart Data Access GUI Related Components (dacvclXXX.bpl)
- Devart DataSet Manager (DataSetManagerXXX.bpl)
- Oracle Data Access Components (dclodacXXX.bpl)
- Oracle Data Access GUI Related Components (odacvclXXX.bpl)
- OraProvider Package (oraprovXXX.bpl)

Lazarus

To uninstall Source versions of ODAC, start the IDE, select **Install/Uninstall Packages** from the **Packages** menu, and remove the following packages.
- dac10 xx.xx.xx.xx
- dacvcl10 xx.xx.xx.xx
- dcldac10 xx.xx.xx.xx
- dclodac10 xx.xx.xx.xx
3.2 Migrating from BDE and DOA

Note: Migration Wizard is only available for Delphi.

Migration Wizard allows you to convert your BDE or DOA projects to ODAC. This wizard replaces BDE or DOA components in a specified project (.dfm and .pas files) with ODAC components.

To convert a project, perform the following steps.

- Select Migration Wizard from the ODAC menu
- Select Replace BDE components to replace the corresponding components with ODAC ones and click the Next button. If you need to replace DOA components, select DOA from the drop-down list on the right and click Next.
- Select the location of the files to search - current open project or disc folder.
- If you have selected Disc folder on the previous step, specify the required folder and specify whether to process subfolders. Press the Next button.
- Select whether to make backup (it is highly recommended to make a backup), backup location, and log parameters, and press the Next button. Default backup location is RBackup folder in your project folder.
- Check your settings and press the Finish button to start the conversion operation.
- The project should be saved before conversion. You will be asked before saving it. Click Yes to continue project conversion.

After the project conversion it will be reopened.

The Wizard just replaces all standard BDE components. Probably you will need to make some changes manually to compile your application successfully.

If some problems occur while making changes, you can restore your project from backup file. To do this, perform the following steps.

- Select Migration Wizard from the ODAC menu
- Select Restore original files from backup and press the Next button.
3.3 Connecting to Oracle

Contents
- Requirements
- General information
- Creating OraSession
  - Design time creation
  - Run time creation
- Opening connection
- Closing connection
- Modifying connection
- Additional information
- See Also

Requirements
In order to connect to Oracle server you need the server itself running, ODAC installed and IDE running. If you have Oracle Client Software installed and want to use it, you need to know TNS alias name, login and password. If you do not wish to use OCI, you have to know host name or IP address, Oracle System Identifier (SID) or Oracle Service Name, port, login and password.

General information
To establish a connection to server you have to provide some connection parameters to ODAC. This information is used by OraSession component to find the server and login with credentials of your account. The parameters are represented as connection string. You can compose the connection string manually or have ODAC construct it for you.
There are two ways to connect to server: with and without Oracle Client Interface. This is controlled by Direct property. It indicates whether the Oracle Client Interface will be used for connecting to server. By default Direct mode is disabled to preserve maximal functionality. Switch to Direct mode if you want to work in a system without Oracle Client Software installed.

**Creating OraSession**

**Design time creation**

The following assumes that you have IDE running, and you are currently focused on a form designer.

1. Find the OraSession component on the ODAC tab of the component palette.
2. Double-click the component. Notice that new object appears on the designer underneath the form. If this is first time you create the OraSession in this application it is named OraSession1.
3. Click on the OraSession1 object and press F11 to focus on object's properties. Or double-click on OraSession1 to open the dialog.
4. If you connect through OCI, in the Server property provide TNS alias of the server.
5. If you use Direct mode, perform the following assignments:
   - set Direct to true
   - Set the Server property to a string that contains the host address of the database server, port number, and the Oracle System Identifier (SID) or Oracle Service Name in the following format: Host:Port:SID or Host:Port:sn=ServiceName
6. In the Username property specify your login. For example, scott.
7. In the Password property specify your password. For example, tiger.

**Run time creation**

Same operations performed in runtime look as follows (note that you have to add DB, DBAccess, Ora units to the uses clause):

```delphi
uses DB, DBAccess, Ora;
...
var
  OraSession1: TOraSession;
begint
  OraSession1 := TOraSession.Create(nil);
  OraSession1.Server := 'ORASERVER';
  OraSession1.Username := 'SCOTT';
  OraSession1.Password := 'TIGER';
```
[Delphi Direct]

uses DB, DBAccess, Ora;
...
var
  OraSession1: TOraSession;
begin
  OraSession1 := TOraSession.Create(nil);
  OraSession1.Options.Direct := True;
  OraSession1.Server := 'LOCALHOST:1521:ORASERVER';
  OraSession1.Username := 'SCOTT';
  OraSession1.Password := 'TIGER';

[C++ Builder OCI]

#pragma link "DBAccess"
#pragma link "Ora"
...
  TOraSession *OraSession1 = new TOraSession(NULL);
  OraSession1->Server = "ORASERVER";
  OraSession1->Username = "SCOT";
  OraSession1->Password = "TIGER";

[C++ Builder Direct]

#pragma link "DBAccess"
#pragma link "Ora"
...
  TOraSession *OraSession1 = new TOraSession(NULL);
  OraSession1->Options->Direct = True;
  OraSession1->Server = "LOCALHOST:1521:ORASERVER";
  OraSession1->Username = "SCOT";
  OraSession1->Password = "TIGER";

You can do this all in single assignment. It actually does not matter whether connection string
is assigned directly or composed with particular properties. After you assign a value to
ConnectionString property all other properties are populated with parsed values. So you can
choose what is more convenient for you.

[Delphi OCI]

  OraSession1.ConnectionString := 'SCOTT/TIGER@ORASERVER';

[Delphi Direct]

  OraSession1.ConnectionString := 'SCOTT/TIGER@LOCALHOST:1521:ORASERVER';

[C++ Builder OCI]

  OraSession1->ConnectionString = "SCOTT/TIGER@ORASERVER";

[C++ Builder Direct]

  OraSession1->ConnectionString = "SCOTT/TIGER@LOCALHOST:1521:ORASERVER";
Opening connection

Opening a connection is as simple as that:

[Delphi]

    OraSession1.Connect;

[C++ Builder]

    OraSession1->Connect();

Of course, the OraSession1 must have valid connection string assigned earlier. When you call Connect, ODAC tries to find the host and connect to server. If any problem occurs it raises an exception with brief explanation on what is wrong. Finally, when connection is established, the Connect method returns and Connected property is changed to True.

In design time you can connect to server in few steps:
1. In the dialog window provide necessary logon information.
2. Click Connect button to establish connection.

Or you can simply change Connected property to True in Properties window to establish connection using current connection string.

Closing connection

To close a connection call its Disconnect method, or set its Connected property to False.

The following example summarizes aforementioned information and shows how to create, setup, open, use and then close the connection.

[Delphi]

    var
    OraSession1: TOraSession;
    begin
        OraSession1 := TOraSession.Create(nil);
        OraSession1.ConnectString := 'SCOTT/TIGER@ORASERVER';
        OraSession1.Connect;
        ShowMessage(OraSession1.OracleVersion);
        OraSession1.Disconnect;
    end;

[C++ Builder]

    #pragma link "DBAccess"
    #pragma link "Ora"

    ...    
    TOraSession *OraSession1 = new TOraSession(NULL);
    OraSession1->Options->Direct = True;
    OraSession1->Server = "LOCALHOST:1521:ORASERVER";
Modifying connection

You can modify connection by changing properties of OraSession object. Keep in mind that while some of the properties can be altered freely, most of them close connection when new value is assigned. For example, if you change Server property, it gets closed immediately, and you have to reopen it manually.

Additional information

ODAC has wide set of features you can take advantage of. The following list enumerates some of them so you can explore the advanced techniques to achieve better performance, balance network load or enable additional capabilities.

Asynchronous connection opening Connection pooling (refer to MSDN documentation for information about connection pooling).
• Asynchronous connection opening
• Connection pooling

See Also

• TOraSession

© 1997-2021 Devart. All Rights Reserved.
Request Support    DAC Forum    Provide Feedback

3.4 Creating Database Objects

This tutorial describes how to create tables, stored procedures and other objects in Oracle.

Contents

• Requirements
• General information
• Using SQL*Plus
• Additional information

Requirements

In order to create database objects you have to connect to server. This process is described in details in Connecting to Oracle.
General information

Database objects are created using Data Definition Language (DDL), which is a part of SQL. The DDL statements can be executed on server by account that has necessary privileges.

There are two ways to manipulate a database. You can build DDL statements manually and run them within Oracle SQL*Plus or component like OraQuery. Another way is to use IDE - visual shells that provide graphical user interface to manage database. We will discuss both ways.

Using SQL*Plus
1. Launch the SQL*Plus and authorize yourself.
2. Type the following code and press Enter. This will create first of the tables we'll use for tutorial purposes.:

```sql
CREATE TABLE dept (  deptno INT PRIMARY KEY,  dname VARCHAR2(14),  loc VARCHAR2(13)
);
```
3. Run the following query. This is another table we'll use.

```sql
CREATE TABLE emp (  empno INT PRIMARY KEY,  ename VARCHAR2(10),  job VARCHAR2(9),  mgr INT,  hiredate DATE,  sal FLOAT,  comm FLOAT,  deptno INT REFERENCES dept
);
```
4. These two tables are enough to demonstrate basic functionality. Now you can type exit to exit the SQL*Plus.

Additional information

Actually there are lots of ways to create tables on server. Any tool or component that is capable of running a SQL query, can be used to manage database objects. For example, OracleCommand suits fine for creating objects one by one, while OracleScript is designed for executing series of DDL/DML statements. For information on DDL statements syntax refer to Oracle documentation.
3.5 Retrieving and Modifying Data

Introducing
This tutorial describes how to use OraQuery component.

Requirements
This walkthrough supposes that you know how to connect to server, how to create the necessary objects on the server, and how to insert the data to the created tables.

Retrieving and Updating Data
In this sample we are using OraQuery to retrieve and manipulate data. For more information, refer to the description of this class in our documentation.

[Delphi]

```delphi
program Project1;
{$APPTYPE CONSOLE}
uses
  SysUtils,
  DB,
  DBAccess,
  Ora;

procedure PrintDept(OraSession: TOraSession);
var
  OraQuery: TOraQuery;
  i: integer;
begin
  OraQuery := TOraQuery.Create(nil);
  OraQuery.Session := OraSession;
  OraQuery.SQL.Text := 'SELECT * FROM dept';
  try
    OraQuery.Open;
    for i:= 0 to OraQuery.Fields.Count - 1 do
      write(OraQuery.Fields[i]._DisplayName+#09);
    writeln;
    while not OraQuery.Eof do
      begin
        for i:= 0 to OraQuery.Fields.Count - 1 do
          write(OraQuery.Fields[i].AsString+#09);
        writeln;
        OraQuery.Next;
      end;
    finally
      OraQuery.close;
      OraQuery.Free;
  end;
end;
procedure ModifyDept(OraSession: TOraSession; SQLText: string; action: string);
var
  OraQuery: TOraQuery;
```
begin
  OraQuery := TOraQuery.Create(nil);
  OraQuery.Session := OraSession;
  OraQuery.SQL.Text := SQLText;
  try
    OraQuery.Execute;
    writeln;
    write(format('Rows in DEPT %s: %d', [action,OraQuery.RowsAffected]));
  finally
    OraQuery.Free;
  end;
end;

OraSession: TораSession;
begin
OraSession := TОraSession.Create(nil);
OraSession.ConnectionString := 'SCOTT/TIGER@ORCL1200';
try
  OraSession.Connect;
  PrintDept(OraSession);
  ModifyDept(OraSession,'UPDATE DEPT SET LOC=''VEGAS'' WHERE DEPTNO > 20',
             OraQuery.RowsAffected);
  ModifyDept(OraSession, 'INSERT INTO dept (deptno, dname, loc) VALUES (50,
                          'TWO','RENO');
  ModifyDept(OraSession,'DELETE FROM dept WHERE deptno = 50','deleted');
  Readln;
finally
  OraSession.Free;
end;
end.

[C++ Builder]

#include <vcl.h>
#pragma hdrstop
#include <tchar.h>
#include <stdio.h>
#include <DBAccess.hpp>
#include <Ora.hpp>
#pragma argsused

void PrintDept(TOraSession *OraSession)
{
  TOraQuery *OraQuery = new TOraQuery(NULL);
  OraQuery->SQL->Text = "SELECT * FROM dept";
  OraQuery->Session = OraSession;
  int i;
  try
  {
    OraQuery->Open();
    for(i = 0; i< OraQuery->Fields->Count; i++)
      printf("%s\t",OraQuery->Fields->operator []()[i]->DisplayName.t_str());
    printf("\n");
    while (!OraQuery->Eof)
    {
      for(i = 0; i< OraQuery->Fields->Count; i++)
        printf("%s\t",OraQuery->Fields->operator []()[i]->AsString.t_str());
      printf("\n");
      OraQuery->Next();
    }
  }
  catch (Exception e)
  {  
    OraQuery->Free();
  }
}
3.6 Inserting Data Into Tables

This tutorial describes how to use OraQuery component to insert data into tables by means of executing SQL queries.

- **Requirements**
- **General information**
Getting Started

Getting Started

- Inserting data in run time
- Design time setup
- Additional information

Requirements

This walkthrough supposes that you know how to connect to server (tutorial Logging onto the server) and that necessary objects are already created on the server (tutorial Creating database objects).

General information

Data on server can be modified (inserted, changed or deleted) using Data Manipulation Language (DML), which is a part of SQL. The DML statements can be executed on server by account that has necessary privileges.

There are two ways to manipulate a database. You can build DML statements manually and run them within some component like OraQuery. Another way is to use design-time features that provide graphical user interface to manage database. We will discuss both ways.

The goal of this tutorial is to insert the following data into tables dept and emp:

**Table dept**

<table>
<thead>
<tr>
<th>deptno</th>
<th>dname</th>
<th>loc</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>ACCOUNTING</td>
<td>NEW YORK</td>
</tr>
<tr>
<td>20</td>
<td>RESEARCH</td>
<td>DALLAS</td>
</tr>
<tr>
<td>30</td>
<td>SALES</td>
<td>CHICAGO</td>
</tr>
<tr>
<td>40</td>
<td>OPERATIONS</td>
<td>BOSTON</td>
</tr>
</tbody>
</table>

**Table emp**

<table>
<thead>
<tr>
<th>empno</th>
<th>ename</th>
<th>job</th>
<th>mgr</th>
<th>hiredate</th>
<th>sal</th>
<th>comm</th>
<th>deptno</th>
</tr>
</thead>
<tbody>
<tr>
<td>7369</td>
<td>SMIT</td>
<td>CLE</td>
<td>7902</td>
<td>17.12</td>
<td>800</td>
<td>NULL</td>
<td>20</td>
</tr>
</tbody>
</table>
Inserting data in run time

To insert the first row into table dept you can use the following statement:

```
INSERT INTO dept (deptno, dname, loc) VALUES (10,'ACCOUNTING','NEW YORK')
```

The following code fragment executes the query:
[Delphi]

```delphi
var
  OraSession1: TOraSession;
  OraQuery1: TOraQuery;
begin
  OraSession1 := TOraSession.Create(nil);
  OraQuery1 := TOraQuery.Create(nil);
  OraSession1.ConnectionString := 'SCOTT/TIGER@ORASERVER';
  OraQuery1.SQL.Text := 'INSERT INTO dept (deptno, dname, loc) VALUES (10, ''',
  OraQuery1.Session := OraSession1;
  OraSession1>LoginPrompt := False;
  try
    OraSession1.Connect;
    try
      OraQuery1.Execute;
      ShowMessage(IntToStr(OraQuery1.RowsAffected)+' rows were affected.');
    except
      ShowMessage('Error encountered during INSERT operation.');
    end;
  finally
    OraSession1.disconnect;
    OraQuery1.Free;
    OraSession1.Free;
  end;
end;
```

[C++ Builder]

```cpp
TOraSession *OraSession1 = new TOraSession(NULL);
TOraQuery *OraQuery1 = new TOraQuery(NULL);
OraSession1->ConnectionString = "SCOTT/TIGER@ORCL1020";
OraQuery1->SQL->Text = "INSERT INTO dept (deptno, dname, loc) VALUES (10,
OraQuery1->Session = OraSession1;
OraSession1->LoginPrompt = false;
  try
  {
    OraSession1->Connect();
    try
    {
      OraQuery1->Execute();
      ShowMessage(IntToStr(OraQuery1->RowsAffected)+" rows were affected.");
    }
    catch(const Exception& e)
    {
      ShowMessage("Error encountered during INSERT operation.");
    }
  }
  finally
  {
    OraSession1->Disconnect();
    OraQuery1->Free();
    OraSession1->Free();
  }
```

The sample first creates a connection with hardcoded connection string. Then it creates OraQuery object, assigns the query text and connection to the OraQuery instance.
Connection is opened then. The Execute method of OraQuery runs SQL statement in the Text property. The RowsAffected property stores the number of rows affected by the query. This method is not intended to run SELECT statements. We will discuss retrieving data in other tutorials.

If the query is executed successfully you are notified about number of affected rows. If some error occurs you get the error message. The connection is closed anyway. It is recommended that you use `try ... finally` clauses to make sure the connections are closed properly.

**Design time setup**

Same operations in design time include following steps:

1. Place OraSession component on a designer.
   1. Setup its properties and open connection by changing the Connected property to True or double-click on the component, in the dialog window provide necessary logon information and press Connect button.
2. Place OraQuery component on the designer.
3. In its Session property select name of the OraSession instance on the designer.
4. Click on the ellipsis in SQL property in Properties window double-click the component in the OraQuery editor on the SQL tab and enter the following query:

   ```sql
   INSERT INTO dept VALUES (20,'SALES','DALLAS')
   ```

   and press the Execute button.

**Additional information**

Actually there are lots of ways to insert data into tables. Any tool or component that is capable of running a SQL query, can be used to manage data. Some components are best for performing certain tasks. For example, OraLoader is the fastest way to insert data, OraScript is designed for executing series of statements. For more information on these components refer to ODAC reference.

**See Also**

- Getting Started
- OraQuery Class
- OraLoader Class
- OraScript Class
3.7 Working With Oracle Stored Procedures

This section describes how to create and use Oracle stored procedures and functions with ODAC.

It supposes that you know how to connect to server, how to create the necessary objects on the server, and how to manipulate with the data stored in database tables. The section describes stored procedures using for typed DataSets.

This section contains the following articles:

- Stored Procedures - General Information
  Contains general information about stored procedures and functions and describes how to create them.
- Using Stored Procedures via the TOraStoredProc Class
  Contains an information about using stored procedures with the help of TOraStoredProc class.
- Using Package Procedures
  Describes approaches of the stored procedure usage when they are included into the Oracle packages.

See Also

- Stored Procedures - General Information
- Using Stored Procedures via the TOraStoredProc class
- Using Package Procedures

3.7.1 Stored Procedures - General Information

This section contains information about general aspects of stored procedures usage.

A stored procedure is a schema object that consists of a set of SQL statements and other PL/SQL constructs, grouped together, stored in the database, and run as a unit to solve a specific problem or perform a set of related tasks. Procedures let you combine the ease and flexibility of SQL with the procedural functionality of a structured programming language. Large or complex processing that might require the execution of several SQL statements is moved into stored procedures, and all applications call the procedures only.
Objects similar to stored procedures are stored functions. Almost everything that is true for procedures, holds for functions as well. The main difference between these objects is that function has a return value, and procedure has not.

A stored procedures and functions may have input, output, and input/output parameters.

Input parameter is a parameter whose value is passed into a stored procedure/function module. The value of an IN parameter is a constant; it can’t be changed or reassigned within the module.

For example, the following procedure inserts a row into the Dept table:

```sql
CREATE PROCEDURE dept_insert (pDeptno INTEGER, pDname VARCHAR2, pLoc VARCHAR2) AS BEGIN  INSERT INTO dept(deptno, dname, loc) VALUES (pDeptno, pDname, pLoc); END;
```

It needs to receive the values to be inserted into the new record, and thus the procedure has three input parameters, corresponding to each field of the table. The procedure may be executed inside a PL/SQL block like follows:

```sql
begin  dept_insert (10, 'Accounting', 'New York');end;
```

Output parameter is a parameter whose value is passed out of the stored procedure/function module, back to the calling PL/SQL block. An OUT parameter must be a variable, not a constant. It can be found only on the left-hand side of an assignment in the module. You cannot assign a default value to an OUT parameter outside of the module’s body. In other words, an OUT parameter behaves like an uninitialized variable. In the following sample, the stored procedure returns the count of records in table Dept:

```sql
CREATE PROCEDURE dept_count (cnt OUT INTEGER) AS BEGIN  SELECT COUNT(*) INTO cnt FROM dept; END;
```

An input/output parameter is a parameter that functions as an IN or an OUT parameter or both. The value of the IN/OUT parameter is passed into the stored procedure/function and a new value can be assigned to the parameter and passed out of the module. An IN/OUT parameter must be a variable, not a constant. However, it can be found on both sides of an assignment. In other words, an IN/OUT parameter behaves like an initialized variable.

Besides scalar variables, a stored procedure can return result sets, i.e. the results of a SELECT statement. In Oracle, the cursor variables are used for this case. A cursor may be interpreted as a reference to the result set. The following sample demonstrates how a
simplest select statement can be wrapped in a stored procedure:

```sql
CREATE PROCEDURE get_all_depts_proc (cur OUT SYS_REFCURSOR) AS
BEGIN
  OPEN cur FOR SELECT * FROM dept;
END;
```

The same SELECT statement can be used via a stored function as follows:

```sql
CREATE OR REPLACE FUNCTION get_all_depts_func RETURN SYS_REFCURSOR AS
  cur SYS_REFCURSOR;
BEGIN
  OPEN cur FOR SELECT * FROM dept;
  RETURN cur;
END;
```

Here the cursor is passed as the return value instead of being an output parameter.

**See Also:**
- Using Stored Procedures via the TOraStoredProc class
- Package Procedures

3.7.2 Using Stored Procedures via the TOraStoredProc Class

This topic describes how to use Oracle stored procedures and functions with ODAC by the help of TOraStoredProc class.

The following sample demonstrates the work with an Oracle stored procedure using the get_all_depts_proc procedure from the previous section. Since the out parameter of the procedure is the cursor, it is possible to work with the procedure as with a simple DataSet.

**Note:** If several out parameters in the procedure are cursors, then TOraStoredProc will work only with the first one of them as with a DataSet.

**[Delphi]**

```delphi
program Project1;  // {$APPTYPE CONSOLE}
uses
  SysUtils,
  DB,
  DBAccess,
  Ora;
procedure PrintDept(OraSession: TOraSession);
var
```
OraStoredProc: TOraStoredProc;
i: integer;
begin
  //procedure creation
  OraStoredProc := TOraStoredProc.Create(nil);
  OraStoredProc.Session := OraSession;
  //setting the stored procedure name
  OraStoredProc.StoredProcName := 'get_all_depts_proc';
  //The ParamCheck property must be set to True for automatic
definition of parameters used in the stored procedure
  OraStoredProc.ParamCheck := True;
  try
    //execution of the stored procedure
    OraStoredProc.Execute;
    //retrieving data from the cursor returned by the procedure
    for i := 0 to OraStoredProc.FieldCount - 1 do
      Write(OraStoredProc.Fields[i].DisplayName+#09);
      WriteLn;
    while not OraStoredProc.Eof do
      begin
      for i := 0 to OraStoredProc.Fields.Count - 1 do
        Write(OraStoredProc.Fields[i].AsString+#09);
      WriteLn;
        OraStoredProc.Next;
      end;
    finally
      OraStoredProc.close;
      OraStoredProc.Free;
    end;
  end;
var
  OraSession: TOraSession;
begin
  OraSession := TOraSession.Create(nil);
  OraSession.ConnectionString := 'SCOTT/TEGR@ORCL1020';
  try
    OraSession.Connect;
    PrintDept(OraSession);
    readln;
  finally
    OraSession.Free;
  end;
end.

[C++ Builder]

#include <vcl.h>
#pragma hdrstop
#include <tchar.h>
#include <stdio.h>
#include <DBAccess.hpp>
#include <Ora.hpp>
#pragma argsused
void PrintDept(TOraSession *OraSession)
{
  TOraStoredProc *OraStoredProc = new TOraStoredProc(NULL);
  OraStoredProc->StoredProcName = "get_all_depts_proc";
```
OraStoredProc->Session = OraSession;
int i;
try
{
OraStoredProc->Execute();
for(i = 0; i < OraStoredProc->Fields->Count; i++)
    printf("%s\t",OraStoredProc->Fields->operator [][](i)->DisplayName.t_str());
printf("\n");
while (!OraStoredProc->Eof)
{
    for(i = 0; i < OraStoredProc->Fields->Count; i++)
        printf("%s\t",OraStoredProc->Fields->operator [][](i)->AsString.t_str());
    printf("\n");
    OraStoredProc->Next();
}
}  
__finally
{
OraStoredProc->Free();
}
}
int _tmain(int argc, _TCHAR* argv[])
{
    TOraSession *OraSession = new TOraSession(NULL);
    int i;
    try
    {
        OraSession->ConnectionString = "SCOTT/TIGER@ORCL1020";
        OraSession->Connect();
        PrintDept(OraSession);
        system("pause");
    }  
    __finally
    {
        OraSession->Free();
    }
    return 0;
}
```

**See Also**

- Stored Procedures - General Information
- Using Package Procedures

© 1997-2021
Devar, All Rights Reserved.

**3.7.3 Using Package Procedures**

This topic describes how to create and use Oracle stored procedures and functions within the Oracle packages with ODAC.

In Oracle databases, stored procedures and functions may be grouped into specific sets
which are called packages. To call a package procedure, one needs only to add the package name before the procedure's name, like "package.get_all_depts_proc". However, with ODAC using of packages may be even easier due to typed and untyped. The first ones are the classes generated by the Package Wizard, the second ones are instances of the TOraPackage class.

Untyped OraPackage may be set to represent any package specified by the name, provided that this package is available for the connection used. TOraPackage class has a set of methods intended to execute procedures and retrieve their descriptions.

Typed OraPackage is a class representing the only specific package. For each procedure or function of this package, instances of corresponding typed OraPackage have a special method. Such approach allows to invoke stored procedures just like usual object methods. Typed Oracle packages can be created using Oracle Package Wizard. For more information on this see Using Package Wizard for working with PL/SQL Packages.

See Also

- Stored Procedures - General Information
- Using Stored Procedures via the TOraStoredProc class

3.8 Working With PL/SQL

This section describes how to create and use Oracle PL/SQL blocks ODAC.

It supposes that you know how to connect to server, how to create the necessary objects on the server, and how to manipulate with the data stored in database tables. The section describes stored procedures using for typed DataSets.

This section contains the following articles:

- PL/SQL - General Information

Contains general information PL/SQL blocks and describes how to create them.

- Using PL/SQL via the TOraSQL Class

Contains an information about using PL/SQL with the help of TOraSQL class.
3.8.1 PL/SQL - General Information

PL/SQL (Procedural Language/Structured Query Language, also known as Pretty Lazy/Structured Query Language) is Oracle Corporation’s procedural extension language for SQL and the Oracle relational database. PL/SQL's general syntax resembles that of Ada. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java. PL/SQL is available in Oracle Database (since version 7).

- Introduction
- Basic Code Structure
- Data Types
- Control Operators
- Application Sample

Introduction

PL/SQL supports variables, conditions, loops and exceptions. Arrays are also supported, though in a somewhat unusual way, involving the use of PL/SQL collections. PL/SQL collections are a slightly advanced topic. Implementations from version 8 of Oracle Database onwards have included features associated with object-orientation. PL/SQL program units (essentially code containers) can be compiled into the Oracle database. Programmers can thus embed PL/SQL units of functionality into the database directly. They also can write scripts containing PL/SQL program units that can be read into the database using the Oracle SQL*Plus tool. Once the program units have been stored into the database, they become available for execution at a later time. While programmers can readily embed Data Manipulation Language (DML) statements directly into their PL/SQL code using straightforward SQL statements, Data Definition Language (DDL) requires more complex "Dynamic SQL" statements to be written in the PL/SQL code. However, DML statements underpin the majority of PL/SQL code in typical software applications. In the case of PL/SQL dynamic SQL, early versions of the Oracle Database required the use of a complicated Oracle DBMS_SQL package library. More recent versions have however introduced a simpler "Native Dynamic SQL", along with an associated EXECUTE IMMEDIATE syntax. Oracle Corporation customarily extends package functionality with each successive release of the Oracle Database.

Basic Code Structure

An application on PL/SQL consists of blocks (anonymous and named). A block can include nested blocks, aka subblocks. The general shape of a PL/SQL-block:

<<label>>
DECLARE        TYPE / item / FUNCTION / PROCEDURE declarations
BEGIN      Statements
EXCEPTION        EXCEPTION handlers
END label;

Data Types
The PL/SQL language supports the following type categories:
• nested data types, including collections and records;
• scalar;
• compound;
• reference;
• LOB-types;
• Object data types.

Control Operators
• selection statements:
  IF - THEN - END IF;
  IF - THEN - ELSE - END IF;
  IF - THEN - ELSIF - END IF;
  CASE - WHEN - THEN - END CASE;

• loop statements:
  LOOP - END LOOP;
  WHILE - LOOP - END LOOP;
  FOR - LOOP - END LOOP;
  EXIT;
  EXIT WHEN;

• GO TO statements:
  GOTO label_name;

Application Sample
Example of a program that updates data in the table and displays the number of changed records in the console

DECLARE
  cnt NUMBER;
BEGIN
  SELECT DEPTNO
INTO cnt
FROM DEPT
WHERE DEPTNO = 10;
UPDATE DEPT SET LOC='VEGAS' WHERE DEPTNO = 10;
DBMS_OUTPUT.PUT_LINE('Rows in DEPT updated '||sql%rowcount);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Row for update in DEPT no found');
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.put_line('Query return more that one row');
END;

See also:
- Using PL/SQL via the TOraSQL Class

3.8.2 Using PL/SQL via the TOraSQL Class

This topic describes how to use PL/SQL with ODAC by the help of TOraSQL class.

For work with PL/SQL blocks, the TOraSQL component is used, that allows using parameters and macros in PL/SQL code.

A sample application updating data in a table and returning the number of modified records to the console:

[Delphi]

OraSQL1.SQL.Text := 'DECLARE
  cnt NUMBER;
BEGIN
  SELECT DEPTNO
  INTO cnt
  FROM DEPT
  WHERE DNAME = :DNAME;
  UPDATE EMP SET SAL=SAL + 100 WHERE DEPTNO = cnt;
  :RES := sql%rowcount;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Row for update in EMP no found');
  WHEN TOO_MANY_ROWS THEN
    DBMS_OUTPUT.put_line('Query return more that one row');
END;

OraSQL1.ParamByName('DNAME').DataType := ftString;
OraSQL1.ParamByName('DNAME').ParamType := ptInput;
OraSQL1.ParamByName('DNAME').AsString := 'ACCOUNT';
OraSQL1.ParamByName('RES').DataType := ftInteger;
OraSQL1.ParamByName('RES').ParamType := ptOutput;
OraSQL1.Execute;
ShowMessage(Format('Rows in EMP updated: %d',[OraSQL1.ParamByName('RES').AsInteger]));
3.9 Using Transactions

Understanding Transactions

A transaction is one or several operations considered as a single unit of work which is completed entirely or have no effect at all ("all-or-nothing"). If a failure occurs at one point in the transaction, all of the updates can be rolled back to their pre-transaction state. A transaction must conform to the ACID properties - atomicity, consistency, isolation, and durability-in order to guarantee data consistency.

If a transaction involves multiple tables in the same database, then explicit transactions in PL/SQL often perform better. You can use COMMIT and ROLLBACK statements in your SQL to fix and discard respectively the previous commands in your current PL/SQL block. For more information, see Oracle PL/SQL documentation.

Otherwise, a transaction with plain SQL can be implemented via TOraTransaction TOraSession classes. For example, you can use TOraSession: start transaction on TOraSession, execute several SQL statements via TOraDataSet, and commit/rollback all operations when it is necessary. See the sample from the Local Transaction topic.

This article describes the way to manipulate transactions from your application (without
envolving PL/SQL transactions) - this is the most common case of working with transactions. Concerning your task, you can choose the type of transaction to implement - local or distributed. A transaction considered to be a local transaction when it is a single-phase transaction and is handled by the database directly. A distributed transaction is a transaction that affects several resources, it is coordinated by a transaction monitor and uses fail-safe mechanisms (such as two-phase commit) for transaction resolution.

Note: transaction will be global if either TransactionId or TransactionName property is set or if GlobalCoordinator property is gcMTS.

Local Transactions

To start local transaction with TObject transaction component, set DefaultSession property of the component to a session on which transaction will be performed. Set IsolationLevel property optionally. Then call StartTransaction method of the TObjectTransaction component. To manage transaction use Commit, Rollback, Savepoint, RollbackToSavepoint methods.

[Delphi]

```delphi
var
    OraSession: TOraSession;
    OraTransaction: TOraTransaction;
begin
    OraSession := TOraSession.Create(nil);
    OraTransaction := TOraTransaction.Create(nil);
    try
        OraSession.ConnectionString := 'login/password@SID';
        OraSession.Connect;
        OraTransaction.AddSession(OraSession);
        OraTransaction.IsolationLevel := ilReadCommitted;
        OraTransaction.StartTransaction;
        try
            OraSession.ExecSQL('INSERT INTO Dept(DeptNo, DName) Values(50, ''DEVELOP'');
            OraSession.ExecSQL('INSERT INTO Dept(DeptNo, DName) Values(60, ''PRODUCE'');
            OraTransaction.Commit;
            ShowMessage('Both records are written to database.');
        except
            OraTransaction.Rollback;
            ShowMessage('Neither record was written to database.');
        end;
    finally
        OraTransaction.Free;
        OraSession.Free;
    end;
end;
```

[C++ Builder]

```cpp
TOraSession *OraSession = new TOraSession(NULL);
TOraTransaction *OraTransaction = new TOraTransaction(NULL);
try
```
3.10 Demo Projects

ODAC includes a number of demo projects that show off the main ODAC functionality and development patterns.

The ODAC demo projects consist of one large project called OdacDemo with demos for all main ODAC components, use cases, and data access technologies, and a number of smaller projects on how to use ODAC in different IDEs and how to integrate ODAC with third-party components.

Most demo projects are built for Delphi and Embarcadero RAD Studio. There are only two ODAC demos for C++Builder. However, the C++Builder distribution includes source code for all other demo projects as well.

Where are the ODAC demo projects located?

In most cases all the ODAC demo projects are located in "%Odac%\Demos\".

In Delphi 2007 for Win32 under Windows Vista all the ODAC demo projects are located in "My Documents\Devar\Odac for Delphi 2007\Demos\", for example "C:\Documents and Settings \All Users\Documents\Devar\Odac for Delphi 2007\Demos\".

```cpp
{ 
  OraSession->ConnectString = "SCOTT/TIGER@ORCL1020";
  OraSession->Connect();
  OraTransaction->AddSession(OraSession);
  OraTransaction->IsolationLevel = ilReadCommitted;
  OraTransaction->StartTransaction();
  try
  { 
    OraSession->ExecSQL("INSERT INTO Dept(DeptNo, DName) Values(50, 'DEVELOPMENT');
    OraSession->ExecSQL("INSERT INTO Dept(DeptNo, DName) Values(60, 'PRODUCTS');
    ShowMessage("Both records are written to database.");
  } catch(const Exception& e)
  { 
    OraTransaction->Rollback();
    ShowMessage("Neither record was written to database.");
  }
  finally
  { 
    OraSession->Disconnect();
    OraTransaction->Free();
    OraSession->Free();
  }
```

© 1997-2021
Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback
The structure of the demo project directory depends on the IDE version you are using.

For most new IDEs the structure will be as following.

Demos

|—OdacDemo  [The main ODAC demo project]
|—Performance  [Demo project, that compares performance of ODAC with another components (BDE, ADO, dbExpress)]
|—ThirdParty
  |— [A collection of demo projects on integration with third-party components]
|—Miscellaneous
  |— [Some other demo projects on design technologies]

*OdacDemo* is the main demo project that shows off all the ODAC functionality. The other directories contain a number of supplementary demo projects that describe special use cases. The list of all samples in the ODAC demo project and the description for the supplementary projects is provided in the following section.

**Note:** This documentation describes ALL the ODAC demo projects. The actual demo projects you will have installed on your computer depends on your ODAC version, ODAC edition, and the IDE version you are using. The integration demos may require installation of third-party components to compile and work properly.

**Instructions for using the ODAC demo projects**

To explore an ODAC demo project,
1. Launch your IDE.
2. In your IDE, choose File|Open Project from the menu bar.
3. Find the directory you have installed ODAC to and open the Demos folder.
4. Browse through the demo project folders located here and open the project file of the demo you would like to use.
5. Compile and launch the demo. If it exists, consult the *ReadMe.txt* file for more details.

The executed version of the demo will contain a sample application written with ODAC or a navigable list of samples and sample descriptions. To use each sample properly, you will need to connect to a working Oracle server.

The included sample applications are fully functional. To use the demos, you have to set up a
connection to Oracle first. You can do that by clicking on the "Connect" button.

Many demos may also use some database objects. If so, they will have two object manipulation buttons, "Create" and "Drop". If your demo requires additional objects, click "Create" to create necessary database objects. When you are done with the demo, click "Drop" to remove all the objects used for the demo from your Oracle database.

**Note:** The ODAC demo directory includes two sample SQL scripts for creating and dropping all the test schema objects used in the ODAC demos. You can modify and execute this script manually, if you want. This will not change the behavior of the demos.

You can find a complete walkthrough for the main ODAC demo project in the [Getting Started](#) topic. Other ODAC demo projects include a `ReadMe.txt` file with individual building and launching instructions.

**Demo project descriptions**

**OdacDemo**

`OdacDemo` is one large project which includes three collections of demos.

**Working with components**

A collection of samples that show how to work with the basic ODAC components.

**General demos**

A collection of samples that show off the ODAC technology and demonstrate some ways to work with data.

**Oracle-specific demos**

A collection of samples that demonstrate how to incorporate Oracle features in database applications.

`OdacDemo` can be opened from `%Odac%/Demos/OdacDemo/odacdemo.dpr (.bdsproj). The following table describes all demos contained in this project.

### Working with Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alerter</td>
<td>Uses the <code>TOraAlerter</code> component to send messages between sessions through <code>DBMS_ALERT</code> and <code>DBMSPIPE</code> Oracle package</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ChangeNotification</strong></td>
<td>Demonstrates how to subscribe, receive, and reflect DML or DDL changes on objects associated with queries. Note: Requires execute privileges on DBMS_ALERT.</td>
</tr>
<tr>
<td><strong>ConnectDialog</strong></td>
<td>Demonstrates how to customize the ODAC connect dialog. Changes the standard ODAC connect dialog to two custom connect dialogs. The first customized sample dialog is inherited from the TForm class, and the second one is inherited from the default ODAC connect dialog class.</td>
</tr>
<tr>
<td><strong>CRDBGrid</strong></td>
<td>Demonstrates how to work with the TCRDBGrid component. Shows off main TCRDBGrid features, like filtering, searching, stretching, using compound headers, and more.</td>
</tr>
<tr>
<td><strong>ErrorHandler</strong></td>
<td>Demonstrates using the TOraErrorHandler for exception handling and translating error messages.</td>
</tr>
<tr>
<td><strong>Loader</strong></td>
<td>Uses the TOraLoader component to load data into a server table quickly. Demonstrates Direct and DML modes for loading data. In Direct mode, data is loaded through DirectPath API, which loads big volumes of data into a table faster than while using INSERT statement. In DML mode, data is processed with the DML array feature of Oracle. It also compares two TOraLoader data loading handlers: GetColumnData and PutData.</td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td>Demonstrates working with TOraQuery, which is one of the most useful ODAC components. Includes many TOraQuery usage scenarios. Demonstrates how to execute queries in both standard and NonBlocking mode and how to edit data and export it to XML files. Note: This is a very good introductory demo. We recommend starting with it when first becoming familiar with ODAC.</td>
</tr>
<tr>
<td><strong>Queue</strong></td>
<td>Demonstrates how to use ODAC to work with Oracle Streams Advanced Queuing. Implements notification on new messages. Shows how to create and manage Oracle Queues. Demonstrates the TOraQueue, TOraQueueTable, and TOraQueueAdmin components. Note: Requires DBMS_AQ and DBMS_AQADM privileges.</td>
</tr>
<tr>
<td><strong>Smart</strong></td>
<td>Uses TSmartQuery to customize refreshing, sorting, and server-side record management in a data grid. Shows how to perform local filtering, demonstrates several different kinds of record locking and refreshing, and working with FetchAll mode.</td>
</tr>
<tr>
<td><strong>Sql</strong></td>
<td>Uses TOraSQL to execute SQL statements and PL/SQL blocks. Demonstrates how to work in standard and NonBlocking modes, how to work with parameters in SQL, and how to break long-duration query execution.</td>
</tr>
<tr>
<td><strong>StoredProc</strong></td>
<td>Uses TOraStoredProc to access an editable recordset represented by an Oracle cursor from an Oracle stored procedure in the client.</td>
</tr>
</tbody>
</table>
Table
Demonstrates how to use TOraTable to work with data from a single table on the server without writing any SQL queries manually. Performs server-side data sorting and filtering and retrieves results for browsing and editing.

Trace
Uses TOraTrace to work with Oracle SQL and PL/SQL tracing.

Transaction
Demonstrates main approaches for setting up distributed transactions with the TOraTransaction component. Shows how to manage transactions, tune the transaction isolation level, and select the coordinator for a distributed transaction.

UpdateSQL
Demonstrates using the TOraUpdateSQL component to customize update commands. Lets you optionally use TOraSQL and TOraQuery objects for carrying out insert, delete, query, and update commands.

VirtualTable
Demonstrates working with the TVirtualTable component. This sample shows how to fill virtual dataset with data from other datasets, filter data by a given criteria, locate specified records, perform file operations, and change data and table structure.

General Demos

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>Demonstrates how to perform the most important tasks of working with data in CachedUpdates mode, including highlighting uncommitted changes, managing transactions, and committing changes in a batch.</td>
</tr>
<tr>
<td>FilterAndIndex</td>
<td>Demonstrates ODAC's local storage functionality. This sample shows how to perform local filtering, sorting and locating by multiple fields, including by calculated and lookup fields.</td>
</tr>
<tr>
<td>MasterDetail</td>
<td>Uses ODAC functionality to work with master/detail relationships. This sample shows how to use local master/detail functionality. Demonstrates different kinds of master/detail linking, including linking by SQL, simple fields, and calculated fields.</td>
</tr>
<tr>
<td>Pictures</td>
<td>Uses ODAC functionality to work with BLOB fields and graphics. The sample demonstrates how to retrieve binary data from Oracle database and display it on visual components. Sample also shows how to load and save pictures to files and to the database.</td>
</tr>
<tr>
<td>Threads</td>
<td>Demonstrates how ODAC can be used in multithreaded applications. This sample allows you to set up several threads and test ODAC's performance with multithreading.</td>
</tr>
</tbody>
</table>

Oracle-specific Demos
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrays</td>
<td>Demonstrates working with Oracle <strong>arrays</strong>. This sample lets you view and control how arrays are represented in dataset fields by the SparseArrays and ObjectView properties.</td>
</tr>
<tr>
<td>BFile</td>
<td>Shows the basics of working with file binary data stored in file systems located outside Oracle databases. This sample uses the TBFileField field type of ODAC.</td>
</tr>
<tr>
<td>BlobPictures</td>
<td>Demonstrates working with Oracle BLOB data types. The sample shows how to get binary data from the table, how to change BLOB fields using UPDATE statements, and how to insert a new record by executing stored procedure with a BLOB parameter. Also it shows off some extended BLOB handling functionality like local caching control, compression type changing, and more.</td>
</tr>
<tr>
<td>Clob</td>
<td>Demonstrates working with Oracle CLOB data types. The sample shows how to get a character stream from a table, how to change CLOB fields using UPDATE statements, and how to save and load data to/from a file. It also demonstrates several different ways of performing CLOB insertion.</td>
</tr>
<tr>
<td>Cursor</td>
<td>Uses ODAC functionality to work with Oracle <strong>Cursors</strong>. Shows how to fetch data from a Cursor parameter by setting <code>TOrDataSet.Cursor</code> to the <code>TOrParam.AsCursor</code>.</td>
</tr>
<tr>
<td>DMLArray</td>
<td>Demonstrates how to multiply execute SQL statements with different parameters by using ODAC functionality for the Oracle DML array feature.</td>
</tr>
<tr>
<td>FetchCursors</td>
<td>Uses <code>TOrQuery</code> to retrieve several Oracle cursors at once, fetch data in a batch, and close the cursors.</td>
</tr>
<tr>
<td>Long</td>
<td>Demonstrates working with Oracle LONG data types. The sample shows how to get character string from a table, update LONG fields and insert a new record. Also shows how to perform file operations with LONG fields.</td>
</tr>
<tr>
<td>LongStrings</td>
<td>Demonstrates ODAC functionality for working with long string fields (fields that have more than 256 characters). Shows different ways of their displaying as memo fields and string fields.</td>
</tr>
<tr>
<td>MultiCursors</td>
<td>Shows how to use one <code>TOrQuery</code> object to retrieve and update data from several tables by using several REF CURSOR parameters.</td>
</tr>
<tr>
<td>MultiQueries</td>
<td>Shows how Oracle queries are handled in multithreaded applications. This sample project lets you compare opening multiple queries within a single session or different sessions by setting <code>NonBlocking</code> and <code>FetchAll</code> =False modes.</td>
</tr>
<tr>
<td>NestedTables</td>
<td>Demonstrates using the <code>TOrNestedTable</code> component to work with Oracle nested tables. This sample project fills a TOrNestedTable dataset instance with the result set of a query to a table with a nested table field. It shows how to work with the data contained in nested table fields. Note that the nested table accessing interface is similar</td>
</tr>
</tbody>
</table>
to the interface for accessing cursor data in a dataset representation of a table with CURSOR fields.

Objects
Demonstrates working with Oracle object fields. This sample shows how to clone objects and access and modify object field properties.

Pipes
Uses TOraAlerter in Pipe mode to organize cross-session message exchanging.

PLSQLTable
Demonstrates using PL/SQL Table types as parameters. Working with PL/SQL Table types in PL/SQL lets you imitate array functionality provided by other programming languages.

Progressor
Uses TOraAlerter in Pipe mode to indicate the progress of long-duration Oracle processes.

ProxySession
Demonstrates connecting to Oracle with the Oracle Proxy session functionality. This type of connection allows to quickly establish connections without specifying a password. Note: Requires the CONNECT THROUGH privilege

Refls
Demonstrates using the REF Oracle data type in queries.

SmartRefresh
Uses the Smart Refresh features of TSmartQuery to notify your data changes to other subscribed users. This sample is based on the TOraAlerter component. Note: Smart Refresh is only available in ODAC Professional Edition and ODAC Developer Edition.

XMLType
Uses the TOraXMLField class to work with Oracle SYS.XMLTYPE type LOB or Schema-based data. This sample project shows how to perform all the basic operations with this data type.

Supplementary Demo Projects

ODAC also includes a number of additional demo projects that describe some special use cases, show how to use ODAC in different IDEs and give examples of integrating it with third-party components. These supplementary ODAC demo projects are sorted into subfolders in the %Odac%\Demos\ directory.

<table>
<thead>
<tr>
<th>Location</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThirdParty</td>
<td>FastReport</td>
<td>Demonstrates how ODAC can be used with FastReport components. This project consists of two parts. The first part consists of several packages that integrate ODAC components into the FastReport editor. The second part is a demo application that lets you design and preview reports with ODAC technology in the FastReport editor.</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>InfoPower</strong></td>
<td>Uses InfoPower components to display recordsets retrieved with ODAC. This demo project displays an InfoPower grid component and fills it with the result of an ODAC query. Shows how to link ODAC data sources to InfoPower components.</td>
<td></td>
</tr>
<tr>
<td><strong>IntraWeb</strong></td>
<td>A collection of sample projects that show how to use ODAC components as data sources for IntraWeb applications. Contains IntraWeb samples for setting up a connection, querying a database and modifying data and working with <code>CachedUpdates</code> and <code>MasterDetail</code> relationships. Starting with Oracle 10.2g and higher lets you see the effect of setting <code>TOraCachedUpdates</code>.</td>
<td></td>
</tr>
<tr>
<td><strong>QuickReport</strong></td>
<td>Lets you launch and view a QuickReport application based on ODAC. This demo project lets you modify the application in design-time.</td>
<td></td>
</tr>
<tr>
<td><strong>ReportBuilder</strong></td>
<td>Uses ODAC data sources to create a ReportBuilder report that takes data from an Oracle database. Shows how to set up a ReportBuilder document in design-time and how to integrate ODAC components into the Report Builder editor to perform document design in run-time.</td>
<td></td>
</tr>
<tr>
<td><strong>CBuilder</strong></td>
<td>General demo project that shows how to create ODAC-based applications with C++Builder. Lets you execute SQL scripts and work with result sets in a grid. This is one of the two ODAC demos for C++Builder.</td>
<td></td>
</tr>
<tr>
<td><strong>DII</strong></td>
<td>Demonstrates creating and loading DLLs for ODAC-based projects. This demo project consists of two parts - an OraDII project that creates...</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>a DLL of a form that sends a query to the server and displays its results, and an OraExe project that can be executed to display a form for loading and running this DLL. Allows you to build a dll for one ODAC-based project and load and test it from a separate application.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uses external procedures to save LOB data to file on an Oracle server and store the file name and file date in a database. This demo project uses the external procedure DLL file described in the Writing Oracle external procedures with ODAC topic.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demonstrates the recommended approach to working with unstable networks. This sample lets you perform transactions and updates in several different modes, simulate sudden session termination, and view what happens to your data state when connections to the server are unexpectedly lost. Shows off CachedUpdates, LocalMasterDetail, FetchAll, Pooling, and different Failover modes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demonstrates using the SDO_GEOMETRY Oracle type with ODAC. This project reads SDO_GEOMETRY objects from the database and draws figures stored in these objects. The project allows you to edit figures and save them to the database.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demonstrates using MIDAS technology with ODAC. This project consists of two parts: a MIDAS server that processes requests to the database and a thin MIDAS client that displays an interactive grid. This demo shows how to build thin clients that display interactive components and delegate all</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Performance

Measures ODAC performance on several types of queries. This project lets you compare ODAC performance to BDE, ADO, and dbExpress. Tests the following functionality: Fetch, Master/Detail, Stored Procedure Call, Data Loading, Multi Executing, and Insert/Post.

### VirtualTableCB

Demonstrates working with the TVirtualTable component. This sample shows how to fill virtual dataset with data from other datasets, filter data by a given criteria, locate specified records, perform file operations, and change data and table structure. This is one of the two demo projects for C++Builder.

<table>
<thead>
<tr>
<th>OdacDemo</th>
<th>OdacDemo</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Win32 version of the main ODAC demo project - see above]</td>
<td></td>
</tr>
</tbody>
</table>

---

## 3.11 Deployment

ODAC applications can be built and deployed with or without run-time libraries. Using run-time libraries is managed with the "Build with runtime packages" check box in the Project Options dialog box.

### Deploying Windows applications built without run-time packages

You do not need to deploy any files with ODAC-based applications built without run-time packages, provided you are using a registered version of ODAC.

You can check your application does not require run-time packages by making sure the "Build with runtime packages" check box is not selected in the Project Options dialog box.

**Trial Limitation Warning**
If you are evaluating deploying Windows applications with ODAC Trial Edition, you will need to deploy the following DAC BPL files:

<table>
<thead>
<tr>
<th>File</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dacXX.bpl</td>
<td>always</td>
</tr>
<tr>
<td>odacXX.bpl</td>
<td>always</td>
</tr>
</tbody>
</table>

and their dependencies (required IDE BPL files) with your application, even if it is built without run-time packages:

<table>
<thead>
<tr>
<th>File</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtlXX.bpl</td>
<td>always</td>
</tr>
<tr>
<td>dbrtlXX.bpl</td>
<td>always</td>
</tr>
<tr>
<td>vcldbXXX.bpl</td>
<td>always</td>
</tr>
</tbody>
</table>

### Deploying Windows applications built with run-time packages

You can set your application to be built with run-time packages by selecting the "Build with runtime packages" check box in the Project Options dialog box before compiling your application.

In this case, you will also need to deploy the following BPL files with your Windows application:

<table>
<thead>
<tr>
<th>File</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>dacXX.bpl</td>
<td>always</td>
</tr>
<tr>
<td>odacXX.bpl</td>
<td>always</td>
</tr>
<tr>
<td>dacvclXX.bpl</td>
<td>always</td>
</tr>
<tr>
<td>odacvclXX.bpl</td>
<td>if your application uses the OdacVcl unit</td>
</tr>
<tr>
<td>crcontrolsXX.bpl</td>
<td>if your application uses the CRDBGrid component</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

### 4 Using ODAC

This section describes basics of using Oracle Data Access Components

- Updating Data with ODAC Dataset Components
- Connecting in Direct Mode
- Master/Detail Relationships
- Data Type Mapping
- Data Encryption
4.1 Updating Data with ODAC Dataset Components

ODAC dataset components which descend from TCustomDADataset provide different ways for reflecting local changes on the server.
The first approach is to use automatic generation of update SQL statements. When using this approach you should either specify Key Fields (the KeyFields property) or include RowID field into you SELECT SQL statement to avoid requesting of KeyFields from the server. When SELECT statement uses multiple tables, you can use UpdatingTable property to specify which table will be updated. If UpdatingTable is blank, the first table of the FROM clause will be used. When using sophisticated SELECT SQL statements (statements that use multiple tables, Synonyms, DBLinks, aggregated fields) we recommend to enable ExtendedFieldsInfo option. When this option is enabled, additional requests to the server may be performed to obtain more information about updating objects. This helps to generate correct updating SQL statements but may result in performance decrease. To avoid editing the fields that will not be used in update SQL statements use the SetFieldsReadOnly option. You can increase performance by refreshing fields using RETURNING clause when insert or update is performed. To enable this feature enable DMLRefresh and ReturnParams options.

Another approach is to set update SQL statements using SQLInsert, SQLUpdate and SQLDelete properties. Use them to specify SQL statements that will be used for corresponding data modifications. It is useful when generating data modification statements is not possible (for example when working with data of a cursor, returned by a stored procedure) or you need to execute some specific statements. You may also assign TOraUpdateSQL component to the UpdateObject property. TOraUpdateSQL component holds all updating SQL statements in one place. You can generate all these SQL statements using ODAC design time editors. For more careful customization of data update operations you can use InsertObject, ModifyObject and DeleteObject properties of TOraUpdateSQL component.

See Also
- TSmartQuery
- TOraQuery
- TOraStoredProc
- TOraTable
- TOraUpdateSQL

4.2 Connecting in Direct Mode

ODAC Professional Edition allows you to connect to Oracle in two ways: in the Client mode, using Oracle Client software, or in the Direct mode, over TCP/IP. The Direct mode can be enabled using the TOraSession.Options.Direct property.
ODAC Connection Modes

By default, ODAC, like most applications that work with Oracle, uses the Oracle Call Interface (OCI) to connect to the Oracle database server. This is referred to as connecting in the Client mode, and is the usual way to develop Oracle applications with a third-generation language. All OCI routines are stored in external libraries, so the executables for applications that work through OCI are small. However, working through OCI requires Oracle client software to be installed on client machines. It is rather inconvenient and causes additional installation and administration expenses. Furthermore, there are some situations where the installation of Oracle client is not advisable or may be even impossible—for example, if you deploy an application to remote machines that are not overseen by a proficient system administrator.

To overcome these challenges, ODAC Professional Edition includes an option to connect to Oracle directly over the network using the TCP/IP protocol. This is referred to as connecting in the Direct mode. Connecting in the Direct mode does not require Oracle client software to be installed on client machines. The only requirement for running an application that uses ODAC in the Direct mode, is that the operating system must support the TCP/IP protocol.

Connecting in Direct Mode

To connect to Oracle server in the Direct mode, set up your TOraSession instance as follows:

- set the Options.Direct property of your TOraSession instance to True;
- set the Server property of your TOraSession instance, to a string that contains the host address of the database server, port number, and Oracle Service Name or Oracle System Identifier (SID) in the following format:

  - if you connect to Oracle using Service Name:
    
    Host:Port/ServiceName
    Host:Port:sn=ServiceName (deprecated format)
  
  - if you connect to Oracle using SID:
    
    Host:Port:SID
    Host:Port:sid=SID (deprecated format)

  **Host** is the server's IP address or DNS name.

  **Port** is the port number that the server listens to.

  **SID** is a system identifier that specifies the name of an Oracle database instance.

  **ServiceName** is a system alias for an Oracle database instance (or multiple instances).
Note that the syntax used to set up the `Server` property in the Direct mode is different from the Client mode. In the Client mode, this property must be set to the TNS name of the Oracle server.

Note that if the port number is followed by a colon, and the service name prefix (`sn=`) or the SID prefix (`sid=`) is not defined, then by default, the connection will be established using SID.

An example below illustrates the connection to Oracle in the Direct mode. The IP address of the Oracle server is 205.227.44.44, the port number is 1521 (the most commonly used port for Oracle), and the SID is `orcl` (standard Oracle SID):

```pascal
var
  Session: TOraSession;

Session.Options.Direct := True;
Session.Username := 'Scott';
Session.Password := 'tiger';
Session.Server := '205.227.44.44:1521:orcl';
Session.Connect;
```

connecting to Oracle with Service Name:

```pascal
...
Session.Server := '205.227.44.44:1521/orcl';
...
or...
Session.Server := '205.227.44.44:1521:sid=orcl';
...
```

connecting to Oracle with SID:

```pascal
...
Session.Server := '205.227.44.44:1521:orcl';
...
or...
Session.Server := '205.227.44.44:1521:sn=orcl';
...
```

This is all you need to do to enable the Direct mode in your application. You do not have to rewrite other parts of your code.

To return to the OCI mode, set `TOraSession.Options.Direct` to `False` and `Session.Server` to the TNS name of your server.

You can connect to Multi-Threaded Server using the Direct mode. The server must be configured to use a specific port and the TTC protocol. This can help you avoid firewall conflicts.

**Note:** The Direct mode is available in ODAC Professional Edition and Oracle Trial Edition. An
Client Mode vs Direct Mode

Applications that use the Client mode and those that use the Direct mode have similar performance and file size. In terms of security, using the Direct mode is the same as using Oracle Client without Oracle Advanced Security. In the Direct mode, ODAC uses DES authentication and does not support Oracle Advanced Security.

Advantages of the Direct mode:
- No need to install and administer Oracle client.
- Reduced system requirements.

Limitations of the Direct mode:
- Only TCP/IP connections supported.
- TOracleLoader direct loading is not supported.
- Some issues may occur when using firewalls.
- NLS conversion is not supported on the client side.
- Transparent Application Failover is not supported.
- Change notifications (TOracleChangeNotification) are not supported.
- OS Authentication supported for Windows only.

A connection in the Direct mode is managed transparently by an instance of TOracleSession, and you can easily switch back to OCI in the Client mode at any time if the above limitations become critical to you.

See Also
- TOracleSession.Server
- TOracleSession.Options
- MDevart.ODac.TOracleSQL.BreakExec()
- TCustomDADDataSet.BreakExec
4.3 Master/Detail Relationships

Master/detail (MD) relationship between two tables is a very widespread one. So it is very important to provide an easy way for database application developer to work with it. Let's examine how ODAC implements this feature.

Suppose we have classic MD relationship between "Department" and "Employee" tables.

"Department" table has field Dept_No. Dept_No is a primary key.

"Employee" table has a primary key EmpNo and foreign key Dept_No that binds "Employee" to "Department".

It is necessary to display and edit these tables.

ODAC provides two ways to bind tables. First code example shows how to bind two TCustomOraDataSet components (TOraQuery, TSmartQuery, TOraTable or even TOraStoredProc) into MD relationship via parameters.

```delphi
procedure TForm1.Form1Create(Sender: TObject);
var
  Master, Detail: TOraQuery;
  MasterSource: TDataSource;
begin
  // create master dataset
  Master := TOraQuery.Create(Self);
  Master.SQL.Text := 'SELECT * FROM Department';
  // create detail dataset
  Detail := TOraQuery.Create(Self);
  Detail.SQL.Text := 'SELECT * FROM Employee WHERE Dept_No = :Dept_No';
  // connect detail dataset with master via TDataSource component
  MasterSource := TDataSource.Create(Self);
  MasterSource.DataSet := Master;
  Detail.MasterSource := MasterSource;
  // open master dataset and only then detail dataset
  Master.Open;
  Detail.Open;
end;
```

Pay attention to one thing: parameter name in detail dataset SQL must be equal to the field name or the alias in the master dataset that is used as foreign key for detail table. After opening detail dataset always holds records with Dept_No field value equal to the one in the current master dataset record.

There is an additional feature: when inserting new records to detail dataset it automatically fills foreign key fields with values taken from master dataset.

Now suppose that detail table "Department" foreign key field is named DepLink but not Dept_No. In such case detail dataset described in above code example will not autofill
DepLink field with current "Department".Dept_No value on insert. This issue is solved in second code example.

```delphi
procedure TForm1.Form1Create(Sender: TObject);
var
  Master, Detail: TOraQuery;
  MasterSource: TDataSource;
begin
  // create master dataset
  Master := TOraQuery.Create(Self);
  Master.SQL.Text := 'SELECT * FROM Department';
  // create detail dataset
  Detail := TOraQuery.Create(Self);
  Detail.SQL.Text := 'SELECT * FROM Employee';
  // setup MD
  Detail.MasterFields := 'Dept_No'; // primary key in Department
  Detail.DetailFields := 'DepLink'; // foreign key in Employee
  // connect detail dataset with master via TDataSource component
  MasterSource := TDataSource.Create(Self);
  MasterSource.DataSet := Master;
  MasterSource := MasterSource;
  // open master dataset and only then detail dataset
  Master.Open;
  Detail.Open;
end;
```

In this code example MD relationship is set up using `MasterFields` and `DetailFields` properties. Also note that there are no WHERE clause in detail dataset SQL.

To defer refreshing of detail dataset while master dataset navigation you can use `DetailDelay` option.

Such MD relationship can be local and remote, depending on the `TCustomDADataset.Options.LocalMasterDetail` option. If this option is set to True, dataset uses local filtering for establishing master-detail relationship and does not refer to the server. Otherwise detail dataset performs query each time when record is selected in master dataset. Using local MD relationship can reduce server calls number and save server resources. It can be useful for slow connection. `CachedUpdates` mode can be used for detail dataset only for local MD relationship. Using local MD relationship is not recommended when detail table contains too many rows, because in remote MD relationship only records that correspond to the current record in master dataset are fetched. So, this can decrease network traffic in some cases.

See Also
- `TCustomDADataset.Options`
- `TMemDataSet.CachedUpdates`
4.4 Data Type Mapping

Overview

Data Type Mapping is a flexible and easily customizable gear, which allows mapping between DB types and Delphi field types.

In this article there are several examples, which can be used when working with all supported DBs. In order to clearly display the universality of the Data Type Mapping gear, a separate DB will be used for each example.

Data Type Mapping Rules

In versions where Data Type Mapping was not supported, ODAC automatically set correspondence between the DB data types and Delphi field types. In versions with Data Type Mapping support the correspondence between the DB data types and Delphi field types can be set manually.

Here is the example with the numeric type in the following table of a Oracle database:

```
CREATE TABLE NUMBER_TYPES
(
    ID NUMBER NOT NULL,
    VALUE1 NUMBER(4,0),
    VALUE2 NUMBER(10,0),
    VALUE3 NUMBER(15,0),
    VALUE4 NUMBER(5,2),
    VALUE5 NUMBER(10,4),
    VALUE6 NUMBER(15,6),
    CONSTRAINT PK_NUMBER_TYPES PRIMARY KEY (ID)
)
```

And Data Type Mapping should be used so that:

- the numeric fields with Scale=0 in Delphi would be mapped to one of the field types: TSmallintField, TIntegerField or TlargeintField, depending on Precision
- to save precision, the numeric fields with Precision>=10 and Scale<=4 would be mapped to TBCTDField
- and the numeric fields with Scale>=5 would be mapped to TFMFTBCDField.

The above in the form of a table:

<table>
<thead>
<tr>
<th>Oracle data type</th>
<th>Default Delphi field type</th>
<th>Destination Delphi field</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To specify that numeric fields with Precision <= 4 and Scale = 0 must be mapped to ftSmallint, such a rule should be set:

```pascal
var
  DBType: Word;
  MinPrecision: Integer;
  MaxPrecision: Integer;
  MinScale: Integer;
  MaxScale: Integer;
  FieldType: TFieldType;
begin
  DBType := oraNumber;
  MinPrecision := 0;
  MaxPrecision := 4;
  MinScale := 0;
  MaxScale := 0;
  FieldType := ftSmallint;
  OraSession.DataTypeMap.AddDBTypeRule(DBType, MinPrecision, MaxPrecision, MinScale, MaxScale, FieldType);
end;
```

This is an example of the detailed rule setting, and it is made for maximum visualization. Usually, rules are set much shorter, e.g. as follows:

```pascal
OraSession.DataTypeMap.Clear;
// rule for numeric(4,0)
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, 4, 0, 0, ftSmallint);  // rule for numeric(10,0)
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 5, 10, 0, 0, ftInteger);    // rule for numeric(15,0)
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 11, rlAny, 0, 0, ftLargeint); // rule for numeric(5,2)
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, 9, 1, rlAny, ftFloat);    // rule for numeric(10,4)
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 10, rlAny, 1, rlAny, ftBCD);  // rule for numeric(15,6)
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 10, rlAny, 5, rlAny, ftFMTBCD);
```

**Rules order**

When setting rules, there can occur a situation when two or more rules that contradict each other are set for one type in the database. In this case, only one rule will be applied —
the one, which was set first.

For example, there is a table in an Oracle database:

```sql
CREATE TABLE PERSON
(
    ID                NUMBER        NOT NULL,
    FIRSTNAME         VARCHAR2(20)          ,
    LASTNAME         VARCHAR2(30)          ,
    GENDER_CODE       VARCHAR2(1)           ,
    BIRTH_DTTM        DATE                  ,
    CONSTRAINT PK_PERSON PRIMARY KEY (ID)
)
```

TBCDField should be used for `NUMBER(10,4)`, and TFMTBCDField - for `NUMBER(15,6)` instead of default fields:

<table>
<thead>
<tr>
<th>Oracle data type</th>
<th>Default Delphi field type</th>
<th>Destination field type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER(5,2)</td>
<td>ftFloat</td>
<td>ftFloat</td>
</tr>
<tr>
<td>NUMBER(10,4)</td>
<td>ftFloat</td>
<td>ftBCD</td>
</tr>
<tr>
<td>NUMBER(15,6)</td>
<td>ftFloat</td>
<td>ftFMTBCD</td>
</tr>
</tbody>
</table>

If rules are set in the following way:

```delphi
OraSession.DataTypeMap.Clear;
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, 9, rlAny, rlAny, ..., 0, rlAny, 0, rlAny, ftFloat);
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, rlAny, 0, rlAny, ftBCD);
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, rlAny, 0, rlAny, ftFMTBCD);
```

it will lead to the following result:

<table>
<thead>
<tr>
<th>Oracle data type</th>
<th>Delphi field type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER(5,2)</td>
<td>ftFloat</td>
</tr>
<tr>
<td>NUMBER(10,4)</td>
<td>ftBCD</td>
</tr>
<tr>
<td>NUMBER(15,6)</td>
<td>ftFMTBCD</td>
</tr>
</tbody>
</table>

But if rules are set in the following way:

```delphi
OraSession.DataTypeMap.Clear;
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, rlAny, 0, rlAny, ftFMTBCD);
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, rlAny, 0, rlAny, ftBCD);
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, 9, rlAny, rlAny, ftFloat);
```

it will lead to the following result:

<table>
<thead>
<tr>
<th>Oracle data type</th>
<th>Delphi field type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMBER(5,2)</td>
<td>ftFMTBCD</td>
</tr>
</tbody>
</table>
This happens because the rule

```c
OraSession.DataTypeMap.AddDBTypeRule(oraNumber, 0, rlAny, 0, rlAny, ftFMTBCD);
```

will be applied for the NUMBER fields, whose Precision is from 0 to infinity, and Scale is from 0 to infinity too. This condition is met by all NUMBER fields with any Precision and Scale.

When using Data Type Mapping, first matching rule is searched for each type, and it is used for mapping. In the second example, the first set rule appears to be the first matching rule for all three types, and therefore the ftFMTBCD type will be used for all fields in Delphi.

If to go back to the first example, the first matching rule for the NUMBER(5,2) type is the first rule, for NUMBER(10,4) - the second rule, and for NUMBER(15,6) - the third rule. So in the first example, the expected result was obtained.

So it should be remembered that if rules for Data Type Mapping are set so that two or more rules that contradict to each other are set for one type in the database, the rules will be applied in the specified order.

**Defining rules for Connection and Dataset**

Data Type Mapping allows setting rules for the whole connection as well as for each DataSet in the application.

For example, such table is created in Oracle:

```sql
CREATE TABLE PERSON
(
    ID                NUMBER              NOT NULL  ,
    FIRSTNAME         VARCHAR2(20)      ,
    LASTNAME          VARCHAR2              ,
    GENDER_CODE       VARCHAR2(1)       ,
    BIRTH_DTTM        DATE         ,
    CONSTRAINT PK_PERSON PRIMARY KEY (ID)
)
```

It is exactly known that the birth_dttm field contains birth day, and this field should be ftDate in Delphi, and not ftDateTime. If such rule is set:
OraSession.DataTypeMap.Clear;
OraSession.DataTypeMap.AddDBTypeRule(OraDate, ftDate);

all DATETIME fields in Delphi will have the ftDate type, that is incorrect. The ftDate type was expected to be used for the DATETIME type only when working with the person table. In this case, Data Type Mapping should be set not for the whole connection, but for a particular DataSet:

OraQuery.DataTypeMap.Clear;
OraQuery.DataTypeMap.AddDBTypeRule(OraDate, ftDate);

Or the opposite case. For example, DATETIME is used in the application only for date storage, and only one table stores both date and time. In this case, the following rules setting will be correct:

OraSession.DataTypeMap.Clear;
OraSession.DataTypeMap.AddDBTypeRule(OraDate, ftDate);
OraQuery.DataTypeMap.Clear;
OraQuery.DataTypeMap.AddDBTypeRule(OraDate, ftDateTime);

In this case, in all DataSets for the DATETIME type fields with the ftDate type will be created, and for OraQuery - with the ftDateTime type.

The point is that the priority of the rules set for the DataSet is higher than the priority of the rules set for the whole connection. This allows both flexible and convenient setting of Data Type Mapping for the whole application. There is no need to set the same rules for each DataSet, all the general rules can be set once for the whole connection. And if a DataSet with an individual Data Type Mapping is necessary, individual rules can be set for it.

Rules for a particular field

Sometimes there is a need to set a rule not for the whole connection, and not for the whole dataset, but only for a particular field.

e.g. there is such table in a Oracle database:

```sql
CREATE TABLE ITEM
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(50) NOT NULL,
  GUID VARCHAR2(38),
  CONSTRAINT PK_ITEM PRIMARY KEY (ID)
)
```

The guid field contains a unique identifier. For convenient work, this identifier is expected to be mapped to the TGuidField type in Delphi. But there is one problem, if to set the rule like
this:

```csharp
OraQuery.DataTypeMap.Clear;
OraQuery.DataTypeMap.AddDBTypeRule(oraVarchar2, ftGuid);
```

then both **name** and **guid** fields will have the ftGuid type in Delphi, that does not correspond to what was planned. In this case, the only way is to use Data Type Mapping for a particular field:

```csharp
OraQuery.DataTypeMap.AddFieldNameRule('GUID', ftGuid);
```

In addition, it is important to remember that setting rules for particular fields has the highest priority. If to set some rule for a particular field, all other rules in the Connection or DataSet will be ignored for this field.

### Ignoring conversion errors

Data Type Mapping allows mapping various types, and sometimes there can occur the problem with that the data stored in a DB cannot be converted to the correct data of the Delphi field type specified in rules of Data Type Mapping or vice-versa. In this case, an error will occur, which will inform that the data cannot be mapped to the specified type.

For example:

<table>
<thead>
<tr>
<th>Database value</th>
<th>Destination field type</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>'text value'</td>
<td>ftInteger</td>
<td>String cannot be converted to Integer</td>
</tr>
<tr>
<td>1000000</td>
<td>ftSmallint</td>
<td>Value is out of range</td>
</tr>
<tr>
<td>15,1</td>
<td>ftInteger</td>
<td>Cannot convert float to integer</td>
</tr>
</tbody>
</table>

But when setting rules for Data Type Mapping, there is a possibility to ignore data conversion errors:

```csharp
OraSession.DataTypeMap.AddDBTypeRule(oraVarchar2, ftInteger, True);
```

In this case, the correct conversion is impossible. But because of ignoring data conversion errors, Data Type Mapping tries to return values that can be set to the Delphi fields or DB fields depending on the direction of conversion.

<table>
<thead>
<tr>
<th>Database value</th>
<th>Destination field type</th>
<th>Result</th>
<th>Result description</th>
</tr>
</thead>
<tbody>
<tr>
<td>'text value'</td>
<td>ftInteger</td>
<td>0</td>
<td>0 will be returned if</td>
</tr>
</tbody>
</table>
4.5 Data Encryption

ODAC has built-in algorithms for data encryption and decryption. To enable encryption, you should attach the `TCREncryptor` component to the dataset, and specify the encrypted fields. When inserting or updating data in the table, information will be encrypted on the client side in accordance with the specified method. Also when reading data from the server, the components decrypt the data in these fields "on the fly".

For encryption, you should specify the data encryption algorithm (the `EncryptionAlgorithm` property) and password (the `Password` property). On the basis of the specified password, the key is generated, which encrypts the data. There is also a possibility to set the key directly using the `SetKey` method.

When storing the encrypted data, in addition to the initial data, you can also store additional information: the GUID and the hash. (The method is specified in the `TCREncryptor.DataHeader` property).

If data is stored without additional information, it is impossible to determine whether the data is encrypted or not. In this case, only the encrypted data should be stored in the column, otherwise, there will be confusion because of the inability to distinguish the nature of the data. Also in this way, the similar source data will be equivalent in the encrypted form, that is not good from the point of view of the information protection. The advantage of this method is the size of the initial data equal to the size of the encrypted data.

To avoid these problems, it is recommended to store, along with the data, the appropriate GUID, which is necessary for specifying that the value in the record is encrypted and it must be decrypted when reading data. This allows you to avoid confusion and keep in the same
column both the encrypted and decrypted data, which is particularly important when using an existing table. Also, when doing in this way, a random initializing vector is generated before the data encryption, which is used for encryption. This allows you to receive different results for the same initial data, which significantly increases security.

The most preferable way is to store the hash data along with the GUID and encrypted information to determine the validity of the data and verify its integrity. In this way, if there was an attempt to falsify the data at any stage of the transmission or data storage, when decrypting the data, there will be a corresponding error generated. For calculating the hash the SHA1 or MD5 algorithms can be used (the HashAlgorithm property).

The disadvantage of the latter two methods - additional memory is required for storage of the auxiliary information.

As the encryption algorithms work with a certain size of the buffer, and when storing the additional information it is necessary to use additional memory, TCREncryptor supports encryption of string or binary fields only (ftString, ftWideString, ftBytes, ftVarBytes, ftBlob, ftMemo, ftWideMemo). If encryption of string fields is used, firstly, the data is encrypted, and then the obtained binary data is converted into hexadecimal format. In this case, data storage requires two times more space (one byte = 2 characters in hexadecimal).

Therefore, to have the possibility to encrypt other data types (such as date, number, etc.), it is necessary to create a field of the binary or BLOB type in the table, and then convert it into the desired type on the client side with the help of data mapping.

It should be noted that the search and sorting by encrypted fields become impossible on the server side. Data search for these fields can be performed only on the client after decryption of data using the Locate and LocateEx methods. Sorting is performed by setting the TMemDataSet.IndexFieldNames property.

**Example.**

Let's say there is an employee list of an enterprise stored in the table with the following data: full name, date of employment, salary, and photo. We want all these data to be stored in the encrypted form. Write a script for creating the table:

```sql
CREATE TABLE EMP (
    EMPNO NUMBER,
    ENAME VARCHAR2(2000),
    HIREDATE VARCHAR2(200),
    SAL VARCHAR2(200),
    FOTO BLOB,
    CONSTRAINT PK_EMP PRIMARY KEY (EMPNO));
```
As we can see, the fields for storage of the textual information, date, and floating-point number are created with the VARCHAR2 type. This is for the ability to store encrypted information, and in the case of the text field - to improve performance. Write the code to process this information on the client.

OraQuery.SQL.Text := 'SELECT * FROM EMP';
OraQuery.Encryption.Encryptor := OraEncryptor;
OraQuery.Encryption.Fields := 'ENAME, HIREDATE, SAL, FOTO';
OraEncryptor.Password := '111111';
OraQuery.DataTypeMap.AddFieldNameRule ('ENAME', ftString);
OraQuery.DataTypeMap.AddFieldNameRule ('HIREDATE', ftDateTime);
OraQuery.DataTypeMap.AddFieldNameRule ('SAL', ftFloat);
OraQuery.Open;

© 1997-2021
Devart. All Rights Reserved.

4.6 Working in an Unstable Network

The following settings are recommended for working in an unstable network:

TCustomDAConnection.Options.LocalFailover = True
TCustomDAConnection.Options.DisconnectedMode = True
TDataSet.CachedUpdates = True
TCustomDADataSet.FetchAll = True
TCustomDADataSet.Options.LocalMasterDetail = True
AutoCommit = True

These settings minimize the number of requests to the server. Using
TCustomDAConnection.Options.DisconnectedMode allows DataSet to work without an active connection. It minimizes server resource usage and reduces connection break probability. I.e. in this mode connection automatically closes if it is not required any more. But every explicit operation must be finished explicitly. That means each explicit connect must be followed by explicit disconnect. Read Working with Disconnected Mode topic for more information.

Setting the FetchAll property to True allows to fetch all data after cursor opening and to close connection. If you are using master/detail relationship, we recommend to set the LocalMasterDetail option to True.

It is not recommended to prepare queries explicitly. Use the CachedUpdates mode for DataSet data editing. Use the TCustomDADataSet.Options.UpdateBatchSize property to reduce the number of requests to the server.

If a connection breaks, a fatal error occurs, and the OnConnectionLost event will be raised if the following conditions are fulfilled:
• There are no active transactions;
• There are no opened and not fetched datasets;
• There are no explicitly prepared datasets or SQLs.

If the user does not refuse suggested RetryMode parameter value (or does not use the OnConnectionLost event handler), ODAC can implicitly perform the following operations:

```csharp
Connect;
DataSet.ApplyUpdates;
DataSet.Open;
```

I.e. when the connection breaks, implicit reconnect is performed and the corresponding operation is reexecuted. We recommend to wrap other operations in transactions and fulfill their reexecuting yourself.

The using of Pooling in Disconnected Mode allows to speed up most of the operations because of connecting duration reducing.

See Also
• FailOver demo
• Working with Disconnected Mode
• TCustomDAConnection.Options
• TCustomDAConnection.Pooling

4.7 Secure Connections

This section describes how to establish a secure connection between a client application and Oracle Database.
• Connecting via SSL
• Connecting via SSH
• HTTP/HTTPS Network Tunneling

4.7.1 Connecting via SSL

Connecting to Oracle Database Using SSL
This section discusses how to connect a client application to Oracle Database using SSL (Secure Sockets Layer), which is an industry standard protocol for secure access to a remote machine over untrusted networks. It runs on top of TCP/IP to secure client-server communications by allowing an SSL-enabled client to authenticate itself to an SSL-enabled server and vice versa. During server authentication, an SSL-enabled client application uses standard techniques of public-key cryptography to verify the server's identity by checking that the server's certificate is issued by a trusted certificate authority (CA) and proves the ownership of the public key.

Conversely, SSL client authentication allows the server to validate the client's identity. The client and server can also authenticate each other using self-signed certificates, however, you will almost never want to use a self-signed certificate, except for an Intranet or a development server. After establishing an SSL connection, the client and server can exchange messages that are symmetrically encrypted with the shared secret key. SSL is the recommended method to establish a secure connection to Oracle due to easier configuration and higher performance, compared to SSH.

To establish an SSL connection to the server with ODAC, you must compile and install the TCRSSLIOHandler component, which is distributed with SecureBridge and is required to bind ODAC with SecureBridge. The installation instructions for the component are provided in the Readme.html file, which is located by default in "My Documents\Devart\ODAC for RAD Studio XX\Demos\TechnologySpecific\SecureBridge".

Connecting to Oracle Database Using Oracle Wallet
1. Place the following components on the form: TOraSession, TOraQuery, TOraDataSource, TDBGGrid, TButton, TCRSSLIOHandler, TCRSsoFileStorage.

2. Select the TOraSession component and set the IOHandler property to an instance of
3. Select the TDBGrid component and set the DataSource property to an instance of TOraDataSource.

4. Select the TOraDataSource component and set the DataSet property to an instance of TOraQuery.

5. Select the TOraQuery component and set the Session property to an instance of TOraSession.

6. Double-click the TOraQuery component and specify a SQL query to execute against Oracle Database.

7. Select the TCRSSoFileStorage component and specify the path to the wallet file. A wallet is a container for storing authentication and signing credentials, including keys and certificates needed by SSL. See this document for information on creating an Oracle wallet. If you are using Oracle Cloud, see this document for information on obtaining wallet files.

8. Select the TCRSSoFileStorage component and set the Storage property to an instance of TCRSSoFileStorage.

9. Select the TButton component and create an OnClick event. Add the code to call the Open method of TOraQuery when the button is clicked.

10. Select the TOraSession component and specify the server’s distinguished name (DN) in the ServerCertDN property of SSLOptions to enable server DN matching. It is used to check whether the server is genuine by matching the server’s global database name against the DN from the server certificate. See this document for information on editing the client network configuration files.
11. Double-click the ToraSession component and specify the server address, port, username and password.

12. Compile and run the application.
Connecting to Oracle Database Using SSL Certificates and Keys

The steps are similar to the above, except that you specify the server and client SSL certificates and the private client key instead of wallet files, thus you do not need the TCRSsoFileStorage component.

Select the TOraSession component and expand SSLOptions. Specify the server certificate in the CACert property, the client certificate in the Cert property, the private client key in the Key property and the server's distinguished name (DN) in the ServerCertDN property.

Connecting to Oracle Database Using the OpenSSL Library

Another way to embed SSL client functionality into your Delphi app, which uses ODAC components to access Oracle Database, is by using the OpenSSL library that implements the SSL protocol and enables servers to securely communicate with their clients. The description of the SSL connection features without using the SecureBridge's IOHandler:

The following options must be set for an SSL connection:
• SSLCACert — the server CA certificate;
• SSLCert — the client certificate;
• SSLKey — the private client key;
• SSLCipherList — the cipher suite;
• ServerCertDN — the server's distinguished name (DN).

Note: The ssleay32.dll and libeay32.dll files are required to use the SSL protocol with the OpenSSL library.

4.7.2 Connecting via SSH

Connecting to Oracle Database Through SSH

This section dicusses how to connect a client application to Oracle Database through SSH. SSH is a network protocol for secure remote login to another system over the Internet by connecting the SSH client to the SSH server. SSH provides a mechanism for establishing a secure connection between the client and the remote server, which authenticate each other and exchange messages. It employs different forms of symmetrical encryption, asymmetrical encryption, and hashing. The SSH client initiates a connection and uses public key cryptography to verify the identity of the SSH server.

It is possible to use SSH an an encryption method to secure the connection between a Delphi application and an Oracle server. You can embed the SSH client functionality into your application and install the SSH server on a remote machine where your Oracle server resides. The SSH client connects to the SSH server, which sends all commands to Oracle Database.

SSH key-based authentication is done by public and private keys that a client uses to authenticate itself when logging into an SSH server. The server key is used is used by the client to authenticate the SSH server and is specified in the TScSSHClient.HostKeyName property. The client key is used by the SSH server to authenticate the client and is specified in the TScSSHClient.PrivateKeyName property. Note that the private key contains the public key. See SecureBridge tutorial on configuring the SSH server.

The SSH server is required to replicate the steps in this tutorial and encrypt the network connection between the client application and Oracle Database. You can build the SSH server demo project, which is distributed with SecureBridge ("Documents\Devart\ODAC for RAD Studio\Demos\TechnologySpecific\SecureBridge\Demo\SSH") and run the executable
To establish a connection to a remote SSH server, you must compile and install the TCRSSHIOHandler component, which is distributed with SecureBridge and is required to bind ODAC with SecureBridge. The installation instructions for the component are provided in the readme.html file, which is located by default in "My Documents\Devart\ODAC for RAD Studio XX\Demos\TechnologySpecific\SecureBridge".

Sample Application That Connects to Oracle Database Through SSH

1. Place the following components on the form: TOraSession, TOraQuery, TOraDataSource, TDBGGrid, TButton, TCRSSHIOHandler, TScSSHClient, and TScFileStorage.

2. Select the TDBGGrid component and set the DataSource property to an instance TOraDataSource.

3. Select the TOraDataSource component and set the DataSet property to an instance of TOraQuery.

4. Select the TOraQuery component and set the Session property to an instance of TOraSession. Double-click TOraQuery and specify a SQL query to execute against Oracle Database.

5. Select the TButton component and create an OnClick event. Add the code to call the Open method of TOraQuery when the button is clicked.

6. Select the TCRSSHIOHandler component and set the Client property to TScSSHClient.

7. Select the TScFileStorage component and specify the directory for storing information about keys and users in the Path property. Follow the instructions to generate a pair of keys for authenticating the server by the client.

8. Select the TScSSHClient component and specify the server public key in the HostKeyName...
property and the client private key in the PrivateKeyName property. Specify the address of the SSH server in the HostName property and the port, user, and password in corresponding properties. Set the KeyStorage property to an instance of TScFileStorage. If you are connecting to Oracle Cloud, leave the Password and HostKeyName properties empty — only specify PrivateKeyName and User (the default username is opc). See the Oracle documentation for information on generating SSH keys.

9. Select the TOraSession component and set the IOHandler property to an instance of TCRSSHIOHandler. Double-click TOraSession and specify the server address, port, service name, and user credentials.

10. Compile and run the application.
It is not mandatory to use the TcSSHServer component as the SSH server — you can use any other server that implements the SSH protocol.

© 1997-2021 Devart. All Rights Reserved.

4.7.3 Network Tunneling

Connecting to Oracle Database Through HTTP Tunnel

This section discusses how to connect a client application to Oracle Database in two ways: directly and through an HTTP tunnel. If you need to connect to Oracle Database in conditions of restricted connectivity, e.g. when a database server is hidden behind a firewall, or you need to transmit private network data through a public network, you can set up an HTTP tunnel to create a direct network link between two locations. The tunnel is created by an intermediary called a proxy server.

- Direct Connection
- Connecting Through HTTP Tunnel
  - Connecting Through Proxy and HTTP Tunnel
- Sample Application That Uses HTTP/HTTPS Tunneling
- Additional Information

Direct Connection

Direct connection implies that a client connects to a server through a directly connected network, without IP routing: you only need to specify the server address, port number, service
name, and user credentials. This is also the fastest and preferred way to communicate with an Oracle server.

Code sample for a direct connection:

```pascal
var
  OraSession: TOraSession;
...
OraSession := TOraSession.Create(self);
OraSession.Options.Direct = True;
OraSession.Username := 'Scott';
OraSession.Password := 'Tiger';
OraSession.Server := '205.227.44.44:1521/ORCL1020';
OraSession.Connect;
```

Connecting Through HTTP Tunnel

When an Oracle server is hidden behind a firewall, the client is not able to connect to the server directly on a specified port. If your firewall allows HTTP connections, you can use ODAC with a properly configured web server to connect to the database server. ODAC supports HTTP tunneling based on the PHP script.

A possible scenario of using HTTP tunneling: the client needs to access the database of a website from a remote machine, but access to the designated port of the database server is forbidden—only connections on the HTTP port 80 are allowed. To establish a connection in this scenario, you must deploy the tunnel.php script, which is distributed with the provider package, on the web server. It enables access to the database server through an HTTP tunnel. The script must be accessible through HTTP. You can verify script accessibility using any web browser. The script file is located in the HTTP folder of the installed provider: `%Program Files%\Devart\ODAC for RAD Studio XX\HTTP\tunnel.php`. The only requirement to the server is support for PHP 5.

To connect to the database, you must set the `TOraSession` parameters as you do for a direct connection, then set the `HttpOptions.Enabled` property to `True`, and set the following parameters, specific to the HTTP tunneling:

<table>
<thead>
<tr>
<th>Property</th>
<th>Mandatory</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HttpOptions.Url</td>
<td>Yes</td>
<td>The URL of the PHP script for HTTP tunneling. For example, if the script is located in the root directory, the URL may look like this: <code>https://host/tunnel.php</code>.</td>
</tr>
<tr>
<td>HttpOptions.Username</td>
<td>No</td>
<td>The username and password for the password-protected directory that contains the HTTP tunneling script.</td>
</tr>
</tbody>
</table>
Connecting Through Proxy and HTTP Tunnel

The HTTP tunneling server may be not be directly accessible from the client machine—for example, the client address is 10.0.0.2, the server address is 192.168.0.10, and the Oracle server listens on port 1521. The client and server reside in different networks, so the client can only reach it through the proxy server at 10.0.0.1, which listens on port 808. In this case, in addition to `TSession.HttpOptions`, you have to set values for `HttpOptions.ProxyOptions`, for example:

```pascal
var
  OraSession: TOraSession;
  ...OraSession := TOraSession.Create(self);
  OraSession.Server := '192.168.0.10';
  OraSession.Port := 1521;
  OraSession.Username := 'Scott';
  OraSession.Password := 'Tiger';
  OraSession.HttpOptions.Enabled := True;
  OraSession.HttpOptions.ProxyOptions.Hostname := '10.0.0.1';
  OraSession.HttpOptions.ProxyOptions.Port := 808;
  OraSession.Connect;
```

Note that setting the parameters for `OraSession.HttpOptions.ProxyOptions` automatically enables the use of the proxy server.

Sample Application That Uses HTTP/HTTPS Tunneling

1. Open your browser and visit the URL of the `tunnel.php` script on your server to verify that the script has been properly installed.

2. Run RAD Studio and create a new VCL application.

3. Place the following components on the form: `TSession`, `TQuery`, `DataSource`, `DBGrid`, `Button`, and `TCRSSLIOHandler`. The last component is required when connecting through HTTPS. `TCRSSLIOHandler` is distributed with SecureBridge and is required for binding ODAC with SecureBridge. The installation instructions for the component are provided in `Readme.html`, which is located by default in "My Documents\Devart\ODAC for RAD Studio XX
4. Select TDBGrid and set the DataSource property to an instance of TDataSource.
5. Select the TDataSource component and set the DataSet property to an instance of TOraQuery.
6. Select TOraQuery and set the Session property to an instance of TOraSession. Double-click the component and enter an SQL statement to be executed against Oracle Database.
7. Double-click TButton to switch to the code view. Add the code to call the Open method of TOraQuery when the button is clicked.

8. Select the TOraSession Component. If you use an HTTPS tunnel, set the IOHandler property to CRSSLIOHandler1. Expand the HttpOptions and enter the URL of the tunnel.php script on your server.
9. Double-click the ToraSession component. Specify your server address, port, service name, username and password for the Oracle user. Click Connect to test connection to the Oracle server.

10. Press F9 to compile and run the project, and click the button to run the query against the database through HTTPS and display the data in the form.
Additional Information

There is one more way to tunnel network traffic. The Secure Shell forwarding, or SSH, can be used for data forwarding. However, SSH is designed to encrypt traffic rather than traverse firewalls. The Connecting via SSH document describes how to set up an SSH connection in ODAC.

Keep in mind that traffic tunneling or encryption always increases the CPU usage and bandwidth utilization. It is recommended that you use direct connection whenever possible.

© 1997-2021 Devart. All Rights Reserved.

4.8 Disconnected Mode

In disconnected mode a connection opens only when it is required. After performing all server calls connection closes automatically until next server call is required. Datasets remain opened when connection closes. Disconnected Mode may be useful for saving server resources and operating in an unstable or expensive network. Drawback of using disconnected mode is that each connection establishing requires some time for authorization. If connection is often closed and opened it can slow down application work. We recommend to use pooling to solve this problem. For additional information see TCustomDAConnection.Pooling.

To enable disconnected mode set TCustomDAConnection.Options.DisconnectedMode to True.
In disconnected mode a connection is opened for executing requests to the server (if it was not opened already) and is closed automatically if it is not required any more. If the connection was explicitly opened (the Connect method was called or the Connected property was explicitly set to True), it does not close until the Disconnect method is called or the Connected property is set to False explicitly.

The following settings are recommended to use for working in disconnected mode:

```plaintext
TDataSet.CachedUpdates = True
TCustomDADataset.FetchAll = True
TCustomDADataset.Options.LocalMasterDetail = True
AutoCommit = True
```

These settings minimize the number of requests to the server.

**Disconnected mode features**

If you perform a query with the FetchAll option set to True, connection closes when all data is fetched if it is not used by someone else. If the FetchAll option is set to false, connection does not close until all data blocks are fetched.

If explicit transaction was started, connection does not close until the transaction is committed or rolled back.

If the query was prepared explicitly, connection does not close until the query is unprepared or its SQL text is changed.

If dataset uses TCustomDADataset.LockMode set to lmLockImmediate, connection opens when user starts to edit the record, and closes after user posts or cancels changes.

**See Also**

- TCustomDAConnection.Options
- FetchAll
- Devart.Odac.TOraQuery.LockMode
- TCustomDAConnection.Pooling
- TCustomDAConnection.Connect
- TCustomDAConnection.Disconnect
- Connected
- Working in unstable network

© 1997-2021
Devart. All Rights Reserved.
4.9 Batch Operations

Data amount processed by modern databases grows steadily. In this regard, there is an acute problem — database performance. Insert, Update and Delete operations have to be performed as fast as possible. Therefore Devart provides several solutions to speed up processing of huge amounts of data. So, for example, insertion of a large portion of data to a DB is supported in the TOraLoader. Unfortunately, TOraLoader allows to insert data only – it can't be used for updating and deleting data.

The new version of Devart Delphi Data Access Components introduces the new mechanism for large data processing — Batch Operations. The point is that just one parametrized Modify SQL query is executed. The plurality of changes is due to the fact that parameters of such a query will be not single values, but a full array of values. Such approach increases the speed of data operations dramatically. Moreover, in contrast to using TOraLoader, Batch operations can be used not only for insertion, but for modification and deletion as well.

Let’s have a better look at capabilities of Batch operations with an example of the BATCH_TEST table containing attributes of the most popular data types.

Batch_Test table generating script

```sql
CREATE TABLE BATCH_TEST  
(  
  ID INTEGER NOT NULL PRIMARY KEY,  
  F_INTEGER INTEGER,  
  F_FLOAT FLOAT,  
  F_STRING VARCHAR2(250),  
  F_DATE DATE  
);  
```

Batch operations execution

To insert records into the BATCH_TEST table, we use the following SQL query:

```sql
```

When a simple insertion operation is used, the query parameter values look as follows:

<table>
<thead>
<tr>
<th>Parameters</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>2.5</td>
<td>'String Value 1'</td>
<td>01.09.2015</td>
<td></td>
</tr>
</tbody>
</table>

After the query execution, one record will be inserted into the BATCH_TEST table.

When using Batch operations, the query and its parameters remain unchanged. However, parameter values will be enclosed in an array:
Now, 5 records are inserted into the table at a time on query execution.

How to implement a Batch operation in the code?

Batch INSERT operation sample

Let’s try to insert 1000 rows to the BATCH_TEST table using a Batch Insert operation:

```pascal
var i: Integer;
begin
  // describe the SQL query
  OraQuery1.SQL.Text := 'INSERT INTO BATCH_TEST VALUES (:ID, :F_INTEGER, ... := Now;
  end;
  // insert 1000 rows into the BATCH_TEST table
  OraQuery1.Execute(1000);
end;
```

This command will insert 1000 rows to the table with one SQL query using the prepared array of parameter values. The number of inserted rows is defined in the Iters parameter of the Execute(Iters: integer; Offset: integer = 0) method. In addition, you can pass another parameter – Offset (0 by default) – to the method. The Offset parameter points the array element, which the Batch operation starts from.

We can insert 1000 records into the BATCH_TEST table in 2 ways.
All 1000 rows at a time:

```
Query1.Execute(1000);
```

2×500 rows:

```
// insert first 500 rows
OraQuery1.Execute(500, 0);
// insert next 500 rows
OraQuery1.Execute(500, 500);
```

500 rows, then 300, and finally 200:

```
// insert 500 rows
Query1.Execute(500, 0);
// insert next 300 rows starting from 500
OraQuery1.Execute(300, 500);
// insert next 200 rows starting from 800
Query1.Execute(200, 800);
```

Batch UPDATE operation sample

With Batch operations we can modify all 1000 rows of our BATCH_TEST table just this simple:

```
var  
i: Integer;
begin  
  // describe the SQL query
  Query1.SQL.Text := 'UPDATE BATCH_TEST SET F_INTEGER=:F_INTEGER, ... 
  // define parameter types passed to the query:
  Query1.Params[0].DataType := ftInteger;
  Query1.Params[1].DataType := ftFloat;
  Query1.Params[2].DataType := ftString;
  Query1.Params[3].DataType := ftDateTime;
  Query1.Params[4].DataType := ftInteger;
  
  // specify the array dimension:
  Query1.Params.ValueCount := 1000;
  // populate the array with parameter values:
  for i := 0 to 1000 - 1 do begin 
    Query1.Params[0][i].AsInteger := i - 2000 + 1;
    Query1.Params[1][i].AsFloat := (i + 1) / 100;
    Query1.Params[2][i].AsString := 'New Values ' + IntToStr(i + 1);
    Query1.Params[3][i].AsDateTime := Now;
    Query1.Params[4][i].AsInteger := i + 1;
  end;
  // update 1000 rows in the BATCH_TEST table
  Query1.Execute(1000);
end;
```

Batch DELETE operation sample

Deleting 1000 rows from the BATCH_TEST table looks like the following operation:

```
var
i: Integer;
begin
  // describe the SQL query
  OraQuery1.SQL.Text := 'DELETE FROM BATCH_TEST WHERE ID=:ID';
  // define parameter types passed to the query:
  OraQuery1.Params[0].DataType := ftInteger;
  // specify the array dimension
  OraQuery1.Params.ValueCount := 1000;
  // populate the arrays with parameter values
  for i := 0 to 1000 - 1 do
    OraQuery1.Params[0][i].AsInteger := i + 1;
  // delete 1000 rows from the BATCH_TEST table
  OraQuery1.Execute(1000);
end;

Performance comparison

The example with BATCH_TEST table allows to analyze execution speed of normal operations with a database and Batch operations:

<table>
<thead>
<tr>
<th>Operation Type</th>
<th>25 000 records</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Standard Operation (sec.)</td>
</tr>
<tr>
<td>Insert</td>
<td>17.64</td>
</tr>
<tr>
<td>Update</td>
<td>18.28</td>
</tr>
<tr>
<td>Delete</td>
<td>16.19</td>
</tr>
</tbody>
</table>

The less, the better.

It should be noted, that the retrieved results may differ when modifying the same table on different database servers. This is due to the fact that operations execution speed may differ depending on the settings of a particular server, its current workload, throughput, network connection, etc.

Thing you shouldn't do when accessing parameters in Batch operations!

When populating the array and inserting records, we accessed query parameters by index. It would be more obvious to access parameters by name:

    for i := 0 to 9999 do begin
      OraQuery1.Params.ParamByName('ID')[i].AsInteger := i + 1;
      OraQuery1.Params.ParamByName('F_INTEGER')[i].AsInteger := i + 2000 + 1;
      OraQuery1.Params.ParamByName('F_FLOAT')[i].AsFloat := (i + 1) / 12;
      OraQuery1.Params.ParamByName('F_STRING')[i].AsString := 'Values ' + IntToStr(i + 1);
      OraQuery1.Params.ParamByName('F_DATE')[i].AsDateTime := Now;
    end;

However, the parameter array would be populated slower, since you would have to define the
ordinal number of each parameter by its name in each loop iteration. If a loop is executed 10000 times – **performance loss can become quite significant**.

© 1997-2021 Devart. All Rights Reserved.

4.10 Increasing Performance

This topic considers basic stages of working with DataSet and ways to increase performance on each of these stages.

**Connect**

If your application performs Connect/Disconnect operations frequently, additional performance can be gained using pooling mode (TCustomDACConnection.Pooling = True). It reduces connection reopening time greatly (hundreds times). Such situation usually occurs in web applications.

**Execute**

If your application executes the same query several times, you can use the TCustomDADataset.Prepare method or set the TDatasetOptions.AutoPrepare property to increase performance. For example, it can be enabled for Detail dataset in Master/Detail relationship or for update objects in TCustomDATAUpdateSQL. The performance gain achieved this way can be anywhere from several percent to several times, depending on the situation.

To execute SQL statements a TOraSQL component is more preferable than TOraQuery. It can give several additional percent performance gain.

Using DML Array parameters, combined with query preparing can give a considerable performance gain. For example if you need to perform an insert query plenty of times, it is recommended to do it the following way to get the best performance. At first, set parameter data types, and then prepare the query. After that, set param Lengths and fill them with values. Finally, execute the query. For instance:

```delphi
OraSQL1.ParamByName('Param1').DataType := ftInteger;
OraSQL1.Prepare;
OraSQL1.ParamByName('Param1').Param[0].Length := 1000;
for i := 1 to 1000
  OraSQL1.ParamByName('Param1').Param[0].ItemAsInteger[i] := 123;
OraSQL1.Execute;
```

If you execute many different SELECT statements, you can gain additional performance by setting the TOraSQL.StatementCache property to True. This feature is available only with
Oracle 9.2i and higher. The `TOraSession.Options.StatementCache` property should be set to True to use this feature. But using the StatementCache property you may easy step over maximum open cursors on Oracle server. And thus you must be attentive when using this option.

If the `TCustomDADataSet.Options.StrictUpdate` option is set to False, the `RowsAffected` property is not calculated and becomes equal zero. This can improve performance of query executing, so if you need to execute many data updating statements at once and you don't mind affected rows count, set this option to False.

**Fetch**

In some situations you can increase performance a bit by using `TCustomDADataSet.Options.CompressBlobMode`. Sometimes using `TOraDataSet.Options.DeferredLobRead` can give some additional performance, because Lobs will be read when they are required. You can also set `TOraSession.Options.OptimizerMode` to adjust the fetch performance.

Oracle optimizer can be tuned to increase performance of SQL statements executing and data fetching.

You can also tweak your application performance by using the following properties of `TCustomDADataSet` descendants:

- `FetchRows`
- `Options.FlatBuffers`
- `Options.LongStrings`
- `UniDirectional`

See the descriptions of these properties for more details and recommendations.

**Navigate**

The `Locate` function works faster when dataset is locally sorted on KeyFields fields. Local dataset sorting can be set with the `IndexFieldNames` property. Performance gain can be large if the dataset contains a large number of rows.

Lookup fields work faster when lookup dataset is locally sorted on lookup Keys.

Setting the `TDADatasetOptions.CacheCalcFields` property can improve performance when locally sorting and locating on calculated and lookup fields. It can be also useful when calculated field expressions contain complicated calculations.
Setting the **TDADatasetOptions.LocalMasterDetail** option can improve performance greatly by avoiding server requests on detail refreshes. Setting the **TDADatasetOptions.DetailDelay** option can be useful for avoiding detail refreshes when switching master DataSet records frequently.

**Update**

If your application updates datasets in the CachedUpdates mode, then setting the **TCustomDADataSet.Options.UpdateBatchSize** option to more than 1 can improve performance several hundred times more by reducing the number of requests to the server.

Specifying update SQL statements in a dataset improves performance because of omitting SQL statements generation and automatic preparation of internal updating datasets that are created for every kind of update SQL statements.

You can also increase the data sending performance a bit (several percents) by using Dataset.UpdateObject.ModifyObject, Dataset.UpdateObject, etc. Little additional performance improvement can be reached by setting the **AutoPrepare** property for these objects.

**Insert**

If you are about to insert a large number of records into a table, you should use the **T:Devart.Odac.TOraLoader** component instead of Insert/Post methods, or execution of the INSERT commands multiple times in a cycle. Sometimes usage of **T:Devart.Odac.TOraLoader** improves performance several times.

### 4.11 Macros

Macros help you to change SQL statements dynamically. They allow partial replacement of the query statement by user-defined text. Macros are identified by their names which are then referred from SQL statement to replace their occurrences for associated values.

First step is to assign macros with their names and values to a dataset object.

Then modify SQL statement to include macro names into desired insertion points. Prefix each name with & ("at") sign to let ODAC discriminate them at parse time. Resolved SQL statement will hold macro values instead of their names but at the right places of their occurrences. For example, having the following statement with the TableName macro name:

```sql
SELECT * FROM &TableName
```
You may later assign any actual table name to the macro value property leaving your SQL statement intact.

```delphi
Query1.SQL.Text := 'SELECT * FROM &TableName';
Query1.MacroByName('TableName').Value := 'Dept';
Query1.Open;
```

ODAC replaces all macro names with their values and sends SQL statement to the server when SQL execution is requested.

Note that there is a difference between using TMacro AsString and Value properties. If you set macro with the AsString property, it will be quoted. For example, the following statements will result in the same result Query1.SQL property value.

```delphi
Query1.MacroByName('StringMacro').Value := '''A string''';
Query1.MacroByName('StringMacro').AsString := 'A string';
```

Macros can be especially useful in scripts that perform similar operations on different objects. You can use macros that will be replaced with an object name. It allows you to have the same script text and to change only macro values. For example, the following is a script that creates a new user account and grants required privileges.

```delphi
Script1.SQL.Add('CREATE USER &Username IDENTIFIED BY &Password;');
Script1.SQL.Add('GRANT &Privileges TO &Username;');
```

To execute the script for another user you do not have to change the script SQL property, you can just set required macro values.

You may also consider using macros to construct adaptable conditions in WHERE clauses of your statements.

**See Also**
- TMacro
- TCustomDADataset.MacroByName
- TCustomDADataset.Macros

© 1997-2021 Devart. All Rights Reserved.

Request Support DAC Forum Provide Feedback

4.12 **DataSet Manager**

**DataSet Manager window**

The DataSet Manager window displays the datasets in your project. You can use the DataSet Manager window to create a user interface (consisting of data-bound controls) by dragging
items from the window onto forms in your project. Each item has a drop-down control list where you can select the type of control to create prior to dragging it onto a form. You can customize the control list with additional controls, including the controls you have created.

Using the DataSet Manager window, you can:

- Create forms that display data by dragging items from the DataSet Manager window onto forms.

- Customize the list of controls available for each data type in the DataSet Manager window.

- Choose which control should be created when dragging an item onto a form in your Windows application.

- Create and delete TField objects in the DataSets of your project.

Opening the DataSet Manager window

You can display the DataSet Manager window by clicking DataSet Manager on the Tools menu. You can also use IDE desktop saving/loading to save DataSet Manager window position and restore it during the next IDE loads.
Observing project DataSets in the DataSet Manager Window

By default DataSet Manager shows DataSets of currently open forms. It can also extract DataSets from all forms in the project. To use this, click *Extract DataSets from all forms in project* button. This settings is remembered. Note, that using this mode can slow down opening of the large projects with plenty of forms and DataSets. Opening of such projects can be very slow in Delphi 6 and Borland Developer Studio 2006 and can take up to several tens of minutes.

DataSets can be grouped by form or connection. To change DataSet grouping click the *Grouping mode* button or click a down. You can also change grouping mode by selecting required mode from the DataSet Manager window popup menu.

Creating Data-bound Controls

You can drag an item from the DataSet Manager window onto a form to create a new data-bound control. Each node in the DataSet Manager window allows you to choose the type of control that will be created when you drag it onto a form. You must choose between a Grid layout, where all columns or properties are displayed in a TDataGrid component, or a Details layout, where all columns or properties are displayed in individual controls.

To use grid layout drag the dataset node on the form. By default TDataSource and TDBGrid components are created. You can choose the control to be created prior to dragging by selecting an item in the DataSet Manager window and choosing the control from the item’s drop-down control list.
To use Details layout choose Details from the DataSet node drop-down control list in the DataSet Manager window. Then select required controls in the drop-down control list for each DataSet field. DataSet fields must be created. After setting required options you can drag the DataSet to the form from the DataSet wizard. DataSet Manager will create TDataSource component, and a component and a label for each field.
Adding custom controls to the DataSet Manager window

To add custom control to the list click the Options button on the DataSet Manager toolbar. A DataSet Manager - Customize controls dialog will appear. Using this dialog you can set controls for the DataSets and for the DataSet fields of different types. To do it, click DataSets node or the node of field of required type in DB objects groups box and use Add and Remove buttons to set required control list. You can also set default control by selecting it in the list of assigned DB controls and pressing Default button.

The default configuration can easily be restored by pressing Reset button in the DataSet Manager - Options dialog.

Working with TField objects

DataSet Manager allows you to create and remove TField objects. DataSet must be active to work with its fields in the DataSet Manager. You can add fields, based on the database table columns, create new fields, remove fields, use drag-n-drop to change fields order.
To create a field based on the database table column right-click the Fields node and select *Create Field* from the popup menu or press <Insert>. Note that after you add at least one field manually, DataSet fields corresponding to data fields will not be generated automatically when you drag the DataSet on the form, and you can not drag such fields on the form. To add all available fields right-click the Fields node and select *Add all fields* from the popup menu.

To create new field right-click the Fields node and select *New Field* from the popup menu or press <Ctrl+Insert>. The New Field dialog box will appear. Enter required values and press OK button.

To delete fields select these fields in the DataSet Manager window and press <Delete>.

DataSet Manager allows you to change view of the fields displayed in the main window. Open the *Customize controls* dialog, and jump to the Options page.

You can chose what information will be added to names of the Field and Data Field objects in the main window of DataSet Manager. Below you can see the example.
4.13 Using Connection Pooling

Connection pooling enables an application to use a connection from a pool of connections that do not need to be reestablished for each use. Once a connection has been created and placed in a pool, an application can reuse that connection without performing the complete connection process.

Using a pooled connection can result in significant performance gains, because applications can save the overhead involved in making a connection. This can be particularly significant for middle-tier applications that connect over a network or for applications that connect and disconnect repeatedly, such as Internet applications.

To use connection pooling set the Pooling property of the TCustomDAConnection component to True. Also you should set the PoolingOptions of the TCustomDAConnection. These options include MinPoolSize, MaxPoolSize, Validate, ConnectionLifeTime. Connections belong to the same pool if they have identical values for the following parameters:

- MinPoolSize
- MaxPoolSize
- Validate
- ConnectionLifeTime
- Server, Username, Password
- TOraSession.Username, TOraSession.Server, TOraSession.ConnectMode
- TOraSession.Options.UseOCI7

When a connection component disconnects from the database the connection actually remains active and is placed into the pool. When this or another connection component connects to the database it takes a connection from the pool. Only when there are no connections in the pool, new connection is established.

ODAC connection pool is thread safe. Multiple threads of an application can connect to the database and disconnect from it using TCustomDAConnection components with pooling enabled at the same time. Connection pool uses the most optimal algorithms for fast and reliable work.

Connections in the pool are validated to make sure that a broken connection will not be returned for the TCustomDAConnection component when it connects to the database. The pool validates connection when it is placed to the pool (e.g. when the TCustomDAConnection component disconnects). If connection is broken it is not placed to the pool. Instead the pool frees this connection. Connections that are held in the pool are validated every 30 seconds. All broken connections are freed. If you set the PoolingOptions.Validate to True, a connection also will be validated when the TCustomDAConnection component connects and takes a connection from the pool. When some network problem occurs all connections to the database can be broken. Therefore the pool validates all connections before any of them will be used by a TCustomDAConnection component if a fatal error is detected on one connection.
The pool frees connections that are held in the pool during a long time. If no new connections are placed to the pool it becomes empty after approximately 4 minutes. This pool behaviour is intended to save resources when the count of connections in the pool exceeds the count that is needed by application. If you set the PoolingOptions.MinPoolSize property to a non-zero value, this prevents the pool from freeing all pooled connections. When connection count in the pool decreases to MinPoolSize value, remaining connection will not be freed except if they are broken.

The PoolingOptions.MaxPoolSize property limits the count of connections that can be active at the same time. If maximum count of connections is active and some TCustomDACConnection component tries to connect, it will have to wait until any of TCustomDACConnection components disconnect. Maximum wait time is 30 seconds. If active connections' count does not decrease during 30 seconds, the TCustomDACConnection component will not connect and an exception will be raised.

You can limit the time of connection’s existence by setting the PoolingOptions.ConnectionLifeTime property. When the TCustomDACConnection component disconnects, its internal connection will be freed instead of placing to the pool if this connection is active during the time longer than the value of the PoolingOptions.ConnectionLifeTime property. This property is designed to make load balancing work with the connection pool.

To force freeing of a connection when the TCustomDACConnection component disconnects, the RemoveFromPool method of TCustomDACConnection can be used. You can also free all connection in the pool by using the class procedures Clear or AsyncClear of TOraConnectionPoolManager. These procedures can be useful when you know that all connections will be broken for some reason.

It is recommended to use connection pooling with the DisconnectMode option of the TCustomDACConnection component set to True. In this case internal connections can be shared between TCustomDACConnection components. When some operation is performed on the TCustomDACConnection component (for example, an execution of SQL statement) this component will connect using pooled connection and after performing operation it will disconnect. When an operation is performed on another TCustomDACConnection component it can use the same connection from the pool.

Also, ODAC supports proxy connection pooling. When proxy pooling is used, TOraSession components can connect to a database with different Username and Password properties but all connections in the pool use PoolingOptions.ProxyUsername and PoolingOptions.ProxyPassword. When connecting, TOraSession component creates a new connection. It uses Username, Password properties and a connection from the pool as a
proxy connection. The proxy connection pool allows you to use a single pool for all sessions with different **Username** and **Password** properties.

You can use OCI connection pooling or MTS connection pooling. To use these types of pooling set the **PoolingOptions.PoolType** to ptOCI or ptMTS. In this case the pool is created and managed by the Oracle client or by MTS.

**See Also**
- **TCustomDACConnection.Pooling**
- **TCustomDACConnection.PoolingOptions**
- **TOraSession.PoolingOptions**
- **Working with Disconnected Mode**

**4.14 Automatic Key Field Value Generation**

When editing dataset it is often convenient to generate key field(s) values automatically instead of filling them manually. In the most common way application developer generates primary key value basing it on previously created sequence. There are three ways to do it.

First, application independent way - developer creates AFTER INSERT trigger that fills the field value. But here he faces the problem with getting value inserted by trigger back to dataset. This problem can be easily solved in ODAC by specifying return parameters. For instance:

```sql
// suppose that AFTER INSERT trigger fills DepNo field
OraQuery.SQL.Text := 'SELECT DepNo, DepName, Location FROM Department';
OraQuery.SQLInsert.Text := 'INSERT INTO Department (DepNo, DepName, Location)
VALUES(DepNo, DepName, Location) ' +
'RETURNING DepNo INTO :DepNo';
...```

Second way is custom key field value generation. Developer can fill key field value in TOraDataSet.BeforePost event handler. But in this case he should manually execute query and retrieve sequence value. So this way may be useful only if some special value processing is needed.

The third way, using KeySequence is the simplest. Developer only needs to specify two properties and key field values are generated automatically. There is no need to create trigger or perform custom BeforePost processing.
OraQuery.SQL.Text := 'SELECT DepNo, DepName, Location FROM Department';
OraQuery.KeyFields := 'DepNo';  // key field
OraQuery.KeySequence := 'DepSequence'; // sequence that will generate values

See also
- KeySequence

4.15 TOraLoader Component

There are cases when you need to put large amount of data to Oracle database. Of course, you may construct INSERT SQL statement and execute it with TOraSQL component. But it takes a lot of time. You can greatly speed up loading time of data by using DML array features. Oracle 8i has better way to do it. With Oracle 8i using the direct path load interface is possible. The direct path load interface allows to access the direct path load engine of the Oracle database server to perform the functions of the Oracle SQL*Loader utility. This functionality provides the ability to load data from external files into Oracle database objects, either a table or a partition of a partitioned table.

ODAC simplifies using direct path load interface by TOraLoader component. TOraLoader allows you to load various formatted data. The capability of TOraLoader component to load various formatted data is reached by reading external data in writing method itself.

To write your own loader you should:
- create TOraLoader component;
- set name of loading table to TableName;
- create and customize columns which will be loaded (use TOraLoader component editor at design time);
- write your own event handler: OnGetColumnData or OnPutData
- call Load method to start loading.

See Also
- TOraLoader
- TDPColumn
4.16 **TOraTransaction Component**

TOraTransaction component can be used to manage either local or distributed (global) transactions.

To start local transaction with TOraTransaction component, set DefaultSession property of the component to a session on which transaction will be performed. Set IsolationLevel property optionally. Then call StartTransaction method of the TOraTransaction component. To manage transaction use Commit, Rollback, Savepoint, RollbackToSavepoint methods.

Global transactions can be performed on one or more sessions connected to the same or to the different databases. These sessions can be established from different applications and computers. On each of these sessions a separate branch of transaction is performed. Global transaction can be coordinated either by internal mechanism of TOraTransaction or Microsoft Transaction Manager DTC. This behavior can be tuned by GlobalCoordinator property. In case of using internal mechanism you should specify TransactionId and BranchQualifier for each session to identify global transaction. Global transaction can be managed using Commit, Rollback, Savepoint, RollbackToSavepoint, Detach and Resume methods of TOraTransaction. If an OraSession uses global transaction, it must have non-empty *InternalName* property. For more information about global transaction please refer to Oracle documentation.

**Note:** transaction will be global if either TransactionId or TransactionName property is set or if GlobalCoordinator property is gcMTS.

Here is a sample code that starts and commits global transaction:

```pascal
var
  Id: TBytes;
begin
  OraSession1.InternalName := 'SampleName1';
  OraSession2.InternalName := 'SampleName2';
  OraSession1.Connect;
  OraSession2.Connect;
  SetLength(Id, 2);
  id[0] := 7; id[1] := 3;
  OraTransaction.TransactionId := Id;
  SetLength(Id, 1);
  id[0] := 1;
  OraTransaction.AddSession(OraSession1, Id);
  id[0] := 2;
  OraTransaction.AddSession(OraSession2, Id);
  OraTransaction.StartTransaction;
  OraSQL1.Session := OraSession1;
  OraSQL2.Session := OraSession2;
  OraSQL1.Execute;
  OraSQL2.Execute;
```

© 2021 Devart
The following example demonstrates a global transaction with two branches created from different applications. After performing update these applications detach their transaction branches. Then third application (transaction manager) resumes all branches of the transaction and performs two-phase commit.

```
// Application 1
var
  Id: TBytes;
begin
  OraSession.InternalName := 'SampleName1';
  OraSession.Connect;
  SetLength(Id, 2);
  id[0] := 7; id[1] := 3;
  OraTransaction.TransactionId := Id;
  SetLength(Id, 1);
  id[0] := 1;
  OraTransaction.AddSession(OraSession, Id);
  OraTransaction.StartTransaction;
  OraSQL.Execute;
  OraTransaction.Detach;
end;

// Application 2
var
  Id: TBytes;
begin
  OraSession.InternalName := 'SampleName2';
  OraSession.Connect;
  SetLength(Id, 2);
  id[0] := 7; id[1] := 3;
  OraTransaction.TransactionId := Id;
  SetLength(Id, 1);
  id[0] := 2;
  OraTransaction.AddSession(OraSession, Id);
  OraTransaction.StartTransaction;
  OraSQL.Execute;
  OraTransaction.Detach;
end;

// Application 3 (transaction manager that commits transaction)
var
  Id: TBytes;
begin
  OraSession1.Connect;
  OraSession2.Connect;
  SetLength(Id, 2);
  id[0] := 7; id[1] := 3;
  OraTransaction.TransactionId := Id;
  SetLength(Id, 1);
  id[0] := 1;
  OraTransaction.AddSession(OraSession1, Id);
  id[0] := 2;
  OraTransaction.AddSession(OraSession2, Id);
  OraTransaction.Resume;
```
The next example demonstrates using distributed transaction coordinated by MTS DTC:

```pascal
begin
  OraSession1.InternalName := 'SampleName1';
  OraSession2.InternalName := 'SampleName2';
  OraSession1.Connect;
  OraSession2.Connect;
  OraTransaction.GlobalCoordinator := gcMTS;
  OraTransaction.AddSession(OraSession1);
  OraTransaction.AddSession(OraSession2);
  OraTransaction.StartTransaction;
  OraSQL1.Session := OraSession1;
  OraSQL2.Session := OraSession2;
  OraSQL1.Execute;
  OraSQL2.Execute;
  OraTransaction.Commit;
end;
```

4.17 TOraQueue, TOraQueueAdmin and TOraQueueTable Components

TOraQueue, TOraQueueAdmin and TOraQueueTable components provide access to Oracle Streams Advanced Queuing. Oracle Streams AQ provides database-integrated message queuing functionality. It is built on top of Oracle Streams and leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S).

Because Oracle Streams AQ is implemented in database tables, all operational benefits of high availability, scalability, and reliability are also applicable to queue data. Standard database features such as recovery, restart, and security are supported by Oracle Streams AQ. Like other database tables, queue tables can be imported and exported.

When applications communicate with each other, producer applications enqueue messages and consumer applications dequeue messages. At the basic level of queuing, one producer enqueues one or more messages into one queue. Each message is dequeued and processed once by one of the consumers. A message stays in the queue until a consumer dequeues it or the message expires. A producer may stipulate the delay before the message is available for the consumption, and the time after which the message expires. Likewise, a consumer may wait when trying to dequeue a message if no message is available. An agent program or application may act both as a producer and a consumer.
**TOraQueue** component provides access to functionality of DBMS_AQ Oracle package. User must have AQ_USER_ROLE to work with this component. TOraQueue component can be used to enqueue and dequeue messages from the given queue. Queue message includes a payload and message properties. Type of payload is defined for each queue. This can be Oracle object type or 'RAW' type. RAW payload contains any array of bytes. To use TOraQueue component set its Session and QueueName properties. Then one of Enqueue method overloads can be used to enqueue message.

To enqueue message with RAW payload use the Enqueue method overload with string or TBytes Payload parameter. For example:

```pascal
var
  MsgProp: TQueueMessageProperties;
begin
  MsgProp := TQueueMessageProperties.Create;
  try
    MsgProp.Priority := 2;
    MsgProp.Delay := 30; // delay in sec before message can be dequeued
    MsgProp.Expiration := 180; // message expiration in sec
    OraQueue.Enqueue('123', MsgProp);
  finally
    MsgProp.Free;
  end;
end;
```

To enqueue message to queue with object payload use Enqueue method overload with TOraObject Payload parameter. For example:

```pascal
var
  MsgProp: TQueueMessageProperties;
  Payload: TOraObject;
begin
  MsgProp := TQueueMessageProperties.Create;
  try
    Payload := TOraObject.Create;
    try
      Payload.AllocObject(OraSession.OCISvcCtx, 'OBJ_MES');
      Payload.AttrAsInteger['A'] := 3;
      Payload.AttrAsString['B'] := 'Hello';
      OraQueue.Enqueue(Payload, MsgProp);
    finally
      Payload.Free;
    end;
  finally
    Payload.Free;
  end;
end;
```

To dequeue message, use one of **TOraQueue.Dequeue** method overloads. If there are no
messages available for dequeuing, Dequeue method will wait until a message will be available. The following example demonstrates the usage of Dequeue method for a queue with object payload:

```pascal
var
  Payload: TOraObject;
  a: integer;
  b: string;
begin
  Payload := TOraObject.Create;
  try
    OraQueue.Dequeue(Payload);
    a := Payload.AttrAsInteger['A'];
    b := Payload.AttrAsString['B'];
  finally
    Payload.Free;
  end;
end;
```

To get notification when a message available to dequeuing appears in the queue use `TOraQueue.Listen` method or set `AsyncNotification` property to True and write `OnMessage` event handler.

Listen method listens on one or more queues on behalf of a list of agents. This method waits until a message is available in one of the queues and then returns the agent that corresponds this queue. Listen method should be called in separate thread to avoid program blocking. For example:

```pascal
procedure TListenThread.Execute;
var
  Agents: TQueueAgents;
  Agent: TQueueAgent;
  Message: string;
begin
  Agents := ...        OraQueue.Dequeue(Message);      end;    finally      Agent.Free;    end;  finally    Agents.Free;  end;end;
```
**TOraQueueAdmin** and **TOraQueueTable** components are used to administrate queues. User must have AQ_ADMINISTRATOR_ROLE to work with these components. TOraQueueAdmin and TOraQueueTable components can be used to create and drop queues and queue tables, alter queue properties.

Queue can be persistent or non-persistent. To create persistent queue first a queue table must be created. Use TOraQueueTable component to create a queue table. Set properties of the component to required values and then call **CreateQueueTable** method. Use **AlterQueueTable** method to alter a queue table properties and **DropQueueTable** method to drop a queue table.

When a queue table is created, TOraQueueAdmin component can be used to create a queue. Set properties of the component to required values and then call **CreateQueue** method. Use **AlterQueue** method of TOraQueueAdmin to alter a queue properties and **DropQueue** method to drop a queue.

Before messages can be enqueued and dequeued enqueuing and dequeuing must be started on the queue. Use **StartQueue** method of TOraQueueAdmin to start enqueuing and dequeuing on the queue.

---

## 4.18 TOraChangeNotification Component

TOraChangeNotification component is used to register queries with the database and receive notifications in response to DML or DDL changes on the objects associated with the queries. The notifications are published by the database when the DML or DDL transaction commits.

When the database issues change notification, it can contain some or all of the following information:

- Names of the modified objects. For example, the notification can specify that the hr.employees table was changed.

- The type of change. For example, the message specifies whether the change was caused by an INSERT, UPDATE, DELETE, ALTER TABLE, or DROP TABLE.

- The ROWIDs of the changed rows and the type of DML that changed them.

- Global events such as STARTUP and SHUTDOWN (consistent only). In a Real Applications Cluster, the database delivers a notification when the first instance on the database starts or the last instance shuts down.
The notification contains only metadata about the changed rows or objects rather than the changed data itself. For example, the database does not notify the client that monthly salary increased from 5000 to 6000. To obtain more recent values for the changed objects or rows, the client must query the database based on the information contained in the notification.

Database Change Notification is useful for an application that caches query result sets on mostly read-only objects in the mid-tier to avoid network round trips to the database. Such application can create a registration on the queries it is interested in caching using the change notification service. On changes to objects referenced inside those queries, the database publishes a change notification when the underlying transaction commits. In response to the notification, the application can refresh its cache by re-executing the queries.

TOraChangeNotification component represents dependency between an application and an Oracle database based on the database events in which the application is interested. To create subscription place on the form TOraChangeNotification component and assign it to ChangeNotification properties of datasets. When you open a dataset subscription on changes to database tables that dataset selects data from will be created. All datasets that use one TOraChangeNotification component share one change notification subscription. A dataset is registered in subscription when you open this dataset.

Components derived from TOraDataSet can automatically refresh their data in response to change notification. To enable autorefresh set ReflectChangeNotify in TOraDataSet.Options to True. Dataset refreshes when data in one of the tables that appear in query FROM clause is changed. If dataset's SQL has ROWID in SELECT clause and changed table corresponds UpdatingTable property of TOraDataSet then only changed rows are refreshed with RefreshRecord method. If TOraDataSet.SQLRefresh property is set it must use ROWID. Otherwise refreshing may be incorrect. If TOraDataSet.SQL property doesn't contain ROWID in SELECT clause or data is changed not in UpdatingTable but in other table that appears in query FROM clause the dataset performs full refresh with Refresh method.

Change notification can be handled manually using OnChange event of TOraChangeNotification component. For example:

```pascal
procedure TForm1.OraChangeNotificationChange(Sender: TObject; NotifyType: TChangeNotifyEventType; TableChanges: TNotifyTableChanges);
var
  i, j: integer;
begin
  if NotifyType <> cneObjChange then Exit;
  for i := 0 to TableChanges.Count - 1 do begin
    Memo.Lines.Add(TableChanges[i].TableName);
    if cnoAllRows in TableChanges[i].Operations then Continue;
    for j := 0 to TableChanges[i].RowChanges.Count - 1 do
```
For the best performance of change notification, the following guidelines are presented. Registered objects are few and mostly read-only and modifications to these objects are rather an exception than a rule. If the object is extremely volatile, then it will cause a large number of invalidation notifications to be sent, and potentially a lot of storage in the invalidation queue on the server. If there are frequent and a large numbers of notifications, the OLTP throughput can be slowed down due to the overhead of the notifications generation. It is also a good idea to keep the number of duplicate registrations on any given object low (ideally one) in order to avoid the same notification message being replicated to multiple recipients.

See Also
- TOraDataSet.Options
- TOraChangeNotification

© 1997-2021 Devart. All Rights Reserved.

4.19 Working With Data

This section describes the basics of working with data. It contains the information on how to work with BLOB, CLOB, XMLTYPE, VARRAY data types, Unicode Character Data, Objects, PL/SQL Tables, etc.

- BLOB and CLOB Data Types
- Unicode Character Data
- Objects
- XMLTYPE Data Type
- VARRAY Data Type
- DML Array
- PL/SQL Tables

© 1997-2021 Devart. All Rights Reserved.

4.19.1 BLOB and CLOB Data Types

ODAC components support Oracle 8 BLOB and CLOB data types. You can retrieve values of LOB fields using TOraQuery component the same way as you do for LONG or LONG RAW fields. The difference with usage of LOB data type becomes evident when you need to
access these fields in SQL DML and PL/SQL statements.

For BLOB and CLOB data types only LOB locators (pointers to data) are stored in table columns; actual BLOB and CLOB data is stored in separate tablespace. This is the difference to the way that data of LONG or LONG RAW types is stored in database – tables hold their immediate values.

When accessing LOB column, it is the locator which is returned, not the value itself as in the case with LONG or LONG RAW data types.

For example consider this table definition.

```sql
CREATE TABLE ClobTable (  Id NUMBER,  Name VARCHAR2(30),  Desc CLOB )
```

To update Desc column of this table we need to create CLOB, get its locator, write CLOB data using this locator and execute UPDATE statement to write the locator into the table field.

The following SQL statement can be used to update ClobTable:

```sql
UPDATE ClobTable
SET
  Name = :Name,
  Desc = EMPTY_CLOB()
WHERE
  Id = :Id
RETURNING
  Desc INTO :Desc
```

You can use EMPTY_BLOB or EMPTY_CLOB Oracle function to create empty LOB. After executing this statement LOB locator is returned into Desc parameter. Then ODAC writes LOB data into database using this locator. You must set ParamType of Desc parameter to ptInput.

It is important for ODAC to use ParamType property of parameters in LOB operations. If ParamType is ptInput ODAC writes data to a server, if ParamType is ptOutput it reads data.

Another way to update ClobTable is using temporary LOB. Set TemporaryLobUpdate in TOraDataSet.Options to True. Then ODAC will create temporary LOB and write data to it before executing SQL statement.

When TemporaryLobUpdate option is True following statement can be used to update ClobTable:
UPDATE ClobTable
SET
  Name = :Name,
  Desc = :Desc
WHERE
  Id = :Id

ODAC will initialize Desc parameter with locator of temporary LOB before executing this statement.

TemporaryLobUpdate option should be set to True when calling stored procedure with IN or IN OUT LOB parameter. To call procedure:

CREATE OR REPLACE
PROCEDURE ClobTableUpdate (p_Id IN NUMBER, p_Name IN VARCHAR2,
                           p_Desc IN CLOB)
IS
BEGIN
  UPDATE ClobTable
  SET
    Name = p_Name,
    Desc = p_Desc
  WHERE
    Id = p_Id;
END;

the following code can be used:

OraStoredProcedure.Options.TemporaryLobUpdate := True;
OraStoredProcedure.StoredProcName := 'ClobTableUpdate';
OraStoredProcedure.Prepare;
OraStoredProcedure.ParamByName('p_Id').AsInteger := Id;
OraStoredProcedure.ParamByName('p_Name').AsString := Name;
OraStoredProcedure.ParamByName('Desc').ParamType := ptInput;
OraStoredProcedure.ParamByName('Desc').AsOraClob.LoadFromFile(FileName);
OraStoredProcedure.Execute;

Note that LOB parameter can have ptInOutOutput type only when TemporaryLobUpdate option is set to True. Otherwise the type of LOB parameter must be ptInput or ptOutput.

You can also use dtBlob and dtMemo datatypes with LOB parameters to write ordinary DML statements. In this case Oracle automatically converts LONG and LONG ROW values to CLOB or BLOB data.

It is possible to control the way LOB objects are handled while the application fetches records from the database. LOBs can be fetched either with other fields to the application or on demand. This is determined by DeferredLobRead option in TOraDataSet component. Setting TOraDataSet.Options.DeferredLobRead to false allows to reduce traffic over the network since LOBs are only transferred on demand and to use less memory on the client side because returned record sets do not hold contents of LOB fields.
For managing LOB compression, use `TCustomDADataset.Options.CompressBlobMode`. LOBs can be stored compressed on the client side, on the server side (in database) or on both sides. By default it has `cbNone` value, that means no compression is provided. Use `cbClient` value to store compressed LOBs on client side. This saves client memory. LOB data is stored unchanged in a database, other applications can read these LOBs as usual. If `cbServer` value is used, LOB data is stored compressed in the database. It's decompressed on the client side. This saves server disk space and network traffic. Other application can't process compressed LOB data as usual. To use compressed LOB data both on the client and server sides use `cbClientServer` value. To use `cbClient`, `cbServer`, `cbClientServer` and `cbNone` constants you should add the MemData unit to the uses clause.

Set `TODaDataset.Options.CacheLobs` to `False` to access streamed LOB values on the server side without caching LOBs on the client side. Only requested portions of data are fetched in that case. Setting `CacheBlobs` to `False` may bring up the following benefits for time-critical applications: reduced traffic over the network since only required data are fetched, less memory is needed on the client side because LOB data is not cached on client side. This option doesn't make sense if `DeferredLobRead` is set to `False` because in that case all LOB values are fetched to the dataset.

**Note:** Internal compression functions are available in Borland Developer Studio 2006, Delphi 2005 and Delphi 7. To use BLOB compression under Delphi 6 and C++Builder you should use your own compression functions. To use them set `CompressProc` and `UncompressProc` variables declared in MemUtils unit.

```pascal
type
  TCompressProc = function(dest: IntPtr; destLen: IntPtr; const source: IntPtr; sourceLen: longint): longint;
  TUncompressProc = function(dest: IntPtr; destLen: IntPtr; source: IntPtr; sourceLen: longint): longint;

var
  CompressProc: TCompressProc;
  UncompressProc: TUncompressProc;
```

You can compress and decompress a single LOB. To do it set the `TOraLob.Compressed` property. Set it to `True` to compress LOB data and to `False` to decompress LOB data.

Note that using compression and decompression operations will raise CPU usage and can reduce application performance.

**See Also**
- `TOraLob`
- `TDAParam.ParamType`
4.19.2 Unicode Character Data

Symbolic information in Oracle can be retrieved for the user as a different character encoding according to the query. Oracle supports a number of encoding formats including Unicode. ODAC components support UTF-16 Unicode encoding formats for data fields with OCI 8.0 or higher. Any character of any language can be represented in UTF-16.

ODAC allows to represent string data using string and WideString types. You can use `TOraSession.UseUnicode` property to enable this behaviour. This property value affects the description of queries and stored procedures. `TOraSession.UseUnicode` property does not influence the parameters' types that were set manually.

Suppose that `SIMPLE_TYPES` table is created as:

```sql
CREATE TABLE SIMPLE_TYPES (  ID NUMBER(6) NOT NULL,  F_CHAR CHAR(250),  F_VARCHAR VARCHAR2(300),  F_RAW RAW(250),)
```

Suppose we open following SELECT statement in dataset

```sql
SELECT a.RowId, a.* FROM SIMPLE_TYPES a
```

If `TOraSession.UseUnicode` is set to False you get the next fields list after dataset open:

- **RowId**: TStringField
- **ID**: TIntegerField
- **F_CHAR**: TStringField
- **F_VARCHAR**: TStringField
- **F_RAW**: TVarBytesField

When you set `TOraSession.UseUnicode` to True the string fields' type changes:

- **RowId**: TWideStringField
- **ID**: TIntegerField
- **F_CHAR**: TWideStringField
- **F_VARCHAR**: TWideStringField
- **F_RAW**: TWideStringField

Fields of TWideStringField type hold rows in UTF-16 Unicode format. To get the value of the
fields you can use TWideStringField.Value property. You can use FlatBuffers, LongString, FieldsAsString, RawAsString, TrimFixedChar options of TOraDataSet which are compatible with TOraDataSession.UseUnicode.

To use Unicode values as parameters previously you need to set the value of data type field to ftWideString or ftFixedWideChar for the fields of VARCHAR or CHAR types accordingly. Otherwise after the execution of AsWideString orAsString operation data type field will be ftString by default.

```pascal
var
  WS: WideString;
begin
...
  with OraQuery1 do begin
    Close;
    SQL.Text:=
      'SELECT * from SIMPLE_TYPES '
      'WHERE '
      '  F_CHAR = :F_CHAR';
    Params[0].DataType := ftFixedWideChar;
    Params[0].AsWideString := WS;
    Open;
...
```

If parameter has Unicode data type value, assigning value by using AsString property converts String to WideString. And vice versa, if parameter doesn't have Unicode data type value, assigning value by AsWideString property converts WideString into String.

Also Unicode encoding is supported for ROWID, NUMBER, INTERVAL, TIMESTAMP, RAW, CLOB.

CLOB data type supports string data in UTF-16 Unicode encoding. You can set TOraDataSession.UseUnicode property to True and get TMemoField of ftOraClob blob type. You can update CLOB field and set its value to Unicode string the following way:

```pascal
var
  WS: WideString;
begin
...
  with OraQuery1 do begin
    SQL.Text:=
      'UPDATE ODAC_CLOB '
      'SET '
      '  Value = EMPTY_CLOB() '
      'WHERE '
      '  ID = :ID '
      'RETURNING '
      '  Value '
      'INTO '
      '  :Value ';
    ParamByName('ID').AsFloat:=1;
```
ODAC allows you to query and update columns of Oracle object type. You can access to attributes of object column as to fields of dataset using TDataSet.FieldByName or TDataSet.Fields. Dataset represents object attributes in two ways depending on the value of ObjectView property. If ObjectView is True attributes are stored hierarchically in the Fields property, that means any attributes of the object column are represented by child field of the object field and don't appear sequentially after the object field in the TFields.Fields array. When ObjectView is False, the attributes are stored sequentially in the Fields property, that means any child fields of the object field are stored after the object field as siblings in the Fields array.

For example we have such types and table

```sql
CREATE TYPE TAddress AS OBJECT (
    Country VARCHAR2(30),
    City VARCHAR2(30),
    Street VARCHAR2(30),
    Apartment NUMBER
);
CREATE TYPE TPerson AS OBJECT (
    Name VARCHAR2(30),
    Address TAddress,
    Phone VARCHAR2(20),
    BirthDate DATE
);
CREATE TABLE ODAC_Emp (
    Person TPerson,
    Job VARCHAR2(9),
    HireDate DATE,
    Sal NUMBER(7,2)
);
```
If you execute this query

```sql
SELECT * FROM ODAC_Emp
```

to learn the name of an employee you can write

```pascal
Value := Query.FieldByName('PERSON.NAME').AsString;
```

If ObjectView is True object column is represented by TADTField and to access the object attribute use child field by TADTField.Fields

```pascal
Value := TADTField(Query.FieldByName('PERSON')).Fields.FieldByName('NAME').AsString;
```

Another way to get the value of an attribute is using TOraObject. Use TOraDataSet.GetObject method to get reference to the needed object.

So, the previous example may be rewritten this way

```pascal
Value := Query.GetObject('PERSON').AttrAsString['NAME'];
```

Also ODAC supports object parameters. Use this feature when writing statements to update dataset rows. So, to insert a new row in ODAC_Emp table it is enough to assign this statement to SQLInsert property.

```sql
INSERT INTO ODAC_Emp
  (PERSON, JOB, HIREDATE, SAL)
VALUES
  (:PERSON, :JOB, :HIREDATE, :SAL)
```

To execute this INSERT statement only by TOraQuery or TOraSQL component use TOraParam.AsObject property to set attributes' value of PERSON parameter. But before you should assign dtObject to TOraParam.DataType property and allocate object handle by TOraParam.AllocObject method.

```pascal
var
  OraSQL: TOraSQL;

  OraSQL.SQL.Text := 'INSERT INTO ODAC_Emp' +
    '(PERSON, JOB, HIREDATE, SAL)' +
    'VALUES (:PERSON, :JOB, :HIREDATE, :SAL)';

  with OraSQL.ParamByName('Person').AsObject do begin
    AllocObject(OraSession.OCISvcCtx, 'TPerson');
    AttrAsString['Name'] := 'JON';
    AttrAsString['Address.Country'] := 'USA';
    AttrAsString['Address.City'] := 'Boston';
    AttrAsInteger['Address.Apartment'] := 133;
    AttrAsDateTime['BirthDate'] := EncodeDate(1970, 7, 23);
  end;

  OraSQL.ParamByName('Job').AsString := 'MANAGER';
  OraSQL.ParamByName('HireDate').AsDateTime := EncodeDate(1998, 5, 14);
  OraSQL.ParamByName('Sal').AsInteger := 1700;

  OraSQL.Execute;
```
4.19.4 XMLTYPE Data Type

Oracle 9i introduced a new data type, XMLTYPE, to facilitate native handling of XML data in the database. XMLTYPE has built-in member functions that operate on XML content. For example, you can use XMLTYPE functions to create, extract, and index XML data stored in Oracle 9i database. XMLTYPE data type can be used as the data type of columns in tables and views. XMLTYPE data type uses the following storage methods:

- Large objects (LOBs). LOB storage maintains content accuracy to the original XML (white spaces and all). Here the XML documents are stored composed as whole documents like files. XML stored as a whole document in the database and retrieve it as a whole document. You do not need to perform piece-wise updates on XML documents.

ODAC can work with fields of XMLTYPE data type. For example, suppose we have the following table with XMLTYPE field:

```sql
CREATE TABLE xml_tab(
    ID NUMBER(10),
    XMLField XMLTYPE
);
```

After creating a TOraQuery object and executing the next SELECT statement:

```sql
SELECT * FROM xml_tab
```

TOraXMLField object will be created for XMLTYPE type which holds retrieved XML document. XMLTYPE fields are cached in ODAC on opening a table. Update of XMLTYPE fields will be posted to the server after the execution of corresponding DML query.

You can update or append XMLTYPE fields to the table the following way:

```sql
Edit; // or Insert, Append;
TOraXMLField(FieldByName('X')).AsXML.AsString := '<root> <node1>v1</node1> <node2 name1=''222''>v2</node2> <node1>v3</node1> </root>'; Post;
```

Or you can write:
You can use `TOraXMLField.AsXML` property to get XMLTYPE document as an `TOraXML` object. Using this object you can query XMLTYPE data and extract its portions by calling `TOraXML.Exists` and `TOraXML.Extract` functions. Both these functions use a subset of the W3C XPath recommendation to navigate through the document. XMLTYPE uses the built-in Oracle XML parser and processor, and that's why it provides better performance and scalability when used inside the server. `TOraXML.Transform` function takes in XMLTYPE instance and XSLT stylesheet. It applies the stylesheet to the XML document and returns a transformed XML instance.

You can treat the XMLTYPE as a parameter in DML statements as shown below.

```
Close;
SQL.Text := 'UPDATE xml_tab SET XMLField = :XMLField WHERE ID = 101';
with Params[0].AsXML do begin
  OCISvcCtx := OraSession1.OCISvcCtx;
  AsString := '<test></test>';  
end;
Execute;
```

Or you can create the XMLTYPE value from `TOraLob`.

```
with Params[0].AsXML do begin
  OraLob := TOraLob.Create(OraSession1.OCISvcCtx);
  try
    OraLob.CreateTemporary(ltClob);
    OraLob.Write(0, Length('<test_lob></test_lob>'), PChar('<test_lob></test_lob>'), 1);
    OraLob.WriteByte(0);
    AllocObject(OraSession1.OCISvcCtx, OraLob);
  finally
    OraLob.Free;
  end;
end;
Execute;
```

XML Schema is a schema definition language written in XML. It can be used to describe the structure and other various semantics of conforming instance documents. When using Oracle XML DB, you must first register your XML schema. Than you can use the XML schema URLs while creating XMLTYPE tables, columns, and views. You can use XML schema to declare which elements and attributes can be used and what kinds of element nesting, and data types are allowed in the XML documents that are being stored or processed. For example, user can create XML document based on the following schema, create table with XMLTYPE field and initialize the field value.

```
declare
doc varchar2(1000) :=
```
ODAC supports schema-based XML documents processing. You can open TOraQuery and get TOraXML object with schema-based document the same way as for LOB-based XMLTYPE. You can get value document calling TOraXMLField.AsString or TOraXML.AsString. Extract, Exists, Transform, GetSchema, Validate, and IsSchemaBased functions of TOraXML are available for this XML document type.

```pascal
with OraQuery1 do begin
  RetDoc := TOraXML.Create();
  RetDoc.OCISvcCtx := OraSession1.OCISvcCtx;
  try
    with TOraXMLField(FieldByName('XMLTYPE')).AsXML do begin
      GetSchema(RetDoc, SchemaURL, RootElem);
      Str := RetDoc.AsString;
    end;
  finally
    // Clean up
  end;
end;
```
4.19.5 VARRAY Data Type

Everything considered in Working with objects is right for Arrays. Some problems appear when you need to use large arrays in dataset. As ODAC creates one field for each array item great number of TField objects are created. As a result the performance decreases. So ODAC has the limitation and creates fields for first 1000 items. However you can access all items with TOraArray object. Another way is to set TOraQuery.SparseArray to True and access array items by TArrayField object.

If such types are created

CREATE TYPE TODACArray1 AS VARRAY (5) OF NUMBER;
CREATE TYPE TODACArray2 AS VARRAY (4) OF CHAR(10);
CREATE TABLE ODAC_Array (  Code NUMBER,  Title VARCHAR2(10),  Arr1 TODACArray1,  Arr2 TODACArray2,);

To access array items you can call FieldByName method. For example

Value := Query.FieldByName('Arr1[0]').AsInteger;
If ObjectField property is True this code is correct

Value := TArrayField(Query.FieldByName('Arr1')).Fields[0].AsInteger;
Using TOraDataSet.GetArray you can access array items through TOraArray object

Value := Query.GetArray('Arr1').ItemAsInteger[0];

You can use VARRAY type for parameters of SQL and PL/SQL statements. You need to assign dtArray to TOraParam.DataType and use TOraParam.AsArray property to access array items.

For example:
Using of array binding feature can greatly speed up the execution of the application on insert or update big volume of data. The main advantage is that array binding allows you to execute several INSERT SQL statements with the different parameters at once. Note that you access server only once - that increases speed of update a lot.

The following is a sample of using DML Array.

The following table is used in this sample.

CREATE TABLE dept
(
    deptno NUMBER(2) CONSTRAINT pk_dept PRIMARY KEY,
    dname VARCHAR2(14),
    loc VARCHAR2(13)
);

At first, you should open a session:

OraSession.UserName := 'scott';
OraSession.Password := 'tiger';
OraSession.Server := 'Ora';
OraSession.Connect;
After that you should specify SQL statement for the execution:

```sql
OraSQL.SQL.Text := 'INSERT INTO dept VALUES(:deptno_p, :dname_p, :loc_p)';
```

Colons in the SQL text mean parameters with the values which will be specified later.

Now you should specify parameter type for each parameter from the SQL text and the `Length` property of the parameters, which should be equal to the number of SQL statement executions.

```pascal
OraSQL.ParamByName('deptno_p').DataType := ftInteger;
OraSQL.ParamByName('deptno_p').Length := 4;

OraSQL.ParamByName('dname_p').DataType := ftString;
OraSQL.ParamByName('dname_p').Length := 4;

OraSQL.ParamByName('loc_p').DataType := ftString;
OraSQL.ParamByName('loc_p').Length := 4;
```

You should call Prepare method before specifying values for the highest efficiency.

```pascal
OraSQL.Prepare;
```

Each item of the array must correspond to the single execution of the SQL statement.

```pascal
OraSQL.ParamByName('deptno_p').ItemAsInteger[1] := 10;
OraSQL.ParamByName('dname_p').ItemAsString[1] := 'ACCOUNTING';
OraSQL.ParamByName('loc_p').ItemAsString[1] := 'NEW YORK';

OraSQL.ParamByName('deptno_p').ItemAsInteger[2] := 20;
OraSQL.ParamByName('dname_p').ItemAsString[2] := 'RESEARCH';
OraSQL.ParamByName('loc_p').ItemAsString[2] := 'DALLAS';

OraSQL.ParamByName('deptno_p').ItemAsInteger[3] := 30;
OraSQL.ParamByName('dname_p').ItemAsString[3] := 'SALES';
OraSQL.ParamByName('loc_p').ItemAsString[3] := 'CHICAGO';

OraSQL.ParamByName('deptno_p').ItemAsInteger[4] := 40;
OraSQL.ParamByName('dname_p').ItemAsString[4] := 'OPERATIONS';
OraSQL.ParamByName('loc_p').ItemAsString[4] := 'BOSTON';
```

After accomplishing previous steps you should call Execute method that assumes a parameter specifying how many times SQL statement will be executed. Note that the value of this method argument must be equal to the number of parameters value elements. Now you can execute `SELECT * FROM Dept` with any Oracle tool (you can use dbForge Studio for Oracle for this purpose) and see four new records appended.

```pascal
OraSQL.Execute(4);
```
4.19.7 Cursors

Since Oracle 7.3 the REF CURSOR type has been available to allow recordsets to be returned from stored procedures and functions.

Oracle 9i introduced the predefined SYS_REFCURSOR type, meaning we no longer have to define our own REF CURSOR types.

Using Ref Cursors

The example below uses a ref cursor to return a subset of records in the EMP table.

The following procedure opens a query using a SYS_REFCURSOR output parameter. Notice the cursor is not closed in the procedure. It is up to the calling code to manage the cursor once it has been opened.

```sql
CREATE OR REPLACE PROCEDURE sp_get_emp (p_deptno IN emp.deptno%TYPE, p_recordset OUT SYS_REFCURSOR) AS
BEGIN
  OPEN p_recordset FOR
    SELECT ename, empno, deptno
    FROM emp
    WHERE deptno = p_deptno
    ORDER BY ename;
END sp_get_emp;
/
```

Using cursors in ODAC

In ODAC work with cursors may be implemented using the following components:

- `TOraQuery`
- `TOraStoredProc`
- `TOraSQL`

Below is a sample working with cursors in `TOraStoredProc`:

**Delphi**

```delphi
program Project1;
{$APPTYPE CONSOLE}
{$R *.res}
uses  System.SysUtils, Ora;
var  OraSession: TOraSession;
    OraStoredProc: TOraStoredProc;
begin
  OraSession := TOraSession.Create(nil);
  try
    OraSession.ConnectString := 'scott/tiger@orc';
    OraSession.Connect;
    WriteLn('Использование TOraStoredProc');
  except
    on E: Exception do
      WriteLn(E.Message);
  end;
end.
```
OraStoredProc := TOra StoredProc.Create(nil);
  try
    OraStoredProc.Session := OraSession;
    OraStoredProc.StoredProcName := 'sp_get_emp';
    OraStoredProc.Prepare;
    OraStoredProc.ParamByName('p_deptno').AsInteger := 10;
    OraStoredProc.Execute;
    while not OraStoredProc.Eof do begin
      Writeln(OraStoredProc.FieldByName('ename').AsString);
      OraStoredProc.Next;
    end;
  finally
    OraStoredProc.Free;
  end;
  finally
    OraSession.Free;
    readln;
  end;
end.

C++Builder

#include <vcl.h>
#pragma hdrstop
#include <tchar.h>
#include <stdio.h>
#include <Ora.hpp>
#pragma argsused
int _tmain(int argc, _TCHAR* argv[])
{
  TORaSession *OraSession = new TORaSession(NULL);
  try
  {
    OraSession->ConnectString = "scott/tiger@orcl1120";
    OraSession->Connect();
    TORaStoredProc *OraStoredProc = new TORaStoredProc(NULL);
    try
    {
      OraStoredProc->StoredProcedureName = "sp_get_emp";
      OraStoredProc->Prepare();
      OraStoredProc->ParamByName("p_deptno") -> AsInteger = 10;
      OraStoredProc->Session = OraSession;
      OraStoredProc->Execute();
      while (!OraStoredProc->Eof)
      {
        printf("%s\n", OraStoredProc->FieldByName("ename") ->AsString . t_str());
      }
    }
    finally
    {  
      OraStoredProc->Free();
    }
  }
  finally
  {  
    OraSession->Free();
  }
```
If several output parameters in the procedure are cursors, then TOraStoredProc will work only with the first one as with a DataSet. To retrieve other DataSets, the asCursor method must be used:

```plsql
OraQuery.Cursor := OraSession.ParamByName('Cur2').AsCursor;
OraQuery.Open;
```

4.19.8 PL/SQL Tables

ODAC allows you to use PL/SQL arrays known as PL/SQL Tables as parameters of anonymous PL/SQL blocks or as parameters of stored procedures. As ordinary arrays, PL/SQL arrays can be used for storing the same data accessible by index.

We will use standard Dept table in our sample. If you don't have this table at your database see SQL script at Demos\InstallDemoObjects.sql folder. Following sample demonstrates how to update several records from Dept table simultaneously using parameter of PL/SQL Table type.

Here is a PL/SQL block used in our sample:

```plsql
DECLARE
    i INTEGER;
BEGIN
    i := 1;
    FOR rec IN (SELECT DeptNo FROM Scott.Dept
                 WHERE RowNum <= 10 ORDER BY DeptNo)
    LOOP
        UPDATE Scott.Dept
        SET DName = :NameArr(i)
        WHERE DeptNo = Rec.DeptNo;
        i := i + 1;
    END LOOP;
END;
```

There is one parameter in the text of the sample PL/SQL block with NameArr name. It has PL/SQL Table type. This SQL updates DName field of Dept table with the values from NameArr array.

At first, you should open a session:

```plsql
OraSession.UserName := 'scott';
OraSession.Password := 'tiger';
OraSession.Server := 'Ora';
```
OraSession.Connect;

After that you should specify SQL statement for the execution:

```sql
OraSQL.SQL.Add('DECLARE');
OraSQL.SQL.Add('  i INTEGER;');
OraSQL.SQL.Add('BEGIN');
OraSQL.SQL.Add('  i:= 1;');
OraSQL.SQL.Add('  FOR rec IN (SELECT DeptNo FROM Scott.Dept WHERE RowNum <= 10 ORDER BY DeptNo)');
OraSQL.SQL.Add('    i:= i + 1;');
OraSQL.SQL.Add('  END LOOP;');
OraSQL.SQL.Add('END;');
```

The NameArr parameter value should be specified later.

Then you need to specify that the parameter with NameArr name has PL/SQL Table type. To do it, you should set `Table` property to True and `Length` property of the parameter to the required value. If Dept table has four records, the size of the array also must be four.

```sql
OraSQL.ParamByName('NameArr').Table := True;
OraSQL.ParamByName('NameArr').DataType := ftString;
OraSQL.ParamByName('NameArr').Length := 4;
```

After that you need to set values for the array items of NameArr parameter. The amount of array items must be equal to the value of `Length` property.

```sql
OraSQL.ParamByName('NameArr').ItemAsString[1] := 'London';
OraSQL.ParamByName('NameArr').ItemAsString[2] := 'Berlin';
OraSQL.ParamByName('NameArr').ItemAsString[3] := 'Geneva';
OraSQL.ParamByName('NameArr').ItemAsString[4] := 'Vienna';
```

Now you can execute SQL by calling `Execute` method of `TOraSQL` component.

```sql
OraSQL.Execute;
```

### 4.20 Writing Oracle External Procedures with ODAC

External procedure is a procedure stored in a dynamic link library (DLL), or libunit in the case of a Java class method. Different programming languages can be used for external procedures creation - C, C++, Object Pascal, Java. External procedure can be called directly from PL/SQL and SQL. You can use ODAC components for writing external procedures for Oracle database. A small example of external procedure using ODAC components is listed below.
For example, let's create an external procedure that saves LOB to file and stores the file name and the file date in a database. Suppose we have the following table to store file names and dates:

```
CREATE TABLE scott.odac_file_list
(
    id integer PRIMARY KEY,
    file_name VARCHAR2(100),
    file_date TIMESTAMP
)
```

Let's create a DLL ExtProc containing our external procedure add_file.

All external procedures and functions in DLL must be listed in the library exports clause. Before calling any OCI functions in DLL InitOCI procedure must be called. When OCI is no longer needed FreeOCI procedure must be called.

In declaration of procedure add_file cdecl directive must be used. It is necessary to call OCIExtProcGetEnv function, that returns environment, service context and error handles. A call to OCIExtProcGetEnv function is required to make OCI callbacks to database.

Then we will create a data module with TOraSession and TOraQuery components. TOraSession component can be linked to external procedure service context by assigning service context pointer to OCISvcCtx property of TOraSession. After such assignment we can execute queries through OraSession.

An external procedure must not raise Delphi exceptions. All these exceptions must be processed inside the procedure and procedure can raise PL/SQL exceptions with OCIExtProcRaiseExcpWithMsg OCI function.

The source code of DLL, that contains add_file procedure is the following:

```pascal
library ExtProc;
uses
    SysUtils,
    Classes,
    DB, Ora, OraCall, OraError, OraClasses,
    Data in 'Data.pas' {dmData: TDataModule};
{$R *.res}
procedure add_file(Context: pOCIEExtProcContext; Id: pOCINumber;
    FileName: PChar; FileDate: pOCIDateTime; FileText: pOCILobLocator); cdecl;
var
    hEnv: pOCIEEnv;
    hSvcCtx: pOCISvcCtx;
    hError: pOCIERror;
    dmData: TdmData;
    OraLob: TOraLob;
begin
    try
        // get OCI service context
```
Check(OCIExtProcGetEnv(Context, hEnv, hSvcCtx, hError));
dmData := TdmData.Create(nil);
try
  // set service context handle in OraSession
  dmData.OraSession.OCISvcCtx := hSvcCtx;
  with dmData.OraSQL do begin
    ParamByName('ID').DataType := TFieldType(ftNumber);
    ParamByName('ID').AsNumber.OCINumberPtr := Id;
    ParamByName('FILE_NAME').AsString := FileName;
    ParamByName('FILE_DATE').DataType := ftTimeStamp;
    ParamByName('FILE_DATE').AsTimeStamps.OCIDateTime := FileDate;
    Execute;
    OraLob := TOraLob.Create(hSvcCtx);
    try
      OraLob.OCILobLocator := FileText;
      OraLob.ReadLob;
      OraLob.SaveToFile(FileName);
    finally
      OraLob.Free;
    end;
  end;
finally
  dmData.Free;
end;
except
  on e: EораError do
    OCIExtProcRaiseExcpWithMsg(Context, e.ErrorCode, PChar(e.Message), Length(e.Message));
  on e: Exception do
    OCIExtProcRaiseExcpWithMsg(Context, 20000, PChar(e.Message), Length(e.Message));
end;
exports
  add_file;
begin
  // Load oci.dll and link OCI functions
  InitOCI;
end.

To use this external procedure compile the DLL and copy it to Oracle server. The DLL must be copied to ORACLE_HOME\bin (Windows) or ORACLE_HOME/lib (UNIX). See Oracle documentation about making the external procedures agent load external procedure libraries from other paths.

External procedures DLL must be defined with CREATE LIBRARY statement. In our ExternalProc Demo the library is created as follows:

```
CREATE OR REPLACE LIBRARY Scott.ExtProcDemo AS 'C:\oracle\product\10.2.0\db_1\bin\ExtProc.dll'
```

Note: the path to the DLL passed to CREATE LIBRARY statement is case sensitive.

Then we define the external procedure:

```
CREATE PROCEDURE scott.add_file(   id NUMBER,
```

file_name VARCHAR2,
file_date TIMESTAMP,
file_text CLOB
)
AS LANGUAGE C
NAME "add_file"
LIBRARY scott.ExtProcDemo
WITH CONTEXT
PARAMETERS (CONTEXT, id OCINUMBER, file_name STRING, file_date OCIDATETIME

The "LANGUAGE C" option shows that it is an external procedure written in the language compatible with the C language call specification. The "NAME" option is the name of the procedure in the DLL. "WITH CONTEXT" option enables OCI callbacks to the database during an external procedure execution. That means an additional CONTEXT parameter is passed to the procedure. It allows the procedure to use a connection to the database.

Now the add_file procedure can be called from an SQL query.

Note to execute external procedures Oracle Net files listener.ora and tnsnames.ora must be configured for external procedures. See Oracle documentation about the configuration Oracle net for external procedures.

SeeAlso
- ExternalProc demo

© 1997-2021 Devart. All Rights Reserved.

4.21 Transparent Application Failover Support

Transparent application failover (TAF) is the ability of applications to automatically reconnect to the database if the connection fails. If the server fails, the connection also fails. The next time the client tries to use the connection to execute a new SQL statement, for example, the operating system displays an error to the client. At this point, the user must log in to the database again. With TAF, however, Oracle automatically obtains a new connection to the database. This allows the user to continue to work using the new connection as if the original connection had never failed. If the client is not involved in a database transaction, then users may not notice the failure of the server. Because this reconnection happens automatically, the client application code may not need changes to use TAF. TAF automatically restores:

- Client-Server Database Connections;
- Users' Database Sessions;
- Executing Commands;
- Open Cursors Used for Fetching;
• Active Transactions;
• Server-Side Program Variables.

Unfortunately, TAF cannot automatically restore some session properties. If the application issued ALTER SESSION commands, then the application must reissue them after TAF processing is complete.

Frequently failure of one instance and failover to another takes time. Because of this delay, you may want to inform users that failover is in progress. Additionally, the session on the initial instance may have received some ALTER SESSION commands. These will not be automatically reissued on the second instance. You may need to reissue these commands on the second instance.

To address such problems, you can use TOraSession.OnFailover event. The event is raised during the session recovery process when connection is lost. When connection failure is detected TOnFailover event is raised first time. Then application keeps raising it until connection is restored or user stops failover process.

Transparent Application Failover Restrictions:
• All PL/SQL package states on the server are lost at failover.
• ALTER SESSION statements are lost.
• If failover occurs when a transaction is in process, then each subsequent call causes an error message until the user issues Rollback call. Then a success message is issued. Be sure to check this informational message to see if you must perform any additional operations.
• Continuing work on failed over cursors may cause an error message.
• If the first command after failover is not a SELECT statement or fetch operation, an error message results.
• Failover only takes effect for Oracle 8.0 or higher.
• At failover time, any queries in progress are reissued and processed again from the beginning. This may result in the next query taking a long time if the original query took a long time.

Preparing and Running the Sample

The tnsnames.ora file should be suitably modified for your database entry so that TAF tries to reconnect when the database connection is lost. The tnsnames.ora file is located in the <Oracle_Home>/network/Admin directory. Your database TNS entry should look like this:

```sql
<DBFAILOVER.US.ORACLE.COM> =
  (DESCRIPTION =
    (ADDRESS_LIST =
```

© 2021 Devart
where `<Oracle_Home>` is the directory where your database or SQL*Plus client is installed. Replace the values for the database parameters highlighted in bold with your database parameters.

Build and run the Query project from the ODAC demos. (Please ensure to perform the following steps). Set `TOrasession.Options.UseOCI7` to False. Write `TOrasession.OnFailover` event as follows.

```pascal
procedure TfmMain.OraSessionFailover(Sender: TObject;  
  FailoverState: TFailoverState; FailoverType: TFailoverType;  
  var Retry: Boolean); 
begin 
  case FailoverState of 
    fsBegin: begin 
      ShowMessage('Failover Begin'); 
      StatusBar1.Panels[0].Text := 'Trying to reconnect, Please wait...'; 
      end; 
    fsAbort: begin 
      ShowMessage('Failover Aborted'); 
      StatusBar1.Panels[0].Text := 'Failover Aborted'; 
      end; 
    fsEnd: 
      ShowMessage('Failover End'); 
    fsAbort: begin 
      StatusBar1.Panels[0].Text := 'Failover Error. Retrying to connect ' + 
        'to database. Please wait... '; 
      Retry:=true; 
    end; 
    fsReauth: begin 
      ShowMessage('Failover reauthenticating'); 
      StatusBar1.Panels[0].Text := 'Failover reauthenticating'; 
    end; 
    else 
      ShowMessage('Bad Failover'); 
      StatusBar1.Panels[0].Text := 'Bad Failover'; 
  end; 
end; 
```

When run, the sample shows a form with a blank data grid. The user should click "Open" button to start fetching the Dept table records.

For demonstrating TAF, user should restart the database from SQL*Plus using following command:

- To login as a DBA user type
SQL> Connect sys/<your sys_password>@<Your_TNSName> as sysdba
• To shutdown and restart database type

SQL> startup force

After restarting the database, the user should return to the application and refresh the data by clicking "RefreshRecords". The Failover event is called and the Failover handler method displays the appropriate messages in a message box and in the status bar of application. The query will be executed again against the database using a new connection, data fetched and displayed in the data grid.

See Also
• TOraSession.OnFailover

4.22 DBMonitor

To extend monitoring capabilities of ODAC applications there is an additional tool called DBMonitor. It is provided as an alternative to Borland SQL Monitor which is also supported by ODAC.

DBMonitor is an easy-to-use tool to provide visual monitoring of your database applications.

DBMonitor has the following features:
• multiple client processes tracing;
• SQL event filtering (by sender objects);
• SQL parameter and error tracing.

DBMonitor is intended to hamper an application being monitored as little as possible.

To trace your application with DB Monitor you should follow these steps:
• drop TOraSQLMonitor component onto the form;
• turn moDBMonitor option on;
• set to True the Debug property for components you want to trace;
• start DBMonitor before running your program.
4.23 Writing GUI Applications with ODAC

Since version 3.80 ODAC GUI part is standalone. This means that to make GUI elements such as SQL cursors, connect form, connect dialog etc. available, you should explicitly include OdacVcl unit in your application. This feature is needed for writing console applications.

*Delphi and C++Builder*

By default ODAC does not require Forms, Controls and other GUI related units. Only TConnectDialog, TOraErrorHandler and TOrAlerter components require the Forms unit.

4.24 Compatibility with Previous Versions

We always try to keep ODAC compatible with previous versions, but sometimes we have to change the behaviour of ODAC in order to enhance its functionality, or avoid bugs. This topic describes such changes, and how to revert the old ODAC behaviour. We strongly recommend not to turn on the old behaviour of ODAC. Use options described below only if changes applied to ODAC crashed your existent application.

Values of the options described below should be assigned in the initialization section of one of the units in your project.

**DBAccess.BaseSQLOldBehavior:**

The BaseSQL property is similar to the SQL property, but it does not store changes made by AddWhere, DeleteWhere, and SetOrderBy methods. After assigning an SQL text and modifying it by one of these methods, all subsequent changes of the SQL property will not be reflected in the BaseSQL property. This behavior was changed in ODAC 5.55.1.26. To restore old behavior, set the BaseSQLOldBehavior variable to True.

**DBAccess.SQLGeneratorCompatibility:**

If the manually assigned RefreshSQL property contains only "WHERE" clause, ODAC uses the value of the BaseSQL property to complete the refresh SQL statement. In this situation all modifications applied to the SELECT query by functions AddWhere, DeleteWhere are not taken into account. This behavior was changed in ODAC 6.00.0.4. To restore the old behavior, set the BaseSQLOldBehavior variable to True.
MemDS.SendDataSetChangeEventAfterOpen:
Starting with ODAC 6.20.0.11, the DataSetChange event is sent after the dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids. This problem appears only under Windows XP when visual styles are enabled.
To disable sending this event, change the value of this variable to False.

MemDS.DoNotRaiseExceptionOnUaFail:
Starting with ODAC 6.20.0.12, if the OnUpdateRecord event handler sets the UpdateAction parameter to uaFail, an exception is raised. The default value of UpdateAction is uaFail. So, the exception will be raised when the value of this parameter is left unchanged.
To restore the old behaviour, set DoNotRaiseExceptionOnUaFail to True.

Ora.OraQueryCompatibilityMode:
Before ODAC 6, TOraQuery could be editable only when InsertSQL, UpdateSQL, and DeleteSQL properties are assigned. The ability to generate update SQL statements with TOraQuery automatically was added in ODAC 6.00.0.4. Therefore, after upgrading your ODAC to the sixth version, all TOraQuery components in your project become editable, and can be modified by the end users. To restore the old behavior, set the OraQueryCompatibilityMode variable to True.

4.25 Oracle Package Wizard

Oracle Package Wizard is designed for creating wrapper classes for PL/SQL Packages. It greatly simplifies working with types and stored procedures containing in PL/SQL Packages.

Oracle Package Wizard supports:
- All native Oracle types.
- PL/SQL tables of any simple data type except boolean.
- PL/SQL records, including nested records.

To create a wrapper class, perform the following steps:
1. Run Oracle Package Wizard from the ODAC menu.
2. Assign properties to connect to your Oracle server.
3. Choose the packages you want to be wrapped.
Note that items with unsupported parameter types cannot be selected. They are grayed out.

4. Select code generation options:

Parameter type and method conventions

*Use Numbers* - when this option is checked, Wizard maps Oracle numbers with the precision larger than 15 to `ftNumber`. Otherwise, they are mapped to `ftFloat`.

*Use Integers* - when this option is enabled, Wizard maps Oracle numbers with the precision less than 10 to `ftInteger`. Otherwise, they are mapped to `ftFloat` or `ftNumber`.

*Use TimeStamps* - when this option is enabled, Wizard maps Oracle timestamps to `ftTimeStamp`, `ftTimeStampTZ`, or `ftTimeStampLTZ`. Otherwise, timestamps are mapped to `ftDateTime`.

*Use DataSets* - when this option is enabled, Wizard uses `TOraDataSet` parameters to return Oracle cursors. Otherwise `TOraCursor` parameters are used.

*Use Unicode* - when this option is enabled, Wizard creates fields of the `ftWideString` data type. Otherwise, `ftString` is used.

*Use variants as parameters* - when this option is enabled, variants are used for all simple parameter types.

*Generate overloaded methods* - when this option is enabled, overloaded methods are created. Otherwise, overloaded subprograms are mapped to the methods with different suffixes (1, 2, 3 and so on).

Identifier generation rules

*Unchangedcase, CapitalizedCase, lowercase, UPPERCASE* - these alternative options define character case in identifier names.

*Remove underscores* - when this option is enabled, Wizard removes underscores from generated identifiers.

*Prefix objects with T* - when this option is enabled, generated class names are prefixed with 'T'.

*Prefix parameters with A* - when this option is enabled, method parameters are prefixed with 'A'.

Target environment

*Generate code for all versions of Delphi* - when this option is enabled, generated code is compatible with the following Delphi versions: Delphi 6, Delphi 7, Borland Developer Studio 2006, CodeGear Delphi 2007 for Win32. Otherwise, generated code will work surely only in
the current version of Delphi.

*Generated code for* - select Win32, CLR or Both to determine environments that generated code will be compatible with.

5. Define files to be generated:
   - Select the target directory.
   - Specify the unit name.
   - Enable the "Add to project" option if you want to add the generated unit to the current project.
   - Enable the "Generate as components" option if you want to generate components registration code.
   - Choice the Component palette tab name that will be used for generating components registration code.
   - Enable the "Generate resources" option to generate resource files (*.res files for Win32 and *.bmp files for CLR). In case of a CLR code generation, you must specify the Images subdirectory name.

Press the Generate button to generate classes for selected packages.

4.26 64-bit Development with Embarcadero RAD Studio XE2

**RAD Studio XE2 Overview**

RAD Studio XE2 is the major breakthrough in the line of all Delphi versions of this product. It allows deploying your applications both on Windows and Mac OS platforms. Additionally, it is now possible to create 64-bit Windows applications to fully benefit from the power of new hardware. Moreover, you can create visually spectacular applications with the help of the FireMonkey GPU application platform.

Its main features are the following:
- Windows 64-bit platform support;
- Mac OS support;
- FireMonkey application development platform;
- Live data bindings with visual components;
- VCL styles for Windows applications.

**Changes in 64-bit Application Development**
64-bit platform support implies several important changes that each developer must keep in mind prior to the development of a new application or the modernization of an old one.

General

RAD Studio XE2 IDE is a 32-bit application. It means that it cannot load 64-bit packages at design-time. So, all design-time packages in RAD Studio XE2 IDE are 32-bit.

Therefore, if you develop your own components, you should remember that for the purpose of developing components with the 64-bit platform support, you have to compile run-time packages both for the 32- and 64-bit platforms, while design-time packages need to be compiled only for the 32-bit platform. This might be a source of difficulties if your package is simultaneously both a run-time and a design-time package, as it is more than likely that this package won't be compiled for the 64-bit platform. In this case, you will have to separate your package into two packages, one of which will be used as run-time only, and the other as design-time only.

For the same reason, if your design-time packages require that certain DLLs be loaded, you should remember that design-time packages can be only 32-bit and that is why they can load only 32-bit versions of these DLLs, while at run-time 64-bit versions of the DLLs will be loaded. Correspondingly, if there are only 64-bit versions of the DLL on your computer, you won't be able to use all functions at design-time and, vice versa, if you have only 32-bit versions of the DLLs, your application won't be able to work at run-time.

Extended type

For this type in a 64-bit applications compiler generates SSE2 instructions instead of FPU, and that greatly improves performance in applications that use this type a lot (where data accuracy is needed). For this purpose, the size and precision of Extended type is reduced:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>32-bit</th>
<th>64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended</td>
<td>10 bytes</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>

The following two additional types are introduced to ensure compatibility in the process of developing 32- and 64-bit applications:

Extended80 – whose size in 32-bit application is 10 bytes; however, this type provides the same precision as its 8-byte equivalent in 64-bit applications.

Extended80Rec – can be used to perform low-level operations on an extended precision floating-point value. For example, the sign, the exponent, and the mantissa can be changed separately. It enables you to perform memory-related operations with 10-bit floating-point
variables, but not extended-precision arithmetic operations.

**Pointer and Integers**

The major difference between 32- and 64-bit platforms is the volume of the used memory and, correspondingly, the size of the pointer that is used to address large memory volumes.

<table>
<thead>
<tr>
<th>TYPE</th>
<th>32-bit</th>
<th>64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pointer</td>
<td>4 bytes</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>

At the same time, the size of the Integer type remains the same for both platforms:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>32-bit</th>
<th>64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>

That is why, the following code will work incorrectly on the 64-bit platform:

```pascal
Ptr := Pointer(Integer(Ptr) + Offset);
```

While this code will correctly on the 64-bit platform and incorrectly on the 32-bit platform:

```pascal
Ptr := Pointer(Int64(Ptr) + Offset);
```

For this purpose, the following platform-dependent integer type is introduced:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>32-bit</th>
<th>64-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>NativeInt</td>
<td>4 bytes</td>
<td>8 bytes</td>
</tr>
<tr>
<td>NativeUInt</td>
<td>4 bytes</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>

This type helps ensure that pointers work correctly both for the 32- and 64-bit platforms:

```pascal
Ptr := Pointer(NativeInt(Ptr) + Offset);
```

However, you need to be extra-careful when developing applications for several versions of Delphi, in which case you should remember that in the previous versions of Delphi the NativeInt type had different sizes:

<table>
<thead>
<tr>
<th>TYPE</th>
<th>Delphi Version</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>NativeInt</td>
<td>D5</td>
<td>N/A</td>
</tr>
<tr>
<td>NativeInt</td>
<td>D6</td>
<td>N/A</td>
</tr>
<tr>
<td>NativeInt</td>
<td>D7</td>
<td>8 bytes</td>
</tr>
<tr>
<td>NativeInt</td>
<td>D2005</td>
<td>8 bytes</td>
</tr>
<tr>
<td>NativeInt</td>
<td>D2006</td>
<td>8 bytes</td>
</tr>
<tr>
<td>NativeInt</td>
<td>D2007</td>
<td>8 bytes</td>
</tr>
</tbody>
</table>
Out parameters

Some WinAPIs have OUT parameters of the SIZE_T type, which is equivalent to NativeInt in Delphi XE2. The problem is that if you are developing only a 32-bit application, you won't be able to pass Integer to OUT, while in a 64-bit application, you will not be able to pass Int64; in both cases you will have to pass NativeInt.

For example:

```delphi
procedure MyProc(out Value: NativeInt);
begin
  Value := 12345;
end;
var
  Value1: NativeInt;
{$IFDEF WIN32}
  Value2: Integer;
{$ENDIF}
{$IFDEF WIN64}
  Value2: Int64;
{$ENDIF}
begin
  MyProc(Value1); // will be compiled;
  MyProc(Value2); // will not be compiled !!!
end;
```

Win API

If you pass pointers to SendMessage/PostMessage/TControl.Perform, the wParam and lParam parameters should be type-casted to the WPARAM/LPARAM type and not to Integer/Longint.

Correct:

```delphi
SendMessage(hWnd, WM_SETTEXT, 0, LPARAM(@MyCharArray));
```

Wrong:

```delphi
SendMessage(hWnd, WM_SETTEXT, 0, Integer(@MyCharArray));
```

Replace SetWindowLong/GetWindowLog with SetWindowLongPtr/GetWindowLongPtr for GWLP_HINSTANCE, GWLP_ID, GWLP_USERDATA, GWLP_HWNDPARENT and GWLP_WNDPROC as they return pointers and handles. Pointers that are passed to SetWindowLongPtr should be type-casted to LONG_PTR and not to Integer/Longint.
Correct:

\[
\text{SetWindowLongPtr}(\text{hWnd}, \text{GWLP_WNDPROC}, \text{LONG_PTR}(\text{@MyWindowProc}));
\]

Wrong:

\[
\text{SetWindowLong}(\text{hWnd}, \text{GWL_WNDPROC}, \text{Longint}(\text{@MyWindowProc}));
\]

Pointers that are assigned to the TMessage.Result field should use a type-cast to LRESULT instead of Integer/Longint.

Correct:

\[
\text{Message.Result} := \text{LRESULT}(\text{Self});
\]

Wrong:

\[
\text{Message.Result} := \text{Integer}(\text{Self});
\]

All TWM...-records for the windows message handlers must use the correct Windows types for the fields:

\[
\text{Msg: UINT; wParam: WPARAM; lParam: LPARAM; Result: LRESULT)}
\]

**Assembler**

In order to make your application (that uses assembly code) work, you will have to make several changes to it:

- rewrite your code that mixes Pascal code and assembly code. Mixing them is not supported in 64-bit applications;
- rewrite assembly code that doesn't consider architecture and processor specifics.

You can use conditional defines to make your application work with different architectures.


**Exception handling**

The biggest difference in exception handling between Delphi 32 and 64-bit is that in Delphi XE2 64-bit you will gain more performance because of different internal exception mechanism. For 32-bit applications, the Delphi compiler (dcc32.exe) generates additional code that is executed any way and that causes performance loss. The 64-bit compiler (dcc64.exe) doesn't generate such code, it generates metadata and stores it in the PDATA section of an executable file instead.
But in Delphi XE2 64-bit it's impossible to have more than 16 levels of nested exceptions. Having more than 16 levels of nested exceptions will cause a Run Time error.

**Debugging**

Debugging of 64-bit applications in RAD Studio XE2 is remote. It is caused by the same reason: RAD Studio XE2 IDE is a 32 application, but your application is 64-bit. If you are trying to debug your application and you cannot do it, you should check that the **Include remote debug symbols** project option is enabled.

To enable it, perform the following steps:
1. Open Project Options (in the main menu **Project->Options**).
2. In the Target combobox, select **Debug configuration - 64-bit Windows platform**. If there is no such option in the combobox, right click "Target Platforms" in Project Manager and select **Add platform**. After adding the 64-bit Windows platform, the **Debug configuration - 64-bit Windows platform** option will be available in the Target combobox.
3. Select **Linking** in the left part of the Project Options form.
4. enable the **Include remote debug symbols** option.

After that, you can run and debug your 64-bit application.

To enable remote debugging, perform the following steps:
1. Install Platform Assistant Server (PAServer) on a remote computer. You can find PAServer in the `%RAD_Studio_XE2_Install_Directory%\PAServer` directory. The `setup_paserver.exe` file is an installation file for Windows, and the `setup_paserver.zip` file is an installation file for MacOS.
2. Run the `PAServer.exe` file on a remote computer and set the password that will be used to connect to this computer.
3. On a local computer with RAD Studio XE2 installed, right-click the target platform that you want to debug in Project Manager and select **Assign Remote Profile**. Click the **Add** button in the displayed window, input your profile name, click the **Next** button, input the name of a remote computer and the password to it (that you assigned when you started PAServer on a remote computer).

After that, you can test the connection by clicking the **Test Connection** button. If your connection failed, check that your firewalls on both remote and local computers do not block your connection, and try to establish a connection once more. If your connection succeeded, click the Next button and then the Finish button. Select your newly created profile and click **OK**.

After performing these steps you will be able to debug your application on a remote computer.
You application will be executed on a remote computer, but you will be able to debug it on your local computer with RAD Studio XE2.

For more information about working with Platform Assistant Server, please refer to http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Running_the_Platform_Assistant_on_Windows

© 1997-2021 Devart. All Rights Reserved.

4.27 Database Specific Aspects of 64-bit Development

Oracle Connectivity Aspects

OCI mode:

Since at design-time Rad Studio XE 2 works only with x32 libraries and if a connection to the server is needed at design-time, you need to install Oracle Client (x32) regardless of the intended platform. (If the x32 client is needed only for development, you can use only Oracle Instant Client). By default, ODAC use DEFAULT of Oracle Client, that is why, if a x64 client is the default client at design-time, you need to specify a x32 client. To prevent conflicts between different versions of Oracle Client on the end-user side, you can leave the Home property empty, in this case, the default client will be used.

DIRECT mode:

Since there is no need to install Oracle Client for the DIRECT mode, the development of applications for the x64 platform does not differ from the development of application for Windows x86.

© 1997-2021 Devart. All Rights Reserved.

5 Reference

This page shortly describes units that exist in ODAC.

Units

<table>
<thead>
<tr>
<th>Unit Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRAccess</td>
<td>This unit contains base</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CRBatchMove</td>
<td>This unit contains implementation of the TCRBatchMove component.</td>
</tr>
<tr>
<td>CREncryption</td>
<td>This unit contains base classes for data encryption.</td>
</tr>
<tr>
<td>CRGrid</td>
<td>This unit contains the TCRDBGrid component.</td>
</tr>
<tr>
<td>CRVio</td>
<td>This unit contains classes for HTTP connections.</td>
</tr>
<tr>
<td>DAAlerter</td>
<td>This unit contains the base class for the TOraAlerter component.</td>
</tr>
<tr>
<td>DADump</td>
<td>This unit contains the base class for the TOraDump component.</td>
</tr>
<tr>
<td>DALoader</td>
<td>This unit contains the base class for the TOraLoader component.</td>
</tr>
<tr>
<td>DAScript</td>
<td>This unit contains the base class for the TOraScript component.</td>
</tr>
<tr>
<td>DASQLMonitor</td>
<td>This unit contains the base class for the TOraSQLMonitor component.</td>
</tr>
<tr>
<td>DBAccess</td>
<td>This unit contains base classes for most of the components.</td>
</tr>
<tr>
<td>MemData</td>
<td>This unit contains classes for storing data in memory.</td>
</tr>
<tr>
<td>MemDS</td>
<td>This unit contains implementation of the TMemDataSet class.</td>
</tr>
<tr>
<td>OdacVcl</td>
<td>This unit contains the visual constituent of ODAC.</td>
</tr>
<tr>
<td>Ora</td>
<td>This unit contains main components of ODAC.</td>
</tr>
<tr>
<td>OraAlerter</td>
<td>This unit contains implementation of the TOraAlerter component.</td>
</tr>
<tr>
<td>OraAQ</td>
<td>This unit contains ODAC components for working with</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OraCall</td>
<td>Defines Oracle Call Interface routines.</td>
</tr>
<tr>
<td>OraClasses</td>
<td>OraClasses unit defines following data type constants: dtRowId, dtCursor, dtOraBlob, dtOraClob, dtBFILE, dtCFILE, dtLabel, dtFixedChar, dtUndefined, dtTimestamp, dtTimestampTZ, dtTimestampLTZ, dtIntervalYM, dtIntervalIDS // obsolete dtBLOBLocator = dtOraBlob, dtCLOBLocator = dtOraClob</td>
</tr>
<tr>
<td>OraConnectionPool</td>
<td>This unit contains the TOraConnectionPoolManager class for managing connection pool.</td>
</tr>
<tr>
<td>OraErrHand</td>
<td>This unit contains the TOraErrorHandler component.</td>
</tr>
<tr>
<td>OraError</td>
<td>This unit contains the EOraError exception class.</td>
</tr>
<tr>
<td>OraLoader</td>
<td>This unit contains implementation of the TOraLoader component.</td>
</tr>
<tr>
<td>OraNet</td>
<td>This unit implements the Direct Mode in ODAC.</td>
</tr>
<tr>
<td>OraObjects</td>
<td>This unit contains classes for Oracle OBJECT, ARRAY, TABLE and XMLTYPE data types.</td>
</tr>
<tr>
<td>OraPackage</td>
<td>This unit contains implementation of the TOraPackage component.</td>
</tr>
<tr>
<td>OraProvider</td>
<td>This unit contains implementation of the TOraProvider component.</td>
</tr>
<tr>
<td>OraScript</td>
<td>This unit contains implementation of the TOraScript component.</td>
</tr>
</tbody>
</table>
| OraSmart        | This unit contains the
5.1 CRAccess

This unit contains base classes for accessing databases.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRCursor</td>
<td>A base class for classes that work with database cursors.</td>
</tr>
</tbody>
</table>

**Types**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBeforeFetchProc</td>
<td>This type is used for the TCustomDADataset.Before Fetch event.</td>
</tr>
</tbody>
</table>

**Enumerations**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRIsolationLevel</td>
<td>Specifies how to handle transactions containing</td>
</tr>
</tbody>
</table>
5.1.1 Classes

Classes in the CRAccess unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRCursor</td>
<td>A base class for classes that work with database cursors.</td>
</tr>
</tbody>
</table>

5.1.1.1 TCRCursor Class

A base class for classes that work with database cursors.

For a list of all members of this type, see TCRCursor members.

Unit

CRAccess

Syntax

```pascal
TCRCursor = class(TSharedObject);
```

Remarks

TCRCursor is a base class for classes that work with database cursors.

Inheritance Hierarchy
5.1.1.1 Members

**TCRCursor** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefCount</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Release</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

5.1.2 Types

Types in the **CRAccess** unit.

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBeforeFetchProc</td>
<td>This type is used for the TCustomDADataset.Before Fetch event.</td>
</tr>
</tbody>
</table>
5.1.2.1 TBeforeFetchProc Procedure Reference

This type is used for the TCustomDADataset.BeforeFetch event.

Unit

CRAccess

Syntax

TBeforeFetchProc = procedure (var Cancel: boolean) of object;

Parameters

Cancel

True, if the current fetch operation should be aborted.

5.1.3 Enumerations

Enumerations in the CRAccess unit.

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRIsolationLevel</td>
<td>Specifies how to handle transactions containing database modifications.</td>
</tr>
<tr>
<td>TCRTransactionAction</td>
<td>Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.</td>
</tr>
<tr>
<td>TCursorState</td>
<td>Used to set cursor state</td>
</tr>
</tbody>
</table>
5.1.3.1 TCRIsolationLevel Enumeration

Specifies how to handle transactions containing database modifications.

Unit

CRAccess

Syntax

TCRIsolationLevel = (ilReadCommitted, ilReadUnCommitted, ilRepeatableRead, ilIsolated, ilSnapshot, ilCustom);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ilCustom</td>
<td>The parameters of the transaction are set manually in the Params property.</td>
</tr>
<tr>
<td>ilIsolated</td>
<td>The most restricted level of transaction isolation. Database server isolates data involved in current transaction by putting additional processing on range locks. Used to put aside all undesired effects observed in the concurrent accesses to the same set of data, but may lead to a greater latency at times of a congested database environment.</td>
</tr>
<tr>
<td>ilReadCommitted</td>
<td>Sets isolation level at which transaction cannot see changes made by outside transactions until they are committed. Only dirty reads (changes made by uncommitted transactions) are eliminated by this state of the isolation level. The default value.</td>
</tr>
<tr>
<td>ilReadUncommitted</td>
<td>The most unrestricted level of the transaction isolation. All types of data access interferences are possible. Mainly used for browsing database and to receive instant data with prospective changes.</td>
</tr>
<tr>
<td>ilRepeatableRead</td>
<td>Prevents concurrent transactions from modifying data in the current uncommitted transaction. This level eliminates dirty reads as well as nonrepeatable reads (repeatable reads of the same data in one transaction before and after outside transactions may have started and committed).</td>
</tr>
<tr>
<td>ilSnapshot</td>
<td>Uses row versioning. Provides transaction-level read consistency. A data snapshot is taken when the snapshot transaction starts, and remains consistent for the duration of a transaction.</td>
</tr>
</tbody>
</table>
5.1.3.2 TCRTransactionAction Enumeration

Specifies the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

Unit
CRAccess

Syntax

\[
\text{TCRTransactionAction} = (\text{taCommit}, \text{taRollback});
\]

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>taCommit</td>
<td>Transaction is committed.</td>
</tr>
<tr>
<td>taRollback</td>
<td>Transaction is rolled back.</td>
</tr>
</tbody>
</table>

5.1.3.3 TCursorState Enumeration

Used to set cursor state

Unit
CRAccess

Syntax

\[
\text{TCursorState} = (\text{csInactive}, \text{csOpen}, \text{csParsed}, \text{csPrepared}, \text{csBound}, \text{csExecuteFetchAll}, \text{csExecuting}, \text{csExecuted}, \text{csFetching}, \text{csFetchingAll}, \text{csFetched});
\]

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>csBound</td>
<td>Parameters bound</td>
</tr>
<tr>
<td>csExecuted</td>
<td>Statement successfully executed</td>
</tr>
<tr>
<td>csExecuteFetchAll</td>
<td>Set before FetchAll</td>
</tr>
<tr>
<td>csExecuting</td>
<td>Statement is set before executing</td>
</tr>
<tr>
<td>csFetched</td>
<td>Fetch finished or canceled</td>
</tr>
</tbody>
</table>
5.2 **CRBatchMove**

This unit contains implementation of the TCRBatchMove component.

### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRBatchMove</td>
<td>Transfers records between datasets.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRBatchMoveProgressEvent</td>
<td>This type is used for the TCRBatchMove.OnBatchMoveProgress event.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRBatchMode</td>
<td>Used to set the type of the batch operation that will be executed after calling the TCRBatchMove.Execute method.</td>
</tr>
<tr>
<td>TCRFieldMappingMode</td>
<td>Used to specify the way fields of the destination and source datasets will be mapped to each other if the TCRBatchMove.Mappings list is empty.</td>
</tr>
</tbody>
</table>
5.2.1 Classes

Classes in the **CRBatchMove** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRBatchMove</td>
<td>Transfers records between datasets.</td>
</tr>
</tbody>
</table>

5.2.1.1 TCRBatchMove Class

Transfers records between datasets.

For a list of all members of this type, see [TCRBatchMove members](#).

Unit

**CRBatchMove**

Syntax

```
TCRBatchMove = class(TComponent);
```

Remarks

The TCRBatchMove component transfers records between datasets. Use it to copy dataset records to another dataset or to delete datasets records that match records in another dataset. The **TCRBatchMove.Mode** property determines the desired operation type, the **TCRBatchMove.Source** and **TCRBatchMove.Destination** properties indicate corresponding datasets.

**Note:** A TCRBatchMove component is added to the Data Access page of the component palette, not to the ODAC page.
5.2.1.1.1 Members

**TCRBatchMove** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbortOnKeyViol</td>
<td>Used to specify whether the batch operation should be terminated immediately after key or integrity violation.</td>
</tr>
<tr>
<td>AbortOnProblem</td>
<td>Used to specify whether the batch operation should be terminated immediately when it is necessary to truncate data to make it fit the specified Destination.</td>
</tr>
<tr>
<td>ChangedCount</td>
<td>Used to get the number of records changed in the destination dataset.</td>
</tr>
<tr>
<td>CommitCount</td>
<td>Used to set the number of records to be batch moved before commit occurs.</td>
</tr>
<tr>
<td>Destination</td>
<td>Used to specify the destination dataset for the batch operation.</td>
</tr>
<tr>
<td>FieldMappingMode</td>
<td>Used to specify the way fields of destination and source datasets will be mapped to each other if the TCRBatchMove.Mappings list is empty.</td>
</tr>
<tr>
<td>KeyViolCount</td>
<td>Used to get the number of records that could not be moved to or from the destination dataset because of integrity or key violations.</td>
</tr>
<tr>
<td>Mappings</td>
<td>Used to set field matching between source and destination datasets for the batch operation.</td>
</tr>
<tr>
<td>Mode</td>
<td>Used to set the type of the batch operation that will be executed after calling the TCRBatchMove.Execute method.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovedCount</td>
<td>Used to get the number of records that were read from the source dataset during the batch operation.</td>
</tr>
<tr>
<td>ProblemCount</td>
<td>Used to get the number of records that could not be added to the destination dataset because of the field type mismatch.</td>
</tr>
<tr>
<td>RecordCount</td>
<td>Used to indicate the maximum number of records in the source dataset that will be applied to the destination dataset.</td>
</tr>
<tr>
<td>Source</td>
<td>Used to specify the source dataset for the batch operation.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Performs the batch operation.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnBatchMoveProgress</td>
<td>Occurs when providing feedback to the user about the batch operation in progress is needed.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.2.1.1.2 Properties

Properties of the TCRBatchMove class.

For a complete list of the TCRBatchMove class members, see the TCRBatchMove Members topic.
Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChangedCount</td>
<td>Used to get the number of records changed in the destination dataset.</td>
</tr>
<tr>
<td>KeyViolCount</td>
<td>Used to get the number of records that could not be moved to or from the destination dataset because of integrity or key violations.</td>
</tr>
<tr>
<td>MovedCount</td>
<td>Used to get the number of records that were read from the source dataset during the batch operation.</td>
</tr>
<tr>
<td>ProblemCount</td>
<td>Used to get the number of records that could not be added to the destination dataset because of the field type mismatch.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbortOnKeyViol</td>
<td>Used to specify whether the batch operation should be terminated immediately after key or integrity violation.</td>
</tr>
<tr>
<td>AbortOnProblem</td>
<td>Used to specify whether the batch operation should be terminated immediately when it is necessary to truncate data to make it fit the specified Destination.</td>
</tr>
<tr>
<td>CommitCount</td>
<td>Used to set the number of records to be batch moved before commit occurs.</td>
</tr>
<tr>
<td>Destination</td>
<td>Used to specify the destination dataset for the batch operation.</td>
</tr>
<tr>
<td>FieldMappingMode</td>
<td>Used to specify the way fields of destination and source datasets will be mapped to each other if the</td>
</tr>
</tbody>
</table>
### Mappings

**TCRBatchMove.Mappings**

- **list is empty.**

**Used to set field matching between source and destination datasets for the batch operation.**

### Mode

**Used to set the type of the batch operation that will be executed after calling the TCRBatchMove.Execute method.**

### RecordCount

**Used to indicate the maximum number of records in the source dataset that will be applied to the destination dataset.**

### Source

**Used to specify the source dataset for the batch operation.**

### See Also

- [TCRBatchMove Class](#)
- [TCRBatchMove Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

5.2.1.2.1 AbortOnKeyViol Property

- Used to specify whether the batch operation should be terminated immediately after key or integrity violation.

### Class

**TCRBatchMove**

### Syntax

```
property AbortOnKeyViol: boolean default True;
```

### Remarks

Use the AbortOnKeyViol property to specify whether the batch operation is terminated immediately after key or integrity violation.
5.2.1.1.2.2 AbortOnProblem Property

Used to specify whether the batch operation should be terminated immediately when it is necessary to truncate data to make it fit the specified Destination.

Class

TCRBatchMove

Syntax

```
property AbortOnProblem: boolean default True;
```

Remarks

Use the AbortOnProblem property to specify whether the batch operation is terminated immediately when it is necessary to truncate data to make it fit the specified Destination.

5.2.1.1.2.3 ChangedCount Property

Used to get the number of records changed in the destination dataset.

Class

TCRBatchMove

Syntax

```
property ChangedCount: Integer;
```

Remarks

Use the ChangedCount property to get the number of records changed in the destination dataset. It shows the number of records that were updated in the bmUpdate or bmAppendUpdate mode or were deleted in the bmDelete mode.
5.2.1.1.2.4 CommitCount Property

Used to set the number of records to be batch moved before commit occurs.

Class

TCRBatchMove

Syntax

```
property CommitCount: integer default 0;
```

Remarks

Use the CommitCount property to set the number of records to be batch moved before the commit occurs. If it is set to 0, the operation will be chunked to the number of records to fit 32 Kb.

5.2.1.1.2.5 Destination Property

Used to specify the destination dataset for the batch operation.

Class

TCRBatchMove

Syntax

```
property Destination: TDataSet;
```

Remarks

Specifies the destination dataset for the batch operation.

5.2.1.1.2.6 FieldMappingMode Property

Used to specify the way fields of destination and source datasets will be mapped to each other if the Mappings list is empty.

Class
TCRBatchMove

Syntax

```pascal
property FieldMappingMode: TCRFieldMappingMode default mmFieldIndex;
```

Remarks

Specifies in what way fields of destination and source datasets will be mapped to each other if the Mappings list is empty.

See Also

- `AbortOnKeyViol`

5.2.1.2.7 KeyViolCount Property

Used to get the number of records that could not be moved to or from the destination dataset because of integrity or key violations.

Class

TCRBatchMove

Syntax

```pascal
property KeyViolCount: Integer;
```

Remarks

Use the KeyViolCount property to get the number of records that could not be replaced, added, deleted from the destination dataset because of integrity or key violations.

If `AbortOnKeyViol` is True, then KeyViolCount will never exceed one, because the operation aborts when the integrity or key violation occurs.

See Also

- `AbortOnKeyViol`
5.2.1.1.2.8 Mappings Property

Used to set field matching between source and destination datasets for the batch operation.

Class
TCRBatchMove

Syntax

```delphi
property Mappings: TStrings;
```

Remarks
Use the Mappings property to set field matching between the source and destination datasets for the batch operation. By default, field matching is based on their position in the datasets. To map the column ColName in the source dataset to the column with the same name in the destination dataset, use:

ColName

Example
To map a column named SourceColName in the source dataset to the column named DestColName in the destination dataset, use:

DestColName = SourceColName

5.2.1.1.2.9 Mode Property

Used to set the type of the batch operation that will be executed after calling the `Execute` method.

Class
TCRBatchMove

Syntax

```delphi
property Mode: TCRBatchMode default bmAppend;
```

Remarks
Use the Mode property to set the type of the batch operation that will be executed after calling the `Execute` method.

**Class**

**TCRBatchMove**

**Syntax**

```
property MovedCount: Integer;
```

**Remarks**

Use the MovedCount property to get the number of records that were read from the source dataset during the batch operation. This number includes records that caused key or integrity violations or were trimmed.

**ProblemCount Property**

**Class**

**TCRBatchMove**

**Syntax**

```
property ProblemCount: Integer;
```

**Remarks**

Use the ProblemCount property to get the number of records that could not be added to the destination dataset because of the field type mismatch.
If `AbortOnProblem` is True, then ProblemCount will never exceed one, because the operation aborts when the problem occurs.

**See Also**
- `AbortOnProblem`

---

### 5.2.1.1.2.12 RecordCount Property

Used to indicate the maximum number of records in the source dataset that will be applied to the destination dataset.

**Class**

**TCRBatchMove**

**Syntax**

```plaintext
property RecordCount: Integer default 0;
```

**Remarks**

Determines the maximum number of records in the source dataset, that will be applied to the destination dataset. If it is set to 0, all records in the source dataset will be applied to the destination dataset, starting from the first record. If RecordCount is greater than 0, up to the RecordCount records are applied to the destination dataset, starting from the current record in the source dataset. If RecordCount exceeds the number of records left in the source dataset, batch operation terminates after reaching last record.
property Source: TDataSet;

Remarks
Specifies the source dataset for the batch operation.

Methods of the TCRBatchMove class.

For a complete list of the TCRBatchMove class members, see the TCRBatchMove Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Performs the batch operation.</td>
</tr>
</tbody>
</table>

See Also
- TCRBatchMove Class
- TCRBatchMove Class Members

Perform the batch operation.

Class
TCRBatchMove

Syntax

procedure Execute;

Remarks
Call the Execute method to perform the batch operation.
5.2.1.1.4 Events

Events of the **TCRBatchMove** class.

For a complete list of the **TCRBatchMove** class members, see the [TCRBatchMove Members](#) topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnBatchMoveProgress</strong></td>
<td>Occurs when providing feedback to the user about the batch operation in progress is needed.</td>
</tr>
</tbody>
</table>

### See Also

- [TCRBatchMove Class](#)
- [TCRBatchMove Class Members](#)

### 5.2.1.1.4.1 OnBatchMoveProgress Event

Occurs when providing feedback to the user about the batch operation in progress is needed.

### Class

**TCRBatchMove**

### Syntax

**property** OnBatchMoveProgress: TCRBatchMoveProgressEvent;

### Remarks

Write the OnBatchMoveProgress event handler to provide feedback to the user about the batch operation progress.
Reserved.

5.2.2 Types

Types in the CRBatchMove unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRBatchMoveProgressEvent</td>
<td>This type is used for the TCRBatchMove.OnBatchMoveProgress event.</td>
</tr>
</tbody>
</table>

TCRBatchMoveProgressEvent Procedure Reference

This type is used for the TCRBatchMove.OnBatchMoveProgress event.

Unit

CRBatchMove

Syntax

TCRBatchMoveProgressEvent = procedure (Sender: TObject; Percent: integer) of object;

Parameters

Sender
An object that raised the event.

Percent
Percentage of the batch operation progress.

5.2.3 Enumerations

Enumerations in the CRBatchMove unit.

Enumerations
### TCRBatchMode Enumeration

Used to set the type of the batch operation that will be executed after calling the `TCRBatchMove.Execute` method.

**Unit**

**Syntax**

```plaintext
TCRBatchMode = (bmAppend, bmUpdate, bmAppendUpdate, bmDelete);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmAppend</td>
<td>Appends the records from the source dataset to the destination dataset. The default mode.</td>
</tr>
<tr>
<td>bmAppendUpdate</td>
<td>Replaces records in the destination dataset with the matching records from the source dataset. If there is no matching record in the destination dataset, the record will be appended to it.</td>
</tr>
<tr>
<td>bmDelete</td>
<td>Deletes records from the destination dataset if there are matching records in the source dataset.</td>
</tr>
<tr>
<td>bmUpdate</td>
<td>Replaces records in the destination dataset with the matching records from the source dataset.</td>
</tr>
</tbody>
</table>
5.2.3.2 TCRFieldMappingMode Enumeration

Used to specify the way fields of the destination and source datasets will be mapped to each other if the TCRBatchMove.Mappings list is empty.

Unit

CRBatchMove

Syntax

TCRFieldMappingMode = (mmFieldIndex, mmFieldName);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>mmFieldIndex</td>
<td>Specifies that the fields of the destination dataset will be mapped to the fields of the source dataset by field index.</td>
</tr>
<tr>
<td>mmFieldName</td>
<td>Mapping is performed by field names.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.3 CREncryption

This unit contains base classes for data encryption.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCREncryptor</td>
<td>The class that performs data encryption and decryption in a client application using various encryption algorithms.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCREncDataHeader</td>
<td>Specifies whether the additional information is stored with the encrypted data.</td>
</tr>
</tbody>
</table>
### 5.3.1 Classes

Classes in the `CREncryption` unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TCREncryptor</strong></td>
<td>The class that performs data encryption and decryption in a client application using various encryption algorithms.</td>
</tr>
</tbody>
</table>

**Unit**

`CREncryption`

**Syntax**

```plaintext
TCREncryptor = class(TComponent);
```
### 5.3.1.1.1 Members

**TCREncryptor** class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataHeader</strong></td>
<td>Specifies whether the additional information is stored with the encrypted data.</td>
</tr>
<tr>
<td><strong>EncryptionAlgorithm</strong></td>
<td>Specifies the algorithm of data encryption.</td>
</tr>
<tr>
<td><strong>HashAlgorithm</strong></td>
<td>Specifies the algorithm of generating hash data.</td>
</tr>
<tr>
<td><strong>InvalidHashAction</strong></td>
<td>Specifies the action to perform on data fetching when hash data is invalid.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Used to set a password that is used to generate a key for encryption.</td>
</tr>
</tbody>
</table>

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetKey</strong></td>
<td>Sets a key, using which data is encrypted.</td>
</tr>
</tbody>
</table>

For a complete list of the TCREncryptor class members, see the [TCREncryptor Members](#) topic.

#### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataHeader</strong></td>
<td>Specifies whether the additional information is stored with the encrypted data.</td>
</tr>
<tr>
<td><strong>Property</strong></td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EncryptionAlgorithm</td>
<td>Specifies the algorithm of data encryption.</td>
</tr>
<tr>
<td>HashAlgorithm</td>
<td>Specifies the algorithm of generating hash data.</td>
</tr>
<tr>
<td>InvalidHashAction</td>
<td>Specifies the action to perform on data fetching when hash data is invalid.</td>
</tr>
<tr>
<td>Password</td>
<td>Used to set a password that is used to generate a key for encryption.</td>
</tr>
</tbody>
</table>

### See Also

- [TCREncryptor Class](#)
- [TCREncryptor Class Members](#)

5.3.1.1.2.1 DataHeader Property

Specifies whether the additional information is stored with the encrypted data.

**Class**

[TCREncryptor](#)

**Syntax**

```property
DataHeader: TCREncDataHeader default ehTagAndHash;
```

**Remarks**

Use DataHeader to specify whether the additional information is stored with the encrypted data. Default value is `ehTagAndHash`. 

© 1997-2021 Devart. All Rights Reserved.

[Request Support] [DAC Forum] [Provide Feedback]
5.3.1.2.2 EncryptionAlgorithm Property

Specifies the algorithm of data encryption.

Class

TCREncryptor

Syntax

```pascal
property EncryptionAlgorithm: TCREncryptionAlgorithm default eaBlowfish;
```

Remarks

Use EncryptionAlgorithm to specify the algorithm of data encryption. Default value is eaBlowfish.

5.3.1.2.3 HashAlgorithm Property

Specifies the algorithm of generating hash data.

Class

TCREncryptor

Syntax

```pascal
property HashAlgorithm: TCRHashAlgorithm default haSHA1;
```

Remarks

Use HashAlgorithm to specify the algorithm of generating hash data. This property is used only if hash is stored with the encrypted data (the DataHeader property is set to ehTagAndHash). Default value is haSHA1.
5.3.1.2.4 InvalidHashAction Property

Specifies the action to perform on data fetching when hash data is invalid.

Class
TCREncryptor

Syntax

| property | InvalidHashAction: TCRInvalidHashAction default ihFail; |

Remarks

Use InvalidHashAction to specify the action to perform on data fetching when hash data is invalid. This property is used only if hash is stored with the encrypted data (the DataHeader property is set to ehTagAndHash). Default value is ihFail.

If the DataHeader property is set to ehTagAndHash, then on data fetching from a server the hash check is performed for each record. After data decryption its hash is calculated and compared with the hash stored in the field. If these values don't coincide, it means that the stored data is incorrect, and depending on the value of the InvalidHashAction property one of the following actions is performed:

- **ihFail** - the EInvalidHash exception is raised and further data reading from the server is interrupted.
- **ihSkipData** - the value of the field for this record is set to Null. No exception is raised.
- **ihIgnoreError** - in spite of the fact that the data is not valid, the value is set in the field. No exception is raised.
Remarks

Use Password to set a password that is used to generate a key for encryption.

**Note:** Calling of the `SetKey` method clears the Password property.

### 5.3.1.1.3 Methods

Methods of the `TCREncryptor` class.

For a complete list of the `TCREncryptor` class members, see the `TCREncryptor Members` topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetKey</strong></td>
<td>Sets a key, using which data is encrypted.</td>
</tr>
</tbody>
</table>

#### See Also

- `TCREncryptor Class`
- `TCREncryptor Class Members`

### 5.3.1.1.3.1 SetKey Method

Sets a key, using which data is encrypted.

#### Class

`TCREncryptor`

#### Syntax

```pascal
procedure SetKey(const Key: TBytes; Offset: Integer; Count: Integer);
overload;
procedure SetKey(const Key; Count: Integer);
overload;
```

#### Parameters
**Key**
- Holds bytes that represent a key.

**Offset**
- Offset in bytes to the position, where the key begins.

**Count**
- Number of bytes to use from Key.

**Remarks**
Use SetKey to set a key, using which data is encrypted.

**Note:** Calling of the SetKey method clears the Password property.

---

5.3.2 **Enumerations**

Enumerations in the **CREncryption** unit.

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCREncDataHeader</td>
<td>Specifies whether the additional information is stored with the encrypted data.</td>
</tr>
<tr>
<td>TCREncryptionAlgorithm</td>
<td>Specifies the algorithm of data encryption.</td>
</tr>
<tr>
<td>TCRHashAlgorithm</td>
<td>Specifies the algorithm of generating hash data.</td>
</tr>
<tr>
<td>TCRInvalidHashAction</td>
<td>Specifies the action to perform on data fetching when hash data is invalid.</td>
</tr>
</tbody>
</table>

---

5.3.2.1 **TCREncDataHeader Enumeration**

Specifies whether the additional information is stored with the encrypted data.
**CREncryption**

**Syntax**

```
TCREncDataHeader = (ehTagAndHash, ehTag, ehNone);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ehNone</td>
<td>No additional information is stored.</td>
</tr>
<tr>
<td>ehTag</td>
<td>GUID and the random initialization vector are stored with the encrypted data.</td>
</tr>
<tr>
<td>ehTagAndHash</td>
<td>Hash, GUID, and the random initialization vector are stored with the encrypted data.</td>
</tr>
</tbody>
</table>

---

5.3.2.2 **TCREncryptionAlgorithm Enumeration**

Specifies the algorithm of data encryption.

**Unit**

**CREncryption**

**Syntax**

```
TCREncryptionAlgorithm = (eaTripleDES, eaBlowfish, eaAES128, eaAES192, eaAES256, eaCast128, eaRC4);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>eaAES128</td>
<td>The AES encryption algorithm with key size of 128 bits is used.</td>
</tr>
<tr>
<td>eaAES192</td>
<td>The AES encryption algorithm with key size of 192 bits is used.</td>
</tr>
<tr>
<td>eaAES256</td>
<td>The AES encryption algorithm with key size of 256 bits is used.</td>
</tr>
<tr>
<td>eaBlowfish</td>
<td>The Blowfish encryption algorithm is used.</td>
</tr>
<tr>
<td>eaCast128</td>
<td>The CAST-128 encryption algorithm with key size of 128 bits is used.</td>
</tr>
<tr>
<td>eaRC4</td>
<td>The RC4 encryption algorithm is used.</td>
</tr>
<tr>
<td>eaTripleDES</td>
<td>The Triple DES encryption algorithm is used.</td>
</tr>
</tbody>
</table>
5.3.2.3 TCRHashAlgorithm Enumeration

Specifies the algorithm of generating hash data.

Unit
CREncryption

Syntax

TCRHashAlgorithm = (haSHA1, haMD5);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>haMD5</td>
<td>The MD5 hash algorithm is used.</td>
</tr>
<tr>
<td>haSHA1</td>
<td>The SHA-1 hash algorithm is used.</td>
</tr>
</tbody>
</table>

5.3.2.4 TCRInvalidHashAction Enumeration

Specifies the action to perform on data fetching when hash data is invalid.

Unit
CREncryption

Syntax

TCRInvalidHashAction = (ihFail, ihSkipData, ihIgnoreError);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ihFail</td>
<td>The EInvalidHash exception is raised and further data reading from the server is interrupted.</td>
</tr>
<tr>
<td>ihIgnoreError</td>
<td>In spite of the fact that the data is not valid, the value is set in the field. No exception is raised.</td>
</tr>
<tr>
<td>ihSkipData</td>
<td>The value of the field for this record is set to Null. No exception is raised.</td>
</tr>
</tbody>
</table>
5.4 CRGrid

This unit contains the TCRDBGrid component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRDBGrid</td>
<td>Extends the functionality of the TDBGrid component.</td>
</tr>
</tbody>
</table>

5.4.1 Classes

Classes in the CRGrid unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRDBGrid</td>
<td>Extends the functionality of the TDBGrid component.</td>
</tr>
</tbody>
</table>

5.4.1.1 TCRDBGrid Class

Extends the functionality of the TDBGrid component.

For a list of all members of this type, see TCRDBGrid members.

Unit

CRGrid

Syntax

TCRDBGrid = class(TCustomDBGrid);
TCRDBGrid class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>To provide extended functionality TCRDBGrid component is shipped with a TCRColumn class, which is derived from the standard TColumn class but is tailored to work with TCRDBGrid only.</td>
</tr>
<tr>
<td>Filtered</td>
<td>Set Filtered property to specify whether record entries are filtered according to the conditional statements of individual columns or not.</td>
</tr>
<tr>
<td>LevelDelimiterChar</td>
<td>Set LevelDelimiterChar to set the character symbol which TCRDBGrid looks for in column captions to resolve multilevel nested columns.</td>
</tr>
<tr>
<td>OnMemoClick</td>
<td>Write an OnMemoClick event handler to provide custom processing of Memo fields instead of built-in Memo field editor.</td>
</tr>
<tr>
<td>OptionsEx</td>
<td>OptionsEx property provides control over TCRDBGrid-specific features. They complement inherited options found in Options property.</td>
</tr>
<tr>
<td>OptionsMenu</td>
<td>TCRDBGrid has local menu is situated in the left upper corner. Standard menu has items to show or hide filter bar and search bars. User can change content of this</td>
</tr>
</tbody>
</table>
menu using OptionsMenu property.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ActivateFilterEdit</strong></td>
<td>Use <code>ActivateFilterEdit</code> at run-time to move input focus to the filter bar. It activates edit control that belongs to the column, specified by Column property.</td>
</tr>
<tr>
<td><strong>ActivateSearchEdit</strong></td>
<td>Use <code>ActivateSearchEdit</code> at run-time to move input focus to the search bar. It activates edit control that belongs to the column, specified by Column property.</td>
</tr>
<tr>
<td><strong>AdjustColumns</strong></td>
<td>Call <code>AdjustColumns</code> method at run-time to stretch all columns in a grid so that they become wide enough to accommodate every visible field value. Widths of column captions restrict minimum column sizes for this operation.</td>
</tr>
<tr>
<td><strong>ApplyFilter</strong></td>
<td>Call <code>ApplyFilter</code> method at run-time to update TCRDBGrid component taking into account filter statements defined for every column.</td>
</tr>
<tr>
<td><strong>CalcTitleLevel</strong></td>
<td>Call <code>CalcTitleLevel</code> method to set Top and Bottom values of aRect parameter record depending on Level parameter value.</td>
</tr>
<tr>
<td><strong>ClearFilters</strong></td>
<td>Call <code>ClearFilters</code> method to clear all filter statements for grid columns.</td>
</tr>
<tr>
<td><strong>ClearSorting</strong></td>
<td>Call <code>ClearSorting</code> method to discard sorting previously applied to the grid data.</td>
</tr>
</tbody>
</table>
DataChanged

Call DataChanged method to update displayed data and status information for the grid.

GetGridSize

Call GetGridSize method to obtain width of a client area for the grid component.

GetTitleLevel

GetTitleLevel method returns TRect structure filled with top and bottom coordinates for the specified title level.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGetCellParams</td>
<td>Write an OnGetCellParams event handler to provide custom background colors while the grid redraws individual cells.</td>
</tr>
</tbody>
</table>

Properties of the TCRDBGrid class.

For a complete list of the TCRDBGrid class members, see the TCRDBGrid Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptionsMenu</td>
<td>TCRDBGrid has local menu situated in the left upper corner. Standard menu has items to show or hide filter bar and search bars. User can change content of this menu using OptionsMenu property.</td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Columns</strong></td>
<td>To provide extended functionality TCRDBGrid component is shipped with a TCRColumn class, which is derived from the standard TColumn class but is tailored to work with TCRDBGrid only.</td>
</tr>
<tr>
<td><strong>Filtered</strong></td>
<td>Set Filtered property to specify whether record entries are filtered according to the conditional statements of individual columns or not.</td>
</tr>
<tr>
<td><strong>LevelDelimiterChar</strong></td>
<td>Set LevelDelimiterChar to set the character symbol which TCRDBGrid looks for in column captions to resolve multilevel nested columns.</td>
</tr>
<tr>
<td><strong>OnMemoClick</strong></td>
<td>Write an OnMemoClick event handler to provide custom processing of Memo fields instead of built-in Memo field editor.</td>
</tr>
<tr>
<td><strong>OptionsEx</strong></td>
<td>OptionsEx property provides control over TCRDBGrid-specific features. They complement inherited options found in Options property.</td>
</tr>
</tbody>
</table>

See Also
- [TCRDBGrid Class](#)
- [TCRDBGrid Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

To provide extended functionality TCRDBGrid component is shipped with a TCRColumn class, which is derived from the standard TColumn class but is tailored to work with TCRDBGrid only.
Class
TCRDBGrid

Syntax

```
property Columns: TCRDBGridColumns stored False;
```

Remarks

When browsing TDBGridColumns remember that its Items property actually returns TCRColumn objects when indexed.

© 1997-2021 Devart. All Rights Reserved.

5.4.1.2.2 Filtered Property

Set Filtered property to specify whether record entries are filtered according to the conditional statements of individual columns or not.

Class
TCRDBGrid

Syntax

```
property Filtered: boolean default True;
```

Remarks

Conditional statements of all columns are combined to form a single filter.

© 1997-2021 Devart. All Rights Reserved.

5.4.1.2.3 LevelDelimiterChar Property

Set LevelDelimiterChar to set the character symbol which TCRDBGrid looks for in column captions to resolve multilevel nested columns.

Class
TCRDBGrid
5.4.1.2.4 OnMemoClick Property

Write an OnMemoClick event handler to provide custom processing of Memo fields instead of built-in Memo field editor.

Class

TCRDBGrid

5.4.1.2.5 OptionsEx Property

OptionsEx property provides control over TCRDBGrid-specific features. They complement inherited options found in Options property.

Class

TCRDBGrid

Remarks

Following values are supported for the OptionsEx set:

*dgeEnableSort* - enables sorting of records by the specified column clicking on the column title.

*dgeFilterBar* - shows filter bar below grid caption where filter expressions for every column
are entered manually.

dgeLocalFilter - records are filtered using associated dataset component facilities. Otherwise
database server processes all filter operations.

dgeLocalSorting - records are sorted using associated dataset component facilities.
Otherwise database server processes all sorting operations.

dgeRecordCount - displays status bar with current record number in respect to the total
number of records. This option is mutually exclusive with dgeSummary option.

dgeSearchBar - shows search bar below grid caption where incremental search expression
is entered for every column.

dgeStretch - makes all columns adjust their widths so that they fin onto entire grid area.

dgeSummary - displays status bar with summary information for every column. See
TCRColumn.SummaryMode property on description of available options for individual
columns. dgeSummary is mutually exclusive with dgeRecordCount option.

Class

TCRDBGrid

Syntax

```pascal
property OptionsMenu: TPopupMenu;
```

5.4.1.2.6 OptionsMenu Property

TCRDBGrid has local menu is situated in the left upper corner. Standard menu has items to
show or hide filter bar and search bars. User can change content of this menu using
OptionsMenu property.

Class

TCRDBGrid

5.4.1.3 Methods

Methods of the TCRDBGrid class.

For a complete list of the TCRDBGrid class members, see the TCRDBGrid Members topic.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActivateFilterEdit</td>
<td>Use ActivateFilterEdit at run-time to move input focus to the filter bar. It activates edit control that belongs to the column, specified by Column property.</td>
</tr>
<tr>
<td>ActivateSearchEdit</td>
<td>Use ActivateSearchEdit at run-time to move input focus to the search bar. It activates edit control that belongs to the column, specified by Column property.</td>
</tr>
<tr>
<td>AdjustColumns</td>
<td>Call AdjustColumns method at run-time to stretch all columns in a grid so that they become wide enough to accommodate every visible field value. Widths of column captions restrict minimum column sizes for this operation.</td>
</tr>
<tr>
<td>ApplyFilter</td>
<td>Call ApplyFilter method at run-time to update TCRDBGrid component taking into account filter statements defined for every column.</td>
</tr>
<tr>
<td>CalcTitleLevel</td>
<td>Call CalcTitleLevel method to set Top and Bottom values of aRect parameter record depending on Level parameter value.</td>
</tr>
<tr>
<td>ClearFilters</td>
<td>Call CalcFilters method to clear all filter statements for grid columns.</td>
</tr>
<tr>
<td>ClearSorting</td>
<td>Call ClearSorting method to discard sorting previously applied to the grid data.</td>
</tr>
<tr>
<td>DataChanged</td>
<td>Call DataChanged method to update displayed data and status information for the grid.</td>
</tr>
</tbody>
</table>
GetGridSize

Call GetGridSize method to obtain width of a client area for the grid component.

GetTitleLevel

GetTitleLevel method returns TRect structure filled with top and bottom coordinates for the specified title level.

See Also
- TCRDBGrid Class
- TCRDBGrid Class Members

5.4.1.1.3.1  ActivateFilterEdit Method

Use ActivateFilterEdit at run-time to move input focus to the filter bar. It activates edit control that belongs to the column, specified by Column property.

Class
TCRDBGrid

Syntax

\texttt{procedure ActivateFilterEdit(Column: TColumn);}

Parameters

\texttt{Column}

5.4.1.1.3.2  ActivateSearchEdit Method

Use ActivateSearchEdit at run-time to move input focus to the search bar. It activates edit control that belongs to the column, specified by Column property.

Class
TCRDBGrid

Syntax
procedure ActivateSearchEdit(Column: TColumn);

Parameters

Column

Call AdjustColumns method at run-time to stretch all columns in a grid so that they become wide enough to accommodate every visible field value. Widths of column captions restrict minimum column sizes for this operation.

Class

TCRDBGrid

Syntax

procedure AdjustColumns;

Remarks

Visual feedback is immediate if OptionsEx property includes dgeLocalFilter option.

Note: Since filtering is performed at session level then other data-aware controls may also be updated.

Call ApplyFilter method at run-time to update TCRDBGrid component taking into account filter statements defined for every column.

Class

TCRDBGrid

Syntax

procedure ApplyFilter;
Remarks

Visual feedback is immediate if OptionsEx property includes dgeLocalFilter option.

Note: Since filtering is performed at session level then other data-aware controls may also be updated.

5.4.1.3.5 CalcTitleLevel Method

Call CalcTitleLevel method to set Top and Bottom values of aRect parameter record depending on Level parameter value.

Class

TCRDBGrid

Syntax

procedure CalcTitleLevel(Level: integer; var aRect: TRect);

Parameters

Level
aRect

5.4.1.3.6 ClearFilters Method

Call ClearFilters method to clear all filter statements for grid columns.

Class

TCRDBGrid

Syntax

procedure ClearFilters;

© 1997-2021 Devart. All Rights Reserved.

Request Support    DAC Forum    Provide Feedback
5.4.1.3.7 ClearSorting Method

Call ClearSorting method to discard sorting previously applied to the grid data.

Class

TCRDBGrid

Syntax

```pascal
procedure ClearSorting;
```

5.4.1.3.8 DataChanged Method

Call DataChanged method to update displayed data and status information for the grid.

Class

TCRDBGrid

Syntax

```pascal
procedure DataChanged;
```

5.4.1.3.9 GetGridSize Method

Call GetGridSize method to obtain width of a client area for the grid component.

Class

TCRDBGrid

Syntax

```pascal
function GetGridSize: integer;
```

Remarks

Client area accommodates cell values only without column grid lines.
5.4.1.3.10 GetTitleLevel Method

GetTitleLevel method returns TRect structure filled with top and bottom coordinates for the specified title level.

Class
TCRDBGrid

Syntax

```
function GetTitleLevel(Level: integer): TRect;
```

Parameters

Level

5.4.1.4 Events

Events of the TCRDBGrid class.

For a complete list of the TCRDBGrid class members, see the TCRDBGrid Members topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGetCellParams</td>
<td>Write an OnGetCellParams event handler to provide custom background colors while the grid redraws individual cells.</td>
</tr>
</tbody>
</table>

See Also

- TCRDBGrid Class
- TCRDBGrid Class Members

© 1997-2021
Devart. All Rights Reserved.
5.4.1.1.4.1 OnGetCellParams Event

Write an OnGetCellParams event handler to provide custom background colors while the grid redraws individual cells.

Class

TCRDBGrid

Syntax

property OnGetCellParams: TGetCellParamsEvent;

5.5 CRVio

This unit contains classes for HTTP connections.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>THttpOptions</td>
<td>This class is used to establish an HTTP connection.</td>
</tr>
<tr>
<td>TProxyOptions</td>
<td>This class is used to establish an HTTP connection through a proxy server.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIPVersion</td>
<td>Specifies Internet Protocol version.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
5.5.1 Classes

Classes in the CRVio unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>THttpOptions</td>
<td>This class is used to establish an HTTP connection.</td>
</tr>
<tr>
<td>TProxyOptions</td>
<td>This class is used to establish an HTTP connection through a proxy server.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.5.1.1 THttpOptions Class

This class is used to establish an HTTP connection.

For a list of all members of this type, see THttpOptions members.

Unit
CRVio

Syntax

THttpOptions = class(TPersistent);

Remarks

The THttpOptions class is used to establish an HTTP connection.

For more information about HTTP tunneling, see Network Tunneling.

See Also
- Network Tunneling

© 1997-2021 Devart. All Rights Reserved.
5.5.1.1.1 Members

**THttpOptions** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Enables an HTTP connection.</td>
</tr>
<tr>
<td>Password</td>
<td>Holds the password for HTTP authorization.</td>
</tr>
<tr>
<td>ProxyOptions</td>
<td>Holds a TProxyOptions object that contains settings for a proxy connection.</td>
</tr>
<tr>
<td>TrustServerCertificate</td>
<td>Verifies the server certificate during an SSL handshake.</td>
</tr>
<tr>
<td>Url</td>
<td>Holds the URL of the PHP script for HTTP tunneling.</td>
</tr>
<tr>
<td>Username</td>
<td>Holds the username for HTTP authorization.</td>
</tr>
</tbody>
</table>

5.5.1.1.2 Properties

Properties of the **THttpOptions** class.

For a complete list of the **THttpOptions** class members, see the **THttpOptions Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Enables an HTTP connection.</td>
</tr>
<tr>
<td>ProxyOptions</td>
<td>Holds a TProxyOptions object that contains settings for a proxy connection.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Password</strong></td>
<td>Holds the password for HTTP authorization.</td>
</tr>
<tr>
<td><strong>TrustServerCertificate</strong></td>
<td>Verifies the server certificate during an SSL handshake.</td>
</tr>
<tr>
<td><strong>Url</strong></td>
<td>Holds the URL of the PHP script for HTTP tunneling.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Holds the username for HTTP authorization.</td>
</tr>
</tbody>
</table>

**See Also**
- THttpOptions Class
- THttpOptions Class Members

```
property Enabled: boolean default False;
```

**Remarks**

The Enabled property specifies that a connection is established through HTTP.

**5.5.1.2.2 Password Property**

Holds the password for HTTP authorization.

**Class**

THttpOptions

**Syntax**
**property** Password: *string*;

**Remarks**
The Password property holds the password for the password-protected directory that contains the HTTP tunneling script.

**5.5.1.1.2.3 ProxyOptions Property**

Holds a TProxyOptions object that contains settings for a proxy connection.

**Class**

*THttpOptions*

**Syntax**

```delphi
property ProxyOptions: TProxyOptions;
```

**Remarks**
The ProxyOptions property holds a TProxyOptions object that contains settings for a proxy connection.

If it is necessary to connect to the server that resides in a different network, sometimes the client can only connect to it through a proxy server. In this case, besides the connection string, you have to set up ProxyOptions.

**5.5.1.1.2.4 TrustServerCertificate Property**

Verifies the server certificate during an SSL handshake.

**Class**

*THttpOptions*

**Syntax**

```delphi
property TrustServerCertificate: boolean default False;
```
Remarks
The TrustServerCertificate property specifies whether to verify the server certificate during an SSL handshake. When True, the Odac bypasses walking the certificate chain to verify the certificate. The default value is False.

Class
THttpOptions

Syntax

```markdown
property Url: string;
```

Remarks
The Url property holds the URL of the PHP script for HTTP tunneling. For example, if the script is located in the server root, the URL can be the following: http://server/tunnel.php.

Class
THttpOptions

Syntax

```markdown
property Username: string;
```

Remarks
The Username property holds the username for HTTP authorization.

Class
THttpOptions

Syntax

```markdown
property Username: string;
```

Remarks
The Username property holds the username for the password-protected directory that contains the HTTP tunneling script.
5.5.1.2 **TProxyOptions Class**

This class is used to establish an HTTP connection through a proxy server.

For a list of all members of this type, see [TProxyOptions](https://devart.com/docs/ddi/TProxyOptions) members.

**Unit**

**CRVio**

**Syntax**

```pascal
TProxyOptions = class(TPersistent);
```

**Remarks**

The **TProxyOptions** class is used to establish an HTTP connection through a proxy server.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Holds the hostname or IP address of the proxy server.</td>
</tr>
<tr>
<td>Password</td>
<td>Holds the proxy password.</td>
</tr>
<tr>
<td>Port</td>
<td>Holds the port number of the proxy server.</td>
</tr>
<tr>
<td>Username</td>
<td>Holds the proxy username.</td>
</tr>
</tbody>
</table>
Properties of the **TProxyOptions** class.

For a complete list of the **TProxyOptions** class members, see the [TProxyOptions Members](#) topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hostname</td>
<td>Holds the hostname or IP address of the proxy server.</td>
</tr>
<tr>
<td>Password</td>
<td>Holds the proxy password.</td>
</tr>
<tr>
<td>Port</td>
<td>Holds the port number of the proxy server.</td>
</tr>
<tr>
<td>Username</td>
<td>Holds the proxy username.</td>
</tr>
</tbody>
</table>

### See Also
- [TProxyOptions Class](#)
- [TProxyOptions Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

### 5.5.1.2.1 Hostname Property

Holds the hostname or IP address of the proxy server.

### Class

**TProxyOptions**

### Syntax

```
property Hostname: string;
```

### Remarks

The Hostname property holds the hostname or IP address of the proxy server.

© 1997-2021 Devart. All Rights Reserved.
5.5.1.2.2.2 Password Property

Holds the proxy password.

Class

TProxyOptions

Syntax

```plaintext
property Password: string;
```

Remarks

The Password property holds the proxy password.

© 1997-2021 Devart. All Rights Reserved.

5.5.1.2.2.3 Port Property

Holds the port number of the proxy server.

Class

TProxyOptions

Syntax

```plaintext
property Port: integer default 0;
```

Remarks

Use the Port property to specify the port number of the proxy server.

© 1997-2021 Devart. All Rights Reserved.

5.5.1.2.2.4 Username Property

Holds the proxy username.

Class

TProxyOptions
Syntax

```plaintext
property Username: string;
```

Remarks

The Username property holds the proxy username.

5.5.2 Enumerations

Enumerations in the CRVio unit.

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIPVersion</td>
<td>Specifies Internet Protocol version.</td>
</tr>
</tbody>
</table>

5.5.2.1 TIPVersion Enumeration

Specifies Internet Protocol version.

Unit

CRVio

Syntax

```plaintext
TIPVersion = (ivIPv4, ivIPv6, ivIPBoth);
```

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ivIPBoth</td>
<td>Specifies that either IPv6 or IPv4 Internet Protocol version is used</td>
</tr>
<tr>
<td>ivIPv4</td>
<td>Specifies that the IPv4 Internet Protocol version is used</td>
</tr>
<tr>
<td>ivIPv6</td>
<td>Specifies that the IPv6 Internet Protocol version is used</td>
</tr>
</tbody>
</table>
Remarks

Note: When the TIPVersion property is set to `ivIPBoth`, a connection attempt is made via IPv6 if it is enabled in the operating system settings. If the connection attempt fails, a new connection attempt is made via IPv4.

See Also

- `TOraSessionOptions.IPVersion`

5.6 **DAAlerter**

This unit contains the base class for the TOraAlerter component.

### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDAAlerter</strong></td>
<td>A base class that defines functionality for database event notification.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TAlertErrorEvent</strong></td>
<td>This type is used for the TDAAlerter.OnError event.</td>
</tr>
</tbody>
</table>

### Classes in the DAAlerter unit.

#### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDAAlerter</strong></td>
<td>A base class that defines functionality for database event notification.</td>
</tr>
</tbody>
</table>
5.6.1.1 TDAAlerter Class

A base class that defines functionality for database event notification.

For a list of all members of this type, see TDAAlerter members.

Unit

`DAAlerter`

Syntax

```plaintext
TDAAlerter = class(TComponent);
```

Remarks

TDAAlerter is a base class that defines functionality for descendant classes support database event notification. Applications never use TDAAlerter objects directly. Instead they use descendants of TDAAlerter.

The TDAAlerter component allows you to register interest in and handle events posted by a database server. Use TDAAlerter to handle events for responding to actions and database changes made by other applications. To get events, an application must register required events. To do this, set the Events property to the required events and call the Start method. When one of the registered events occurs OnEvent handler is called.

5.6.1.1.1 Members

**TDAAlerter** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to determine if TDAAlerter waits for messages.</td>
</tr>
<tr>
<td>AutoRegister</td>
<td>Used to automatically register events whenever</td>
</tr>
</tbody>
</table>
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendEvent</td>
<td>Sends an event with Name and content Message.</td>
</tr>
<tr>
<td>Start</td>
<td>Starts waiting process.</td>
</tr>
<tr>
<td>Stop</td>
<td>Stops waiting process.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Occurs if an exception occurs in waiting process</td>
</tr>
</tbody>
</table>

Properties of the `TDAAlerter` class.

For a complete list of the `TDAAlerter` class members, see the `TDAAlerter Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to determine if TDAAlerter waits for messages.</td>
</tr>
<tr>
<td>AutoRegister</td>
<td>Used to automatically register events whenever connection opens.</td>
</tr>
<tr>
<td>Connection</td>
<td>Used to specify the connection for TDAAlerter.</td>
</tr>
</tbody>
</table>

See Also
- `TDAAlerter Class`
5.6.1.1.2.1 Active Property

Used to determine if TDAAlerter waits for messages.

Class

TDAAlerter

Syntax

```plaintext
property Active: boolean default False;
```

Remarks

Check the Active property to know whether TDAAlerter waits for messages or not. Set it to True to register events.

See Also

- Start
- Stop
5.6.1.2.3 Connection Property

Used to specify the connection for TDAAlerter.

Class

TDAAlerter

Syntax

```property` Connection: TCustomDAConnection;```

Remarks

Use the Connection property to specify the connection for TDAAlerter.

5.6.1.3 Methods

Methods of the TDAAlerter class.

For a complete list of the TDAAlerter class members, see the TDAAlerter Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendEvent</td>
<td>Sends an event with Name and content Message.</td>
</tr>
<tr>
<td>Start</td>
<td>Starts waiting process.</td>
</tr>
<tr>
<td>Stop</td>
<td>Stops waiting process.</td>
</tr>
</tbody>
</table>

See Also

- TDAAlerter Class
- TDAAlerter Class Members
5.6.1.1.3.1  SendEvent Method

Sends an event with Name and content Message.

Class
TDAAlerter

Syntax

\begin{verbatim}
procedure SendEvent(const EventName: string; const Message: string);
\end{verbatim}

Parameters

\begin{itemize}
  \item \textit{EventName}
    \begin{itemize}
      \item Holds the event name.
    \end{itemize}
  \item \textit{Message}
    \begin{itemize}
      \item Holds the content Message of the event.
    \end{itemize}
\end{itemize}

Remarks

Use SendEvent procedure to send an event with Name and content Message.

5.6.1.1.3.2  Start Method

Starts waiting process.

Class
TDAAlerter

Syntax

\begin{verbatim}
procedure Start;
\end{verbatim}

Remarks

Call the Start method to run waiting process. After starting TDAAlerter waits for messages with names defined by the Events property.

See Also

- Stop
5.6.1.3.3 Stop Method

Stops waiting process.

Class

TDAAlerter

Syntax

```
procedure Stop;
```

Remarks

Call Stop method to end waiting process.

See Also

- Start

5.6.1.4 Events

Events of the TDAAlerter class.

For a complete list of the TDAAlerter class members, see the TDAAlerter Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Occurs if an exception occurs in waiting process</td>
</tr>
</tbody>
</table>

See Also

- TDAAlerter Class
- TDAAlerter Class Members
5.6.1.4.1 OnError Event

Occurs if an exception occurs in waiting process

Class

TDAAlerter

Syntax

```plaintext
property OnError: TAlerterErrorEvent;
```

Remarks

The OnError event occurs if an exception occurs in waiting process. Alerter stops in this case. The exception can be accessed using the E parameter.

5.6.2 Types

Types in the DAAlerter unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAlerterErrorEvent</td>
<td>This type is used for the TDAAlerter.OnError event.</td>
</tr>
</tbody>
</table>

5.6.2.1 TAlerterErrorEvent Procedure Reference

This type is used for the TDAAlerter.OnError event.

Unit

DAAlerter
### Syntax

```
TAlertErrorEvent = procedure (Sender: TDAAlert; E: Exception) of object;
```

### Parameters

**Sender**
- An object that raised the event.

**E**
- Exception object.

### 5.7 DADump

This unit contains the base class for the TOraDump component.

### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDADump</td>
<td>A base class that defines functionality for descendant classes that dump database objects to a script.</td>
</tr>
<tr>
<td>TDADumpOptions</td>
<td>This class allows setting up the behaviour of the TDADump class.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDABackupProgressEvent</td>
<td>This type is used for the TDADump.OnBackupProgress event.</td>
</tr>
<tr>
<td>TDARestoreProgressEvent</td>
<td>This type is used for the TDADump.OnRestoreProgress event.</td>
</tr>
</tbody>
</table>
5.7.1 **Classes**

Classes in the **DADump** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDADump</td>
<td>A base class that defines functionality for descendant classes that dump database objects to a script.</td>
</tr>
<tr>
<td>TDADumpOptions</td>
<td>This class allows setting up the behaviour of the TDADump class.</td>
</tr>
</tbody>
</table>

5.7.1.1 **TDADump Class**

A base class that defines functionality for descendant classes that dump database objects to a script.

For a list of all members of this type, see **TDADump** members.

**Unit**

**DADump**

**Syntax**

```
TDADump = class(TComponent);
```

**Remarks**

TDADump is a base class that defines functionality for descendant classes that dump database objects to a script. Applications never use TDADump objects directly. Instead they use descendants of TDADump.

Use TDADump descendents to dump database objects, such as tables, stored procedures, and functions for backup or for transferring the data to another SQL server. The dump contains SQL statements to create the table or other database objects and/or populate the table.
5.7.1.1.1 Members

**TDADump** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Used to specify a connection object that will be used to connect to a data store.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to specify the behaviour of a TDADump component.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to set or get the dump script.</td>
</tr>
<tr>
<td>TableNames</td>
<td>Used to set the names of the tables to dump.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup</td>
<td>Dumps database objects to the <strong>TDADump.SQL</strong> property.</td>
</tr>
<tr>
<td>BackupQuery</td>
<td>Dumps the results of a particular query.</td>
</tr>
<tr>
<td>BackupToFile</td>
<td>Dumps database objects to the specified file.</td>
</tr>
<tr>
<td>BackupToStream</td>
<td>Dumps database objects to the stream.</td>
</tr>
<tr>
<td>Restore</td>
<td>Executes a script contained in the SQL property.</td>
</tr>
<tr>
<td>RestoreFromFile</td>
<td>Executes a script from a file.</td>
</tr>
<tr>
<td>RestoreFromStream</td>
<td>Executes a script received from the stream.</td>
</tr>
</tbody>
</table>
Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Occurs when Oracle raises some error on TDADump.Restore.</td>
</tr>
<tr>
<td>OnRestoreProgress</td>
<td>Occurs to indicate the TDADump.Restore, TDADump.RestoreFromFile, or TDADump.RestoreFromStream method execution progress.</td>
</tr>
</tbody>
</table>

5.7.1.2 Properties

Properties of the TDADump class.

For a complete list of the TDADump class members, see the TDADump Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Used to specify a connection object that will be used to connect to a data store.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to specify the behaviour of a TDADump component.</td>
</tr>
</tbody>
</table>
Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to set or get the dump script.</td>
</tr>
<tr>
<td>TableName</td>
<td>Used to set the names of the tables to dump.</td>
</tr>
</tbody>
</table>

See Also

- **TDADump Class**
- **TDADump Class Members**

5.7.1.2.1.2 Connection Property

Used to specify a connection object that will be used to connect to a data store.

Class

**TDADump**

Syntax

```property
property Connection: TCustomDAConnection;
```

Remarks

Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, link an instance of a TCustomDAConnection descendant to the Connection property.

See Also
5.7.1.1.2.2 Debug Property

Used to display the statement that is being executed and the values and types of its parameters.

Class

TDADump

Syntax

```plaintext
property Debug: boolean default False;
```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the OdacVcl unit to the uses clause of any unit in your project to make the Debug property work.

**Note**: If TOraSQLMonitor is used in the project and the TOraSQLMonitor.Active property is set to False, the debug window is not displayed.

See Also

- TCustomDADataSet.Debug
- TCustomDASQL.Debug
property Options: TDADumpOptions;

Remarks
Use the Options property to specify the behaviour of a TDADump component.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDrop</td>
<td>Used to add drop statements to a script before creating statements.</td>
</tr>
<tr>
<td>CompleteInsert</td>
<td>Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.</td>
</tr>
<tr>
<td>GenerateHeader</td>
<td>Used to add a comment header to a script.</td>
</tr>
<tr>
<td>QuoteNames</td>
<td>Used for TDADump to quote all database object names in generated SQL statements.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.7.1.1.2.4 SQL Property

Used to set or get the dump script.

Class
TDADump

Syntax
property SQL: TStrings;

Remarks
Use the SQL property to get or set the dump script. The SQL property stores script that is executed by the Restore method. This property will store the result of Backup and BackupQuery. At design time the SQL property can be edited by invoking the String List editor in Object Inspector.

See Also
- Restore
- Backup
5.7.1.2.5 TableNames Property

Used to set the names of the tables to dump.

Class

TDADump

Syntax

```plaintext
property TableNames: string;
```

Remarks

Use the TableNames property to set the names of the tables to dump. Table names must be separated with semicolons. If the property is empty, the `Backup` method will dump all available tables.

See Also

- `Backup`

5.7.1.3 Methods

Methods of the `TDADump` class.

For a complete list of the `TDADump` class members, see the [TDADump Members] topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup</td>
<td>Dumps database objects to the <code>TDADump.SQL</code> property.</td>
</tr>
<tr>
<td>BackupQuery</td>
<td>Dumps the results of a particular query.</td>
</tr>
<tr>
<td>BackupToFile</td>
<td>Dumps database objects to the specified file.</td>
</tr>
</tbody>
</table>
**Backup Method**

Dumps database objects to the SQL property.

**Class**

**TDADump**

**Syntax**

```
procedure Backup;
```

**Remarks**

Call the Backup method to dump database objects. The result script will be stored in the SQL property.

**See Also**

- **SQL**
- **Restore**
- **BackupToFile**
- **BackupToStream**
- **BackupQuery**

© 1997-2021 Devart. All Rights Reserved.
5.7.1.3.2 BackupQuery Method

Dumps the results of a particular query.

Class
TDADump

Syntax

```
procedure BackupQuery(const Query: string);
```

Parameters

* **Query**
  
  Holds a query used for data selection.

Remarks

Call the BackupQuery method to dump the results of a particular query. Query must be a valid select statement. If this query selects data from several tables, only data of the first table in the from list will be dumped.

See Also

- [Restore](#)
- [Backup](#)
- [BackupToFile](#)
- [BackupToStream](#)

© 1997-2021 Devart. All Rights Reserved.

5.7.1.3.3 BackupToFile Method

Dumps database objects to the specified file.

Class
TDADump

Syntax

```
procedure BackupToFile(const FileName: string; const Query: string = '');
```

Parameters

© 2021 Devart
FileName
Holds the file name to dump database objects to.

Query
Your query to receive the data for dumping.

Remarks
Call the BackupToFile method to dump database objects to the specified file.

See Also
- RestoreFromStream
- Backup
- BackupToStream

5.7.1.1.3.4 BackupToStream Method

Dumps database objects to the stream.

Class
TDADump

Syntax

```delphi
procedure BackupToStream(Stream: TStream; const Query: string = '');
```

Parameters

Stream
Holds the stream to dump database objects to.

Query
Your query to receive the data for dumping.

Remarks
Call the BackupToStream method to dump database objects to the stream.

See Also
- RestoreFromStream
- Backup
5.7.1.1.3.5 Restore Method

Executes a script contained in the SQL property.

Class
TDADump

Syntax

```pascal
procedure Restore;
```

Remarks
Call the Restore method to execute a script contained in the SQL property.

See Also
- RestoreFromFile
- RestoreFromStream
- Backup
- SQL

5.7.1.1.3.6 RestoreFromFile Method

Executes a script from a file.

Class
TDADump

Syntax

```pascal
procedure RestoreFromFile(const FileName: string);
```

Parameters

- `FileName`  
  Holds the file name to execute a script from.
Remarks

Call the RestoreFromFile method to execute a script from the specified file.

See Also

- Restore
- RestoreFromStream
- BackupToFile

Executes a script received from the stream.

Class

TDADump

Syntax

```delphi
procedure RestoreFromStream(Stream: TStream);
```

Parameters

- **Stream**
  - Holds a stream to receive a script to be executed.

Remarks

Call the RestoreFromStream method to execute a script received from the stream.

See Also

- Restore
- RestoreFromFile
- BackupToStream
Events of the **TDADump** class.

For a complete list of the **TDADump** class members, see the [TDADump Members](#) topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnError</strong></td>
<td>Occurs when Oracle raises some error on <strong>TDADump.Restore</strong>.</td>
</tr>
<tr>
<td><strong>OnRestoreProgress</strong></td>
<td>Occurs to indicate the <strong>TDADump.Restore</strong>, <strong>TDADump.RestoreFromFile</strong>, or <strong>TDADump.RestoreFromStream</strong> method execution progress.</td>
</tr>
</tbody>
</table>

See Also
- **TDADump Class**
- **TDADump Class Members**

© 1997-2021
Devar. All Rights Reserved.

5.7.1.4.1 OnBackupProgress Event


Class
TDADump

Syntax

property OnBackupProgress: TDABackupProgressEvent;

Remarks

The OnBackupProgress event occurs several times during the dumping process of the Backup, M:Devart.Dac.TDADump.BackupToFile(System.String), or M:Devart.Dac.TDADump.BackupToStream(Borland.Vcl.TStream) method execution and indicates its progress. ObjectName parameter indicates the name of the currently dumping database object. ObjectNum shows the number of the current database object in the backup queue starting from zero. ObjectCount shows the quantity of database objects to dump. Percent parameter shows the current percentage of the current table data dumped, not the current percentage of the entire dump process.

See Also

• Backup
• BackupToFile
• BackupToStream

© 1997-2021 Devart. All Rights Reserved.

5.7.1.1.4.2 OnError Event

Occurs when Oracle raises some error on Restore.

Class

TDADump

Syntax

property OnError: TOnErrorEvent;

Remarks

The OnError event occurs when Oracle raises some error on Restore.

Action indicates the action to take when the OnError handler exits. On entry into the handler, Action is always set to eaException.
**Note:** You should add the DAScript module to the 'uses' list to use the OnError event handler.

5.7.1.4.3 OnRestoreProgress Event

Occurs to indicate the [Restore](#), [RestoreFromFile](#), or [RestoreFromStream](#) method execution progress.

**Class**

`TDADump`

**Syntax**

```delphi
property OnRestoreProgress: TDARestoreProgressEvent;
```

**Remarks**

The OnRestoreProgress event occurs several times during the dumping process of the [Restore](#), [RestoreFromFile](#), or [RestoreFromStream](#) method execution and indicates its progress. The Percent parameter of the OnRestoreProgress event handler indicates the percentage of the whole restore script execution.

**See Also**

- [Restore](#)
- [RestoreFromFile](#)
- [RestoreFromStream](#)

5.7.1.2 TDADumpOptions Class

This class allows setting up the behaviour of the TDADump class.

For a list of all members of this type, see [TDADumpOptions](#) members.

**Unit**

`DADump`

**Syntax**
5.7.1.2.1 Members

**TDADumpOptions** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDrop</td>
<td>Used to add drop statements to a script before creating statements.</td>
</tr>
<tr>
<td>CompleteInsert</td>
<td>Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.</td>
</tr>
<tr>
<td>GenerateHeader</td>
<td>Used to add a comment header to a script.</td>
</tr>
<tr>
<td>QuoteNames</td>
<td>Used for TDADump to quote all database object names in generated SQL statements.</td>
</tr>
</tbody>
</table>

5.7.1.2.2 Properties

Properties of the **TDADumpOptions** class.

For a complete list of the **TDADumpOptions** class members, see the **TDADumpOptions Members** topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddDrop</td>
<td>Used to add drop statements to a script before creating statements.</td>
</tr>
<tr>
<td>CompleteInsert</td>
<td>Used to explicitly specify the table fields names when creating statements.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GenerateHeader</td>
<td>Used to add a comment header to a script.</td>
</tr>
<tr>
<td>QuoteNames</td>
<td>Used for TDADump to quote all database object names in generated SQL statements.</td>
</tr>
</tbody>
</table>

See Also
- TDADumpOptions Class
- TDADumpOptions Class Members

### 5.7.1.2.2.1 AddDrop Property

Used to add drop statements to a script before creating statements.

**Class**

TDADumpOptions

**Syntax**

```plaintext
property AddDrop: boolean default True;
```

**Remarks**

Use the AddDrop property to add drop statements to a script before creating statements.

### 5.7.1.2.2.2 CompleteInsert Property

Used to explicitly specify the table fields names when generating the INSERT SQL query. The default value is False.

**Class**

TDADumpOptions
Syntax

```plaintext
property CompleteInsert: boolean default False;
```

Remarks

If the CompleteInsert property is set to True, SQL query will include the field names, for example:

```
INSERT INTO dept(deptno, dname, loc) VALUES ('10', 'ACCOUNTING', 'NEW YORK');
```

If False, it won't include the field names, for example:

```
INSERT INTO dept VALUES ('10', 'ACCOUNTING', 'NEW YORK');
```

5.7.1.2.2.3 GenerateHeader Property

Used to add a comment header to a script.

Class

TDADumpOptions

Syntax

```plaintext
property GenerateHeader: boolean default True;
```

Remarks

Use the GenerateHeader property to add a comment header to a script. It contains script generation date, DAC version, and some other information.

5.7.1.2.2.4 QuoteNames Property

Used for TDADump to quote all database object names in generated SQL statements.

Class

TDADumpOptions
## Syntax

```plaintext
property QuoteNames: boolean default False;
```

## Remarks

If the QuoteNames property is True, TDADump quotes all database object names in generated SQL statements.

## 5.7.2 Types

Types in the **DADump** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDABackupProgressEvent</td>
<td>This type is used for the <strong>TDADump.OnBackupProgress</strong> event.</td>
</tr>
<tr>
<td>TDAOverwriteProgressEvent</td>
<td>This type is used for the <strong>TDADump.OnOverwriteProgress</strong> event.</td>
</tr>
</tbody>
</table>

## 5.7.2.1 TDABackupProgressEvent Procedure Reference

This type is used for the **TDADump.OnBackupProgress** event.

## Unit

**DADump**

## Syntax

```plaintext
TDABackupProgressEvent = procedure (Sender: TObject; ObjectName: string; ObjectNum: integer; ObjectCount: integer; Percent: integer) of object;
```

## Parameters
5.7.2.2 TDARestoreProgressEvent Procedure Reference

This type is used for the TDADump.OnRestoreProgress event.

Unit

DADump

Syntax

TDARestoreProgressEvent = procedure (Sender: TObject; Percent: integer) of object;

Parameters

Sender
An object that raised the event.

Percent
The percentage of the whole restore script execution.

5.8 DALoader

This unit contains the base class for the TOraLoader component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Classes

Classes in the **DALoader** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDAColumn</td>
<td>Represents the attributes for column loading.</td>
</tr>
<tr>
<td>TDAColumns</td>
<td>Holds a collection of TDAColumn objects.</td>
</tr>
<tr>
<td>TDALoader</td>
<td>This class allows loading external data into database.</td>
</tr>
<tr>
<td>TDALoaderOptions</td>
<td>Allows loading external data into database.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDAPutDataEvent</td>
<td>This type is used for the TDALoader.OnPutData event.</td>
</tr>
<tr>
<td>TGetColumnDataEvent</td>
<td>This type is used for the TDALoader.OnGetColumnData</td>
</tr>
<tr>
<td>TLoaderProgressEvent</td>
<td>This type is used for the TDALoader.OnProgress event.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
5.8.1.1  TDAColumn Class

Represents the attributes for column loading.

For a list of all members of this type, see TDAColumn members.

Unit

DALoader

Syntax

TDAColumn = class(TCollectionItem);

Remarks

Each TDALoader uses TDAColumns to maintain a collection of TDAColumn objects. TDAColumn object represents the attributes for column loading. Every TDAColumn object corresponds to one of the table fields with the same name as its TDAColumn.Name property.

To create columns at design-time use the column editor of the TDALoader component.

See Also

- TDALoader
- TDAColumns

© 1997-2021

Devart. All Rights Reserved.  Request Support  DAC Forum  Provide Feedback

5.8.1.1.1  Members

TDAColumn class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldType</td>
<td>Used to specify the types of values that will be loaded.</td>
</tr>
<tr>
<td>Name</td>
<td>Used to specify the field name of loading table.</td>
</tr>
</tbody>
</table>

© 1997-2021

Devart. All Rights Reserved.  Request Support  DAC Forum  Provide Feedback
5.8.1.1.2 Properties

Properties of the TDAColumn class.

For a complete list of the TDAColumn class members, see the TDAColumn Members topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldType</td>
<td>Used to specify the types of values that will be loaded.</td>
</tr>
<tr>
<td>Name</td>
<td>Used to specify the field name of loading table.</td>
</tr>
</tbody>
</table>

See Also

- TDAColumn Class
- TDAColumn Class Members

5.8.1.1.2.1 FieldType Property

Used to specify the types of values that will be loaded.

Class

TDAColumn

Syntax

```pascal
property FieldType: TFieldType default ftString;
```

Remarks

Use the FieldType property to specify the types of values that will be loaded. Field types for columns may not match data types for the corresponding fields in the database table. TDALoader will cast data values to the types of their fields.
5.8.1.2.2 Name Property

Used to specify the field name of loading table.

Class

TDAColumn

Syntax

```
property Name: string;
```

Remarks

Each TDAColumn corresponds to one field of the loading table. Use the Name property to specify the name of this field.

See Also

- FieldType

5.8.1.2 TDAColumns Class

Holds a collection of TDAColumn objects.

For a list of all members of this type, see TDAColumns members.

Unit

DALoader

Syntax

```
TDAColumns = class(TOwnedCollection);
```

Remarks

Each TDAColumns holds a collection of TDAColumn objects. TDAColumns maintains an index of the columns in its Items array. The Count property contains the number of columns in the collection. At design-time, use the Columns editor to add, remove, or modify columns.

See Also

- TDALoader
• **TDAColumn**

© 1997-2021
Devart. All Rights Reserved.

5.8.1.2.1 Members

**TDAColumns** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access individual columns.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

5.8.1.2.2 Properties

Properties of the **TDAColumns** class.

For a complete list of the **TDAColumns** class members, see the [TDAColumns Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access individual columns.</td>
</tr>
</tbody>
</table>

### See Also

• **TDAColumns Class**

© 1997-2021
Devart. All Rights Reserved.

5.8.1.2.2.1 Items Property (Indexer)

Used to access individual columns.

### Class
**TDAColumns**

**Syntax**

```property
Items[Index: integer]: TDAColumn; default;
```

**Parameters**

- `Index`
  - Holds the Index of TDAColumn to refer to.

**Remarks**

Use the Items property to access individual columns. The value of the Index parameter corresponds to the Index property of TDAColumn.

**See Also**

- TDAColumn

---

**5.8.1.3 TDALoader Class**

This class allows loading external data into database.

For a list of all members of this type, see TDALoader members.

**Unit**

DALoader

**Syntax**

```TDALoader = class(TComponent);
```

**Remarks**

TDALoader allows loading external data into database. To specify the name of loading table, set the TDALoader.TableName property. Use the TDALoader.Columns property to access individual columns. Write the TDALoader.OnGetColumnData or TDALoader.OnPutData event handlers to read external data and pass it to the database. Call the TDALoader.Load method to start loading data.

**See Also**
5.8.1.3.1 Members

**TDALoader** class overview.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>Used to add a TDAColumn object for each field that will be loaded.</td>
</tr>
<tr>
<td>Connection</td>
<td>See the TOraLoader.Session property.</td>
</tr>
<tr>
<td>TableName</td>
<td>Used to specify the name of the table to which data will be loaded.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateColumns</td>
<td>Creates TDAColumn objects for all fields of the table with the same name as TDALoader.TableName.</td>
</tr>
<tr>
<td>Load</td>
<td>Starts loading data.</td>
</tr>
<tr>
<td>LoadFromDataSet</td>
<td>Loads data from the specified dataset.</td>
</tr>
<tr>
<td>PutColumnData</td>
<td>Overloaded. Puts the value of individual columns.</td>
</tr>
</tbody>
</table>

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGetColumnData</td>
<td>Occurs when it is needed to put column values.</td>
</tr>
<tr>
<td>OnProgress</td>
<td>Occurs if handling data</td>
</tr>
</tbody>
</table>
loading progress of the `TDALoader.LoadFromDataSet` method is needed.

**OnPutData**

Occurs when putting loading data by rows is needed.

### 5.8.1.3.2 Properties

Properties of the **TDALoader** class.

For a complete list of the **TDALoader** class members, see the [TDALoader Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>Used to add a <strong>TDAColumn</strong> object for each field that will be loaded.</td>
</tr>
<tr>
<td>Connection</td>
<td>See the <strong>TOraLoader.Session</strong> property.</td>
</tr>
<tr>
<td>TableName</td>
<td>Used to specify the name of the table to which data will be loaded.</td>
</tr>
</tbody>
</table>

**See Also**

- **TDALoader Class**
- **TDALoader Class Members**

© 1997-2021 Devart. All Rights Reserved.
Syntax

```pascal
property Columns: TDAColumns stored IsColumnsStored;
```

Remarks

Use the Columns property to add a `TDAColumn` object for each field that will be loaded.

See Also

- `TDAColumns`

5.8.1.3.2.2 Connection Property

See the `TOraLoader.Session` property.

Class

`TDALoader`

Syntax

```pascal
property Connection: TCustomDADevelopment;
```

Remarks

Use the Connection property to specify `TCustomDADevelopment` in which `TDALoader` will be executed. If Connection is not connected, the `Load` method calls `TCustomDADevelopment.Connect`.

See Also

- `TCustomDADevelopment`
**TDALoader**

**Syntax**

```
property TableName: string;
```

**Remarks**

Set the TableName property to specify the name of the table to which data will be loaded. Add TDAColumn objects to `Columns` for the fields that are needed to be loaded.

**See Also**

- [TDAColumn](#)
- [TCustomDACConnection.GetTableNames](#)

---

### 5.8.1.3.3 Methods

Methods of the **TDALoader** class.

For a complete list of the **TDALoader** class members, see the **TDALoader Members** topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateColumns</td>
<td>Creates <a href="#">TDAColumn</a> objects for all fields of the table with the same name as <code>TDALoader.TableName</code>.</td>
</tr>
<tr>
<td>Load</td>
<td>Starts loading data.</td>
</tr>
<tr>
<td>LoadFromDataSet</td>
<td>Loads data from the specified dataset.</td>
</tr>
<tr>
<td>PutColumnData</td>
<td>Overloaded. Puts the value of individual columns.</td>
</tr>
</tbody>
</table>

**See Also**

- [TDALoader Class](#)
- [TDALoader Class Members](#)
5.8.1.3.3.1 CreateColumns Method

Creates TDAColumn objects for all fields of the table with the same name as TableName.

Class
TDALoader

Syntax

```pascal
procedure CreateColumns;
```

Remarks
Call the CreateColumns method to create TDAColumn objects for all fields of the table with the same name as TableName. If columns were created before, they will be recreated. You can call CreateColumns from the component popup menu at design-time. After you can customize column loading by setting properties of TDAColumn objects.

See Also
- TDAColumn
- TableName

© 1997-2021 Devart. All Rights Reserved.
Request Support    DAC Forum    Provide Feedback

5.8.1.3.3.2 Load Method

Starts loading data.

Class
TDALoader

Syntax

```pascal
procedure Load; virtual;
```

Remarks
Call the Load method to start loading data. At first it is necessary to create columns and write one of the OnPutData or OnGetColumnData event handlers.
5.8.1.3.3.3 LoadFromDataSet Method

Loads data from the specified dataset.

Class
TDALoader

Syntax

```plaintext
procedure LoadFromDataSet(DataSet: TDataSet);
```

Parameters

- **DataSet**: Holds the dataset to load data from.

Remarks

Call the LoadFromDataSet method to load data from the specified dataset. There is no need to create columns and write event handlers for OnPutData and OnGetColumnData before calling this method.

5.8.1.3.3.4 PutColumnData Method

Puts the value of individual columns.

Class
TDALoader

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PutColumnData(Col: integer; Row: integer;)</td>
<td>Puts the value of individual columns by the</td>
</tr>
</tbody>
</table>
**const** Value: variant) column index.

**PutColumnData(const ColName: string; Row: integer; const Value: variant)** Puts the value of individual columns by the column name.

Puts the value of individual columns by the column index.

**Class**

**TDALoader**

**Syntax**

```pascal
procedure PutColumnData(Col: integer; Row: integer; **const** Value: variant); overload; **virtual**;
```

**Parameters**

- **Col**
  - Holds the index of a loading column. The first column has index 0.

- **Row**
  - Holds the number of loading row. Row starts from 1.

- **Value**
  - Holds the column value.

**Remarks**

Call the PutColumnData method to put the value of individual columns. PutColumnData can be only called from the OnPutData event handler. The Col parameter indicates the index of loading column. The first column has index 0. The Row parameter indicates the number of the loading row. Row starts from 1.

This overloaded method works faster because it searches the right index by its index, not by the index name.

The value of a column should be assigned to the Value parameter.

**See Also**

- **TDALoader.OnPutData**
Puts the value of individual columns by the column name.

Class

**TDALoader**

Syntax

```pascal
procedure PutColumnData(const ColName: string; Row: integer; const Value: variant); overload;
```

Parameters

- **ColName**
  - Holds the name of a loading column.

- **Row**
  - Holds the number of loading row. Row starts from 1.

- **Value**
  - Holds the column value.

5.8.1.3.4  Events

Events of the **TDALoader** class.

For a complete list of the **TDALoader** class members, see the [TDALoader Members](#) topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnGetColumnData</strong></td>
<td>Occurs when it is needed to put column values.</td>
</tr>
<tr>
<td><strong>OnProgress</strong></td>
<td>Occurs if handling data loading progress of the <strong>TDALoader.LoadFromData</strong> Set method is needed.</td>
</tr>
<tr>
<td><strong>OnPutData</strong></td>
<td>Occurs when putting loading data by rows is needed.</td>
</tr>
</tbody>
</table>

See Also

- [TDALoader Class](#)
- [TDALoader Class Members](#)
5.8.1.3.4.1 OnGetColumnData Event

Occurs when it is needed to put column values.

Class

**TDALoader**

Syntax

```
property OnGetColumnData: TGetColumnDataEvent;
```

Remarks

Write the OnGetColumnData event handler to put column values. **TDALoader** calls the OnGetColumnData event handler for each column in the loop. Column points to a **TDAColumn** object that corresponds to the current loading column. Use its Name or Index property to identify what column is loading. The Row parameter indicates the current loading record. TDALoader increments the Row parameter when all the columns of the current record are loaded. The first row is 1. Set EOF to True to stop data loading. Fill the Value parameter by column values. To start loading call the **Load** method.

Another way to load data is using the **OnPutData** event.

Example

This handler loads 1000 rows.

```
procedure TfmMain.GetColumnData(Sender: TObject; Column: TDAColumn; Row: Integer; var Value: Variant; var EOF: Boolean);
begin
  if Row <= 1000 then begin
    case Column.Index of
      0: Value := Row;
      1: Value := Random(100);
      2: Value := Random*100;
      3: Value := 'abc01234567890123456789';
      4: Value := Date;
    else
      Value := Null;
    end;
  end;
else
  EOF := True;
end;
```
5.8.1.3.4.2 OnProgress Event

Occurs if handling data loading progress of the `LoadFromDataSet` method is needed.

**Class**

`TDALoader`

**Syntax**

```pascal
property OnProgress: TLoaderProgressEvent;
```

**Remarks**

Add a handler to this event if you want to handle data loading progress of the `LoadFromDataSet` method.

**See Also**

- `OnPutData`
- `Load`

5.8.1.3.4.3 OnPutData Event

Occurs when putting loading data by rows is needed.

**Class**

`TDALoader`

**Syntax**

```pascal
property OnPutData: TDAPutDataEvent;
```

**Remarks**

Add a handler to this event if you want to handle data loading progress of the `LoadFromDataSet` method.
Write the OnPutData event handler to put loading data by rows.

Note that rows should be loaded from the first in the ascending order.

To start loading, call the Load method. It is more effective way to load data in comparison with using OnGetColumnData. The OnPutData event handler must send column data by the TDALoader.PutColumnData method. TDALoader will flush data to Oracle when it is needed.

5.8.1.4  TDALoaderOptions Class

Allows loading external data into database.

For a list of all members of this type, see TDALoaderOptions members.

Unit

DALoader

Syntax

```
TDALoaderOptions = class(TPersistent);
```

5.8.1.4.1 Members

TDALoaderOptions class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseBlankValues</td>
<td>Forces ODAC to fill the buffer with null values after loading a row to the database.</td>
</tr>
</tbody>
</table>
Properties of the `TDALoaderOptions` class.

For a complete list of the `TDALoaderOptions` class members, see the `TDALoaderOptions Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseBlankValues</td>
<td>Forces ODAC to fill the buffer with null values after loading a row to the database.</td>
</tr>
</tbody>
</table>

See Also
- `TDALoaderOptions Class`
- `TDALoaderOptions Class Members`

5.8.1.4.2.1 UseBlankValues Property

Forces ODAC to fill the buffer with null values after loading a row to the database.

Class
`TDALoaderOptions`

Syntax

```plaintext
property UseBlankValues: boolean default True;
```

Remarks

Used to force ODAC to fill the buffer with null values after loading a row to the database.
5.8.2 Types

Types in the **DALoader** unit.

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDAPutDataEvent</strong></td>
<td>This type is used for the <strong>TDALoader.OnPutData</strong> event.</td>
</tr>
<tr>
<td><strong>TGetColumnDataEvent</strong></td>
<td>This type is used for the <strong>TDALoader.OnGetColumnData</strong> event.</td>
</tr>
<tr>
<td><strong>TLoaderProgressEvent</strong></td>
<td>This type is used for the <strong>TDALoader.OnProgress</strong> event.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

### 5.8.2.1 TDAPutDataEvent Procedure Reference

This type is used for the **TDALoader.OnPutData** event.

**Unit**

**DALoader**

**Syntax**

```pascal
TDAPutDataEvent = procedure (Sender: TDALoader) of object;
```

**Parameters**

- **Sender**
  - An object that raised the event.

© 1997-2021 Devart. All Rights Reserved.

### 5.8.2.2 TGetColumnDataEvent Procedure Reference

This type is used for the **TDALoader.OnGetColumnData** event.

**Unit**
### DALoader

#### Syntax

```pascal
TGetColumnDataEvent = procedure (Sender: TObject; Column: TDAColumn; Row: integer; var Value: variant; var IsEOF: boolean) of object;
```

#### Parameters

- **Sender**
  - An object that raised the event.

- **Column**
  - Points to `TDAColumn` object that corresponds to the current loading column.

- **Row**
  - Indicates the current loading record.

- **Value**
  - Holds column values.

- **IsEOF**
  - True, if data loading needs to be stopped.

---

#### TLoaderProgressEvent Procedure Reference

This type is used for the `TDALoader.OnProgress` event.

#### Unit

`DALoader`

#### Syntax

```pascal
TLoaderProgressEvent = procedure (Sender: TObject; Percent: integer) of object;
```

#### Parameters

- **Sender**
  - An object that raised the event.

- **Percent**
  - Percentage of the load operation progress.
## 5.9 DAScript

This unit contains the base class for the TOraScript component.

### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDAScript</td>
<td>Makes it possible to execute several SQL statements one by one.</td>
</tr>
<tr>
<td>TDAStatement</td>
<td>This class has attributes and methods for controlling single SQL statement of a script.</td>
</tr>
<tr>
<td>TDAStatements</td>
<td>Holds a collection of TDAStatement objects.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAfterStatementExecuteEvent</td>
<td>This type is used for the TDAScript.AfterExecute event.</td>
</tr>
<tr>
<td>TBeforeStatementExecuteEvent</td>
<td>This type is used for the TDAScript.BeforeExecute event.</td>
</tr>
<tr>
<td>TOnErrorEvent</td>
<td>This type is used for the TDAScript.OnError event.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TErrorAction</td>
<td>Indicates the action to take when the OnError handler exits.</td>
</tr>
</tbody>
</table>
5.9.1 Classes

Classes in the DAScript unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDAScript</td>
<td>Makes it possible to execute several SQL statements one by one.</td>
</tr>
<tr>
<td>TDAStatement</td>
<td>This class has attributes and methods for controlling single SQL statement of a script.</td>
</tr>
<tr>
<td>TDAStatements</td>
<td>Holds a collection of TDAStatement objects.</td>
</tr>
</tbody>
</table>

5.9.1.1 TDAScript Class

Makes it possible to execute several SQL statements one by one.

For a list of all members of this type, see TDAScript members.

Unit

DAScript

Syntax

TDAScript = class(TComponent);

Remarks

Often it is necessary to execute several SQL statements one by one. This can be performed using a lot of components such as TCustomDASQL descendants. Usually it isn't the best solution. With only one TDAScript descendent component you can execute several SQL statements as one. This sequence of statements is called script. To separate single statements use semicolon (;) or slash (/) and for statements that can contain semicolon, only slash. Note that slash must be the first character in line.

Errors that occur during execution can be processed in the TDAScript.OnError event handler. By default, on error TDAScript shows exception and continues execution.
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Used to specify the connection in which the script will be executed.</td>
</tr>
<tr>
<td>DataSet</td>
<td>Refers to a dataset that holds the result set of query execution.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the script execution and all its parameter values.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Used to set the delimiter string that separates script statements.</td>
</tr>
<tr>
<td>EndLine</td>
<td>Used to get the current statement last line number in a script.</td>
</tr>
<tr>
<td>EndOffset</td>
<td>Used to get the offset in the last line of the current statement.</td>
</tr>
<tr>
<td>EndPos</td>
<td>Used to get the end position of the current statement.</td>
</tr>
<tr>
<td>Macros</td>
<td>Used to change SQL script text in design- or run-time easily.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to get or set script text.</td>
</tr>
<tr>
<td>StartLine</td>
<td>Used to get the current statement start line number in a script.</td>
</tr>
<tr>
<td>StartOffset</td>
<td>Used to get the offset in the first line of the current statement.</td>
</tr>
</tbody>
</table>

**See Also**
- `TCustomDASQL`
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StartPos</td>
<td>Used to get the start position of the current statement in a script.</td>
</tr>
<tr>
<td>Statements</td>
<td>Contains a list of statements obtained from the SQL property.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BreakExec</td>
<td>Stops script execution.</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td>Used to get the offset of the statement if the Execute method raised an exception.</td>
</tr>
<tr>
<td>Execute</td>
<td>Executes a script.</td>
</tr>
<tr>
<td>ExecuteFile</td>
<td>Executes SQL statements contained in a file.</td>
</tr>
<tr>
<td>ExecuteNext</td>
<td>Executes the next statement in the script and then stops.</td>
</tr>
<tr>
<td>ExecuteStream</td>
<td>Executes SQL statements contained in a stream object.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Finds a macro with the specified name.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a SQL script execution.</td>
</tr>
<tr>
<td>BeforeExecute</td>
<td>Occurs when taking a specific action before executing the current SQL statement is needed.</td>
</tr>
<tr>
<td>OnError</td>
<td>Occurs when Oracle raises an error.</td>
</tr>
</tbody>
</table>
5.9.1.2.3 Properties

Properties of the TDAScript class.

For a complete list of the TDAScript class members, see the TDAScript Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Used to specify the connection in which the script will be executed.</td>
</tr>
<tr>
<td>DataSet</td>
<td>Refers to a dataset that holds the result set of query execution.</td>
</tr>
<tr>
<td>EndLine</td>
<td>Used to get the current statement last line number in a script.</td>
</tr>
<tr>
<td>EndOffset</td>
<td>Used to get the offset in the last line of the current statement.</td>
</tr>
<tr>
<td>EndPos</td>
<td>Used to get the end position of the current statement.</td>
</tr>
<tr>
<td>StartLine</td>
<td>Used to get the current statement start line number in a script.</td>
</tr>
<tr>
<td>StartOffset</td>
<td>Used to get the offset in the first line of the current statement.</td>
</tr>
<tr>
<td>StartPos</td>
<td>Used to get the start position of the current statement in a script.</td>
</tr>
<tr>
<td>Statements</td>
<td>Contains a list of statements obtained from the SQL property.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug</td>
<td>Used to display the script execution and all its parameter values.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Used to set the delimiter string that separates script statements.</td>
</tr>
</tbody>
</table>
Macros

Used to change SQL script text in design- or run-time easily.

SQL

Used to get or set script text.

See Also

- **TDAScript Class**
- **TDAScript Class Members**

5.9.1.1.2.1 Connection Property

Used to specify the connection in which the script will be executed.

**Class**

**TDAScript**

**Syntax**

```pascal
property Connection: TCustomDACConnection;
```

**Remarks**

Use the Connection property to specify the connection in which the script will be executed. If Connection is not connected, the **Execute** method calls the `Connect` method of Connection.

Set at design-time by selecting from the list of provided **TCustomDACConnection** objects.

At run-time, set the Connection property to reference an existing TCustomDACConnection object.

See Also

- **TCustomDACConnection**
5.9.1.1.2.2  DataSet Property

Refers to a dataset that holds the result set of query execution.

Class
TDAScript

Syntax

```property DataSet: TCustomDADataset;```

Remarks

Set the DataSet property to assign a component that will be used by TOraScript to execute statements and to retrieve the results of the SELECT statements execution inside a script.

See Also
- ExecuteNext
- Execute

5.9.1.1.2.3  Debug Property

Used to display the script execution and all its parameter values.

Class
TDAScript

Syntax

```property Debug: boolean default False;```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the OdacVcl unit to the uses clause of any unit in your project to make the Debug property work.

**Note:** If TOraSQLMonitor is used in the project and the TOraSQLMonitor.Active property is
set to False, the debug window is not displayed.

5.9.1.2.4 Delimiter Property

Used to set the delimiter string that separates script statements.

Class
TDAScript

Syntax

| property Delimiter: string stored IsDelimiterStored; |

Remarks
Use the Delimiter property to set the delimiter string that separates script statements. By default it is semicolon (;). You can use slash (/) to separate statements that can contain semicolon if the Delimiter property's default value is semicolon. Note that slash must be the first character in line.

5.9.1.2.5 EndLine Property

Used to get the current statement last line number in a script.

Class
TDAScript

Syntax

| property EndLine: Int64; |

Remarks
Use the EndLine property to get the current statement last line number in a script.
5.9.1.1.2.6  EndOffset Property

Used to get the offset in the last line of the current statement.

Class
TDAScript

Syntax

```property
EndOffset: Int64;
```

Remarks

Use the EndOffset property to get the offset in the last line of the current statement.

© 1997-2021
Devart. All Rights Reserved.

5.9.1.1.2.7  EndPos Property

Used to get the end position of the current statement.

Class
TDAScript

Syntax

```property
EndPos: Int64;
```

Remarks

Use the EndPos property to get the end position of the current statement (the position of the last character in the statement) in a script.

© 1997-2021
Devart. All Rights Reserved.

5.9.1.1.2.8  Macros Property

Used to change SQL script text in design- or run-time easily.

Class
TDAScript

© 2021 Devart
5.9.1.2.9 SQL Property

Used to get or set script text.

Class
TDAScript

Syntax

property SQL: TStrings;

Remarks
Use the SQL property to get or set script text.

5.9.1.2.10 StartLine Property

Used to get the current statement start line number in a script.

Class
TDAScript

Syntax

```property StartLine: Int64;```

Remarks

Use the StartLine property to get the current statement start line number in a script.

5.9.1.1.2.11 StartOffset Property

Used to get the offset in the first line of the current statement.

Class

TDAScript

Syntax

```property StartOffset: Int64;```

Remarks

Use the StartOffset property to get the offset in the first line of the current statement.

5.9.1.1.2.12 StartPos Property

Used to get the start position of the current statement in a script.

Class

TDAScript

Syntax

```property StartPos: Int64;```

Remarks
Use the StartPos property to get the start position of the current statement (the position of the first statement character) in a script.

Contains a list of statements obtained from the SQL property.

Class

TDAScript

Syntax

property Statements: TDAStatements;

Remarks

Contains a list of statements that are obtained from the SQL property. Use the Access Statements property to view SQL statement, set parameters or execute the specified statement. Statements is a zero-based array of statement records. Index specifies the array element to access.

For example, consider the following script:

```
CREATE TABLE A (FIELD1 INTEGER);
INSERT INTO A VALUES(1);
INSERT INTO A VALUES(2);
INSERT INTO A VALUES(3);
CREATE TABLE B (FIELD1 INTEGER);
INSERT INTO B VALUES(1);
INSERT INTO B VALUES(2);
INSERT INTO B VALUES(3);
```

Note: The list of statements is created and filled when the value of Statements property is requested. That's why the first access to the Statements property can take a long time.

Example

You can use the Statements property in the following way:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
  with Script do
  begin
```
```delphi
for i := 0 to Statements.Count - 1 do
  if Copy(Statements[i].SQL, 1, 6) <> 'CREATE' then
    Statements[i].Execute;
end;
end;
```

See Also
- `TDAScript`

© 1997-2021 Devart. All Rights Reserved.

5.9.1.3 Methods

Methods of the `TDAScript` class.

For a complete list of the `TDAScript` class members, see the `TDAScript Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BreakExec</td>
<td>Stops script execution.</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td>Used to get the offset of the statement if the Execute method raised an exception.</td>
</tr>
<tr>
<td>Execute</td>
<td>Executes a script.</td>
</tr>
<tr>
<td>ExecuteFile</td>
<td>Executes SQL statements contained in a file.</td>
</tr>
<tr>
<td>ExecuteNext</td>
<td>Executes the next statement in the script and then stops.</td>
</tr>
<tr>
<td>ExecuteStream</td>
<td>Executes SQL statements contained in a stream object.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Finds a macro with the specified name.</td>
</tr>
</tbody>
</table>

See Also
- `TDAScript Class`
- `TDAScript Class Members`

© 1997-2021 Devart. All Rights Reserved.

Request Support    DAC Forum    Provide Feedback
5.9.1.1.3.1  BreakExec Method

Stops script execution.

Class
TDAScript

Syntax

```pascal
procedure BreakExec; virtual;
```

Remarks

Call the BreakExec method to stop script execution.

5.9.1.1.3.2  ErrorOffset Method

Used to get the offset of the statement if the Execute method raised an exception.

Class
TDAScript

Syntax

```pascal
function ErrorOffset: Int64;
```

Return Value

offset of an error.

Remarks

Call the ErrorOffset method to get the offset of the statement if the Execute method raised an exception.

See Also

- OnError
5.9.1.3.3 Execute Method

Executes a script.

Class

**TDAScript**

Syntax

```plaintext
procedure Execute; virtual;
```

Remarks

Call the Execute method to execute a script. If Oracle raises an error, the OnError event occurs.

See Also

- [ExecuteNext](#)
- [OnError](#)
- [ErrorOffset](#)

© 1997-2021 Devart. All Rights Reserved.

5.9.1.3.4 ExecuteFile Method

Executes SQL statements contained in a file.

Class

**TDAScript**

Syntax

```plaintext
procedure ExecuteFile(const FileName: string);
```

Parameters

**FileName**

Holds the file name.

Remarks

Call the ExecuteFile method to execute SQL statements contained in a file. Script doesn't load full content into memory. Reading and execution is performed by blocks of 64k size.
Therefore, it is optimal to use it for big files.

5.9.1.3.5 ExecuteNext Method

Executes the next statement in the script and then stops.

Class
TDAScript

Syntax

```delphi
function ExecuteNext: boolean; virtual;
```

Return Value
True, if there are any statements left in the script, False otherwise.

Remarks
Use the ExecuteNext method to execute the next statement in the script statement and stop. If Oracle raises an error, the OnError event occurs.

See Also
- Execute
- OnError
- ErrorOffset

5.9.1.3.6 ExecuteStream Method

Executes SQL statements contained in a stream object.

Class
TDAScript

Syntax

```delphi
procedure ExecuteStream(Stream: TStream);
```
Parameters

Stream
Holds the stream object from which the statements will be executed.

Remarks

Call the ExecuteStream method to execute SQL statements contained in a stream object. Reading from the stream and execution is performed by blocks of 64k size.

Class

TDAScript

Syntax

function FindMacro(Name: string): TMacro;

Parameters

Name
Holds the name of a macro to search for.

Return Value

TMacro object if a match is found, nil otherwise.

Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the TMacros.Items property to avoid depending on the order of the items.

See Also

- TMacro
- Macros
- MacroByName
5.9.1.3.8 MacroByName Method

Finds a macro with the specified name.

Class

**TDAScript**

Syntax

```pascal
function MacroByName(Name: string): TMacro;
```

Parameters

*Name*

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found.

Remarks

Call the MacroByName method to find a macro with the specified name. If a match is found, MacroByName returns the macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the **TMacros.Items** property to avoid depending on the order of the items.

To locate a parameter by name without raising an exception if the parameter is not found, use the **FindMacro** method.

To set a value to a macro, use the **TMacro.Value** property.

See Also

- **TMacro**
- **Macros**
- **FindMacro**

© 1997-2021 Devart. All Rights Reserved.

5.9.1.4 Events

Events of the **TDAScript** class.

For a complete list of the **TDAScript** class members, see the **TDAScript Members** topic.
### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a SQL script execution.</td>
</tr>
<tr>
<td>BeforeExecute</td>
<td>Occurs when taking a specific action before executing the current SQL statement is needed.</td>
</tr>
<tr>
<td>OnError</td>
<td>Occurs when Oracle raises an error.</td>
</tr>
</tbody>
</table>

### See Also
- TDAScript Class
- TDAScript Class Members

### Syntax

```property
property AfterExecute: TAfterStatementExecuteEvent;
```

### Remarks

Occurs after a SQL script has been executed.

### See Also
- Execute

© 1997-2021
Devar. All Rights Reserved.
5.9.1.1.4.2 BeforeExecute Event

Occurs when taking a specific action before executing the current SQL statement is needed.

Class

TDAScript

Syntax

property BeforeExecute: TBeforeStatementExecuteEvent;

Remarks

Write the BeforeExecute event handler to take specific action before executing the current SQL statement. SQL holds text of the current SQL statement. Write SQL to change the statement that will be executed. Set Omit to True to skip statement execution.

© 1997-2021 Devart. All Rights Reserved.

5.9.1.1.4.3 OnError Event

Occurs when Oracle raises an error.

Class

TDAScript

Syntax

property OnError: TOnErrorEvent;

Remarks

Occurs when Oracle raises an error.

Action indicates the action to take when the OnError handler exits. On entry into the handler, Action is always set to eaFail.

See Also

• ErrorOffset

© 1997-2021 Devart. All Rights Reserved.
5.9.1.2 TDAStatement Class

This class has attributes and methods for controlling single SQL statement of a script.

For a list of all members of this type, see TDAStatement members.

Unit
DAScript

Syntax

TDAStatement = class(TCollectionItem);

Remarks
TDAScript contains SQL statements, represented as TDAStatement objects. The TDAStatement class has attributes and methods for controlling single SQL statement of a script.

See Also
- TDAScript
- TDAStatements

© 1997-2021 Devart. All Rights Reserved.

5.9.1.2.1 Members

TDAStatement class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndLine</td>
<td>Used to determine the number of the last statement line in a script.</td>
</tr>
<tr>
<td>EndOffset</td>
<td>Used to get the offset in the last line of the statement.</td>
</tr>
<tr>
<td>EndPos</td>
<td>Used to get the end position of the statement in a script.</td>
</tr>
<tr>
<td>Omit</td>
<td>Used to avoid execution of a statement.</td>
</tr>
<tr>
<td>Params</td>
<td>Contains parameters for an</td>
</tr>
</tbody>
</table>
Oracle Data Access Components

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script</td>
<td>Used to determine the TDAScript object the SQL Statement belongs to.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to get or set the text of an SQL statement.</td>
</tr>
<tr>
<td>StartLine</td>
<td>Used to determine the number of the first statement line in a script.</td>
</tr>
<tr>
<td>StartOffset</td>
<td>Used to get the offset in the first line of a statement.</td>
</tr>
<tr>
<td>StartPos</td>
<td>Used to get the start position of the statement in a script.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Executes a statement.</td>
</tr>
</tbody>
</table>

Properties of the TDAStatement class.

For a complete list of the TDAStatement class members, see the TDAStatement Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndLine</td>
<td>Used to determine the number of the last statement line in a script.</td>
</tr>
<tr>
<td>EndOffset</td>
<td>Used to get the offset in the last line of the statement.</td>
</tr>
<tr>
<td>EndPos</td>
<td>Used to get the end position of the statement in a script.</td>
</tr>
<tr>
<td>Omit</td>
<td>Used to avoid execution of a statement.</td>
</tr>
</tbody>
</table>
| Params   | Contains parameters for an
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Script</strong></td>
<td>SQL statement. Used to determine the TDAScript object the SQL Statement belongs to.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Used to get or set the text of an SQL statement.</td>
</tr>
<tr>
<td><strong>StartLine</strong></td>
<td>Used to determine the number of the first statement line in a script.</td>
</tr>
<tr>
<td><strong>StartOffset</strong></td>
<td>Used to get the offset in the first line of a statement.</td>
</tr>
<tr>
<td><strong>StartPos</strong></td>
<td>Used to get the start position of the statement in a script.</td>
</tr>
</tbody>
</table>

See Also

- **TDAScript Class**
- **TDAScript Class Members**

## 5.9.1.2.2.1 EndLine Property

Used to determine the number of the last statement line in a script.

### Class

**TDAScript**

### Syntax

```pascal
property EndLine: integer;
```

### Remarks

Use the EndLine property to determine the number of the last statement line in a script.
5.9.1.2.2.2 EndOffset Property

Used to get the offset in the last line of the statement.

Class

TDAStatement

Syntax

[property] EndOffset: integer;

Remarks

Use the EndOffset property to get the offset in the last line of the statement.

5.9.1.2.2.3 EndPos Property

Used to get the end position of the statement in a script.

Class

TDAStatement

Syntax

[property] EndPos: integer;

Remarks

Use the EndPos property to get the end position of the statement (the position of the last character in the statement) in a script.

5.9.1.2.2.4 Omit Property

Used to avoid execution of a statement.

Class

TDAStatement
5.9.1.2.2.5 Params Property

Contains parameters for an SQL statement.

Class

TDAScript

Syntax

```
property Params: TDAParams;
```

Remarks

Contains parameters for an SQL statement.

Access Params at runtime to view and set parameter names, values, and data types dynamically. Params is a zero-based array of parameter records. Index specifies the array element to access.

See Also

- TDAParam

5.9.1.2.2.6 Script Property

Used to determine the TDAScript object the SQL Statement belongs to.

Class

TDAScript
5.9.1.2.2.7 SQL Property

Used to get or set the text of an SQL statement.

Class

TDAScript

Syntax

```property` SQL: `string;```

Remarks

Use the SQL property to get or set the text of an SQL statement.

5.9.1.2.2.8 StartLine Property

Used to determine the number of the first statement line in a script.

Class

TDAScript

Syntax

```property` StartLine: integer;```

Remarks

Use the StartLine property to determine the number of the first statement line in a script.
5.9.1.2.2.9 StartOffset Property

Used to get the offset in the first line of a statement.

Class

TDADeclaration

Syntax

\textbf{property} \texttt{StartOffset: integer;}

Remarks

Use the StartOffset property to get the offset in the first line of a statement.

© 1997-2021

Devart. All Rights Reserved.

5.9.1.2.2.10 StartPos Property

Used to get the start position of the statement in a script.

Class

TDADeclaration

Syntax

\textbf{property} \texttt{StartPos: integer;}

Remarks

Use the StartPos property to get the start position of the statement (the position of the first statement character) in a script.

© 1997-2021

Devart. All Rights Reserved.

5.9.1.2.3 Methods

Methods of the \texttt{TDADeclaration} class.

For a complete list of the \texttt{TDADeclaration} class members, see the \texttt{TDADeclaration Members}
5.9.1.3 Execute Method

Executes a statement.

Class

TDAStatement

Syntax

procedure Execute;

Remarks

Use the Execute method to execute a statement.

5.9.3 TDAStatements Class

Holds a collection of TDAStatement objects.

For a list of all members of this type, see TDAStatements members.

Unit

DAScript

Syntax
TDAStatements = class(TCollection);

Remarks
Each TDAStatements holds a collection of TDAStatement objects. TDAStatements maintains an index of the statements in its Items array. The Count property contains the number of statements in the collection. Use TDAStatements class to manipulate script SQL statements.

See Also
- TDAScript
- TDAStatement

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access separate script statements.</td>
</tr>
</tbody>
</table>

See Also
## TDAStatements Class Members

### 5.9.1.3.2.1 Items Property (Indexer)

Used to access separate script statements.

**Class**

**TDAStatements**

**Syntax**

```delphi
property Items[Index: Integer]: TDAStatement; default;
```

**Parameters**

- **Index**
  
  Holds the index value.

**Remarks**

Use the Items property to access individual script statements. The value of the Index parameter corresponds to the Index property of **TDAStatement**.

**See Also**

- **TDAStatement**

### 5.9.2 Types

Types in the **DAScript** unit.

**Types**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TAfterStatementExecuteEvent</strong></td>
<td>This type is used for the <strong>TDAScript.AfterExecute</strong> event.</td>
</tr>
<tr>
<td><strong>TBeforeStatementExecuteEvent</strong></td>
<td>This type is used for the <strong>TDAScript.BeforeExecute</strong> event.</td>
</tr>
</tbody>
</table>
### 5.9.2.1 TAfterStatementExecuteEvent Procedure Reference

This type is used for the `TDAScript.AfterExecute` event.

**Unit**

`DAScript`

**Syntax**

```delphi
TAfterStatementExecuteEvent = procedure (Sender: TObject; SQL: string) of object;
```

**Parameters**

- `Sender`
  - An object that raised the event.
- `SQL`
  - Holds the passed SQL statement.

### 5.9.2.2 TBeforeStatementExecuteEvent Procedure Reference

This type is used for the `TDAScript.BeforeExecute` event.

**Unit**

`DAScript`

**Syntax**

```delphi
TBeforeStatementExecuteEvent = procedure (Sender: TObject; var SQL: string; var Omit: boolean) of object;
```

**Parameters**

- `Sender`

© 1997-2021 Devart. All Rights Reserved.
An object that raised the event.

SQL
Holds the passed SQL statement.

Omit
True, if the statement execution should be skipped.

5.9.3.3 TOnErrorEvent Procedure Reference

This type is used for the TDAScript.OnError event.

Unit
DAScript

Syntax

TOnErrorEvent = procedure (Sender: TObject; E: Exception; SQL: string; var Action: TErrorAction) of object;

Parameters

Sender
An object that raised the event.

E
The error code.

SQL
Holds the passed SQL statement.

Action
The action to take when the OnError handler exits.

5.9.3 Enumerations

Enumerations in the DAScript unit.

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TErrorAction</td>
<td>Indicates the action to take when the OnError handler exits</td>
</tr>
</tbody>
</table>
5.9.3.1 TErrorAction Enumeration

Indicates the action to take when the OnError handler exits.

Unit
DAScript

Syntax

TErrorAction = (eaAbort, eaFail, eaException, eaContinue);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>eaAbort</td>
<td>Abort execution without displaying an error message.</td>
</tr>
<tr>
<td>eaContinue</td>
<td>Continue execution.</td>
</tr>
<tr>
<td>eaException</td>
<td>In Delphi 6 and higher exception is handled by the Application.HandleException method.</td>
</tr>
<tr>
<td>eaFail</td>
<td>Abort execution and display an error message.</td>
</tr>
</tbody>
</table>

5.10 DASQLMonitor

This unit contains the base class for the TOraSQLMonitor component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomDASQLMonitor</td>
<td>A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.</td>
</tr>
<tr>
<td>TDBMonitorOptions</td>
<td>This class holds options for dbMonitor.</td>
</tr>
</tbody>
</table>
Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDATraceFlags</td>
<td>Represents the set of TDATraceFlag.</td>
</tr>
<tr>
<td>TMonitorOptions</td>
<td>Represents the set of TMonitorOption.</td>
</tr>
<tr>
<td>TOnSQLEvent</td>
<td>This type is used for the TCustomDASQLMonitor.On SQL event.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDATraceFlag</td>
<td>Use TraceFlags to specify which database operations the monitor should track in an application at runtime.</td>
</tr>
<tr>
<td>TMonitorOption</td>
<td>Used to define where information from SQLMonitor will be displayed.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.10.1 Classes

Classes in the DASQLMonitor unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomDASQLMonitor</td>
<td>A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.</td>
</tr>
<tr>
<td>TDBMonitorOptions</td>
<td>This class holds options for dbMonitor.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

Request Support   DAC Forum   Provide Feedback
5.10.1.1  TCustomDASQLMonitor Class

A base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively.

For a list of all members of this type, see TCustomDASQLMonitor members.

Unit

DASQLMonitor

Syntax

TCustomDASQLMonitor = class(TComponent);

Remarks

TCustomDASQLMonitor is a base class that introduces properties and methods to monitor dynamic SQL execution in database applications interactively. TCustomDASQLMonitor provides two ways of displaying debug information. It monitors either by dialog window or by Borland's proprietary SQL Monitor. Furthermore to receive debug information use the TCustomDASQLMonitor.OnSQL event.

In applications use descendants of TCustomDASQLMonitor.

© 1997-2021
Devart. All Rights Reserved.

5.10.1.1.1  Members

TCustomDASQLMonitor class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to activate monitoring of SQL.</td>
</tr>
<tr>
<td>DBMonitorOptions</td>
<td>Used to set options for dbMonitor.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to include the desired properties for TCustomDASQLMonitor.</td>
</tr>
<tr>
<td>TraceFlags</td>
<td>Used to specify which</td>
</tr>
</tbody>
</table>
database operations the monitor should track in an application at runtime.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnSQL</td>
<td>Occurs when tracing of SQL activity on database components is needed.</td>
</tr>
</tbody>
</table>

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to activate monitoring of SQL.</td>
</tr>
<tr>
<td>DBMonitorOptions</td>
<td>Used to set options for dbMonitor.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to include the desired properties for TCustomDASQLMonitor.</td>
</tr>
<tr>
<td>TraceFlags</td>
<td>Used to specify which database operations the monitor should track in an application at runtime.</td>
</tr>
</tbody>
</table>

See Also

- TCustomDASQLMonitor Class
- TCustomDASQLMonitor Class Members

© 1997-2021 Devart. All Rights Reserved.

Request Support   DAC Forum   Provide Feedback
5.10.1.1.2.1  Active Property

Used to activate monitoring of SQL.

Class

TCustomDASQLMonitor

Syntax

```
property Active: boolean default True;
```

Remarks

Set the Active property to True to activate monitoring of SQL.

See Also

- OnSQL

© 1997-2021

Devart. All Rights Reserved.

5.10.1.1.2.2  DBMonitorOptions Property

Used to set options for dbMonitor.

Class

TCustomDASQLMonitor

Syntax

```
property DBMonitorOptions: TDBMonitorOptions;
```

Remarks

Use DBMonitorOptions to set options for dbMonitor.

© 1997-2021

Devart. All Rights Reserved.

5.10.1.1.2.3  Options Property

Used to include the desired properties for TCustomDASQLMonitor.

Class

© 2021 Devart
**TCustomDASQLMonitor**

**Syntax**

```property Options: TMonitorOptions default [moDialog, moSQLMonitor, moDBMonitor, moCustom];```

**Remarks**

Set Options to include the desired properties for TCustomDASQLMonitor.

**See Also**

- [OnSQL](#)

---

5.10.1.2.4 TraceFlags Property

Used to specify which database operations the monitor should track in an application at runtime.

**Class**

**TCustomDASQLMonitor**

**Syntax**

```property TraceFlags: TDATraceFlags default [tfQPrepare, tfQExecute, tfError, tfConnect, tfTransact, tfParams, tfMisc];```

**Remarks**

Use the TraceFlags property to specify which database operations the monitor should track in an application at runtime.

**See Also**

- [OnSQL](#)
5.10.1.1.3 Events

Events of the TCustomDASQLMonitor class.

For a complete list of the TCustomDASQLMonitor class members, see the TCustomDASQLMonitor Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnSQL</td>
<td>Occurs when tracing of SQL activity on database components is needed.</td>
</tr>
</tbody>
</table>

See Also
- TCustomDASQLMonitor Class
- TCustomDASQLMonitor Class Members

© 1997-2021 Devart. All Rights Reserved.

5.10.1.1.3.1 OnSQL Event

Occurs when tracing of SQL activity on database components is needed.

Class

TCustomDASQLMonitor

Syntax

```property
OnSQL: TOnSQLEvent;
```

Remarks

Write the OnSQL event handler to let an application trace SQL activity on database components. The Text parameter holds the detected SQL statement. Use the Flag parameter to make selective processing of SQL in the handler body.

See Also
- TraceFlags

© 1997-2021 Devart. All Rights Reserved.
5.10.1.2 TDBMonitorOptions Class

This class holds options for dbMonitor.

For a list of all members of this type, see TDBMonitorOptions members.

Unit
DASQLMonitor

Syntax

TDBMonitorOptions = class(TPersistent);

5.10.1.2.1 Members

TDBMonitorOptions class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>Used to set the host name or IP address of the computer where dbMonitor application runs.</td>
</tr>
<tr>
<td>Port</td>
<td>Used to set the port number for connecting to dbMonitor.</td>
</tr>
<tr>
<td>ReconnectTimeout</td>
<td>Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.</td>
</tr>
<tr>
<td>SendTimeout</td>
<td>Used to set timeout for sending events to dbMonitor.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
5.10.1.2.2 Properties

Properties of the TDBMonitorOptions class.

For a complete list of the TDBMonitorOptions class members, see the TDBMonitorOptions Members topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>Used to set the host name or IP address of the computer where dbMonitor runs.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>Used to set the port number for connecting to dbMonitor.</td>
</tr>
<tr>
<td><strong>ReconnectTimeout</strong></td>
<td>Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.</td>
</tr>
<tr>
<td><strong>SendTimeout</strong></td>
<td>Used to set timeout for sending events to dbMonitor.</td>
</tr>
</tbody>
</table>

See Also

- TDBMonitorOptions Class
- TDBMonitorOptions Class Members

© 1997-2021 Devart. All Rights Reserved.

5.10.1.2.2.1 Host Property

Used to set the host name or IP address of the computer where dbMonitor application runs.

Class

TDBMonitorOptions

Syntax

```
property Host: string;
```

Remarks
Use the Host property to set the host name or IP address of the computer where dbMonitor application runs.

dbMonitor supports remote monitoring. You can run dbMonitor on a different computer than monitored application runs. In this case you need to set the Host property to the corresponding computer name.

### 5.10.1.2.2.2 Port Property

Used to set the port number for connecting to dbMonitor.

**Class**

*TDBMonitorOptions*

**Syntax**

```delphi
property Port: integer default DBMonitorPort;
```

**Remarks**

Use the Port property to set the port number for connecting to dbMonitor.

### 5.10.1.2.2.3 ReconnectTimeout Property

Used to set the minimum time that should be spent before reconnecting to dbMonitor is allowed.

**Class**

*TDBMonitorOptions*

**Syntax**

```delphi
property ReconnectTimeout: integer default DefaultReconnectTimeout;
```

**Remarks**
Use the ReconnectTimeout property to set the minimum time (in milliseconds) that should be spent before allowing reconnecting to dbMonitor. If an error occurs when the component sends an event to dbMonitor (dbMonitor is not running), next events are ignored and the component does not restore the connection until ReconnectTimeout is over.

5.10.1.2.2.4 SendTimeout Property

Used to set timeout for sending events to dbMonitor.

Class

**TDBMonitorOptions**

Syntax

```property` SendTimeout: integer `default` DefaultSendTimeout;```

Remarks

Use the SendTimeout property to set timeout (in milliseconds) for sending events to dbMonitor. If dbMonitor does not respond in the specified timeout, event is ignored.

5.10.2 Types

Types in the **DASQLMonitor** unit.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDATraceFlags</strong></td>
<td>Represents the set of <strong>TDATraceFlag</strong>.</td>
</tr>
<tr>
<td><strong>TMonitorOptions</strong></td>
<td>Represents the set of <strong>TMonitorOption</strong>.</td>
</tr>
<tr>
<td><strong>TOnSQLEvent</strong></td>
<td>This type is used for the <strong>TCustomDASQLMonitor.On</strong> SQL event.</td>
</tr>
</tbody>
</table>
5.10.2.1 TDATraceFlags Set

Represents the set of TDATraceFlag.

Unit
DASQLMonitor

Syntax

\[
\text{TDATraceFlags} = \text{set of TDATraceFlag};
\]

5.10.2.2 TMonitorOptions Set

Represents the set of TMonitorOption.

Unit
DASQLMonitor

Syntax

\[
\text{TMonitorOptions} = \text{set of TMonitorOption};
\]

5.10.2.3 TOnSQLEvent Procedure Reference

This type is used for the TCustomDASQLMonitor.OnSQL event.

Unit
DASQLMonitor

Syntax

\[
\text{TOnSQLEvent} = \text{procedure (Sender: TObject; Text: string; Flag: TDATraceFlag) of object;}
\]
**Sender**
An object that raised the event.

**Text**
Holds the detected SQL statement.

**Flag**
Use the Flag parameter to make selective processing of SQL in the handler body.

---

## 5.10.3 Enumerations

Enumerations in the **DasqlMonitor** unit.

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDATraceFlag</strong></td>
<td>Use TraceFlags to specify which database operations the monitor should track in an application at runtime.</td>
</tr>
<tr>
<td><strong>TMonitorOption</strong></td>
<td>Used to define where information from SQLMonitor will be displayed.</td>
</tr>
</tbody>
</table>

---

### 5.10.3.1 TDATraceFlag Enumeration

Use TraceFlags to specify which database operations the monitor should track in an application at runtime.

**Unit**
**DasqlMonitor**

**Syntax**

```
TDATraceFlag = (tfQPrepare, tfQExecute, tfQFetch, tfError, tfStmt, tfConnect, tfTransact, tfBlob, tfService, tfMisc, tfParams, tfObjDestroy, tfPool);
```
Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>tfBlob</td>
<td>This option is declared for future use.</td>
</tr>
<tr>
<td>tfConnect</td>
<td>Establishing a connection.</td>
</tr>
<tr>
<td>tfError</td>
<td>Errors of query execution.</td>
</tr>
<tr>
<td>tfMisc</td>
<td>This option is declared for future use.</td>
</tr>
<tr>
<td>tfObjDestroy</td>
<td>Destroying of components.</td>
</tr>
<tr>
<td>tfParams</td>
<td>Representing parameter values for tfQPrepare and tfQExecute.</td>
</tr>
<tr>
<td>tfPool</td>
<td>Connection pool operations.</td>
</tr>
<tr>
<td>tfQExecute</td>
<td>Execution of the queries.</td>
</tr>
<tr>
<td>tfQFetch</td>
<td>This option is declared for future use.</td>
</tr>
<tr>
<td>tfQPrepare</td>
<td>Queries preparation.</td>
</tr>
<tr>
<td>tfService</td>
<td>This option is declared for future use.</td>
</tr>
<tr>
<td>tfStmt</td>
<td>This option is declared for future use.</td>
</tr>
<tr>
<td>tfTransact</td>
<td>Processing transactions.</td>
</tr>
</tbody>
</table>

5.10.3.2 TMonitorOption Enumeration

Used to define where information from SQLMonitor will be displayed.

Unit

DASQLMonitor

Syntax

```
TMonitorOption = (moDialog, moSQLMonitor, moDBMonitor, moCustom, moHandled);
```

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>moCustom</td>
<td>Monitoring of SQL for individual components is allowed. Set Debug properties in SQL-related components to True to let TCustomDASQLMonitor instance to monitor their behavior. Has effect when moDialog is included.</td>
</tr>
<tr>
<td>moDBMonitor</td>
<td>Debug information is displayed in DBMonitor.</td>
</tr>
</tbody>
</table>
5.11 DBAccess

This unit contains base classes for most of the components.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>EDAError</strong></td>
<td>A base class for exceptions that are raised when an error occurs on the server side.</td>
</tr>
<tr>
<td><strong>TCRDataSource</strong></td>
<td>Provides an interface between a DAC dataset components and data-aware controls on a form.</td>
</tr>
<tr>
<td><strong>TCustomConnectDialog</strong></td>
<td>A base class for the connect dialog components.</td>
</tr>
<tr>
<td><strong>TCustomDAConnection</strong></td>
<td>A base class for components used to establish connections.</td>
</tr>
<tr>
<td><strong>TCustomDADataset</strong></td>
<td>Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.</td>
</tr>
<tr>
<td><strong>TCustomDASQL</strong></td>
<td>A base class for components executing SQL statements that do not return result sets.</td>
</tr>
<tr>
<td><strong>TCustomDAUpdateSQL</strong></td>
<td>A base class for components that provide DML statements for more flexible control over data modifications.</td>
</tr>
<tr>
<td><strong>TDACCondition</strong></td>
<td>Represents a condition from the TDACConditions list.</td>
</tr>
<tr>
<td><strong>TDACConditions</strong></td>
<td>Holds a collection of</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TDACCondition</strong></td>
<td>Represents TDACCondition objects.</td>
</tr>
<tr>
<td><strong>TDAConnectionOptions</strong></td>
<td>This class allows setting up the behaviour of the TDAConnection class.</td>
</tr>
<tr>
<td><strong>TDAConnectionSSLOptions</strong></td>
<td>This class is used to set up the SSL options.</td>
</tr>
<tr>
<td><strong>TDADatasetOptions</strong></td>
<td>This class allows setting up the behaviour of the TDADataset class.</td>
</tr>
<tr>
<td><strong>TDAEncryption</strong></td>
<td>Used to specify the options of the data encryption in a dataset.</td>
</tr>
<tr>
<td><strong>TDAMapRule</strong></td>
<td>Class that forms rules for Data Type Mapping.</td>
</tr>
<tr>
<td><strong>TDAMapRules</strong></td>
<td>Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.</td>
</tr>
<tr>
<td><strong>TDAMetaData</strong></td>
<td>A class for retrieving metainformation of the specified database objects in the form of dataset.</td>
</tr>
<tr>
<td><strong>TDAParam</strong></td>
<td>A class that forms objects to represent the values of the parameters set.</td>
</tr>
<tr>
<td><strong>TDAParams</strong></td>
<td>This class is used to manage a list of TDAParam objects for an object that uses field parameters.</td>
</tr>
<tr>
<td><strong>TDATransaction</strong></td>
<td>A base class that implements functionality for controlling transactions.</td>
</tr>
<tr>
<td><strong>TMacro</strong></td>
<td>Object that represents the value of a macro.</td>
</tr>
<tr>
<td><strong>TMacros</strong></td>
<td>Controls a list of TMacro objects for the TCustomDASQL.Macros or TCustomDADataset components.</td>
</tr>
<tr>
<td><strong>TPoolingOptions</strong></td>
<td>This class allows setting up the behaviour of the connection pool.</td>
</tr>
<tr>
<td><strong>TSmartFetchOptions</strong></td>
<td>Smart fetch options are</td>
</tr>
</tbody>
</table>
used to set up the behavior of the SmartFetch mode.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAfterExecuteEvent</td>
<td>This type is used for the TCustomDADataSet.AfterExecute and TCustomDASQL.AfterExecute events.</td>
</tr>
<tr>
<td>TAfterFetchEvent</td>
<td>This type is used for the TCustomDADataSet.AfterFetch event.</td>
</tr>
<tr>
<td>TBeforeFetchEvent</td>
<td>This type is used for the TCustomDADataSet.BeforeFetch event.</td>
</tr>
<tr>
<td>TConnectionLostEvent</td>
<td>This type is used for the TCustomDAConnection.OnConnectionLost event.</td>
</tr>
<tr>
<td>TDAConnectionErrorEvent</td>
<td>This type is used for the TCustomDAConnection.OnError event.</td>
</tr>
<tr>
<td>TDATransactionErrorEvent</td>
<td>This type is used for the TDATransaction.OnError event.</td>
</tr>
<tr>
<td>TRefreshOptions</td>
<td>Represents the set of TRefreshOption.</td>
</tr>
<tr>
<td>TUpdateExecuteEvent</td>
<td>This type is used for the TCustomDADataSet.AfterUpdateExecute and TCustomDADataSet.BeforeUpdateExecute events.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCheckMode</td>
<td>Specifies the action to take when another user makes modifications to a record.</td>
</tr>
<tr>
<td>TLabelSet</td>
<td>Sets the language of labels in the connect dialog.</td>
</tr>
</tbody>
</table>
TLockMode | Specifies the lock mode.  
TRefreshOption | Indicates when the editing record will be refreshed.  
TRetryMode | Specifies the application behavior when connection is lost.

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQLOldBehavior</td>
<td>After assigning SQL text and modifying it by AddWhere, DeleteWhere, and SetOrderBy, all subsequent changes of the SQL property will not be reflected in the BaseSQL property.</td>
</tr>
<tr>
<td>ChangeCursor</td>
<td>When set to True allows data access components to change screen cursor for the execution time.</td>
</tr>
<tr>
<td>SQLGeneratorCompatibility</td>
<td>The value of the TCustomDADataSet.BaseSQL property is used to complete the refresh SQL statement, if the manually assigned TCustomDAUpdateSQL.RefreshSQL property contains only WHERE clause.</td>
</tr>
</tbody>
</table>

5.11.1 Classes

Classes in the DBAccess unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| EDAError | A base class for exceptions that are raised when an error
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCRDataSource</td>
<td>Provides an interface between a DAC dataset components and data-aware controls on a form.</td>
</tr>
<tr>
<td>TCustomConnectDialog</td>
<td>A base class for the connect dialog components.</td>
</tr>
<tr>
<td>TCustomDAConnection</td>
<td>A base class for components used to establish connections.</td>
</tr>
<tr>
<td>TCustomDADataset</td>
<td>Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.</td>
</tr>
<tr>
<td>TCustomDASQL</td>
<td>A base class for components executing SQL statements that do not return result sets.</td>
</tr>
<tr>
<td>TCustomDAUpdateSQL</td>
<td>A base class for components that provide DML statements for more flexible control over data modifications.</td>
</tr>
<tr>
<td>TDACCondition</td>
<td>Represents a condition from the TDACConditions list.</td>
</tr>
<tr>
<td>TDACConditions</td>
<td>Holds a collection of TDACCondition objects.</td>
</tr>
<tr>
<td>TDAConnectionOptions</td>
<td>This class allows setting up the behaviour of the TDACConnection class.</td>
</tr>
<tr>
<td>TDAConnectionSSLOptions</td>
<td>This class is used to set up the SSL options.</td>
</tr>
<tr>
<td>TDADatasetOptions</td>
<td>This class allows setting up the behaviour of the TDADataset class.</td>
</tr>
<tr>
<td>TDAEncryption</td>
<td>Used to specify the options of the data encryption in a dataset.</td>
</tr>
<tr>
<td>TDAMapRule</td>
<td>Class that forms rules for Data Type Mapping.</td>
</tr>
<tr>
<td>TDAMapRules</td>
<td>Used for adding rules for DataSet fields mapping with both identifying by field</td>
</tr>
</tbody>
</table>
name and by field type and Delphi field types.

**TDAMetaData**

A class for retrieving metainformation of the specified database objects in the form of dataset.

**TDAParam**

A class that forms objects to represent the values of the parameters set.

**TDAParams**

This class is used to manage a list of TDAParam objects for an object that uses field parameters.

**TDATransaction**

A base class that implements functionality for controlling transactions.

**TMacro**

Object that represents the value of a macro.

**TMacros**

Controls a list of TMacro objects for the TCustomDASQL.Macros or TCustomDADataSet components.

**TPoolingOptions**

This class allows setting up the behaviour of the connection pool.

**TSmartFetchOptions**

Smart fetch options are used to set up the behavior of the SmartFetch mode.

---

### 5.11.1.1 EDAError Class

A base class for exceptions that are raised when an error occurs on the server side.

For a list of all members of this type, see [EDAError members](#).

**Unit**

DBAccess

**Syntax**

© 2021 Devart
**EDAError** = `class` (EDatabaseError);

**Remarks**

EDAError is a base class for exceptions that are raised when an error occurs on the server side.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Contains the component that caused the error.</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>Determines the error code returned by the server.</td>
</tr>
</tbody>
</table>

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>Contains the component that caused the error.</td>
</tr>
<tr>
<td>ErrorCode</td>
<td>Determines the error code returned by the server.</td>
</tr>
</tbody>
</table>

**See Also**

- [EDAError Class](#)
- [EDAError Class Members](#)
5.11.1.1.2.1 Component Property

Contains the component that caused the error.

Class

EDAError

Syntax

```pascal
property Component: TObject;
```

Remarks

The Component property contains the component that caused the error.

5.11.1.1.2.2 ErrorCode Property

Determines the error code returned by the server.

Class

EDAError

Syntax

```pascal
property ErrorCode: integer;
```

Remarks

Use the ErrorCode property to determine the error code returned by Oracle. This value is always positive.

See Also

- TOraErrorHandler.OnError
- TOraErrorHandler.OnError
5.11.1.2 TCRDataSource Class

Provides an interface between a DAC dataset components and data-aware controls on a form.

For a list of all members of this type, see TCRDataSource members.

Unit

DBAccess

Syntax

TCRDataSource = class(TDataSource);

Remarks

TCRDataSource provides an interface between a DAC dataset components and data-aware controls on a form.

TCRDataSource inherits its functionality directly from the TDataSource component.

At design time assign individual data-aware components' DataSource properties from their drop-down listboxes.

© 1997-2021 Devart. All Rights Reserved.

Request Support   DAC Forum   Provide Feedback

5.11.1.3 TCustomConnectDialog Class

A base class for the connect dialog components.

For a list of all members of this type, see TCustomConnectDialog members.

Unit

DBAccess

Syntax
TCustomConnectDialog = class(TComponent);

Remarks

TCustomConnectDialog is a base class for the connect dialog components. It provides functionality to show a dialog box where user can edit username, password and server name before connecting to a database. You can customize captions of buttons and labels by their properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelButton</td>
<td>Used to specify the label for the Cancel button.</td>
</tr>
<tr>
<td>Caption</td>
<td>Used to set the caption of dialog box.</td>
</tr>
<tr>
<td>ConnectButton</td>
<td>Used to specify the label for the Connect button.</td>
</tr>
<tr>
<td>DialogClass</td>
<td>Used to specify the class of the form that will be displayed to enter login information.</td>
</tr>
<tr>
<td>LabelSet</td>
<td>Used to set the language of buttons and labels captions.</td>
</tr>
<tr>
<td>PasswordLabel</td>
<td>Used to specify a prompt for password edit.</td>
</tr>
<tr>
<td>Retries</td>
<td>Used to indicate the number of retries of failed connections.</td>
</tr>
<tr>
<td>SavePassword</td>
<td>Used for the password to be displayed in ConnectDialog in asterisks.</td>
</tr>
<tr>
<td>ServerLabel</td>
<td>Used to specify a prompt for the server name edit.</td>
</tr>
<tr>
<td>StoreLogInfo</td>
<td>Used to specify whether the login information should be kept in system registry after</td>
</tr>
</tbody>
</table>
a connection was established.

**UsernameLabel**

Used to specify a prompt for username edit.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execute</strong></td>
<td>Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.</td>
</tr>
<tr>
<td><strong>GetServerList</strong></td>
<td>Retrieves a list of available server names.</td>
</tr>
</tbody>
</table>

### Properties of the TCustomConnectDialog class.

For a complete list of the TCustomConnectDialog class members, see the TCustomConnectDialog Members topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CancelButton</strong></td>
<td>Used to specify the label for the Cancel button.</td>
</tr>
<tr>
<td><strong>Caption</strong></td>
<td>Used to set the caption of dialog box.</td>
</tr>
<tr>
<td><strong>ConnectButton</strong></td>
<td>Used to specify the label for the Connect button.</td>
</tr>
<tr>
<td><strong>DialogClass</strong></td>
<td>Used to specify the class of the form that will be displayed to enter login information.</td>
</tr>
<tr>
<td><strong>LabelSet</strong></td>
<td>Used to set the language of buttons and labels captions.</td>
</tr>
<tr>
<td><strong>PasswordLabel</strong></td>
<td>Used to specify a prompt for password edit.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
| **Retries** | Used to indicate the number of retries of failed connections. |
| **SavePassword** | Used for the password to be displayed in ConnectDialog in asterisks. |
| **ServerLabel** | Used to specify a prompt for the server name edit. |
| **StoreLogInfo** | Used to specify whether the login information should be kept in system registry after a connection was established. |
| **UsernameLabel** | Used to specify a prompt for username edit. |

See Also

- [TCustomConnectDialog Class](#)
- [TCustomConnectDialog Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

5.11.1.3.2.1 CancelButton Property

Used to specify the label for the Cancel button.

**Class**

**TCustomConnectDialog**

**Syntax**

```plaintext
property CancelButton: string;
```

**Remarks**

Use the CancelButton property to specify the label for the Cancel button.
5.11.1.3.2.2 Caption Property

Used to set the caption of dialog box.

Class

TCustomConnectDialog

Syntax

property Caption: string;

Remarks

Use the Caption property to set the caption of dialog box.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.3.2.3 ConnectButton Property

Used to specify the label for the Connect button.

Class

TCustomConnectDialog

Syntax

property ConnectButton: string;

Remarks

Use the ConnectButton property to specify the label for the Connect button.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.3.2.4 DialogClass Property

Used to specify the class of the form that will be displayed to enter login information.

Class

TCustomConnectDialog
Syntax

```property`` DialogClass: `string`;

Remarks

Use the DialogClass property to specify the class of the form that will be displayed to enter login information. When this property is blank, TCustomConnectDialog uses the default form - TConnectForm. You can write your own login form to enter login information and assign its class name to the DialogClass property. Each login form must have ConnectDialog: TCustomConnectDialog published property to access connection information. For details see the implementation of the connect form which sources are in the Lib subdirectory of the ODAC installation directory.

See Also

- `GetServerList`

© 1997-2021 Devart. All Rights Reserved.

5.11.1.3.2.5 LabelSet Property

Used to set the language of buttons and labels captions.

Class

`TCustomConnectDialog`

Syntax

```property`` LabelSet: `TLabelSet` default lsEnglish;

Remarks

Use the LabelSet property to set the language of labels and buttons captions.

The default value is lsEnglish.
5.11.1.3.2.6 PasswordLabel Property

Used to specify a prompt for password edit.

Class

TCustomConnectDialog

Syntax

property PasswordLabel: string;

Remarks

Use the PasswordLabel property to specify a prompt for password edit.

5.11.1.3.2.7 Retries Property

Used to indicate the number of retries of failed connections.

Class

TCustomConnectDialog

Syntax

property Retries: word default 3;

Remarks

Use the Retries property to determine the number of retries of failed connections.

5.11.1.3.2.8 SavePassword Property

Used for the password to be displayed in ConnectDialog in asterisks.

Class

TCustomConnectDialog
5.11.1.3.2.9  ServerLabel Property

Used to specify a prompt for the server name edit.

Class

TCustomConnectDialog

Syntax

property ServerLabel: string;

Remarks

Use the ServerLabel property to specify a prompt for the server name edit.

5.11.1.3.2.10  StoreLogInfo Property

Used to specify whether the login information should be kept in system registry after a
connection was established.

Class

TCustomConnectDialog

Syntax

property StoreLogInfo: boolean default True;

Remarks
Use the StoreLogInfo property to specify whether to keep login information in system registry after a connection was established using provided username, password and servername.

Set this property to True to store login information.

The default value is True.

5.11.1.3.2.11 UsernameLabel Property

Used to specify a prompt for username edit.

Class

TCustomConnectDialog

Syntax

property UsernameLabel: string;

Remarks

Use the UsernameLabel property to specify a prompt for username edit.

5.11.3.3 Methods

Methods of the TCustomConnectDialog class.

For a complete list of the TCustomConnectDialog class members, see the TCustomConnectDialog Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.</td>
</tr>
<tr>
<td>GetServerList</td>
<td>Retrieves a list of available server names.</td>
</tr>
</tbody>
</table>
5.11.1.3.3.1 Execute Method

Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.

Class

TCustomConnectDialog

Syntax

function Execute: boolean; virtual;

Return Value

True, if connected.

Remarks

Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button. Returns True if connected. If user clicks Cancel, Execute returns False.

In the case of failed connection Execute offers to connect repeat \textit{Retries} times.

5.11.1.3.3.2 GetServerList Method

Retrieves a list of available server names.

Class

TCustomConnectDialog

Syntax

procedure GetServerList(List: TStrings); virtual;
Parameters

List
Holds a list of available server names.

Remarks
Call the GetServerList method to retrieve a list of available server names. It is particularly relevant for writing custom login form.

See Also
- DialogClass

TCustomDAConnection Class

A base class for components used to establish connections.

For a list of all members of this type, see TCustomDAConnection members.

Unit
DBAccess

Syntax

TCustomDAConnection = class(TCustomConnection);

Remarks
TCustomDAConnection is a base class for components that establish connection with database, provide customised login support, and perform transaction control.

Do not create instances of TCustomDAConnection. To add a component that represents a connection to a source of data, use descendants of the TCustomDAConnection class.

© 1997-2021
Devart. All Rights Reserved.

Provide Feedback

5.11.1.4.1 Members

TCustomDAConnection class overview.

Properties
### Name | Description
--- | ---
ConnectDialog | Allows to link a TCustomConnectDialog component.
ConnectString | Used to specify the connection information, such as: UserName, Password, Server, etc.
ConvertEOL | Allows customizing line breaks in string fields and parameters.
InTransaction | Indicates whether the transaction is active.
LoginPrompt | Specifies whether a login dialog appears immediately before opening a new connection.
Options | Specifies the connection behavior.
Password | Serves to supply a password for login.
Pooling | Enables or disables using connection pool.
PoolingOptions | Specifies the behaviour of connection pool.
Server | Serves to supply the server name for login.
Username | Used to supply a user name for login.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyUpdates</td>
<td>Overloaded. Applies changes in datasets.</td>
</tr>
<tr>
<td>Commit</td>
<td>Commits current transaction.</td>
</tr>
<tr>
<td>Connect</td>
<td>Establishes a connection to the server.</td>
</tr>
<tr>
<td>CreateSQL</td>
<td>Creates a component for queries execution.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Performs disconnect.</td>
</tr>
<tr>
<td>ExecProc</td>
<td>Allows to execute stored</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ExecProcEx</td>
<td>Allows to execute a stored procedure or function.</td>
</tr>
<tr>
<td>ExecSQL</td>
<td>Executes a SQL statement with parameters.</td>
</tr>
<tr>
<td>ExecSQLEx</td>
<td>Executes any SQL statement outside the TQuery or TSQL components.</td>
</tr>
<tr>
<td>GetDatabaseNames</td>
<td>Returns a database list from the server.</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetStoredProcNames</td>
<td>Returns a list of stored procedures from the server.</td>
</tr>
<tr>
<td>GetTableNames</td>
<td>Provides a list of available tables names.</td>
</tr>
<tr>
<td>MonitorMessage</td>
<td>Sends a specified message through the TCustomDASQLMonitor component.</td>
</tr>
<tr>
<td>Ping</td>
<td>Used to check state of connection to the server.</td>
</tr>
<tr>
<td>RemoveFromPool</td>
<td>Marks the connection that should not be returned to the pool after disconnect.</td>
</tr>
<tr>
<td>Rollback</td>
<td>Discards all current data changes and ends transaction.</td>
</tr>
<tr>
<td>StartTransaction</td>
<td>Begins a new user transaction.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnConnectionLost</td>
<td>This event occurs when connection was lost.</td>
</tr>
<tr>
<td>OnError</td>
<td>This event occurs when an error has arisen in the connection.</td>
</tr>
</tbody>
</table>
Properties of the `TCustomDACConnection` class.

For a complete list of the `TCustomDACConnection` class members, see the `TCustomDACConnection Members` topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ConnectDialog</code></td>
<td>Allows to link a <code>TCustomConnectDialog</code> component.</td>
</tr>
<tr>
<td><code>ConnectString</code></td>
<td>Used to specify the connection information, such as: <code>UserName</code>, <code>Password</code>, <code>Server</code>, etc.</td>
</tr>
<tr>
<td><code>ConvertEOL</code></td>
<td>Allows customizing line breaks in string fields and parameters.</td>
</tr>
<tr>
<td><code>InTransaction</code></td>
<td>Indicates whether the transaction is active.</td>
</tr>
<tr>
<td><code>LoginPrompt</code></td>
<td>Specifies whether a login dialog appears immediately before opening a new connection.</td>
</tr>
<tr>
<td><code>Options</code></td>
<td>Specifies the connection behavior.</td>
</tr>
<tr>
<td><code>Password</code></td>
<td>Serves to supply a password for login.</td>
</tr>
<tr>
<td><code>Pooling</code></td>
<td>Enables or disables using connection pool.</td>
</tr>
<tr>
<td><code>PoolingOptions</code></td>
<td>Specifies the behaviour of connection pool.</td>
</tr>
<tr>
<td><code>Server</code></td>
<td>Serves to supply the server name for login.</td>
</tr>
<tr>
<td><code>Username</code></td>
<td>Used to supply a user name for login.</td>
</tr>
</tbody>
</table>

**See Also**

- [TCustomDACConnection Class](#)
• **TCustomDAConnection Class Members**

5.11.1.4.2.1 ConnectDialog Property

Allows to link a [TCustomConnectDialog](#) component.

**Class**

**TCustomDAConnection**

**Syntax**

```property ConnectDialog: TCustomConnectDialog;```

**Remarks**

Use the ConnectDialog property to assign to connection a [TCustomConnectDialog](#) component.

**See Also**

• [TCustomConnectDialog](#)

5.11.1.4.2.2 ConnectString Property

Used to specify the connection information, such as: UserName, Password, Server, etc.

**Class**

**TCustomDAConnection**

**Syntax**

```property ConnectString: string stored False;```

**Remarks**

ODAC recognizes an ODBC-like syntax in provider string property values. Within the string, elements are delimited by using a semicolon. Each element consists of a keyword, an equal sign character, and the value passed on initialization. For example:
Server=London1;User ID=nancyd

Connection parameters

The following connection parameters can be used to customize connection:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoginPrompt</td>
<td>Specifies whether a login dialog appears immediately before opening a new connection.</td>
</tr>
<tr>
<td>Pooling</td>
<td>Enables or disables using connection pool.</td>
</tr>
<tr>
<td>ConnectionLifeTime</td>
<td>Used to specify the maximum time during which an opened connection can be used by connection pool.</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Used to specify the maximum number of connections that can be opened in connection pool.</td>
</tr>
<tr>
<td>MinPoolSize</td>
<td>Used to specify the minimum number of connections that can be opened in connection pool.</td>
</tr>
<tr>
<td>Validate Connection</td>
<td>Used for a connection to be validated when it is returned from the pool.</td>
</tr>
<tr>
<td>Server</td>
<td>Serves to supply the server name for login.</td>
</tr>
<tr>
<td>Username</td>
<td>Used to supply a user name for login.</td>
</tr>
<tr>
<td>Password</td>
<td>Used to supply a user name for login.</td>
</tr>
<tr>
<td>Charset</td>
<td>Used to set the character set that ODAC uses to read and write character data.</td>
</tr>
<tr>
<td>UseUnicode</td>
<td>Used to enable or disable Unicode support.</td>
</tr>
<tr>
<td>Port</td>
<td>Used to specify the port number for the connection. Available in Direct Mode only.</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Used to specify the amount of time before an attempt to make a connection is considered unsuccessful.</td>
</tr>
<tr>
<td>Direct</td>
<td>Used for ODAC to connect directly over TCP/IP (in Direct mode) and without requiring Oracle software on the client side.</td>
</tr>
<tr>
<td>IPVersion</td>
<td>Used to specify the version of the Internet Protocol.</td>
</tr>
<tr>
<td>SID</td>
<td>Used to specify the security identifier.</td>
</tr>
<tr>
<td>ServiceName</td>
<td>Used to specify the name of the service.</td>
</tr>
<tr>
<td>ConnectMode</td>
<td>Used to specify the system privileges to use when a user connects to the server.</td>
</tr>
<tr>
<td>HomeName</td>
<td>Used to select the Oracle client to use with</td>
</tr>
</tbody>
</table>
# Schema

Used to change the current schema of the session to the specified schema.

## Old-style connection string

The old connection string format is also supported:

```
user_name/passwd@database
```

where: user_name - username for logging in to the server; passwd - password to log in; database - the database alias from the tnsnames.ora file.

## EZCONNECT connection string

ODAC also supports Oracle's [easy connect naming method](https://www.oracle.com/database/technologies/app-development/easy-connect-naming-method.html). An example of EZCONNECT connection string:

```
username/password@host:port/service_name
```

## See Also

- [Password](#)
- [Username](#)
- [Server](#)
- [Connect](#)

© 1997-2021 Devart. All Rights Reserved.

## 5.11.1.4.2.3 ConvertEOL Property

Allows customizing line breaks in string fields and parameters.

### Class

`TCustomDAConnection`

### Syntax

```
property ConvertEOL: boolean default False;
```

### Remarks

Affects the line break behavior in string fields and parameters. When fetching strings
(including the CLOB and LONG fields) with ConvertEOL = True, dataset converts their line breaks from the LF to CRLF form. And when posting strings to server with ConvertEOL turned on, their line breaks are converted from CRLF to LF form. By default, strings are not converted.

5.11.1.4.2.4  InTransaction Property

Indicates whether the transaction is active.

Class

**TCustomDAConnection**

Syntax

```plaintext
property InTransaction: boolean;
```

Remarks

Examine the InTransaction property at runtime to determine whether user transaction is currently in progress. In other words InTransaction is set to True when user explicitly calls `StartTransaction`. Calling `Commit` or `Rollback` sets InTransaction to False. The value of the InTransaction property cannot be changed directly.

**Important note:** The InTransaction property always shows actual user transaction state on the server. This means that if transaction was implicitly ended by server-side logic, InTransaction becomes False.

See Also

- `StartTransaction`
- `Commit`
- `Rollback`
5.11.1.4.2.5 LoginPrompt Property

Specifies whether a login dialog appears immediately before opening a new connection.

Class
TCustomDAConnection

Syntax

```
property LoginPrompt default DefValLoginPrompt;
```

Remarks
Specifies whether a login dialog appears immediately before opening a new connection. If `ConnectDialog` is not specified, the default connect dialog will be shown. The connect dialog will appear only if the OdacVcl unit appears to the `uses` clause.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.4.2.6 Options Property

Specifies the connection behavior.

Class
TCustomDAConnection

Syntax

```
property Options: TDAConnectionOptions;
```

Remarks
Set the properties of `Options` to specify the behaviour of the connection.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowImplicitConnect</td>
<td>Specifies whether to allow or not implicit connection opening.</td>
</tr>
<tr>
<td>DefaultSortType</td>
<td>Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the</td>
</tr>
</tbody>
</table>
### Password Property

Serves to supply a password for login.

**Class**

`TCustomDAConnection`

**Syntax**

```pascal
property Password: string stored False;
```

**Remarks**

Use the Password property to supply a password to handle server's request for a login.

Application server can use the `TOraSession.ProxySession` property and verify user password itself.

**Warning:** Storing hard-coded user name and password entries as property values or in code for the OnLogin event handler can compromise server security.

**See Also**

- [Username](#)
- [Server](#)
5.11.1.4.2.8  Pooling Property

Enables or disables using connection pool.

Class

TCustomDAConnection

Syntax

property Pooling: boolean default DefValPooling;

Remarks

Normally, when TCustomDAConnection establishes connection with the server it takes server memory and time resources for allocating new server connection. For example, pooling can be very useful when using disconnect mode. If an application has wide user activity that forces many connect/disconnect operations, it may spend a lot of time on creating connection and sending requests to the server. TCustomDAConnection has software pool which stores open connections with identical parameters.

Connection pool uses separate thread that validates the pool every 30 seconds. Pool validation consists of checking each connection in the pool. If a connection is broken due to a network problem or another reason, it is deleted from the pool. The validation procedure removes also connections that are not used for a long time even if they are valid from the pool.

Set Pooling to True to enable pooling. Specify correct values for PoolingOptions. Two connections belong to the same pool if they have identical values for the parameters: MinPoolSize, MaxPoolSize, Validate, ConnectionLifeTime, TOraSession.Username, TOraSession.Server, TOraSession.ConnectMode, TOraSession.Options.

Note: Using Pooling := True can cause errors with working with temporary tables.

See Also

- Username
- Password
- PoolingOptions
5.11.4.2.9 PoolingOptions Property

Specifies the behaviour of connection pool.

Class

TCustomDACConnection

Syntax

property PoolingOptions: TPoolingOptions;

Remarks

Set the properties of PoolingOptions to specify the behaviour of connection pool.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionLifetime</td>
<td>Used to specify the maximum time during which an opened connection can be used by connection pool.</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Used to specify the maximum number of connections that can be opened in connection pool.</td>
</tr>
<tr>
<td>MinPoolSize</td>
<td>Used to specify the minimum number of connections that can be opened in the connection pool.</td>
</tr>
<tr>
<td>Validate</td>
<td>Used for a connection to be validated when it is returned from the pool.</td>
</tr>
</tbody>
</table>

See Also

- Pooling
5.11.1.4.2.10 Server Property

Serves to supply the server name for login.

Class

TCustomDAConnection

Syntax

```property Server: string;```

Remarks

Use the Server property to supply server name to handle server's request for a login.

See Also

- Username
- Password

© 1997-2021
Devart. All Rights Reserved.

5.11.1.4.2.11 Username Property

Used to supply a user name for login.

Class

TCustomDAConnection

Syntax

```property Username: string;```

Remarks

Use the Username property to supply a user name to handle server's request for login. If this property is not set, and OS authentication is used, a connection can be established. Otherwise an error is arised.

Warning: Storing hard-coded user name and password entries as property values or in code for the OnLogin event handler can compromise server security.

See Also
## 5.11.1.4.3 Methods

Methods of the **TCustomDAConnection** class.

For a complete list of the **TCustomDAConnection** class members, see the **TCustomDAConnection Members** topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyUpdates</strong></td>
<td>Overloaded. Applies changes in datasets.</td>
</tr>
<tr>
<td><strong>Commit</strong></td>
<td>Commits current transaction.</td>
</tr>
<tr>
<td><strong>Connect</strong></td>
<td>Establishes a connection to the server.</td>
</tr>
<tr>
<td><strong>CreateSQL</strong></td>
<td>Creates a component for queries execution.</td>
</tr>
<tr>
<td><strong>Disconnect</strong></td>
<td>Performs disconnect.</td>
</tr>
<tr>
<td><strong>ExecProc</strong></td>
<td>Allows to execute stored procedure or function providing its name and parameters.</td>
</tr>
<tr>
<td><strong>ExecProcEx</strong></td>
<td>Allows to execute a stored procedure or function.</td>
</tr>
<tr>
<td><strong>ExecSQL</strong></td>
<td>Executes a SQL statement with parameters.</td>
</tr>
<tr>
<td><strong>ExecSQLEx</strong></td>
<td>Executes any SQL statement outside the TQuery or TSQL components.</td>
</tr>
<tr>
<td><strong>GetDatabaseNames</strong></td>
<td>Returns a database list from the server.</td>
</tr>
<tr>
<td><strong>GetKeyFieldNames</strong></td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td><strong>GetStoredProcNames</strong></td>
<td>Returns a list of stored procedures from the server.</td>
</tr>
<tr>
<td><strong>GetTableNames</strong></td>
<td>Provides a list of available</td>
</tr>
</tbody>
</table>
MonitorMessage

Sends a specified message through the TCustomDASQLMonitor component.

Ping

Used to check state of connection to the server.

RemoveFromPool

Marks the connection that should not be returned to the pool after disconnect.

Rollback

Discards all current data changes and ends transaction.

StartTransaction

Begins a new user transaction.

See Also

- TCustomDAConnection Class
- TCustomDAConnection Class Members

5.11.1.4.3.1 ApplyUpdates Method

Applies changes in datasets.

Class

TCustomDAConnection

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyUpdates</td>
<td>Applies changes from all active datasets.</td>
</tr>
<tr>
<td>ApplyUpdates(const DataSets: array of TCustomDADataset)</td>
<td>Applies changes from the specified datasets.</td>
</tr>
</tbody>
</table>
Applies changes from all active datasets.

Class

TCustomDAConnection

Syntax

procedure ApplyUpdates; overload; virtual;

Remarks

Call the ApplyUpdates method to write all pending cached updates from all active datasets attached to this connection to a database or from specific datasets. The ApplyUpdates method passes cached data to the database for storage, takes care of committing or rolling back transactions, and clearing the cache when the operation is successful.

Using ApplyUpdates for connection is a preferred method of updating datasets rather than calling each individual dataset's ApplyUpdates method.

See Also

- TMemDataSet.CachedUpdates
- TMemDataSet.ApplyUpdates

Applies changes from the specified datasets.

Class

TCustomDAConnection

Syntax

procedure ApplyUpdates(DataSets: array of TCustomDADataSet); overload; virtual;

Parameters

DataSets

A list of datasets changes in which are to be applied.

Remarks
Call the ApplyUpdates method to write all pending cached updates from the specified datasets. The ApplyUpdates method passes cached data to the database for storage, takes care of committing or rolling back transactions and clearing the cache when operation is successful.

Using ApplyUpdates for connection is a preferred method of updating datasets rather than calling each individual dataset's ApplyUpdates method.

**Class**

`TCustomDAConnection`

**Syntax**

```plaintext
procedure Commit; virtual;
```

**Remarks**

Call the Commit method to commit current transaction. On commit server writes permanently all pending data updates associated with the current transaction to the database and then ends the transaction. The current transaction is the last transaction started by calling `StartTransaction`.

**See Also**

- [Rollback](#)
- [StartTransaction](#)
- `[TOraDataSet.FetchAll](#)`

---

5.11.1.4.3.2 Commit Method

Commits current transaction.

5.11.1.4.3.3 Connect Method

Establishes a connection to the server.

Class
**TCustomDAConnection**

**Syntax**

```
procedure Connect; overload; procedure Connect(const ConnectString: string); overload;
```

**Remarks**

Call the Connect method to establish a connection to the server. Connect sets the Connected property to True. If ConnectPrompt is True, Connect prompts user for login information as required by the server, or otherwise tries to establish a connection using values provided in the Username, Password, and Server properties.

If the Username and Password properties are not specified, then Oracle uses authentication information supplied at the operating system login process. For this feature to work make sure that your Oracle instance is appropriately tuned (see the Oracle documentation).

**See Also**

- Disconnect
- Username
- Password
- Server
- ConnectDialog

© 1997-2021 Devart. All Rights Reserved.

Request Support DAC Forum Provide Feedback

5.11.1.4.3.4  CreateSQL Method

Creates a component for queries execution.

**Class**

`TCustomDAConnection`

**Syntax**

```
function CreateSQL: TCustomDASQL; virtual;
```

**Return Value**

A new instance of the class.
Remarks
Call the CreateSQL to return a new instance of the TCustomDASQL class and associates it with this connection object. In the descendant classes this method should be overridden to create an appropriate descendant of the TCustomDASQL component.

Class
TCustomDAConnection

Syntax

```plaintext
procedure Disconnect;
```

Remarks
Call the Disconnect method to drop a connection to database. Before the connection component is deactivated, all associated datasets are closed. Calling Disconnect is similar to setting the Connected property to False.

In most cases, closing a connection frees system resources allocated to the connection.

If user transaction is active, e.g. the InTransaction flag is set, calling to Disconnect commits the current user transaction.

**Note:** If a previously active connection is closed and then reopened, any associated datasets must be individually reopened; reopening the connection does not automatically reopen associated datasets.

See Also
- Connect
5.11.1.4.3.6 ExecProc Method

Allows to execute stored procedure or function providing its name and parameters.

Class

TCustomDAConnection

Syntax

```
function ExecProc(const Name: string; const Params: array of variant): variant;
virtual;
```

Parameters

Name

Holds the name of the stored procedure or function.

Params

Holds the parameters of the stored procedure or function.

Return Value

the result of the stored procedure.

Remarks

Allows to execute stored procedure or function providing its name and parameters.

Use the following Name value syntax for executing specific overloaded routine:
"StoredProcName:1" or "StoredProcName:5". The first example executes the first overloaded
stored procedure, while the second example executes the fifth overloaded procedure.

Assign parameters' values to the Params array in exactly the same order and number as they
appear in the stored procedure declaration. Out parameters of the procedure can be
accessed with the ParamByName procedure.

If the value of an input parameter was not included to the Params array, parameter default
value is taken. Only parameters at the end of the list can be unincluded to the Params array. If
the parameter has no default value, the NULL value is sent.

**Note:** Stored functions unlike stored procedures return result values that are obtained
internally through the RESULT parameter. You will no longer have to provide anonymous
value in the Params array to describe the result of the function. The stored function result is
obtained from the Params[0] indexed property or with the ParamByName("RESULT") method
call.

For further examples of parameter usage see [ExecSQL](#), [ExecSQLEx](#).
Example

For example, having stored function declaration presented in Example 1), you may execute it and retrieve its result with commands presented in Example 2):

Example 1)

```sql
CREATE procedure MY_SUM (  
    A INTEGER,  
    B INTEGER)  
RETURNS (  
    RESULT INTEGER)  
as  
begin  
    Result = a + b;  
end;  
```

Example 2)

```sql
Label2.Caption:= MyOraConnection1.ParamByName('Result').AsString;  
```

See Also
- ExecProcEx
- ExecSQL
- ExecSQLEX
- TOraSession.ParamByName
- TOraSession.SQL

5.11.1.4.3.7 ExecProcEx Method

Allows to execute a stored procedure or function.

Class

TCustomDACConnection

Syntax

```
function ExecProcEx(const Name: string; const Params: array of variant): variant; virtual;
```

Parameters

Name
- Holds the stored procedure name.

Params
Holds an array of pairs of parameters' names and values.

**Return Value**
the result of the stored procedure.

**Remarks**
Allows to execute a stored procedure or function. Provide the stored procedure name and its parameters to the call of ExecProcEx.

Use the following Name value syntax for executing specific overloaded routine: "StoredProcName:1" or "StoredProcName:5". The first example executes the first overloaded stored procedure, while the second example executes the fifth overloaded procedure.

Assign pairs of parameters' names and values to a Params array so that every name comes before its corresponding value when an array is being indexed.

Out parameters of the procedure can be accessed with the ParamByName procedure. If the value for an input parameter was not included to the Params array, the parameter default value is taken. If the parameter has no default value, the NULL value is sent.

**Note:** Stored functions unlike stored procedures return result values that are obtained internally through the RESULT parameter. You will no longer have to provide anonymous value in the Params array to describe the result of the function. Stored function result is obtained from the Params[0] indexed property or with the ParamByName("RESULT") method call.

For an example of parameters usage see ExecSQLEx.

**Example**
If you have some stored procedure accepting four parameters, and you want to provide values only for the first and fourth parameters, you should call ExecProcEx in the following way:

```plaintext
Connection.ExecProcEx('Some_Stored_Procedure', ['Param_Name1', 'Param_Value1', 'Param_Name4', 'Param_Value4']);
```

**See Also**
- ExecSQL
- ExecSQLEx
- ExecProc
5.11.1.4.3.8 ExecSQL Method

Executes a SQL statement with parameters.

Class

TCustomDAConnection

Syntax

```pascal
function ExecSQL(const Text: string): variant;
overload; function ExecSQL(const Text: string; const Params: array of variant): variant; overload; virtual;
```

Parameters

- **Text**
  - a SQL statement to be executed.
- **Params**
  - Array of parameter values arranged in the same order as they appear in SQL statement.

Return Value

- Out parameter with the name Result will hold the result of function having data type dtString. Otherwise returns Null.

Remarks

Use the ExecSQL method to execute any SQL statement outside the TCustomDADataset or TCustomDASQL components. Supply the Params array with the values of parameters arranged in the same order as they appear in a SQL statement which itself is passed to the Text string parameter.

The Params array must contain all IN and OUT parameters defined in SQL statement. For OUT parameters provide any values of valid types so that they are explicitly defined before call to an ExecSQL method.

Out parameter with the name Result will hold the result of function having data type dtString. If none of the parameters in the Text statement is named Result, ExecSQL will return Null.

To get the values of OUT parameters use the ParamByName function.

Example

```pascal
OraSession.ExecSQL('begin :A:= :B + :C; end;', [0, 5, 3]);
A:= OraSession.ParamByName('A').AsInteger;
```

See Also
5.11.1.4.3.9  ExecSQLEx Method

Executes any SQL statement outside the TQuery or TSQL components.

Class

TCustomDAConnection

Syntax

```
function ExecSQLEx(const Text: string; const Params: array of variant): variant; virtual;
```

Parameters

* **Text**
  a SQL statement to be executed.

* **Params**
  Array of parameter values arranged in the same order as they appear in SQL statement.

Return Value

Out parameter with the name Result will hold the result of a function having data type dtString. Otherwise returns Null.

Remarks

Call the ExecSQLEx method to execute any SQL statement outside the TQuery or TSQL components. Supply the Params array with values arranged in pairs of parameter name and its value. This way each parameter name in the array is found on even index values whereas parameter value is on odd index value but right after its parameter name. The parameter pairs must be arranged according to their occurrence in a SQL statement which itself is passed in the Text string parameter.

The Params array must contain all IN and OUT parameters defined in the SQL statement. For OUT parameters provide any values of valid types so that they are explicitly defined before call to the ExecSQLEx method.

Out parameter with the name Result will hold the result of a function having data type dtString. If neither of the parameters in the Text statement is named Result, ExecSQLEx will return
Null.

To get the values of OUT parameters use the ParamByName function.

Example

```pascal
OraConnection.ExecSQLEx('begin :A:= :B + :C; end;', ['A', 0, 'B', 5, 'C', 3]);
A:= OraConnection.ParamByName('A').AsInteger;
```

See Also

- `ExecSQL`

© 1997-2021 Devart. All Rights Reserved.

5.11.1.4.3.10 GetDatabaseNames Method

Returns a database list from the server.

Class

`TCustomDACConnection`

Syntax

```pascal
procedure GetDatabaseNames(List: TStrings); virtual;
```

Parameters

- `List`
  - A TStrings descendant that will be filled with database names.

Remarks

Populates a string list with the names of databases.

**Note:** Any contents already in the target string list object are eliminated and overwritten by data produced by `GetDatabaseNames`.

See Also

- `GetTableNames`
- `GetStoredProcNames`
5.11.1.4.3.11  GetKeyFieldNames Method

Provides a list of available key field names.

Class

TCustomDAConnection

Syntax

```
procedure GetKeyFieldNames(const TableName: string; List: TStrings); virtual;
```

Parameters

- **TableName**
  - Holds the table name
- **List**
  - The list of available key field names

Return Value

- Key field name

Remarks

Call the GetKeyFieldNames method to get the names of available key fields. Populates a string list with the names of key fields in tables.

See Also

- GetTableNames
- GetStoredProcNames

5.11.1.4.3.12  GetStoredProcNames Method

Returns a list of stored procedures from the server.

Class

TCustomDAConnection

Syntax

```
procedure GetStoredProcNames(List: TStrings; AllProcs: boolean = False); virtual;
```
Parameters

List
A TStrings descendant that will be filled with the names of stored procedures in the database.

AllProcs
True, if stored procedures from all schemas or including system procedures (depending on the server) are returned. False otherwise.

Remarks

Call the GetStoredProcNames method to get the names of available stored procedures and functions. GetStoredProcNames populates a string list with the names of stored procs in the database. If AllProcs = True, the procedure returns to the List parameter the names of the stored procedures that belong to all schemas; otherwise, List will contain the names of functions that belong to the current schema.

Note: Any contents already in the target string list object are eliminated and overwritten by data produced by GetStoredProcNames.

See Also
- GetDatabaseNames
- GetTableNames

© 1997-2021 Devart. All Rights Reserved.

5.11.1.4.3.13 GetTableNames Method

Provides a list of available tables names.

Class
TCustomDACConnection

Syntax

procedure GetTableNames(List: TStrings; AllTables: boolean = False; OnlyTables: boolean = False); virtual;

Parameters

List
A TStrings descendant that will be filled with table names.

AllTables
True, if procedure returns all table names including the names of system tables to the List parameter.

Remarks
Call the GetTableNames method to get the names of available tables. Populates a string list with the names of tables in the database. If AllTables = True, procedure returns all table names including the names of system tables to the List parameter, otherwise List will not contain the names of system tables. If AllTables = True, the procedure returns to the List parameter the names of the tables that belong to all schemas; otherwise, List will contain the names of the tables that belong to the current schema.

Note: Any contents already in the target string list object are eliminated and overwritten by the data produced by GetTableNames.

See Also
- GetDatabaseNames
- GetStoredProcNames

5.11.1.4.3.14 MonitorMessage Method

Sends a specified message through the TCustomDASQLMonitor component.

Class
TCustomDAConnection

Syntax

```plaintext
procedure MonitorMessage(const Msg: string);
```

Parameters

*Msg*
Message text that will be sent.

Remarks
Call the MonitorMessage method to output specified message via the TCustomDASQLMonitor component.
5.11.1.4.3.15 Ping Method

Used to check state of connection to the server.

Class

TCustomDAConnection

Syntax

\begin{verbatim}
procedure ping;
\end{verbatim}

Remarks

The method is used for checking server connection state.

5.11.1.4.3.16 RemoveFromPool Method

Marks the connection that should not be returned to the pool after disconnect.

Class

TCustomDAConnection

Syntax

\begin{verbatim}
procedure RemoveFromPool;
\end{verbatim}

Remarks

Call the RemoveFromPool method to mark the connection that should be deleted after disconnect instead of returning to the connection pool.

See Also

• Pooling
5.11.1.4.3.17 Rollback Method

Discards all current data changes and ends transaction.

Class
TCustomDACConnection

Syntax

procedure Rollback; virtual;

Remarks
Call the Rollback method to discard all updates, insertions, and deletions of data associated with the current transaction to the database server and then end the transaction. The current transaction is the last transaction started by calling StartTransaction.

See Also
- Commit
- StartTransaction
- TOraDataSet.FetchAll

5.11.1.4.3.18 StartTransaction Method

Begins a new user transaction.

Class
TCustomDACConnection

Syntax

procedure StartTransaction; virtual;

Remarks
Call the StartTransaction method to begin a new user transaction against the database server. Before calling StartTransaction, an application should check the status of the InTransaction property. If InTransaction is True, indicating that a transaction is already in progress, a subsequent call to StartTransaction without first calling Commit or Rollback to end the current transaction raises EDatabaseError. Calling StartTransaction when connection is closed also raises EDatabaseError.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until an application calls Commit to save the changes, or Rollback to cancel them.

See Also
- Commit
- Rollback
- InTransaction

© 1997-2021 Devart. All Rights Reserved.

Events

Events of the TCustomDACConnection class.

For a complete list of the TCustomDACConnection class members, see the TCustomDACConnection Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnConnectionLost</td>
<td>This event occurs when connection was lost.</td>
</tr>
<tr>
<td>OnError</td>
<td>This event occurs when an error has arisen in the connection.</td>
</tr>
</tbody>
</table>

See Also
- TCustomDACConnection Class
- TCustomDACConnection Class Members

© 1997-2021 Devart. All Rights Reserved.
5.11.1.4.4.1 OnConnectionLost Event

This event occurs when connection was lost.

Class

TCustomDAConnection

Syntax

property OnConnectionLost: TConnectionLostEvent;

Remarks

Write the OnConnectionLost event handler to process fatal errors and perform failover.

Note: To use the OnConnectionLost event handler, you should explicitly add the MemData unit to the 'uses' list and set the TCustomDAConnection.Options.LocalFailover property to True.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.4.4.2 OnError Event

This event occurs when an error has arisen in the connection.

Class

TCustomDAConnection

Syntax

property OnError: TDAConnectionErrorEvent;

Remarks

Write the OnError event handler to respond to errors that arise with connection. Check the E parameter to get the error code. Set the Fail parameter to False to prevent an error dialog from being displayed and to raise the EAbort exception to cancel current operation. The default value of Fail is True.

© 1997-2021
Devart. All Rights Reserved.
5.11.1.5 TCustomDADataSet Class

Encapsulates general set of properties, events, and methods for working with data accessed through various database engines.

For a list of all members of this type, see TCustomDADataSet members.

Unit

DBAccess

Syntax

TCustomDADataSet = class(TMemDataSet);

Remarks

TCustomDADataSet encapsulates general set of properties, events, and methods for working with data accessed through various database engines. All database-specific features are supported by descendants of TCustomDADataSet.

Applications should not use TCustomDADataSet objects directly.

Inheritance Hierarchy

TMemDataSet
   TCustomDADataSet

© 1997-2021 Devart. All Rights Reserved.

5.11.1.5.1 Members

TCustomDADataSet class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong></td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong></td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>IsQuery</strong></td>
<td>Used to check whether SQL statement returns rows.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>KeyFields</strong></td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if</td>
</tr>
</tbody>
</table>
they were empty before updating the database.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocalConstraints</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>MacroCount</strong></td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td><strong>Macros</strong></td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td><strong>MasterFields</strong></td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td><strong>MasterSource</strong></td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td>Used to specify the behaviour of TCustomDADataset object.</td>
</tr>
<tr>
<td><strong>ParamCheck</strong></td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong></td>
<td>Used to indicate how many parameters are there in the Params property.</td>
</tr>
<tr>
<td><strong>Params</strong></td>
<td>Used to view and set parameter names, values, and data types dynamically.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is</td>
</tr>
<tr>
<td><strong>Readonly</strong></td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td><strong>RefreshOptions</strong></td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong></td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td><strong>SQLDelete</strong></td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td><strong>SQLInsert</strong></td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td><strong>SQLLock</strong></td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td><strong>SQLRecCount</strong></td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td><strong>SQLRefresh</strong></td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td><strong>SQLUpdate</strong></td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td><strong>UniDirectional</strong></td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from TMemDataSet)</td>
<td>Used to indicate the update</td>
</tr>
</tbody>
</table>
status for the current record when cached updates are enabled.

| UpdatesPending (inherited from **TMemDataSet**) | Used to check the status of the cached updates buffer. |

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>BreakExec</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td>CancelRange (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>CreateBlobStream</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td>DeferredPost (inherited from <strong>TMemDataSet</strong>)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>DeleteWhere</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td>EditRangeEnd (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td></td>
<td>(inherited from <strong>TMemDataSet</strong>)</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>Used to find out whether TCustomDADataSet has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>Used to learn whether TCustomDADataSet is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>Used to learn whether TCustomDADataSet is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>Moves the cursor to a specific record or to the first record in the dataset</td>
</tr>
<tr>
<td></td>
<td>that matches or is greater than the values specified in the KeyValues</td>
</tr>
<tr>
<td></td>
<td>parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>Determines if a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only</td>
</tr>
<tr>
<td></td>
<td>from <strong>TMemDataSet</strong>)</td>
</tr>
<tr>
<td></td>
<td>its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetDataType</strong></td>
<td>Returns internal field types defined in the MemData and accompanying</td>
</tr>
<tr>
<td></td>
<td>modules.</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong></td>
<td>Returns a multireference shared object from field.</td>
</tr>
<tr>
<td><strong>GetFieldPrecision</strong></td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td>Locate (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Excludes features that don't need to be included to the <strong>TMemDataSet.Locate</strong> method of <strong>TDataSet</strong>.</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>ParamByName</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td>Prepare</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>RefreshRecord</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td>RestoreSQL</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td>RestoreUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord (inherited from <strong>TMemDataSet</strong>)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveSQL</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
</tbody>
</table>
SaveToXML (inherited from TMemDataSet)  Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

SetOrderBy  Builds an ORDER BY clause of a SELECT statement.

SetRange (inherited from TMemDataSet)  Sets the starting and ending values of a range, and applies it.

SetRangeEnd (inherited from TMemDataSet)  Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

SetRangeStart (inherited from TMemDataSet)  Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

SQLSaved  Determines if the SQL property value was saved to the BaseSQL property.

UnLock  Releases a record lock.

UnPrepare (inherited from TMemDataSet)  Frees the resources allocated for a previously prepared query on the server and client sides.

UpdateResult (inherited from TMemDataSet)  Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

UpdateStatus (inherited from TMemDataSet)  Indicates the current update status for the dataset when cached updates are enabled.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
</tbody>
</table>
### AfterFetch
Occurs after dataset finishes fetching data from server.

### AfterUpdateExecute
Occurs after executing insert, delete, update, lock and refresh operations.

### BeforeFetch
Occurs before dataset is going to fetch block of records from the server.

### BeforeUpdateExecute
Occurs before executing insert, delete, update, lock, and refresh operations.

### OnUpdateError (inherited from TMemDataSet)
Occurs when an exception is generated while cached updates are applied to a database.

### OnUpdateRecord (inherited from TMemDataSet)
Occurs when a single update component can not handle the updates.

---

## Properties of the TCustomDADataset class.

For a complete list of the TCustomDADataset class members, see the TCustomDADataset Members topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates</td>
<td>(inherited from TMemDataSet) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td>Connection</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Property</strong></td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used to display the statement that is being executed and the values and</td>
</tr>
<tr>
<td></td>
<td>types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>Used to specify the fields that correspond to the foreign key fields from</td>
</tr>
<tr>
<td></td>
<td>MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>Used to define the number of rows to be transferred across the network at</td>
</tr>
<tr>
<td></td>
<td>the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong></td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong></td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy,</td>
</tr>
<tr>
<td></td>
<td>and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>IsQuery</strong></td>
<td>Used to check whether SQL statement returns rows.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>KeyFields</strong></td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate</td>
</tr>
<tr>
<td></td>
<td>properties if they were empty before updating the database.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>MacroCount</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>MasterFields</td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td>MasterSource</td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to specify the behaviour of TCustomDADataSet object.</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td>ParamCount</td>
<td>Used to indicate how many parameters are there in the Params property.</td>
</tr>
<tr>
<td>Params</td>
<td>Used to view and set parameter names, values, and data types dynamically.</td>
</tr>
<tr>
<td>Prepared</td>
<td>(inherited from <strong>TMemDataSet</strong>) Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td>Ranged</td>
<td>(inherited from <strong>TMemDataSet</strong>) Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>RowsAffected</strong></td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td><strong>SQLDelete</strong></td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td><strong>SQLInsert</strong></td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td><strong>SQLLock</strong></td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td><strong>SQLRecCount</strong></td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td><strong>SQLRefresh</strong></td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td><strong>SQLUpdate</strong></td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td><strong>UniDirectional</strong></td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from TMemDataSet)</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong>  (inherited from TMemDataSet)</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>
5.11.1.5.2.1  BaseSQL Property

Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.

Class

TCustomDADataSet

Syntax

```pascal
property BaseSQL: string;
```

Remarks

Use the BaseSQL property to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL, only macros are expanded. SQL text with all these changes can be returned by `FinalSQL`.

See Also

- `FinalSQL`
- `AddWhere`
- `SaveSQL`
- `SQLSaved`
- `RestoreSQL`

© 1997-2021
Devart. All Rights Reserved.

5.11.1.5.2.2  Conditions Property

Used to add WHERE conditions to a query

Class
**TCustomDADataSet**

**Syntax**

```pascal
property Conditions: TDAConditions stored False;
```

**Remarks**

Use the `Conditions` property to specify conditions that will be applied to the data store.

**See Also**

- `TDAConditions`

© 1997-2021 Devart. All Rights Reserved.

---

**5.11.1.5.2.3  Connection Property**

Used to specify a connection object to use to connect to a data store.

**Class**

`TCustomDADataSet`

**Syntax**

```pascal
property Connection: TCustomDAConnection;
```

**Remarks**

Use the `Connection` property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided `TCustomDAConnection` or its descendant class objects.

At runtime, link an instance of a `TCustomDAConnection` descendant to the `Connection` property.

© 1997-2021 Devart. All Rights Reserved.

---

**5.11.1.5.2.4  DataTypeMap Property**

Used to set data type mapping rules.

**Class**

`TCustomDADataSet`
5.11.1.5.2.5  Debug Property

Used to display the statement that is being executed and the values and types of its parameters.

Class

TCustomDADataset

Syntax

property Debug: boolean default False;

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the OdacVcl unit to the uses clause of any unit in your project to make the Debug property work.

Note: If TOraSQLMonitor is used in the project and the TOraSQLMonitor.Active property is set to False, the debug window is not displayed.

See Also

• TCustomDASQL.Debug
Class

`TCustomDADataSet`

Syntax

```plaintext
property DetailFields: string;
```

Remarks

Use the DetailFields property to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship. DetailFields is a string containing one or more field names in the detail table. Separate field names with semicolons.

Use Field Link Designer to set the value in design time.

See Also

- `MasterFields`
- `MasterSource`

5.11.1.5.2.7  Disconnected Property

Used to keep dataset opened after connection is closed.

Class

`TCustomDADataSet`

Syntax

```plaintext
property Disconnected: boolean;
```

Remarks

Set the Disconnected property to True to keep dataset opened after connection is closed.
5.11.1.5.2.8 FetchRows Property

Used to define the number of rows to be transferred across the network at the same time.

Class

TCustomDADataSet

Syntax

```plaintext
property FetchRows: integer default 25;
```

Remarks

The number of rows that will be transferred across the network at the same time. This property can have a great impact on performance. So it is preferable to choose the optimal value of the FetchRows property for each SQL statement and software/hardware configuration experimentally.

The default value is 25.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.5.2.9 FilterSQL Property

Used to change the WHERE clause of SELECT statement and reopen a query.

Class

TCustomDADataSet

Syntax

```plaintext
property FilterSQL: string;
```

Remarks

The FilterSQL property is similar to the Filter property, but it changes the WHERE clause of SELECT statement and reopens query. Syntax is the same to the WHERE clause.

**Note:** the FilterSQL property adds a value to the WHERE condition as is. If you expect this value to be enclosed in brackets, you should bracket it explicitly.

Example
Query1.FilterSQL := 'Dept >= 20 and DName LIKE ''M%'';

See Also
- AddWhere

5.11.1.5.2.10 FinalSQL Property

Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.

Class
TCustomDADataSet

Syntax

property FinalSQL: string;

Remarks
Use FinalSQL to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros. This is the exact statement that will be passed on to the database server.

See Also
- FinalSQL
- AddWhere
- SaveSQL
- SQLSaved
- RestoreSQL
- BaseSQL

5.11.1.5.2.11 IsQuery Property

Used to check whether SQL statement returns rows.

Class
**TCustomDADataSet**

Syntax

```delphi
property IsQuery: boolean;
```

Remarks

After the TCustomDADataSet component is prepared, the IsQuery property returns True if SQL statement is a SELECT query.

Use the IsQuery property to check whether the SQL statement returns rows or not.

IsQuery is a read-only property. Reading IsQuery on unprepared dataset raises an exception.

**5.11.5.2.12  KeyFields Property**

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database.

Class

**TCustomDADataSet**

Syntax

```delphi
property KeyFields: string;
```

Remarks

TCustomDADataSet uses the KeyFields property to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating the database. For this feature KeyFields may hold a list of semicolon-delimited field names. If KeyFields is not defined before opening a dataset, TCustomDADataSet requests information about primary keys from server sending an additional query.

Assign the KeyFields property with a string containing the name of a field which will be later assigned with Oracle sequenced values. Beforehand Oracle sequence must be created and its name passed to the TOraDataSet.KeySequence property.

Sequences are generated when either Insert or Post method is called. Which of these two methods is used to modify the database is determined by the TOraDataSet.SequenceMode property.
property.

**Note:** Though keys may be created across a number of table fields, sequence is generated only for the first field found in the KeyFields property.

**See Also**
- SQLDelete
- SQLInsert
- SQLRefresh
- SQLUpdate
- TOraDataSet.KeySequence
- TOraDataSet.SequenceMode

**MacroCount Property**

Used to get the number of macros associated with the Macros property.

**Class**

TCustomDADataSet

**Syntax**

```pascal
property MacroCount: word;
```

**Remarks**

Use the MacroCount property to get the number of macros associated with the Macros property.

**See Also**
- Macros
5.11.1.5.2.14 Macros Property

Makes it possible to change SQL queries easily.

Class
TCustomDADataSet

Syntax

```
property Macros: TMacros stored False;
```

Remarks

With the help of macros you can easily change SQL query text at design- or runtime. Marcos extend abilities of parameters and allow to change conditions in a WHERE clause or sort order in an ORDER BY clause. You just insert &MacroName in the SQL query text and change value of macro in the Macro property editor at design time or call the MacroByName function at run time. At the time of opening the query macro is replaced by its value.

Example

```
OraQuery.SQL.Text := 'SELECT * FROM Dept ORDER BY &Order';
OraQuery.MacroByName('Order').Value := 'DeptNo';
OraQuery.Open;
```

See Also

- TMacro
- MacroByName
- Params

© 1997-2021
Devar. All Rights Reserved.

5.11.1.5.2.15 MasterFields Property

Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

Class
TCustomDADataSet

Syntax
property MasterFields: string;

Remarks

Use the MasterFields property after setting the MasterSource property to specify the names of one or more fields that are used as foreign keys for this dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.

MasterFields is a string containing one or more field names in the master table. Separate field names with semicolons.

Each time the current record in the master table changes, the new values in these fields are used to select corresponding records in this table for display.

Use Field Link Designer to set the values at design time after setting the MasterSource property.

See Also

- DetailFields
- MasterSource
- Master/Detail Relationships

© 1997-2021 Devart. All Rights Reserved.

5.11.1.5.2.16 MasterSource Property

Used to specify the data source component which binds current dataset to the master one.

Class

TCustomDADataset

Syntax

property MasterSource: TDataSource;

Remarks

The MasterSource property specifies the data source component which binds current dataset to the master one.

TCustomDADataset uses MasterSource to extract foreign key fields values from the master dataset when building master/detail relationship between two datasets. MasterSource must
point to another dataset; it cannot point to this dataset component.

When MasterSource is not \texttt{nil} dataset fills parameter values with corresponding field values from the current record of the master dataset.

**Note:** Do not set the \texttt{DataSource} property when building master/detail relationships. Although it points to the same object as the MasterSource property, it may lead to undesirable results.

**See Also**
- \texttt{MasterFields}
- \texttt{DetailFields}
- \texttt{Master/Detail Relationships}

---

5.11.5.2.17 Options Property

Used to specify the behaviour of \texttt{TCustomDADataset} object.

**Class**

\texttt{TCustomDADataset}

**Syntax**

\[
\text{property} \ \text{Options: TDADatasetOptions;}
\]

**Remarks**

Set the properties of Options to specify the behaviour of a \texttt{TCustomDADataset} object.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{AutoPrepare}</td>
<td>Used to execute automatic \texttt{Prepare} on the query execution.</td>
</tr>
<tr>
<td>\texttt{CacheCalcFields}</td>
<td>Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td>\texttt{CompressBlobMode}</td>
<td>Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td>\texttt{DefaultValues}</td>
<td>Used to request default values/expressions from the server and assign them to the</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DefaultExpression property</td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td>FieldsOrigin</td>
<td>Used for TCustomDADataSet to fill the Origin property of the TField objects by appropriate value when opening a dataset.</td>
</tr>
<tr>
<td>FlatBuffers</td>
<td>Used to control how a dataset treats data of the ftString and ftVarBytes fields.</td>
</tr>
<tr>
<td>InsertAllSetFields</td>
<td>Used to include all set dataset fields in the generated INSERT statement.</td>
</tr>
<tr>
<td>LocalMasterDetail</td>
<td>Used for TCustomDADataSet to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.</td>
</tr>
<tr>
<td>LongStrings</td>
<td>Used to represent string fields with the length that is greater than 255 as TStringField.</td>
</tr>
<tr>
<td>MasterFieldsNullable</td>
<td>Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).</td>
</tr>
<tr>
<td>NumberRange</td>
<td>Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.</td>
</tr>
<tr>
<td>QueryRecCount</td>
<td>Used for TCustomDADataSet to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td>QuoteNames</td>
<td>Used for TCustomDADataSet to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td>RemoveOnRefresh</td>
<td>Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td>RequiredFields</td>
<td>Used for TCustomDADataSet to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td>SetFieldsReadOnly</td>
<td>Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
</tbody>
</table>
**StrictUpdate**

Used for TCustomDADataSet to raise an exception when the number of updated or deleted records is not equal 1.

**TrimFixedChar**

Specifies whether to discard all trailing spaces in the string fields of a dataset.

**UpdateAllFields**

Used to include all dataset fields in the generated UPDATE and INSERT statements.

**UpdateBatchSize**

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

---

**See Also**

- Master/Detail Relationships
- TMemDataSet.CachedUpdates

© 1997-2021

Devart. All Rights Reserved.

Request Support DAC Forum Provide Feedback

5.11.1.5.2.18  ParamCheck Property

Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

**Class**

**TCustomDADataSet**

**Syntax**

```vbnet
property ParamCheck: boolean default True;
```

**Remarks**

Use the ParamCheck property to specify whether parameters for the Params property are generated automatically after the SQL property was changed.

Set ParamCheck to True to let dataset automatically generate the Params property for the dataset based on a SQL statement.

Setting ParamCheck to False can be used if the dataset component passes to a server the DDL statements that contain, for example, declarations of stored procedures which themselves will accept parameterized values. The default value is True.
5.11.1.5.2.19  ParamCount Property

Used to indicate how many parameters are there in the Params property.

Class

TCustomDADataSet

Syntax

property ParamCount: word;

Remarks

Use the ParamCount property to determine how many parameters are there in the Params property.

See Also

• Params

5.11.1.5.2.20  Params Property

Used to view and set parameter names, values, and data types dynamically.

Class

TCustomDADataSet

Syntax

property Params: TDAParams stored False;

Remarks

Contains the parameters for a query's SQL statement.
Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set the parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

See Also
- ParamByName
- Macros

5.11.1.5.2.21  ReadOnly Property

Used to prevent users from updating, inserting, or deleting data in the dataset.

Class
TCustomDADataset

Syntax

```
property ReadOnly: boolean default False;
```

Remarks

Use the ReadOnly property to prevent users from updating, inserting, or deleting data in the dataset. By default, ReadOnly is False, meaning that users can potentially alter data stored in the dataset.

To guarantee that users cannot modify or add data to a dataset, set ReadOnly to True.

When ReadOnly is True, the dataset's CanModify property is False.
5.11.1.5.2.22 RefreshOptions Property

Used to indicate when the editing record is refreshed.

Class

TCustomDADataSet

Syntax

property RefreshOptions: TRefreshOptions default [];

Remarks

Use the RefreshOptions property to determine when the editing record is refreshed. Refresh is performed by the RefreshRecord method. It queries the current record and replaces one in the dataset. Refresh record is useful when the table has triggers or the table fields have default values. Use roBeforeEdit to get actual data before editing.

The default value is [].

See Also

• RefreshRecord

© 1997-2021 Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback

5.11.1.5.2.23 RowsAffected Property

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

Class

TCustomDADataSet

Syntax

property RowsAffected: integer;

Remarks

Check RowsAffected to determine how many rows were inserted, updated, or deleted during
the last query operation. If RowsAffected is -1, the query has not inserted, updated, or deleted any rows.

5.11.1.5.2.24  SQL Property

Used to provide a SQL statement that a query component executes when its Open method is called.

Class

**TCustomDADataSet**

Syntax

```
property SQL: TStrings;
```

Remarks

Use the SQL property to provide a SQL statement that a query component executes when its Open method is called. At the design time the SQL property can be edited by invoking the String List editor in Object Inspector.

When SQL is changed, TCustomDADataSet calls Close and UnPrepare.

See Also

- SQLInsert
- SQLUpdate
- SQLDelete
- SQLRefresh

5.11.1.5.2.25  SQLDelete Property

Used to specify a SQL statement that will be used when applying a deletion to a record.

Class

**TCustomDADataSet**
Syntax

```
property SQLDelete: TStrings;
```

Remarks

Use the SQLDelete property to specify the SQL statement that will be used when applying a deletion to a record. Statements can be parameterized queries.

To create a SQLDelete statement at design-time, use the query statements editor.

Example

```
DELETE FROM Orders
    WHERE
    OrderID = :old_OrderID
```

See Also

- SQL
- SQLInsert
- SQLUpdate
- SQLRefresh

© 1997-2021 Devart. All Rights Reserved.

5.11.1.5.2.26 SQLInsert Property

Used to specify the SQL statement that will be used when applying an insertion to a dataset.

Class

TCustomDADataSet

Syntax

```
property SQLInsert: TStrings;
```

Remarks

Use the SQLInsert property to specify the SQL statement that will be used when applying an insertion to a dataset. Statements can be parameterized queries. Names of the parameters should be the same as field names. Parameters prefixed with OLD_ allow using current values of fields prior to the actual operation.
Use `ReturnParam` to return OUT parameters back to dataset.

To create a SQLInsert statement at design-time, use the query statements editor.

See Also
- SQL
- SQLUpdate
- SQLDelete
- SQLRefresh

5.11.1.5.2.27 SQLLock Property

Used to specify a SQL statement that will be used to perform a record lock.

Class
- `TCustomDADataset`

Syntax

```property
SQLLock: TStrings;
```

Remarks

Use the SQLLock property to specify a SQL statement that will be used to perform a record lock. Statements can be parameterized queries. Names of the parameters should be the same as field names. The parameters prefixed with OLD_ allow to use current values of fields prior to the actual operation.

To create a SQLLock statement at design-time, the use query statement editor.

See Also
- SQL
- SQLInsert
- SQLUpdate
- SQLDelete
- SQLRefresh
Reserved.

5.11.1.5.2.28 SQLRecCount Property

Used to specify the SQL statement that is used to get the record count when opening a dataset.

Class
TCustomDADataSet

Syntax

```
property SQLRecCount: TStrings;
```

Remarks

Use the SQLRecCount property to specify the SQL statement that is used to get the record count when opening a dataset. The SQL statement is used if the TADADatasetOptions.QueryRecCount property is True, and the TCustomDADataSet.FetchAll property is False. Is not used if the FetchAll property is True.

To create a SQLRecCount statement at design-time, use the query statements editor.

See Also
- SQLInsert
- SQLUpdate
- SQLDelete
- SQLRefresh
- TADADatasetOptions
- FetchingAll

© 1997-2021 Devart. All Rights Reserved.

5.11.1.5.2.29 SQLRefresh Property

Used to specify a SQL statement that will be used to refresh current record by calling the RefreshRecord procedure.

Class
TCustomDADataSet
Syntax

```
property SQLRefresh: TStrings;
```

Remarks

Use the SQLRefresh property to specify a SQL statement that will be used to refresh current record by calling the `RefreshRecord` procedure.

Different behavior is observed when the SQLRefresh property is assigned with a single WHERE clause that holds frequently altered search condition. In this case the WHERE clause from SQLRefresh is combined with the same clause of the SELECT statement in a SQL property and this final query is then sent to the database server.

To create a SQLRefresh statement at design-time, use the query statements editor.

Example

```
SELECT Shipname FROM Orders
  WHERE OrderID = :OrderID
```

See Also

- `RefreshRecord`
- `SQL`
- `SQLInsert`
- `SQLUpdate`
- `SQLDelete`

© 1997-2021 Devart. All Rights Reserved.

5.11.1.5.2.30  SQLUpdate Property

Used to specify a SQL statement that will be used when applying an update to a dataset.

Class

`TCustomDADataset`

Syntax

```
property SQLUpdate: TStrings;
```
Remarks

Use the SQLUpdate property to specify a SQL statement that will be used when applying an update to a dataset. Statements can be parameterized queries. Names of the parameters should be the same as field names. The parameters prefixed with OLD_ allow to use current values of fields prior to the actual operation.

Use ReturnParam to return OUT parameters back to the dataset.

To create a SQLUpdate statement at design-time, use the query statement editor.

Example

```sql
UPDATE Orders
  set ShipName = :ShipName
  WHERE OrderID = :Old_OrderID
```

See Also

- SQL
- SQLInsert
- SQLDelete
- SQLRefresh

5.11.1.5.2.31 UniDirectional Property

Used if an application does not need bidirectional access to records in the result set.

Class

TCustomDADataset

Syntax

```property`` UniDirectional: boolean `default` False;

Remarks

Traditionally SQL cursors are unidirectional. They can travel only forward through a dataset. TCustomDADataset, however, permits bidirectional travelling by caching records. If an application does not need bidirectional access to the records in the result set, set
UniDirectional to True. When UniDirectional is True, an application requires less memory and performance is improved. However, UniDirectional datasets cannot be modified. In FetchAll=False mode data is fetched on demand. When UniDirectional is set to True, data is fetched on demand as well, but obtained rows are not cached except for the current row. In case if the Unidirectional property is True, the FetchAll property will be automatically set to False. And if the FetchAll property is True, the Unidirectional property will be automatically set to False. The default value of UniDirectional is False, enabling forward and backward navigation.

**Note:** Pay attention to the specificity of using the FetchAll property=False

**See Also**
- TOraDataSet.FetchAll

Methods of the **TCustomDADataSet** class.

For a complete list of the **TCustomDADataSet** class members, see the **TCustomDADataSet Members** topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>(inherited from</td>
<td><strong>TMemDataSet</strong>)</td>
</tr>
<tr>
<td>ApplyUpdates</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>(inherited from</td>
<td><strong>TMemDataSet</strong>)</td>
</tr>
<tr>
<td>BreakExec</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td>CancelRange</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>(inherited from</td>
<td><strong>TMemDataSet</strong>)</td>
</tr>
<tr>
<td>CancelUpdates</td>
<td>Clears all pending cached updates from cache and</td>
</tr>
<tr>
<td>(inherited from</td>
<td><strong>TMemDataSet</strong>)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong></td>
<td>Restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong></td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong></td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>DeleteWhere</strong></td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong></td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>Used to find out whether TCustomDADataSet has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>Used to learn whether TCustomDADataSet is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>Used to learn whether TCustomDADataSet is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>Moves the cursor to a specific record or to the first record in the dataset that</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FindParam</td>
<td>Determines if a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td>GetBlob (inherited from TMemDataSet)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td>GetDataType</td>
<td>Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td>GetFieldObject</td>
<td>Returns a multireference shared object from field.</td>
</tr>
<tr>
<td>GetFieldPrecision</td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td>Locate (inherited from TMemDataSet)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx (inherited from TMemDataSet)</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>ParamByName</td>
<td>Sets or uses parameter information for a specific parameter based on its</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Prepare</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>RefreshRecord</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td>RestoreSQL</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveSQL</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetOrderBy</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
<tr>
<td>SetRange</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SetRangeStart</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SQLSaved</td>
<td>Determines if the SQL property value was saved to the BaseSQL property.</td>
</tr>
<tr>
<td>UnLock</td>
<td>Releases a record lock.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>Frees the resources allocated for a previously prepared query on the</td>
</tr>
</tbody>
</table>
5.11.1.5.3.1 AddWhere Method

Adds condition to the WHERE clause of SELECT statement in the SQL property.

Class

TCustomDADataset

Syntax

procedure AddWhere(const Condition: string);

Parameters

Condition
Hold the condition that will be added to the WHERE clause.

Remarks

Call the AddWhere method to add a condition to the WHERE clause of SELECT statement in the SQL property.

If SELECT has no WHERE clause, AddWhere creates it.

Note: the AddWhere method is implicitly called by RefreshRecord. The AddWhere method works for the SELECT statements only.

Note: the AddWhere method adds a value to the WHERE condition as is. If you expect this
value to be enclosed in brackets, you should bracket it explicitly.

See Also
- DeleteWhere

Remarks
Call the BreakExec method to break execution of a SQL statement on the server. Useful when TOraDataSet.NonBlocking is True.

There are some notions to keep in mind when using this procedure:
- execution of the PL/SQL block cannot be interrupted by BreakExec;
- calling BreakExec to interrupt dataset opening in the TOraDataSet.NonBlocking mode may not have effect if a fetch operation has already begun (this happens when BreakExec falls between two fetch operations).

See Also
- TCustomDADataSet.Execute
- TOraDataSet.NonBlocking

Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.
Class

**TCustomDADataSet**

Syntax

```plaintext
function CreateBlobStream(Field: TField; Mode: TBlobStreamMode): TStream; override;
```

**Parameters**

*Field*

Holds the BLOB field for reading data from or writing data to from a stream.

*Mode*

Holds the stream mode, for which the stream will be used.

**Return Value**

The BLOB Stream.

**Remarks**

Call the CreateBlobStream method to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter. It must be a TBlobField component. You can specify whether the stream will be used for reading, writing, or updating the contents of the field with the Mode parameter.

© 1997-2021

© 2021 Devart. All Rights Reserved.

5.11.1.5.3.4 DeleteWhere Method

Removes WHERE clause from the SQL property and assigns the BaseSQL property.

Class

**TCustomDADataSet**

Syntax

```plaintext
procedure DeleteWhere;
```

**Remarks**

Call the DeleteWhere method to remove WHERE clause from the the SQL property and assign BaseSQL.
5.11.1.5.3.5  Execute Method

Executes a SQL statement on the server.

Class

TCustomDADataSet

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Executes a SQL statement on the server.</td>
</tr>
<tr>
<td>Execute(Iters: integer; Offset: integer)</td>
<td>Used to perform Batch operations.</td>
</tr>
</tbody>
</table>

Remarks

Call the Execute method to execute an SQL statement on the server. If SQL statement is a SELECT query, Execute calls the Open method.

Execute implicitly prepares SQL statement by calling the TCustomDADataSet.Prepare method if the TCustomDADataSet.Options option is set to True and the statement has not been prepared yet. To speed up the performance in case of multiple Execute calls, an application should call Prepare before calling the Execute method for the first time.
Used to perform Batch operations.

Class

TCustomDADataSet

Syntax

```pascal
procedure Execute(Iter: integer; Offset: integer = 0); overload;
virtual;
```

Parameters

- **Iter**
  - Specifies the number of inserted rows.

- **Offset**
  - Points the array element, which the Batch operation starts from. 0 by default.

Remarks

The Execute method executes the specified batch SQL query. See the Batch operations article for samples.

See Also

- Batch operations
Syntax

```pascal
function Executing: boolean;
```

**Return Value**

True, if SQL statement is still being executed.

**Remarks**

Check Executing to learn whether TCustomDADataSet is still executing SQL statement. Use the Executing method if NonBlocking is True.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.5.3.7 Fetched Method

Used to find out whether TCustomDADataSet has fetched all rows.

**Class**

TCustomDADataSet

Syntax

```pascal
function Fetched: boolean; virtual;
```

**Return Value**

True, if all rows have been fetched.

**Remarks**

Call the Fetched method to find out whether TCustomDADataSet has fetched all rows.

**See Also**

- Fetching

© 1997-2021
Devart. All Rights Reserved.

5.11.1.5.3.8 Fetching Method

Used to learn whether TCustomDADataSet is still fetching rows.

**Class**
**TCustomDADataSet**

**Syntax**

```pascal
function Fetching: boolean;
```

**Return Value**

True, if TCustomDADataSet is still fetching rows.

**Remarks**

Check Fetching to learn whether TCustomDADataSet is still fetching rows. Use the Fetching method if NonBlocking is True.

**See Also**

- [Executing](#)

---

5.11.1.5.3.9 FetchingAll Method

**Syntax**

```pascal
function FetchingAll: boolean;
```

**Return Value**

True, if TCustomDADataSet is fetching all rows to the end.

**Remarks**

Check FetchingAll to learn whether TCustomDADataSet is fetching all rows to the end.

**See Also**

- [Executing](#)
5.11.1.5.3.10  FindKey Method

Searches for a record which contains specified field values.

Class
TCustomDADataSet

Syntax

```
function FindKey(const KeyValues: array of System.TVarRec): Boolean;
```

Parameters

- **KeyValues**
  
  Holds a key.

Remarks

Call the FindKey method to search for a specific record in a dataset. KeyValues holds a comma-delimited array of field values, that is called a key.

This function is provided for BDE compatibility only. It is recommended to use functions `TMemDataSet.Locate` and `TMemDataSet.LocateEx` for the record search.

5.11.1.5.3.11  FindMacro Method

Finds a macro with the specified name.

Class
TCustomDADataSet

Syntax

```
function FindMacro(const Value: string): TMacro;
```

Parameters

- **Value**
  
  Holds the name of a macro to search for.

Return Value

- TMacro object if a match is found, nil otherwise.
Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the `TMacros.Items` property to avoid depending on the order of the items.

See Also

- `TMacro`
- `Macros`
- `MacroByName`
5.11.1.5.3.13  FindParam Method

Determines if a parameter with the specified name exists in a dataset.

Class

TCustomDADataSet

Syntax

function FindParam(const Value: string): TDAParam;

Parameters

Value

Holds the name of the param for which to search.

Return Value

the TDAParam object for the specified Name. Otherwise it returns nil.

Remarks

Call the FindParam method to determine if a specified param component exists in a dataset. Name is the name of the param for which to search. If FindParam finds a param with a matching name, it returns a TDAParam object for the specified Name. Otherwise it returns nil.

See Also

• Params
• ParamByName

5.11.1.5.3.14  GetDataType Method

Returns internal field types defined in the MemData and accompanying modules.

Class

TCustomDADataSet
Syntax

```pascal
function GetDataType(const FieldName: string): integer; virtual;
```

**Parameters**

`FieldName`
- Holds the name of the field.

**Return Value**
- Internal field types defined in MemData and accompanying modules.

**Remarks**

Call the GetDataType method to return internal field types defined in the MemData and accompanying modules. Internal field data types extend the TFieldType type of VCL by specific database server data types. For example, ftString, ftFile, ftObject.

5.11.1.5.3.15 GetFieldObject Method

Returns a multireference shared object from field.

**Class**

`TCustomDADataset`

**Syntax**

```pascal
function GetFieldObject(Field: TField): TSharedObject;
overload;
function GetFieldObject(Field: TField; RecBuf: TRecordBuffer): TSharedObject;
overload;
function GetFieldObject(FieldDesc: TFieldDesc): TSharedObject;
overload;
function GetFieldObject(FieldDesc: TFieldDesc; RecBuf: TRecordBuffer): TSharedObject;
overload;
function GetFieldObject(const FieldName: string): TSharedObject; overload;
```

**Parameters**

`FieldName`
- Holds the field name.

**Return Value**
- Multireference shared object.

**Remarks**
Call the GetFieldObject method to return a multireference shared object from field. If field does not hold one of the TSharedObject descendants, GetFieldObject raises an exception.

5.11.1.5.3.16 GetFieldPrecision Method

Retrieves the precision of a number field.

Class

TCustomDADataSet

Syntax

function GetFieldPrecision(const FieldName: string): integer;

Parameters

FieldName

Holds the existing field name.

Return Value

precision of number field.

Remarks

Call the GetFieldPrecision method to retrieve the precision of a number field. FieldName is the name of an existing field.

See Also

- GetFieldScale

5.11.1.5.3.17 GetFieldScale Method

Retrieves the scale of a number field.

Class

TCustomDADataSet

Syntax
function GetFieldScale(const FieldName: string): integer;

Parameters

FieldName
  Holds the existing field name.

Return Value
  the scale of the number field.

Remarks

Call the GetFieldScale method to retrieve the scale of a number field. FieldName is the name of an existing field.

See Also
  • GetFieldPrecision

5.11.1.5.3.18 GetKeyFieldNames Method

Provides a list of available key field names.

Class

TCustomDADataset

Syntax

procedure GetKeyFieldNames(List: TStrings);

Parameters

List
  The list of available key field names

Return Value
  Key field name

Remarks

Call the GetKeyFieldNames method to get the names of available key fields. Populates a string list with the names of key fields in tables.

See Also
  • TCustomDACConnection.GetTableNames
5.11.1.5.3.19 GetOrderBy Method

Retrieves an ORDER BY clause from a SQL statement.

Class

TCustomDADataSet

Syntax

```delphi
function GetOrderBy: string;
```

Return Value

an ORDER BY clause from the SQL statement.

Remarks

Call the GetOrderBy method to retrieve an ORDER BY clause from a SQL statement.

Note: GetOrderBy and SetOrderBy methods serve to process only quite simple queries and don't support, for example, subqueries.

See Also

• SetOrderBy

5.11.1.5.3.20 GotoCurrent Method

Sets the current record in this dataset similar to the current record in another dataset.

Class

TCustomDADataSet

Syntax

```delphi
procedure GotoCurrent(DataSet: TCustomDADataSet);
```

Parameters
**Dataset**

Holds the TCustomDADataset descendant to synchronize the record position with.

**Remarks**

Call the GotoCurrent method to set the current record in this dataset similar to the current record in another dataset. The key fields in both these DataSets must be coincident.

**See Also**

- TMemDataSet.Locate
- TMemDataSet.LocateEx

5.11.1.5.3.21  Lock Method

Locks the current record.

**Class**

TCustomDADataset

**Syntax**

```plaintext
procedure Lock; virtual;
```

**Remarks**

Call the Lock method to lock the current record by executing the statement that is defined in the SQLLock property (for the TOraQuery component).

The Lock method sets the savepoint with the name LOCK_ + <component_name>.

TOraQuery uses SQLLock to execute the current record locking. TSmartQuery builds a SELECT FOR UPDATE statement itself.

**See Also**

- TOraDataSet.LockMode
- SQLLock
- UnLock
5.11.1.5.3.22  MacroByName Method

Finds a macro with the specified name.

Class

TCustomDADataset

Syntax

function MacroByName(const Value: string): TMacro;

Parameters

Value

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found.

Remarks

Call the MacroByName method to find a macro with the specified name. If a match is found, MacroByName returns the macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the TMacros.Items property to avoid depending on the order of the items.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To set a value to a macro, use the TMacro.Value property.

Example

OraQuery.SQL := 'SELECT * FROM Scott.Dept ORDER BY &Order';
OraQuery.MacroByName('Order').Value := 'DeptNo';
OraQuery.Open;

See Also

- TMacro
- Macros
- FindMacro
5.11.1.5.3.23  ParamByName Method

Sets or uses parameter information for a specific parameter based on its name.

Class

TCustomDADataSet

Syntax

function ParamByName(const Value: string): TDAParam;

Parameters

Value

Holds the name of the parameter for which to retrieve information.

Return Value

a TDAParam object.

Remarks

Call the ParamByName method to set or use parameter information for a specific parameter based on its name. Name is the name of the parameter for which to retrieve information. ParamByName is used to set a parameter's value at runtime and returns a TDAParam object.

Example

The following statement retrieves the current value of a parameter called "Contact" into an edit box:

```
Edit1.Text := Query1.ParamsByName('Contact').AsString;
```

See Also

- Params
- FindParam

© 1997-2021
Devart. All Rights Reserved.

Provide Feedback
DAC Forum
Request Support
**TCustomDADataSet**

**Syntax**

```plaintext
procedure Prepare; override;
```

**Remarks**

Call the Prepare method to allocate, open, and parse cursor for a query. Calling Prepare before executing a query improves application performance.

TCustomDADataSet automatically prepares a query if it is executed without being prepared first. After execution, TCustomDADataSet unprepares the query. When a query is executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

The UnPrepare method unprepares a query.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

**See Also**

- `TMemDataSet.Prepared`
- `TMemDataSet.UnPrepare`
- `Options`

**RefreshRecord Method**

Actualizes field values for the current record.

**Class**

`TCustomDADataSet`

**Syntax**

```plaintext
procedure RefreshRecord;
```

**Remarks**

Call the RefreshRecord method to actualize field values for the current record.
RefreshRecord performs query to database and refetches new field values from the returned cursor.

See Also
- RefreshOptions
- SQLRefresh

Class
TCustomDADataSet

Syntax

```pascal
procedure RestoreSQL;
```

Remarks

Call the RestoreSQL method to restore the SQL property modified by AddWhere and SetOrderBy.

See Also
- AddWhere
- SetOrderBy
- SaveSQL
- SQLSaved

Class
TCustomDADataSet

Syntax

```pascal
procedure SaveSQL;
```

Remarks

Saves the SQL property value to BaseSQL.
Syntax

```procedure`` SaveSQL;
```

Remarks
Call the SaveSQL method to save the SQL property value to the BaseSQL property.

See Also
- SQLSaved
- RestoreSQL
- BaseSQL

5.11.1.5.3.28 SetOrderBy Method

Builds an ORDER BY clause of a SELECT statement.

Class
```
TCustomDADataSet
```

Syntax

```procedure`` SetOrderBy(`const` Fields: `string`);
```

Parameters
```
Fields
```
Holds the names of the fields which will be added to the ORDER BY clause.

Remarks
Call the SetOrderBy method to build an ORDER BY clause of a SELECT statement. The fields are identified by the comma-delimited field names.

Note: The GetOrderBy and SetOrderBy methods serve to process only quite simple queries and don't support, for example, subqueries.

Example

```Query1.SetOrderBy('DeptNo;DName');```

See Also
5.11.1.5.3.29 SQLSaved Method

Determines if the SQL property value was saved to the BaseSQL property.

Class

TCustomDADataSet

Syntax

function SQLSaved: boolean;

Return Value

True, if the SQL property value was saved to the BaseSQL property.

Remarks

Call the SQLSaved method to know whether the SQL property value was saved to the BaseSQL property.

5.11.1.5.3.30 UnLock Method

Releases a record lock.

Class

TCustomDADataSet

Syntax

procedure UnLock;

Remarks

Call the Unlock method to release the record lock made by the Lock method before.

Unlock is performed by rolling back to the savepoint set by the Lock method.

See Also
Events of the **TCustomDADataset** class.

For a complete list of the **TCustomDADataset** class members, see the [TCustomDADataset Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AfterExecute</strong></td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td><strong>AfterFetch</strong></td>
<td>Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td><strong>AfterUpdateExecute</strong></td>
<td>Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
<tr>
<td><strong>BeforeFetch</strong></td>
<td>Occurs before dataset is going to fetch block of records from the server.</td>
</tr>
<tr>
<td><strong>BeforeUpdateExecute</strong></td>
<td>Occurs before executing insert, delete, update, lock, and refresh operations.</td>
</tr>
<tr>
<td><strong>OnUpdateError</strong></td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td><strong>OnUpdateRecord</strong></td>
<td>Occurs when a single update component can not handle the updates.</td>
</tr>
</tbody>
</table>

See Also
- [TCustomDADataset Class](#)
- [TCustomDADataset Class Members](#)
5.11.1.5.4.1 AfterExecute Event

Occurs after a component has executed a query to database.

Class

TCustomDADataSet

Syntax

property AfterExecute: TAfterExecuteEvent;

Remarks

Occurs after a component has executed a query to database.

See Also

- TCustomDADataSet.Execute

© 1997-2021
Devart. All Rights Reserved.
Request Support DAC Forum Provide Feedback

5.11.1.5.4.2 AfterFetch Event

Occurs after dataset finishes fetching data from server.

Class

TCustomDADataSet

Syntax

property AfterFetch: TAfterFetchEvent;

Remarks

The AfterFetch event occurs after dataset finishes fetching data from server.

See Also

- BeforeFetch
- TOraDataSet.NonBlocking

© 1997-2021
Devart. All Rights Reserved.
Request Support DAC Forum Provide Feedback

© 2021 Devart
5.11.1.5.4.3 AfterUpdateExecute Event

Occurs after executing insert, delete, update, lock, and refresh operations.

Class

TCustomDADataSet

Syntax

property AfterUpdateExecute: TUpdateExecuteEvent;

Remarks

Occurs after executing insert, delete, update, lock, and refresh operations. You can use AfterUpdateExecute to set the parameters of corresponding statements.

5.11.1.5.4.4 BeforeFetch Event

Occurs before dataset is going to fetch block of records from the server.

Class

TCustomDADataSet

Syntax

property BeforeFetch: TBeforeFetchEvent;

Remarks

The BeforeFetch event occurs every time before dataset is going to fetch a block of records from the server. Set Cancel to True to abort current fetch operation.

Note: In the TOraDataSet.NonBlocking mode event handler is called from the fetching thread. Therefore, if you have set the NonBlocking property to True, you should use thread synchronization mechanisms in the code of the BeforeFetch event handler.

See Also

- AfterFetch
- TOraDataSet.NonBlocking
5.11.1.5.4.5 BeforeUpdateExecute Event

Occurs before executing insert, delete, update, lock, and refresh operations.

Class

TCustomDADataSet

Syntax

```property
BeforeUpdateExecute: TUpdateExecuteEvent;
```

Remarks

Occurs before executing insert, delete, update, lock, and refresh operations. You can use BeforeUpdateExecute to set the parameters of corresponding statements.

See Also

- AfterUpdateExecute

© 1997-2021

Devart. All Rights Reserved.

5.11.1.6 TCustomDASQL Class

A base class for components executing SQL statements that do not return result sets.

For a list of all members of this type, see TCustomDASQL members.

Unit

DBAccess

Syntax

```class
TCustomDASQL = TComponent;
```

Remarks

TCustomDASQL is a base class that defines functionality for descendant classes which access database using SQL statements. Applications never use TCustomDASQL objects directly. Instead they use descendants of TCustomDASQL.
Use TCustomDASQL when client application must execute SQL statement or call stored procedure on the database server. The SQL statement should not retrieve rows from the database.

**TCustomDASQL** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChangeCursor</td>
<td>Enables or disables changing screen cursor when executing commands in the NonBlocking mode.</td>
</tr>
<tr>
<td>Connection</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td>FinalSQL</td>
<td>Used to return a SQL statement with expanded macros.</td>
</tr>
<tr>
<td>MacroCount</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.</td>
</tr>
<tr>
<td>ParamCount</td>
<td>Indicates the number of parameters in the Params property.</td>
</tr>
<tr>
<td>Params</td>
<td>Used to contain parameters for a SQL statement.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ParamValues</strong></td>
<td>Used to get or set the values of individual field parameters that are identified by name.</td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td>Used to indicate whether a query is prepared for execution.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong></td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.</td>
</tr>
</tbody>
</table>

**Name**

- **BreakExec**: Breaks execution of an SQL statement on the server.
- **Execute**: Overloaded. Executes a SQL statement on the server.
- **Executing**: Checks whether TCustomDASQL still executes a SQL statement.
- **FindMacro**: Finds a macro with the specified name.
- **FindParam**: Finds a parameter with the specified name.
- **MacroByName**: Finds a macro with the specified name.
- **ParamByName**: Finds a parameter with the specified name.
- **Prepare**: Allocates, opens, and parses cursor for a query.
- **UnPrepare**: Frees the resources allocated for a previously prepared query on the server and client sides.
WaitExecuting: Waits until TCustomDASQL executes a SQL statement.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a SQL statement has been executed.</td>
</tr>
</tbody>
</table>

Properties of the TCustomDASQL class.

For a complete list of the TCustomDASQL class members, see the TCustomDASQL Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChangeCursor</td>
<td>Enables or disables changing screen cursor when executing commands in the NonBlocking mode.</td>
</tr>
<tr>
<td>Connection</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td>FinalSQL</td>
<td>Used to return a SQL statement with expanded macros.</td>
</tr>
<tr>
<td>MacroCount</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>Used to specify whether</td>
</tr>
</tbody>
</table>
Parameters for the `Params` property are implicitly generated when the SQL property is being changed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ParamCount</strong></td>
<td>Indicates the number of parameters in the <code>Params</code> property.</td>
</tr>
<tr>
<td><strong>Params</strong></td>
<td>Used to contain parameters for a SQL statement.</td>
</tr>
<tr>
<td><strong>ParamValues</strong></td>
<td>Used to get or set the values of individual field parameters that are identified by name.</td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td>Used to indicate whether a query is prepared for execution.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong></td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.</td>
</tr>
</tbody>
</table>

See Also
- TCustomDASQL Class
- TCustomDASQL Class Members

5.11.1.6.2.1 ChangeCursor Property

Enables or disables changing screen cursor when executing commands in the NonBlocking mode.

Class

TCustomDASQL

Syntax
**property** ChangeCursor: boolean;

Remarks
Set the ChangeCursor property to False to prevent the screen cursor from changing to crSQLArrow when executing commands in the NonBlocking mode. The default value is True.

5.11.1.6.2.2  Connection Property

Used to specify a connection object to use to connect to a data store.

Class
TCustomDASQL

Syntax

**property** Connection: TCustomDAConnection;

Remarks
Use the Connection property to specify a connection object that will be used to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, link an instance of a TCustomDAConnection descendant to the Connection property.

5.11.1.6.2.3  Debug Property

Used to display the statement that is being executed and the values and types of its parameters.

Class
TCustomDASQL
Syntax

```
property Debug: boolean default False;
```

Remarks

Set the Debug property to True to display the statement that is being executed and the values and types of its parameters.

You should add the OdacVcl unit to the uses clause of any unit in your project to make the Debug property work.

**Note:** If TOraSQLMonitor is used in the project and the TOraSQLMonitor.Active property is set to False, the debug window is not displayed.

See Also

- TCustomDADataSet.Debug

© 1997-2021
Devart. All Rights Reserved.

5.11.1.6.2.4 FinalSQL Property

Used to return a SQL statement with expanded macros.

Class

TCustomDASQL

Syntax

```
property FinalSQL: string;
```

Remarks

Read the FinalSQL property to return a SQL statement with expanded macros. This is the exact statement that will be passed on to the database server.

© 1997-2021
Devart. All Rights Reserved.
5.11.1.6.2.5 MacroCount Property

Used to get the number of macros associated with the Macros property.

Class
TCustomDASQL

Syntax

```property
MacroCount: word;
```

Remarks
Use the MacroCount property to get the number of macros associated with the Macros property.

See Also
- Macros

5.11.1.6.2.6 Macros Property

Makes it possible to change SQL queries easily.

Class
TCustomDASQL

Syntax

```property
Macros: TMacros stored False;
```

Remarks
With the help of macros you can easily change SQL query text at design- or runtime. Marcos extend abilities of parameters and allow to change conditions in a WHERE clause or sort order in an ORDER BY clause. You just insert &MacroName in the SQL query text and change value of macro in the Macro property editor at design time or call the MacroByName function at run time. At the time of opening the query macro is replaced by its value.

See Also
- TMacro
5.11.1.6.2.7 ParamCheck Property

Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

Class

TCustomDASQL

Syntax

property ParamCheck: boolean default True;

Remarks

Use the ParamCheck property to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.

Set ParamCheck to True to let TCustomDASQL generate the Params property for the dataset based on a SQL statement automatically.

Setting ParamCheck to False can be used if the dataset component passes to a server the DDL statements that contain, for example, declarations of the stored procedures that will accept parameterized values themselves. The default value is True.

See Also

- Params

5.11.1.6.2.8 ParamCount Property

Indicates the number of parameters in the Params property.

Class

TCustomDASQL
Syntax

```property``

```
property ParamCount: word;
```  

Remarks

Use the ParamCount property to determine how many parameters are there in the Params property.

Used to contain parameters for a SQL statement.

Class

```
TCustomDASQL
```  

Syntax

```
property Params: TDAParams stored False;
```  

Remarks

Access the Params property at runtime to view and set parameter names, values, and data types dynamically (at design-time use the Parameters editor to set parameter properties).

Params is a zero-based array of parameter records. Index specifies the array element to access. An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

Example

Setting parameters at runtime:

```procedure TForm1.Button1Click(Sender: TObject);
begin
  with OraSQL do
  begin
    SQL.Clear;
    SQL.Add('INSERT INTO Temp_Table(Id, Name)');
    SQL.Add('VALUES (:id, :Name)');
    ParamByName('Id').AsInteger := 55;
    Params[1].AsString := ' Green';
    Execute;
  end;
end;```
5.11.1.6.2.10  ParamValues Property(Indexer)

Used to get or set the values of individual field parameters that are identified by name.

Class

TCustomDASQL

Syntax

property  ParamValues[const  ParamName:  string]:  Variant;  default;

Parameters

ParamName

Holds parameter names separated by semicolon.

Remarks

Use the ParamValues property to get or set the values of individual field parameters that are identified by name.

Setting ParamValues sets the Value property for each parameter listed in the ParamName string. Specify the values as Variants.

Getting ParamValues retrieves an array of variants, each of which represents the value of one of the named parameters.

Note: The Params array is generated implicitly if ParamCheck property is set to True. If ParamName includes a name that does not match any of the parameters in Items, an exception is raised.
5.11.1.6.2.11 Prepared Property

Used to indicate whether a query is prepared for execution.

Class

TCustomDASQL

Syntax

property Prepared: boolean;

Remarks

Check the Prepared property to determine if a query is already prepared for execution. True means that the query has already been prepared. As a rule prepared queries are executed faster, but the preparation itself also takes some time. One of the proper cases for using preparation is parametrized queries that are executed several times.

See Also

• Prepare

5.11.1.6.2.12 RowsAffected Property

Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.

Class

TCustomDASQL

Syntax

property RowsAffected: integer;

Remarks

Check RowsAffected to determine how many rows were inserted, updated, or deleted during the last query operation. If RowsAffected is -1, the query has not inserted, updated, or deleted any rows.
5.11.1.6.2.13 SQL Property

Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.

Class

TCustomDASQL

Syntax

property SQL: TStrings;

Remarks

Use the SQL property to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called. At design time the SQL property can be edited by invoking the String List editor in Object Inspector.

See Also

- FinalSQL
- TCustomDASQL.Execute

Methods

Methods of the TCustomDASQL class.

For a complete list of the TCustomDASQL class members, see the TCustomDASQL Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BreakExec</td>
<td>Breaks execution of an SQL statement on the server.</td>
</tr>
<tr>
<td>Execute</td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td>Executing</td>
<td>Checks whether</td>
</tr>
</tbody>
</table>
TCustomDASQL still executes a SQL statement.

**FindMacro**
Finds a macro with the specified name.

**FindParam**
Finds a parameter with the specified name.

**MacroByName**
Finds a macro with the specified name.

**ParamByName**
Finds a parameter with the specified name.

**Prepare**
Allocates, opens, and parses cursor for a query.

**UnPrepare**
Frees the resources allocated for a previously prepared query on the server and client sides.

**WaitExecuting**
Waits until TCustomDASQL executes a SQL statement.

See Also
- TCustomDASQL Class
- TCustomDASQL Class Members

5.11.1.6.3.1 BreakExec Method

Breaks execution of an SQL statement on the server.

**Class**
TCustomDASQL

**Syntax**

```pascal
procedure BreakExec;
```

**Remarks**
Call the BreakExec method to break execution of an SQL statement on the server. It makes sense to call BreakExec only from another thread. Useful when NonBlocking is True.
5.11.6.3.2 Execute Method

Executes a SQL statement on the server.

Class

TCustomDASQL

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Executes a SQL statement on the server.</td>
</tr>
<tr>
<td>Execute(Iters: integer; Offset: integer)</td>
<td>Used to perform Batch operations.</td>
</tr>
</tbody>
</table>

Remarks

Call the Execute method to execute a SQL statement on the server. If the SQL statement has OUT parameters, use the TCustomDASQL.ParamsByName method or the TCustomDASQL.Params property to get their values. Iters argument specifies the number of times this statement is executed for the DML array operations.
Reserved.

Used to perform [Batch operations](#).

Class

`TCustomDASQL`

Syntax

```pascal
procedure Execute(Iter: integer; Offset: integer = 0); overload;
virtual;
```

Parameters

- **Iter**
  - Specifies the number of inserted rows.
- **Offset**
  - Points the array element, which the Batch operation starts from. 0 by default.

Remarks

The `Execute` method executes the specified batch SQL query. See the [Batch operations](#) article for samples.

See Also

- [Batch operations](#)

© 1997-2021
Devart. All Rights Reserved.

5.11.1.6.3.3 Executing Method

Checks whether `TCustomDASQL` still executes a SQL statement.

Class

`TCustomDASQL`

Syntax

```pascal
function Executing: boolean;
```

Return Value

- True, if a SQL statement is still being executed by `TCustomDASQL`. 
Remarks

Check Executing to find out whether TCustomDASQL still executes a SQL statement. The Executing method is used for nonblocking execution.

Class

TCustomDASQL

Syntax

function FindMacro(const Value: string): TMacro;

Parameters

Value

Holds the name of a macro to search for.

Return Value

TMacro object if a match is found, nil otherwise.

Remarks

Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the TMacros.Items property to avoid depending on the order of the items.

See Also

- TMacro
- Macros
- MacroByName

© 1997-2021
Devart. All Rights Reserved.

Request Support  DAC Forum  Provide Feedback
5.11.1.6.3.5 FindParam Method

Finds a parameter with the specified name.

Class

TCustomDASQL

Syntax

```
function FindParam(const Value: string): TDAParam;
```

Parameters

Value

Holds the parameter name to search for.

Return Value

A TDAParam object, if a parameter with the specified name has been found. If it has not, returns nil.

Remarks

Call the FindParam method to find a parameter with the specified name in a dataset.

See Also

- ParamByName

© 1997-2021

Devart. All Rights Reserved.

Request Support    DAC Forum    Provide Feedback

5.11.1.6.3.6 MacroByName Method

Finds a macro with the specified name.

Class

TCustomDASQL

Syntax

```
function MacroByName(const Value: string): TMacro;
```

Parameters

Value

Holds the name of a macro to search for.

Return Value
TMacro object if a match is found.

Remarks

Call the MacroByName method to find a macro with the specified name. If a match is found, MacroByName returns the macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the TMacros.Items property to avoid depending on the order of the items.

To locate a parameter by name without raising an exception if the parameter is not found, use the FindMacro method.

To set a value to a macro, use the TMacro.Value property.

See Also

- TMacro
- Macros
- FindMacro

© 1997-2021 Devart. All Rights Reserved.

5.11.1.6.3.7 ParamByName Method

Finds a parameter with the specified name.

Class

TCustomDASQL

Syntax

function ParamByName(const Value: string): TDAParam;

Parameters

Value
  Holds the name of the parameter to search for.

Return Value
  a TDAParam object, if a match was found. Otherwise, an exception is raised.

Remarks

Use the ParamByName method to find a parameter with the specified name. If no parameter with the specified name found, an exception is raised.
Example

```pascal
OraSQL.Execute;
Edit1.Text := OraSQL.ParamsByName('Contact').AsString;
```

See Also
- [FindParam](#)

Remarks

Call the Prepare method to allocate, open, and parse cursor for a query. Calling Prepare before executing a query improves application performance.

TCustomDADataSet automatically prepares a query if it is executed without being prepared first. After execution, TCustomDADataSet unprepares the query. When a query is executed a number of times, an application should always explicitly prepare the query to avoid multiple and unnecessary prepares and unprepares.

The UnPrepare method unprepares a query.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

See Also
- [Prepared](#)
- [UnPrepare](#)
5.11.1.6.3.9  UnPrepare Method

Frees the resources allocated for a previously prepared query on the server and client sides.

Class

TCustomDASQL

Syntax

procedure UnPrepare; virtual;

Remarks

Call the UnPrepare method to free resources allocated for a previously prepared query on the server and client sides.

See Also

• Prepare

© 1997-2021 Devart. All Rights Reserved.

5.11.1.6.3.10  WaitExecuting Method

Waits until TCustomDASQL executes a SQL statement.

Class

TCustomDASQL

Syntax

function WaitExecuting(TimeOut: integer = 0): boolean;

Parameters

TimeOut

Holds the time in seconds to wait while TCustomDASQL executes the SQL statement. Zero means infinite time.

Return Value

True, if the execution of a SQL statement was completed in the preset time.

Remarks

Call the WaitExecuting method to wait until TCustomDASQL executes a SQL statement. Use
the WaitExecuting method for nonblocking execution.

See Also
- Executing

5.11.1.6.4 Events

Events of the TCustomDASQL class.

For a complete list of the TCustomDASQL class members, see the TCustomDASQL Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a SQL statement has been executed.</td>
</tr>
</tbody>
</table>

See Also
- TCustomDASQL Class
- TCustomDASQL Class Members

5.11.1.6.4.1 AfterExecute Event

Occurs after a SQL statement has been executed.

Class

TCustomDASQL

Syntax

property AfterExecute: TAfterExecuteEvent;

Remarks

Occurs after a SQL statement has been executed. This event may be used for descendant
components which use multithreaded environment.

See Also
• TCustomDASQL.Execute

5.11.1.7 TCustomDAUpdateSQL Class

A base class for components that provide DML statements for more flexible control over data modifications.

For a list of all members of this type, see TCustomDAUpdateSQL members.

Unit
DBAccess

Syntax

TCustomDAUpdateSQL = class(TComponent);

Remarks

TCustomDAUpdateSQL is a base class for components that provide DML statements for more flexible control over data modifications. Besides providing BDE compatibility, this component allows to associate a separate component for each update command.

See Also
• TOraDataSet.UpdateObject

5.11.1.7.1 Members

TCustomDAUpdateSQL class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSet</td>
<td>Used to hold a reference to the TCustomDADataset</td>
</tr>
</tbody>
</table>
object that is being updated.

**DeleteObject**
Provides ability to perform advanced adjustment of the delete operations.

**DeleteSQL**
Used when deleting a record.

**InsertObject**
Provides ability to perform advanced adjustment of insert operations.

**InsertSQL**
Used when inserting a record.

**LockObject**
Provides ability to perform advanced adjustment of lock operations.

**LockSQL**
Used to lock the current record.

**ModifyObject**
Provides ability to perform advanced adjustment of modify operations.

**ModifySQL**
Used when updating a record.

**RefreshObject**
Provides ability to perform advanced adjustment of refresh operations.

**RefreshSQL**
Used to specify an SQL statement that will be used for refreshing the current record by TCustomDADataSet.RefreshRecord procedure.

**SQL**
Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Apply</strong></td>
<td>Sets parameters for a SQL statement and executes it to update a record.</td>
</tr>
<tr>
<td><strong>ExecSQL</strong></td>
<td>Executes a SQL statement.</td>
</tr>
</tbody>
</table>
Properties of the `TCustomDAUpdateSQL` class.

For a complete list of the `TCustomDAUpdateSQL` class members, see the `TCustomDAUpdateSQL Members` topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DataSet</code></td>
<td>Used to hold a reference to the <code>TCustomDADataset</code> object that is being updated.</td>
</tr>
<tr>
<td><code>SQL</code></td>
<td>Used to return a SQL statement for one of the <code>ModifySQL</code>, <code>InsertSQL</code>, or <code>DeleteSQL</code> properties.</td>
</tr>
</tbody>
</table>

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DeleteObject</code></td>
<td>Provides ability to perform advanced adjustment of the delete operations.</td>
</tr>
<tr>
<td><code>DeleteSQL</code></td>
<td>Used when deleting a record.</td>
</tr>
<tr>
<td><code>InsertObject</code></td>
<td>Provides ability to perform advanced adjustment of insert operations.</td>
</tr>
<tr>
<td><code>InsertSQL</code></td>
<td>Used when inserting a record.</td>
</tr>
<tr>
<td><code>LockObject</code></td>
<td>Provides ability to perform advanced adjustment of lock operations.</td>
</tr>
<tr>
<td><code>LockSQL</code></td>
<td>Used to lock the current record.</td>
</tr>
<tr>
<td><code>ModifyObject</code></td>
<td>Provides ability to perform advanced adjustment of modify operations.</td>
</tr>
<tr>
<td><code>ModifySQL</code></td>
<td>Used when updating a record.</td>
</tr>
</tbody>
</table>
### RefreshObject

**Description**: Provides ability to perform advanced adjustment of refresh operations.

### RefreshSQL

**Description**: Used to specify an SQL statement that will be used for refreshing the current record by `TCustomDADataset.RefreshRecord` procedure.

## See Also
- TCustomDAUpdateSQL Class
- TCustomDAUpdateSQL Class Members

---

5.11.1.7.2.1 **DataSet Property**

Used to hold a reference to the TCustomDADataset object that is being updated.

### Class

**TCustomDAUpdateSQL**

### Syntax

```delphi
property DataSet: TCustomDADataset;
```

### Remarks

The DataSet property holds a reference to the TCustomDADataset object that is being updated. Generally it is not used directly.

---

5.11.1.7.2.2 **DeleteObject Property**

Provides ability to perform advanced adjustment of the delete operations.

### Class

**TCustomDAUpdateSQL**
**DeleteObject Property**

```
property DeleteObject: TComponent;
```

**Remarks**

Assign SQL component or a TOraDataSet descendant to this property to perform advanced adjustment of the delete operations. In some cases this can give some additional performance. Use the same principle to set the SQL property of an object as for setting the `DeleteSQL` property.

**See Also**

- `DeleteSQL`

---

**InsertObject Property**

**Remarks**

Set the DeleteSQL property to a DELETE statement to use when deleting a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

**Class**

`TCustomDAUpdateSQL`
**TCustomDAUpdateSQL**

### Syntax

```property
InsertObject: TComponent;
```

### Remarks

Assign SQL component or TOraDataSet descendant to this property to perform advanced adjustment of insert operations. In some cases this can give some additional performance. Set the SQL property of the object in the same way as used for the `InsertSQL` property.

**See Also**
- `InsertSQL`

---

**InsertSQL Property**

Used when inserting a record.

### Class

**TCustomDAUpdateSQL**

### Syntax

```property
InsertSQL: TStrings;
```

### Remarks

Set the `InsertSQL` property to an INSERT INTO statement to use when inserting a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

---

**LockObject Property**

Provides ability to perform advanced adjustment of lock operations.

### Class
**TCustomDAUpdateSQL**

**Syntax**

```
property LockObject: TComponent;
```

**Remarks**

Assign a SQL component or TOraDataSet descendant to this property to perform advanced adjustment of lock operations. In some cases that can give some additional performance. Set the SQL property of an object in the same way as used for the LockSQL property.

**See Also**

- LockSQL

© 1997-2021 Devart. All Rights Reserved.

---

**5.11.1.7.2.7 LockSQL Property**

Used to lock the current record.

**Class**

**TCustomDAUpdateSQL**

**Syntax**

```
property LockSQL: TStrings;
```

**Remarks**

Use the LockSQL property to lock the current record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2021 Devart. All Rights Reserved.

---

**5.11.1.7.2.8 ModifyObject Property**

Provides ability to perform advanced adjustment of modify operations.

**Class**

**TCustomDAUpdateSQL**

© 2021 Devart
Syntax

```property``` ModifyObject: TComponent;

Remarks

Assign a SQL component or TOracleDataSet descendant to this property to perform advanced adjustment of modify operations. In some cases this can give some additional performance. Set the SQL property of the object in the same way as used for the `ModifySQL` property.

See Also

- `ModifySQL`

© 1997-2021 Devart. All Rights Reserved.

5.11.1.7.2.9 ModifyObject Property

Used when updating a record.

Class

`TCustomDAUpdateSQL`

Syntax

```property``` ModifySQL: TStrings;

Remarks

Set ModifySQL to an UPDATE statement to use when updating a record. Statements can be parameterized queries with parameter names corresponding to the dataset field names.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.7.2.10 RefreshObject Property

Provides ability to perform advanced adjustment of refresh operations.

Class

`TCustomDAUpdateSQL`
5.11.1.7.2.11 RefreshSQL Property

Syntax

```pascal
property RefreshObject: TComponent;
```

Remarks

Assign a SQL component or TOraDataSet descendant to this property to perform advanced adjustment of refresh operations. In some cases that can give some additional performance. Set the SQL property of the object in the same way as used for the RefreshSQL property.

See Also

- RefreshSQL

© 1997-2021 Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback

5.11.1.7.2.11 RefreshSQL Property

Used to specify an SQL statement that will be used for refreshing the current record by TCustomDADataSet.RefreshRecord procedure.

Class

TCustomDAOUpdateSQL

Syntax

```pascal
property RefreshSQL: TStrings;
```

Remarks

Use the RefreshSQL property to specify a SQL statement that will be used for refreshing the current record by the TCustomDADataSet.RefreshRecord procedure.

You can assign to SQLRefresh a WHERE clause only. In such a case it is added to SELECT defined by the SQL property by TCustomDADataSet.AddWhere.

To create a RefreshSQL statement at design time, use the query statements editor.

See Also

- TCustomDADataSet.RefreshRecord

© 1997-2021 Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback
5.11.1.7.2.12  SQL Property(Indexer)

Used to return a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties.

Class

`TCustomDAUpdateSQL`

Syntax

```plaintext
property SQL[UpdateKind: TUpdateKind]: TStrings;
```

Parameters

`UpdateKind`

Specifies which of update SQL statements to return.

Remarks

Returns a SQL statement for one of the ModifySQL, InsertSQL, or DeleteSQL properties, depending on the value of the UpdateKind index.

© 1997-2021

Devart. All Rights Reserved.

5.11.1.7.3  Methods

Methods of the `TCustomDAUpdateSQL` class.

For a complete list of the `TCustomDAUpdateSQL` class members, see the `TCustomDAUpdateSQL Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply</td>
<td>Sets parameters for a SQL statement and executes it to update a record.</td>
</tr>
<tr>
<td>ExecSQL</td>
<td>Executes a SQL statement.</td>
</tr>
</tbody>
</table>

See Also

- `TCustomDAUpdateSQL Class`
- `TCustomDAUpdateSQL Class Members`
5.11.1.7.3.1 Apply Method

Sets parameters for a SQL statement and executes it to update a record.

Class

TCustomDAUpdateSQL

Syntax

procedure Apply(UpdateKind: TUpdateKind); virtual;

Parameters

UpdateKind

Specifies which of update SQL statements to execute.

Remarks

Call the Apply method to set parameters for a SQL statement and execute it to update a record. UpdateKind indicates which SQL statement to bind and execute.

Apply is primarily intended for manually executing update statements from an OnUpdateRecord event handler.

Note: If a SQL statement does not contain parameters, it is more efficient to call ExecSQL instead of Apply.

See Also

• ExecSQL

5.11.1.7.3.2 ExecSQL Method

Executes a SQL statement.

Class

TCustomDAUpdateSQL

Syntax
**procedure** ExecSQL(UpdateKind: TUpdateKind);

**Parameters**

*UpdateKind*

Specifies the kind of update statement to be executed.

**Remarks**

Call the ExecSQL method to execute a SQL statement, necessary for updating the records belonging to a read-only result set when cached updates is enabled. UpdateKind specifies the statement to execute.

ExecSQL is primarily intended for manually executing update statements from the OnUpdateRecord event handler.

**Note:** To both bind parameters and execute a statement, call **Apply**.

**See Also**

- **Apply**

© 1997-2021 Devart. All Rights Reserved.

---

**TDACCondition Class**

Represents a condition from the **TDACConditions** list.

For a list of all members of this type, see **TDACCondition** members.

**Unit**

**DBAccess**

**Syntax**

```plaintext
TDACCondition = class(TCollectionItem);
```

**Remarks**

Manipulate conditions using **TDACConditions**.

**See Also**

- **TDACConditions**

© 1997-2021 Devart. All Rights Reserved.
5.11.1.8.1 Members

**TDACCondition** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Indicates whether the condition is enabled or not</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the condition</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the condition</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable</td>
<td>Disables the condition</td>
</tr>
<tr>
<td>Enable</td>
<td>Enables the condition</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.11.1.8.2 Properties

Properties of the **TDACCondition** class.

For a complete list of the **TDACCondition** class members, see the **TDACCondition Members** topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Indicates whether the condition is enabled or not</td>
</tr>
<tr>
<td>Name</td>
<td>The name of the condition</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the condition</td>
</tr>
</tbody>
</table>

See Also

- **TDACCondition Class**
5.11.1.8.2.1 Enabled Property

Indicates whether the condition is enabled or not

Class
TDACondition

Syntax

```plaintext
property Enabled: Boolean default True;
```

5.11.1.8.2.2 Name Property

The name of the condition

Class
TDACondition

Syntax

```plaintext
property Name: string;
```

5.11.1.8.2.3 Value Property

The value of the condition

Class
TDACondition

Syntax

```plaintext
property Value: string;
```
5.11.1.8.3 Methods

Methods of the **TDACCondition** class.

For a complete list of the **TDACCondition** class members, see the [TDACCondition Members](#) topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable</td>
<td>Disables the condition</td>
</tr>
<tr>
<td>Enable</td>
<td>Enables the condition</td>
</tr>
</tbody>
</table>

See Also

- [TDACCondition Class](#)
- [TDACCondition Class Members](#)

5.11.1.8.3.1 Disable Method

Disables the condition

Class

**TDACCondition**

Syntax

```pascal
procedure Disable;
```

5.11.1.8.3.2 Enable Method

Enables the condition

Class
5.11.1.9  TDAConditions Class

Holds a collection of TDACondition objects.

For a list of all members of this type, see TDAConditions members.

Unit DBAccess

Syntax

TDAConditions = class(TCollection);

Remarks

The given example code

UniTable1.Conditions.Add('1','JOB="MANAGER"');
UniTable1.Conditions.Add('2','SAL>2500');
UniTable1.Conditions.Enable;
UniTable1.Open;

will return the following SQL:

SELECT * FROM EMP
WHERE (JOB="MANAGER")
and
(SAL<2500)

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>

© 1997-2021
Devart. All Rights Reserved.
<table>
<thead>
<tr>
<th><strong>Condition</strong></th>
<th>Used to iterate through all the conditions.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>Indicates whether the condition is enabled</td>
</tr>
<tr>
<td><strong>Items</strong></td>
<td>Used to iterate through all conditions.</td>
</tr>
<tr>
<td><strong>Text</strong></td>
<td>The property returns condition names and values as CONDITION_NAME=CONDITION</td>
</tr>
<tr>
<td><strong>WhereSQL</strong></td>
<td>Returns the SQL WHERE condition added in the Conditions property.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Overloaded. Adds a condition to the WHERE clause of the query.</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Deletes the condition</td>
</tr>
<tr>
<td><strong>Disable</strong></td>
<td>Disables the condition</td>
</tr>
<tr>
<td><strong>Enable</strong></td>
<td>Enables the condition</td>
</tr>
<tr>
<td><strong>Find</strong></td>
<td>Search for TDACCondition (the condition) by its name. If found, the TDACCondition object is returned, otherwise - nil.</td>
</tr>
<tr>
<td><strong>Get</strong></td>
<td>Retrieving a TDACCondition object by its name. If found, the TDACCondition object is returned, otherwise - an exception is raised.</td>
</tr>
<tr>
<td><strong>IndexOf</strong></td>
<td>Retrieving condition index by its name. If found, this condition index is returned, otherwise - the method returns -1.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes the condition</td>
</tr>
</tbody>
</table>
Properties of the **TDAConditions** class.

For a complete list of the **TDAConditions** class members, see the [TDAConditions Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition</td>
<td>Used to iterate through all the conditions.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Indicates whether the condition is enabled</td>
</tr>
<tr>
<td>Items</td>
<td>Used to iterate through all conditions.</td>
</tr>
<tr>
<td>Text</td>
<td>The property returns condition names and values as CONDITION_NAME=CONDITION</td>
</tr>
<tr>
<td>WhereSQL</td>
<td>Returns the SQL WHERE condition added in the Conditions property.</td>
</tr>
</tbody>
</table>

See Also

- [TDAConditions Class](#)
- [TDAConditions Class Members](#)

Used to iterate through all the conditions.

**Class**

**TDAConditions**

**Syntax**
5.11.1.9.2.2  Enabled Property

Indicates whether the condition is enabled

Class
TDAConditions

Syntax

property Enabled: Boolean;

Parameters

Index
Holds an index in the range 0..Count - 1.

Remarks
Use the Items property to iterate through all conditions. Index identifies the index in the range 0..Count - 1. Items can reference a particular condition by its index, but the Condition property is preferred in order to avoid depending on the order of the conditions.
5.11.1.9.2.4 Text Property

The property returns condition names and values as CONDITION_NAME=CONDITION

Class
TDAConditions

Syntax

```pascal
property Text: string;
```

5.11.1.9.2.5 WhereSQL Property

Returns the SQL WHERE condition added in the Conditions property.

Class
TDAConditions

Syntax

```pascal
property WhereSQL: string;
```

5.11.1.9.3 Methods

Methods of the TDAConditions class.

For a complete list of the TDAConditions class members, see the TDAConditions Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Overloaded. Adds a condition to the WHERE clause of the query.</td>
</tr>
</tbody>
</table>
Delete | Deletes the condition
Disables the condition
Enable | Enables the condition
Find | Search for TDACondition (the condition) by its name. If found, the TDACondition object is returned, otherwise - nil.
Get | Retrieving a TDACondition object by its name. If found, the TDACondition object is returned, otherwise - an exception is raised.
IndexOf | Retrieving condition index by its name. If found, this condition index is returned, otherwise - the method returns -1.
Remove | Removes the condition

See Also
- TDAConditions Class
- TDAConditions Class Members

5.11.1.9.3.1 Add Method

Adds a condition to the WHERE clause of the query.

Class
TDAConditions

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(const Value: string; Enabled: Boolean)</td>
<td>Adds a condition to the WHERE clause of the query.</td>
</tr>
<tr>
<td>Add(const Name: string; const Value:</td>
<td>Adds a condition to the WHERE clause of the query.</td>
</tr>
</tbody>
</table>

© 2021 Devart
string; Enabled: Boolean)

Adds a condition to the WHERE clause of the query.

Class
TDAConditions

Syntax

function Add(const Value: string; Enabled: Boolean = True): TDACondition; overload;

Parameters

Value
The value of the condition

Enabled
Indicates that the condition is enabled

Remarks

If you want then to access the condition, you should use Add and its name in the Name parameter.

The given example code will return the following SQL:

SELECT * FROM EMP
WHERE (JOB="MANAGER")
and
(SAL<2500)

Adds a condition to the WHERE clause of the query.

Class
TDAConditions

Syntax

function Add(const Name: string; const Value: string; Enabled:
Boolean = True): **TDACCondition; overload**;

**Parameters**

*Name*
  - Sets the name of the condition

*Value*
  - The value of the condition

*Enabled*
  - Indicates that the condition is enabled

**Remarks**

The given example code will return the following SQL:

```sql
SELECT * FROM EMP
WHERE (JOB="MANAGER")
and
(SAL<2500)
```

5.11.1.9.3.2  Delete Method

Deletes the condition

**Class**

**TDACConditions**

**Syntax**

```pascal
procedure Delete(Index: integer);
```

**Parameters**

*Index*
  - Index of the condition

5.11.1.9.3.3  Disable Method

Disables the condition

**Class**
**5.11.1.9.3.4  Enable Method**

Enables the condition

**Class**

**TDAConditions**

**Syntax**

```plaintext
procedure Enable;
```

**5.11.1.9.3.5  Find Method**

Search for TDACondition (the condition) by its name. If found, the TDACondition object is returned, otherwise - nil.

**Class**

**TDAConditions**

**Syntax**

```plaintext
function Find(const Name: string): TDACondition;
```

**Parameters**

- **Name**

© 1997-2021
Devart. All Rights Reserved.
5.11.1.9.3.6 Get Method

Retrieving a TDACcondition object by its name. If found, the TDACcondition object is returned, otherwise - an exception is raised.

Class

**TDACconditions**

Syntax

```pascal
function Get(const Name: string): TDACcondition;
```

Parameters

Name

© 1997-2021 Devart. All Rights Reserved.

5.11.1.9.3.7 IndexOf Method

Retrieving condition index by its name. If found, this condition index is returned, otherwise - the method returns -1.

Class

**TDACconditions**

Syntax

```pascal
function IndexOf(const Name: string): Integer;
```

Parameters

Name

© 1997-2021 Devart. All Rights Reserved.

5.11.1.9.3.8 Remove Method

Removes the condition

Class

**TDACconditions**
5.11.10 TDACConnectionOptions Class

This class allows setting up the behaviour of the TDACConnection class.

For a list of all members of this type, see TDACConnectionOptions members.

Unit

DBAccess

Syntax

TDACConnectionOptions = class(TPersistent);

5.11.10.1 Members

TDACConnectionOptions class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowImplicitConnect</td>
<td>Specifies whether to allow or not implicit connection opening.</td>
</tr>
<tr>
<td>DefaultSortType</td>
<td>Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNa</td>
</tr>
</tbody>
</table>
## Properties of the `TDAConnectionOptions` class.

For a complete list of the `TDAConnectionOptions` class members, see the [TDAConnectionOptions Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DefaultSortType</code></td>
<td>Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the <code>TMemDataSet.IndexFieldNames</code> property of a dataset.</td>
</tr>
<tr>
<td><code>DisconnectedMode</code></td>
<td>Used to open a connection only when needed for performing a server call and closes after performing the operation.</td>
</tr>
<tr>
<td><code>KeepDesignConnected</code></td>
<td>Used to prevent an application from establishing a connection at the time of startup.</td>
</tr>
<tr>
<td><code>LocalFailover</code></td>
<td>If True, the <code>TCustomDAConnection.OnConnectionLost</code> event occurs and a failover operation can be performed after connection breaks.</td>
</tr>
</tbody>
</table>
### AllowImplicitConnect Property

Specifies whether to allow or not implicit connection opening.

#### Class

TDAConnectionOptions

#### Syntax

```delphi
class TDAConnectionOptions

property AllowImplicitConnect: Boolean default True;
```

#### Remarks

Use the AllowImplicitConnect property to specify whether allow or not implicit connection opening.

If a closed connection has AllowImplicitConnect set to True and a dataset that uses the connection is opened, the connection is opened implicitly to allow opening the dataset.

### LocalFailover

If True, the TCustomDACConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks.
If a closed connection has AllowImplicitConnect set to False and a dataset that uses the connection is opened, an exception is raised.

The default value is True.

5.11.1.10.2.2 DefaultSortType Property

Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.

Class

TDADevelopmentOptions

Syntax

```pascal
property DefaultSortType: TSortType default stCaseSensitive;
```

Remarks

Use the DefaultSortType property to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.

5.11.1.10.2.3 DisconnectedMode Property

Used to open a connection only when needed for performing a server call and closes after performing the operation.

Class

TDADevelopmentOptions

Syntax

```pascal
property DisconnectedMode: boolean default False;
```

Remarks
If True, connection opens only when needed for performing a server call and closes after performing the operation. Datasets remain opened when connection closes. May be useful to save server resources and operate in unstable or expensive network. Drawback of using disconnect mode is that each connection establishing requires some time for authorization. If connection is often closed and opened it can slow down the application work. See the Disconnected Mode topic for more information.

5.11.1.10.2.4  KeepDesignConnected Property

Used to prevent an application from establishing a connection at the time of startup.

Class
TDACConnectionOptions

Syntax

```vbnet
property KeepDesignConnected: boolean default True;
```

Remarks

At the time of startup prevents application from establishing a connection even if the Connected property was set to True at design-time. Set KeepDesignConnected to False to initialize the connected property to False, even if it was True at design-time.

5.11.1.10.2.5  LocalFailover Property

If True, the TCustomDAConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks.

Class
TDACConnectionOptions

Syntax

```vbnet
property LocalFailover: boolean default False;
```
Remarks

If True, the TCustomDAConnection.OnConnectionLost event occurs and a failover operation can be performed after connection breaks. Read the Working in an Unstable Network topic for more information about using failover.

5.11.1.11 TDACollectionSSLOptions Class

This class is used to set up the SSL options.

For a list of all members of this type, see TDACollectionSSLOptions members.

Unit

DBAccess

Syntax

TDACollectionSSLOptions = class(TPersistent);

5.11.1.11.1 Members

TDACollectionSSLOptions class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACert</td>
<td>Holds the path to the certificate authority file.</td>
</tr>
<tr>
<td>Cert</td>
<td>Holds the path to the client certificate.</td>
</tr>
<tr>
<td>CipherList</td>
<td>Holds the list of allowed SSL ciphers.</td>
</tr>
<tr>
<td>Key</td>
<td>Holds the path to the private client key.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.
Properties of the **TDACConnectionSSLOptions** class.

For a complete list of the **TDACConnectionSSLOptions** class members, see the **TDACConnectionSSLOptions Members** topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACert</td>
<td>Holds the path to the certificate authority file.</td>
</tr>
<tr>
<td>Cert</td>
<td>Holds the path to the client certificate.</td>
</tr>
<tr>
<td>CipherList</td>
<td>Holds the list of allowed SSL ciphers.</td>
</tr>
<tr>
<td>Key</td>
<td>Holds the path to the private client key.</td>
</tr>
</tbody>
</table>

### See Also
- **TDACConnectionSSLOptions Class**
- **TDACConnectionSSLOptions Class Members**

---

**CACert Property**

Holds the path to the certificate authority file.

**Class**

**TDACConnectionSSLOptions**

**Syntax**

```csharp
property CACert: string;
```

**Remarks**

Use the CACert property to specify the path to the certificate authority file.
5.11.1.11.2.2  Cert Property

Holds the path to the client certificate.

Class

TDAConnectionSSLOptions

Syntax

```delphi
property Cert: string;
```

Remarks

Use the Cert property to specify the path to the client certificate.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.11.2.3  CipherList Property

Holds the list of allowed SSL ciphers.

Class

TDAConnectionSSLOptions

Syntax

```delphi
property CipherList: string;
```

Remarks

Use the CipherList property to specify the list of allowed SSL ciphers.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.11.2.4  Key Property

Holds the path to the private client key.

Class

TDAConnectionSSLOptions
Syntax

```property``

```key: string;```

Remarks

Use the Key property to specify the path to the private client key.

5.11.1.12 TDADatasetOptions Class

This class allows setting up the behaviour of the TDADataset class.

For a list of all members of this type, see TDADatasetOptions members.

Unit

`DBAccess`

Syntax

```TDADatasetOptions = class(TPersistent);```

5.11.1.12.1 Members

**TDADatasetOptions** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoPrepare</strong></td>
<td>Used to execute automatic TCustomDADataset.Prepare on the query execution.</td>
</tr>
<tr>
<td><strong>CacheCalcFields</strong></td>
<td>Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td><strong>CompressBlobMode</strong></td>
<td>Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td><strong>DefaultValues</strong></td>
<td>Used to request default values/expressions from the server and assign them to the DefaultExpression property.</td>
</tr>
<tr>
<td><strong>DetailDelay</strong></td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td><strong>FieldsOrigin</strong></td>
<td>Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.</td>
</tr>
<tr>
<td><strong>FlatBuffers</strong></td>
<td>Used to control how a dataset treats data of the ftString and ftVarBytes fields.</td>
</tr>
<tr>
<td><strong>InsertAllSetFields</strong></td>
<td>Used to include all set dataset fields in the generated INSERT statement.</td>
</tr>
<tr>
<td><strong>LocalMasterDetail</strong></td>
<td>Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.</td>
</tr>
<tr>
<td><strong>LongStrings</strong></td>
<td>Used to represent string fields with the length that is greater than 255 as TStringField.</td>
</tr>
<tr>
<td><strong>MasterFieldsNullable</strong></td>
<td>Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).</td>
</tr>
<tr>
<td><strong>NumberRange</strong></td>
<td>Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate</td>
</tr>
<tr>
<td><strong>QueryRecCount</strong></td>
<td>Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td><strong>QuoteNames</strong></td>
<td>Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td><strong>RemoveOnRefresh</strong></td>
<td>Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td><strong>RequiredFields</strong></td>
<td>Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong></td>
<td>Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>SetFieldsReadOnly</strong></td>
<td>Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td><strong>StrictUpdate</strong></td>
<td>Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.</td>
</tr>
<tr>
<td><strong>TrimFixedChar</strong></td>
<td>Specifies whether to discard all trailing spaces in the string fields of a dataset.</td>
</tr>
<tr>
<td><strong>UpdateAllFields</strong></td>
<td>Used to include all dataset fields in the generated UPDATE and INSERT statements.</td>
</tr>
</tbody>
</table>
### Properties of the `TDADDataSetOptions` class.

For a complete list of the `TDADDataSetOptions` class members, see the [TDADDataSetOptions Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoPrepare</strong></td>
<td>Used to execute automatic <code>TCustomDADataSet.Prepare</code> on the query execution.</td>
</tr>
<tr>
<td><strong>CacheCalcFields</strong></td>
<td>Used to enable caching of the <code>TField.Calculated</code> and <code>TField.Lookup</code> fields.</td>
</tr>
<tr>
<td><strong>CompressBlobMode</strong></td>
<td>Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td><strong>DefaultValues</strong></td>
<td>Used to request default values/expressions from the server and assign them to the <code>DefaultExpression</code> property.</td>
</tr>
<tr>
<td><strong>DetailDelay</strong></td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td><strong>FieldsOrigin</strong></td>
<td>Used for <code>TCustomDADataSet</code> to fill the <code>Origin</code> property of the <code>TField</code> objects by appropriate value when</td>
</tr>
<tr>
<td><strong>FlatBuffers</strong></td>
<td>opening a dataset.</td>
</tr>
<tr>
<td><strong>InsertAllSetFields</strong></td>
<td>Used to control how a dataset treats data of the ftString and ftVarBytes fields.</td>
</tr>
<tr>
<td><strong>LocalMasterDetail</strong></td>
<td>Used to include all set dataset fields in the generated INSERT statement.</td>
</tr>
<tr>
<td><strong>LongStrings</strong></td>
<td>Used for TCustomDADataSet to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.</td>
</tr>
<tr>
<td><strong>MasterFieldsNullable</strong></td>
<td>Used to represent string fields with the length that is greater than 255 as TStringField.</td>
</tr>
<tr>
<td><strong>NumberRange</strong></td>
<td>Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).</td>
</tr>
<tr>
<td><strong>QueryRecCount</strong></td>
<td>Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.</td>
</tr>
<tr>
<td><strong>QuoteNames</strong></td>
<td>Used for TCustomDADataSet to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td><strong>InsertAllSetFields</strong></td>
<td>Used for TCustomDADataSet to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td><strong>RemoveOnRefresh</strong></td>
<td>Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>RequiredFields</strong></td>
<td>Used for TCustomDADataSet to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong></td>
<td>Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>SetFieldsReadOnly</strong></td>
<td>Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td><strong>StrictUpdate</strong></td>
<td>Used for TCustomDADataSet to raise an exception when the number of updated or deleted records is not equal 1.</td>
</tr>
<tr>
<td><strong>TrimFixedChar</strong></td>
<td>Specifies whether to discard all trailing spaces in the string fields of a dataset.</td>
</tr>
<tr>
<td><strong>UpdateAllFields</strong></td>
<td>Used to include all dataset fields in the generated UPDATE and INSERT statements.</td>
</tr>
<tr>
<td><strong>UpdateBatchSize</strong></td>
<td>Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.</td>
</tr>
</tbody>
</table>

See Also
- TDADatasetOptions Class
- TDADatasetOptions Class Members
5.11.1.12.2.1  AutoPrepare Property

Used to execute automatic TCustomDADataSet.Prepare on the query execution.

Class
TDADatasetOptions

Syntax

| property | AutoPrepare: boolean default False; |

Remarks
Use the AutoPrepare property to execute automatic TCustomDADataSet.Prepare on the query execution. Makes sense for cases when a query will be executed several times, for example, in Master/Detail relationships.

5.11.1.12.2.2  CacheCalcFields Property

Used to enable caching of the TField.Calculated and TField.Lookup fields.

Class
TDADatasetOptions

Syntax

| property | CacheCalcFields: boolean default False; |

Remarks
Use the CacheCalcFields property to enable caching of the TField.Calculated and TField.Lookup fields. It can be useful for reducing CPU usage for calculated fields. Using caching of calculated and lookup fields increases memory usage on the client side.
5.11.1.12.2.3 CompressBlobMode Property

Used to store values of the BLOB fields in compressed form.

Class

TDADatasetOptions

Syntax

```delphi
property CompressBlobMode: TCompressBlobMode default cbNone;
```

Remarks

Use the CompressBlobMode property to store values of the BLOB fields in compressed form. Add the MemData unit to uses list to use this option. Compression rate greatly depends on stored data, for example, usually graphic data compresses badly unlike text.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.12.2.4 DefaultValues Property

Used to request default values/expressions from the server and assign them to the DefaultExpression property.

Class

TDADatasetOptions

Syntax

```delphi
property DefaultValues: boolean default False;
```

Remarks

If True, the default values/expressions are requested from the server and assigned to the DefaultExpression property of TField objects replacing already existent values.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.12.2.5 DetailDelay Property

Used to get or set a delay in milliseconds before refreshing detail dataset while navigating

© 2021 Devart
master dataset.

Class

**TDADatasetOptions**

Syntax

```pascal
property DetailDelay: integer default 0;
```

Remarks

Use the DetailDelay property to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset. If DetailDelay is 0 (the default value) then refreshing of detail dataset occurs immediately. The DetailDelay option should be used for detail dataset.

© 1997-2021 Devart. All Rights Reserved.

5.11.12.2.6 FieldsOrigin Property

Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.

Class

**TDADatasetOptions**

Syntax

```pascal
property FieldsOrigin: boolean;
```

Remarks

If True, TCustomDADataset fills the Origin property of the TField objects by appropriate value when opening a dataset.

© 1997-2021 Devart. All Rights Reserved.

5.11.12.2.7 FlatBuffers Property

Used to control how a dataset treats data of the ftString and ftVarBytes fields.

Class
**TDADatasetOptions**

**Syntax**

```plaintext
property FlatBuffers: boolean default False;
```

**Remarks**

Use the FlatBuffers property to control how a dataset treats data of the ftString and ftVarBytes fields. When set to True, all data fetched from the server is stored in record pData without unused tails.

**5.11.1.12.2.8 InsertAllSetFields Property**

Used to include all set dataset fields in the generated INSERT statement.

**Class**

**TDADatasetOptions**

**Syntax**

```plaintext
property InsertAllSetFields: boolean default False;
```

**Remarks**

If True, all set dataset fields, including those set to NULL explicitly, will be included in the generated INSERT statements. Otherwise, only set fields containing not NULL values will be included to the generated INSERT statement.

**5.11.1.12.2.9 LocalMasterDetail Property**

Used for TCustomDADataSet to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.

**Class**

**TDADatasetOptions**
syntax

**property** LocalMasterDetail: boolean **default** False;

Remarks

If True, for detail dataset in master-detail relationship TCustomDADataset uses local filtering for establishing master/detail relationship and does not refer to the server. Otherwise detail dataset performs query each time a record is selected in master dataset. This option is useful for reducing server calls number, server resources economy. It can be useful for slow connection. The **TMemDataSet.CachedUpdates** mode can be used for detail dataset only when this option is set to true. Setting the LocalMasterDetail option to True is not recommended when detail table contains too many rows, because when it is set to False, only records that correspond to the current record in master dataset are fetched.

Class

**TDADatasetOptions**

Syntax

**property** LongStrings: boolean **default** True;

Remarks

Use the LongStrings property to represent string fields with the length that is greater than 255 as TStringField, not as TMemoField.

Class

**TDADatasetOptions**

5.11.1.12.2.11 MasterFieldsNullable Property

Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).
**TDADatasetOptions**

**Syntax**

```property MasterFieldsNullable: boolean default False;```

**Remarks**

Use the MasterFieldsNullable property to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

**Class**

TDADatasetOptions

**Syntax**

```property NumberRange: boolean default False;```

**Remarks**

Use the NumberRange property to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.

**5.11.1.12.2.13 QueryRecCount Property**

Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.

**Class**

TDADatasetOptions

**Syntax**

```property QueryRecCount: boolean default False;```

**Remarks**

If True, and the FetchAll property is False, TCustomDADataset performs additional query to
get the record count for this SELECT, so the RecordCount property reflects the actual number of records. Does not have any effect if the FetchAll property is True.

5.11.1.12.2.14 QuoteNames Property

Used for TCustomDADataset to quote all database object names in autogenerated SQL statements such as update SQL.

Class

TDADataSetOptions

Syntax

```property QuoteNames: boolean default False;```

Remarks

If True, TCustomDADataset quotes all database object names in autogenerated SQL statements such as update SQL.

5.11.1.12.2.15 RemoveOnRefresh Property

Used for a dataset to locally remove a record that can not be found on the server.

Class

TDADataSetOptions

Syntax

```property RemoveOnRefresh: boolean default True;```

Remarks

When the RefreshRecord procedure can't find necessary record on the server and RemoveOnRefresh is set to True, dataset removes the record locally. Usually RefreshRecord can't find necessary record when someone else dropped the record or changed the key value of it.
This option makes sense only if the StrictUpdate option is set to False. If the StrictUpdate option is True, error will be generated regardless of the RemoveOnRefresh option value.

5.11.1.12.2.16 RequiredFields Property

Used for TCustomDADataset to set the Required property of the TField objects for the NOT NULL fields.

Class

TDADatasetOptions

Syntax

```property
property RequiredFields: boolean default True;
```

Remarks

If True, TCustomDADataset sets the Required property of the TField objects for the NOT NULL fields. It is useful when table has a trigger which updates the NOT NULL fields.

5.11.1.12.2.17 ReturnParams Property

Used to return the new value of fields to dataset after insert or update.

Class

TDADatasetOptions

Syntax

```property
property ReturnParams: boolean default False;
```

Remarks

Use the ReturnParams property to return the new value of fields to dataset after insert or update. The actual value of field after insert or update may be different from the value stored in the local memory if the table has a trigger. When ReturnParams is True, OUT parameters of the SQLInsert and SQLUpdate statements is assigned to the corresponding fields.
5.11.1.12.2.18 SetFieldsReadOnly Property

Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.

Class

TDADatasetOptions

Syntax

```
property SetFieldsReadOnly: boolean default True;
```

Remarks

If True, dataset sets the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated. Set this option for datasets that use automatic generation of the update SQL statements only.

5.11.1.12.2.19 StrictUpdate Property

Used for TCustomDADataSet to raise an exception when the number of updated or deleted records is not equal 1.

Class

TDADatasetOptions

Syntax

```
property StrictUpdate: boolean default True;
```

Remarks

If True, TCustomDADataSet raises an exception when the number of updated or deleted records is not equal 1. Setting this option also causes the exception if the RefreshRecord procedure returns more than one record. The exception does not occur when you execute SQL query, that doesn't return resultset.
**Note**: There can be problems if this option is set to True and triggers for UPDATE, DELETE, REFRESH commands that are defined for the table. So it is recommended to disable (set to False) this option with triggers.

TrimFixedChar specifies whether to discard all trailing spaces in the string fields of a dataset.

```
property TrimFixedChar: boolean default True;
```

**Remarks**

Specifies whether to discard all trailing spaces in the string fields of a dataset.

**UpdateAllFields Property**

Used to include all dataset fields in the generated UPDATE and INSERT statements.

```
property UpdateAllFields: boolean default False;
```

**Remarks**

If True, all dataset fields will be included in the generated UPDATE and INSERT statements. Unspecified fields will have NULL value in the INSERT statements. Otherwise, only updated fields will be included to the generated update statements.
5.11.1.12.22 UpdateBatchSize Property

Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.

Class

TDADatasetOptions

Syntax

```pascal
property UpdateBatchSize: Integer default 1;
```

Remarks

Use the UpdateBatchSize property to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch. Takes effect only when updating dataset in the TMemDataSet.CachedUpdates mode. The default value is 1.

5.11.1.13 TDAEncryption Class

Used to specify the options of the data encryption in a dataset.

For a list of all members of this type, see TDAEncryption members.

Unit

DBAccess

Syntax

```pascal
TDAEncryption = class(TPersistent);
```

Remarks

Set the properties of Encryption to specify the options of the data encryption in a dataset.
### TDAEncryption class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encryptor</strong></td>
<td>Used to specify the encryptor class that will perform the data encryption.</td>
</tr>
<tr>
<td><strong>Fields</strong></td>
<td>Used to set field names for which encryption will be performed.</td>
</tr>
</tbody>
</table>

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Encryptor</strong></td>
<td>Used to specify the encryptor class that will perform the data encryption.</td>
</tr>
</tbody>
</table>

#### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fields</strong></td>
<td>Used to set field names for which encryption will be performed.</td>
</tr>
</tbody>
</table>

See Also
- [TDAEncryption Class](#)
- [TDAEncryption Class Members](#)
5.11.1.13.2.1 Encryptor Property

Used to specify the encryptor class that will perform the data encryption.

Class

**TDAEncryption**

Syntax

```property`` Encryptor: **TCREncryptor**;
```

Remarks

Use the Encryptor property to specify the encryptor class that will perform the data encryption.

5.11.1.13.2.2 Fields Property

Used to set field names for which encryption will be performed.

Class

**TDAEncryption**

Syntax

```property`` Fields: **string**;
```

Remarks

Used to set field names for which encryption will be performed. Field names must be separated by semicolons.
5.11.1.14 TDAMapRule Class

Class that forms rules for Data Type Mapping.

For a list of all members of this type, see TDAMapRule members.

Unit

DBAccess

Syntax

TDAMapRule = class(TMapRule);

Remarks

Using properties of this class, it is possible to change parameter values of the specified rules from the TDAMapRules set.

Inheritance Hierarchy

TMapRule
   TDAMapRule

© 1997-2021
Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback

5.11.1.14.1 Members

TDAMapRule class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBLengthMax</td>
<td>Maximum DB field length, until which the rule is applied.</td>
</tr>
<tr>
<td>DBLengthMin</td>
<td>Minimum DB field length, starting from which the rule is applied.</td>
</tr>
<tr>
<td>DBScaleMax</td>
<td>Maximum DB field scale, until which the rule is applied to the specified DB field.</td>
</tr>
<tr>
<td>DBScaleMin</td>
<td>Minimum DB field Scale, starting from which the rule is applied.</td>
</tr>
</tbody>
</table>
Properties of the `TDAMapRule` class.

For a complete list of the `TDAMapRule` class members, see the [TDAMapRule Members topic](#).

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DBLengthMax</code></td>
<td>Maximum DB field length, until which the rule is applied.</td>
</tr>
<tr>
<td><code>DBLengthMin</code></td>
<td>Minimum DB field length, starting from which the rule is applied.</td>
</tr>
<tr>
<td><code>DBScaleMax</code></td>
<td>Maximum DB field scale, until which the rule is applied to the specified DB field.</td>
</tr>
<tr>
<td><code>DBScaleMin</code></td>
<td>Minimum DB field Scale, starting from which the rule is applied to the specified DB field.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DBType</strong></td>
<td>DB field type, that the rule is applied to.</td>
</tr>
<tr>
<td><strong>FieldLength</strong></td>
<td>The resultant field length in Delphi.</td>
</tr>
<tr>
<td><strong>FieldName</strong></td>
<td>DataSet field name, for which the rule is applied.</td>
</tr>
<tr>
<td><strong>FieldScale</strong></td>
<td>The resultant field Scale in Delphi.</td>
</tr>
<tr>
<td><strong>FieldType</strong></td>
<td>Delphi field type, that the specified DB type or DataSet field will be mapped to.</td>
</tr>
<tr>
<td><strong>IgnoreErrors</strong></td>
<td>Ignoring errors when converting data from DB to Delphi type.</td>
</tr>
</tbody>
</table>

**See Also**
- [TDAMapRule Class](#)
- [TDAMapRule Class Members](#)

### 5.11.1.14.2.1 DBLengthMax Property

Maximum DB field length, until which the rule is applied.

#### Class

**TDAMapRule**

#### Syntax

```delphi
property DBLengthMax: Integer default rlAny;
```

#### Remarks

Setting maximum DB field length, until which the rule is applied to the specified DB field.
5.11.1.14.2.2 DBLengthMin Property

Minimum DB field length, starting from which the rule is applied.

Class
TDAMapRule

Syntax

```property DBLengthMin: Integer default rlAny;```

Remarks
Setting minimum DB field length, starting from which the rule is applied to the specified DB field.

5.11.1.14.2.3 DBScaleMax Property

Maximum DB field scale, until which the rule is applied to the specified DB field.

Class
TDAMapRule

Syntax

```property DBScaleMax: Integer default rlAny;```

Remarks
Setting maximum DB field scale, until which the rule is applied to the specified DB field.

5.11.1.14.2.4 DBScaleMin Property

Minimum DB field Scale, starting from which the rule is applied to the specified DB field.

Class
TDAMapRule
5.11.1.14.2.5  DBScaleMin Property

DB field type, that the rule is applied to.

Class
TDAMapRule

Syntax
```
property DBScaleMin: Integer default rlAny;
```

Remarks
Setting minimum DB field Scale, starting from which the rule is applied to the specified DB field.

5.11.1.14.2.6  DBType Property

DB field type, that the rule is applied to.

Class
TDAMapRule

Syntax
```
property DBType: Word default dtUnknown;
```

Remarks
Setting DB field type, that the rule is applied to. If the current rule is set for Connection, the rule will be applied to all fields of the specified type in all DataSets related to this Connection.

5.11.1.14.2.6  FieldLength Property

The resultant field length in Delphi.

Class
TDAMapRule

Syntax
```
property FieldLength: Integer default rlAny;
```

Remarks
5.11.1.14.2.7 FieldName Property

Dataset field name, for which the rule is applied.

Class

TDAMapRule

Syntax

```pascal
property FieldName: string;
```

Remarks

Specifies the Dataset field name, that the rule is applied to. If the current rule is set for Connection, the rule will be applied to all fields with such name in DataSets related to this Connection.

5.11.1.14.2.8 FieldScale Property

The resultant field Scale in Delphi.

Class

TDAMapRule

Syntax

```pascal
property FieldScale: Integer default rlAny;
```

Remarks

Setting the Delphi field Scale after conversion.
5.11.1.14.2.9  FieldType Property

Delphi field type, that the specified DB type or DataSet field will be mapped to.

Class

TDAMapRule

Syntax

property FieldType: TFieldType stored IsFieldTypeStored default ftUnknown;

Remarks

Setting Delphi field type, that the specified DB type or DataSet field will be mapped to.

5.11.1.14.2.10  IgnoreErrors Property

Ignoring errors when converting data from DB to Delphi type.

Class

TDAMapRule

Syntax

property IgnoreErrors: Boolean default False;

Remarks

Allows to ignore errors while data conversion in case if data or DB data format cannot be recorded to the specified Delphi field type. The default value is false.

5.11.1.15  TDAMapRules Class

Used for adding rules for DataSet fields mapping with both identifying by field name and by field type and Delphi field types.

For a list of all members of this type, see TDAMapRules members.
Unit

**DBAccess**

**Syntax**

```
TDAMapRules = class(TMapRules);
```

**Inheritance Hierarchy**

- TMapRules
  - TDAMapRules

© 1997-2021 Devart. All Rights Reserved.

### 5.11.1.15.1 Members

**TDAMapRules** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IgnoreInvalidRules</td>
<td>Used to avoid raising exception on mapping rules that can't be applied.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

### 5.11.1.15.2 Properties

Properties of the **TDAMapRules** class.

For a complete list of the **TDAMapRules** class members, see the [TDAMapRules Members](#) topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IgnoreInvalidRules</td>
<td>Used to avoid raising exception on mapping rules that can't be applied.</td>
</tr>
</tbody>
</table>
5.11.1.15.2.1 IgnoreInvalidRules Property

Used to avoid raising exception on mapping rules that can't be applied.

Class
TDAMapRules

Syntax

```delphi
property IgnoreInvalidRules: boolean default False;
```

Remarks

Allows to ignore errors (not to raise exception) during data conversion in case if the data or
DB data format cannot be recorded to the specified Delphi field type. The default value is
false.

**Note:** In order to ignore errors occurring during data conversion, use the
TDAMapRule.IgnoreErrors property

See Also

- TDAMapRule.IgnoreErrors

5.11.1.16 TDAMetaData Class

A class for retrieving metainformation of the specified database objects in the form of dataset.

For a list of all members of this type, see TDAMetaData members.

Unit
DBAccess
Syntax

TDAMetaData = class(TMemDataSet);

Remarks

TDAMetaData is a TDataSet descendant standing for retrieving metainformation of the specified database objects in the form of dataset. First of all you need to specify which kind of metainformation you want to see. For this you need to assign the TDAMetaData.MetaDataKind property. Provide one or more conditions in the TDAMetaData.Restrictions property to diminish the size of the resultset and get only information you are interested in.

Use the TDAMetaData.GetMetaDataKinds method to get the full list of supported kinds of meta data. With the TDAMetaData.GetRestrictions method you can find out what restrictions are applicable to the specified MetaDataKind.

Example

The code below demonstrates how to get information about columns of the 'emp' table:

```pascal
MetaData.Connection := Connection;
MetaData.MetaDataKind := 'Columns';
MetaData.Restrictions.Values['TABLE_NAME'] := 'Emp';
MetaData.Open;
```

Inheritance Hierarchy

TMemDataSet
   TDAMetaData

See Also

- TDAMetaData.MetaDataKind
- TDAMetaData.Restrictions
- TDAMetaData.GetMetaDataKinds
- TDAMetaData.GetRestrictions

© 1997-2021
Devart. All Rights Reserved.

TDAMetaData class overview.
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CachedUpdates</strong></td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong></td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong></td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong></td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong></td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>MetaDataKind</strong></td>
<td>Used to specify which kind of metainformation to show.</td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong></td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>Restrictions</strong></td>
<td>Used to provide one or more conditions restricting the list of objects to be described.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong></td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong></td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ApplyRange</strong></td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong></td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>CancelRange</strong></td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong></td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong></td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong></td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong></td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetMetaDataKinds</strong></td>
<td>Used to get values acceptable in the MetaDataKind property.</td>
</tr>
<tr>
<td><strong>GetRestrictions</strong></td>
<td>Used to find out which restrictions are applicable to a certain MetaDataKind.</td>
</tr>
<tr>
<td><strong>Locate</strong></td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><strong>LocateEx</strong></td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td><strong>Prepare</strong></td>
<td>Allocates resources and creates field components for</td>
</tr>
<tr>
<td>Event Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetRange</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SetRangeStart</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>UpdateResult</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
OnUpdateError (inherited from TMemDataSet)  Occurs when an exception is generated while cached updates are applied to a database.

OnUpdateRecord (inherited from TMemDataSet)  Occurs when a single update component can not handle the updates.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.16.2 Properties

Properties of the TDAMetaData class.

For a complete list of the TDAMetaData class members, see the TDAMetaData Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>(inherited from TMemDataSet) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>Connection</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>(inherited from TMemDataSet) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>(inherited from TMemDataSet) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>(inherited from TMemDataSet) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>(inherited from TMemDataSet) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>MetaDataKind</td>
<td>Used to specify which kind of metainformation to show.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>Restrictions</strong></td>
<td>Used to provide one or more conditions restricting the list of objects to be described.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

**See Also**
- **TDAMetaData Class**
- **TDAMetaData Class Members**

5.11.1.16.2.1 Connection Property

Used to specify a connection object to use to connect to a data store.

**Class**

**TDAMetaData**

**Syntax**

```
property Connection: TCustomDAConnection;
```

**Remarks**

Use the Connection property to specify a connection object to use to connect to a data store.

Set at design-time by selecting from the list of provided TCustomDAConnection or its descendant class objects.

At runtime, set the Connection property to reference an instanciated TCustomDAConnection object.
5.11.1.16.2.2  MetaDataKind Property

Used to specify which kind of metainformation to show.

Class

TDAMetaData

Syntax

```plaintext
property MetaDataKind: string;
```

Remarks

This string property specifies which kind of metainformation to show. The value of this property should be assigned before activating the component. If MetaDataKind equals to an empty string (the default value), the full value list that this property accepts will be shown.

They are described in the table below:

<table>
<thead>
<tr>
<th>MetaDataKind</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>show metainformation about columns of existing tables</td>
</tr>
<tr>
<td>Constraints</td>
<td>show metainformation about the constraints defined in the database</td>
</tr>
<tr>
<td>IndexColumns</td>
<td>show metainformation about indexed columns</td>
</tr>
<tr>
<td>Indexes</td>
<td>show metainformation about indexes in a database</td>
</tr>
<tr>
<td>MetaDataKinds</td>
<td>show the acceptable values of this property. You will get the same result if the MetadataKind property is an empty string</td>
</tr>
<tr>
<td>ProcedureParameters</td>
<td>show metainformation about parameters of existing procedures</td>
</tr>
<tr>
<td>Procedures</td>
<td>show metainformation about existing procedures</td>
</tr>
<tr>
<td>Restrictions</td>
<td>generates a dataset that describes which restrictions are applicable to each MetaDataKind</td>
</tr>
<tr>
<td>Tables</td>
<td>show metainformation about existing tables</td>
</tr>
<tr>
<td>Databases</td>
<td>show metainformation about existing databases</td>
</tr>
</tbody>
</table>

If you provide a value that equals neither of the values described in the table, an error will be raised.

See Also

- Restrictions
5.11.1.16.3  Methods

Methods of the **TDAMetaData** class.

For a complete list of the **TDAMetaData** class members, see the **TDAMetaData Members** topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyRange</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>CancelRange</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetMetaDataKinds</strong></td>
<td>Used to get values acceptable in the MetaDataKind property.</td>
</tr>
<tr>
<td><strong>GetRestrictions</strong></td>
<td>Used to find out which restrictions are applicable to a certain MetaDataKind.</td>
</tr>
<tr>
<td><strong>Locate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><strong>LocateEx</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td><strong>Prepare</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td><strong>RevertRecord</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SaveToXML</strong></td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><strong>SetRange</strong></td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><strong>SetRangeEnd</strong></td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><strong>SetRangeStart</strong></td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><strong>UnPrepare</strong></td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><strong>UpdateResult</strong></td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdateStatus</strong></td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

**See Also**
- [TDAMetaData Class](#)
- [TDAMetaData Class Members](#)

© 1997-2021
Devar. All Rights Reserved.
**TDAMetaData**

**Syntax**

```pascal
procedure GetMetaDataKinds(List: TStrings);
```

**Parameters**

- **List**
  - Holds the object that will be filled with metadata kinds (restrictions).

**Remarks**

Call the GetMetaDataKinds method to get values acceptable in the MetaDataKind property. The List parameter will be cleared and then filled with values.

**See Also**

- MetaDataKind

© 1997-2021 Devart. All Rights Reserved.

5.11.1.16.3.2 GetRestrictions Method

**Class**

**TDAMetaData**

**Syntax**

```pascal
procedure GetRestrictions(List: TStrings; const MetaDataKind: string);
```

**Parameters**

- **List**
  - Holds the object that will be filled with metadata kinds (restrictions).
- **MetaDataKind**
  - Holds the metadata kind for which restrictions are returned.

**Remarks**

Call the GetRestrictions method to find out which restrictions are applicable to a certain MetaDataKind. The List parameter will be cleared and then filled with values.
5.11.1.17 TDAParam Class

A class that forms objects to represent the values of the parameters set.

For a list of all members of this type, see TDAParam members.

Unit

DBAccess

Syntax

TDAParam = class(TParam);

Remarks

Use the properties of TDAParam to set the value of a parameter. Objects that use parameters create TDAParam objects to represent these parameters. For example, TDAParam objects are used by TCustomDASQL, TCustomDADataSet.

TDAParam shares many properties with TField, as both describe the value of a field in a dataset. However, a TField object has several properties to describe the field binding and the way the field is displayed, edited, or calculated, that are not needed in a TDAParam object. Conversely, TDAParam includes properties that indicate how the field value is passed as a parameter.

See Also

- TCustomDADataSet
- TCustomDASQL
- TDAParams
### Members

**TDAParam** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsBlob</td>
<td>Used to set and read the value of the BLOB parameter as string.</td>
</tr>
<tr>
<td>AsBlobRef</td>
<td>Used to set and read the value of the BLOB parameter as a TBlob object.</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Used to assign the value for a float field to a parameter.</td>
</tr>
<tr>
<td>AsInteger</td>
<td>Used to assign the value for an integer field to the parameter.</td>
</tr>
<tr>
<td>AsLargeInt</td>
<td>Used to assign the value for a LargeInteger field to the parameter.</td>
</tr>
<tr>
<td>AsMemo</td>
<td>Used to assign the value for a memo field to the parameter.</td>
</tr>
<tr>
<td>AsMemoRef</td>
<td>Used to set and read the value of the memo parameter as a TBlob object.</td>
</tr>
<tr>
<td>AsSQLTimeStamp</td>
<td>Used to specify the value of the parameter when it represents a SQL timestamp field.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to assign the string value to the parameter.</td>
</tr>
<tr>
<td>AsWideString</td>
<td>Used to assign the Unicode string value to the parameter.</td>
</tr>
<tr>
<td>DataType</td>
<td>Indicates the data type of the parameter.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to indicate whether the value assigned to a parameter is NULL.</td>
</tr>
<tr>
<td>ParamType</td>
<td>Used to indicate the type of use for a parameter.</td>
</tr>
</tbody>
</table>
Size | Specifies the size of a string type parameter.
---|---
Value | Used to represent the value of the parameter as Variant.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignField</td>
<td>Assigns field name and field value to a param.</td>
</tr>
<tr>
<td>AssignFieldValue</td>
<td>Assigns the specified field properties and value to a parameter.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>Places the content of a specified file into a TDAParam object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Places the content from a stream into a TDAParam object.</td>
</tr>
<tr>
<td>SetBlobData</td>
<td>Overloaded. Writes the data from a specified buffer to BLOB.</td>
</tr>
</tbody>
</table>

Properties

Properties of the TDAParam class.

For a complete list of the TDAParam class members, see the TDAParam Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsBlob</td>
<td>Used to set and read the value of the BLOB parameter as string.</td>
</tr>
<tr>
<td>AsBlobRef</td>
<td>Used to set and read the value of the BLOB parameter as a TBlob object.</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Used to assign the value for</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AsInteger</strong></td>
<td>Used to assign the value for an integer field to the parameter.</td>
</tr>
<tr>
<td><strong>AsLargeInt</strong></td>
<td>Used to assign the value for a LargeInteger field to the parameter.</td>
</tr>
<tr>
<td><strong>AsMemo</strong></td>
<td>Used to assign the value for a memo field to the parameter.</td>
</tr>
<tr>
<td><strong>AsMemoRef</strong></td>
<td>Used to set and read the value of the memo parameter as a TBlob object.</td>
</tr>
<tr>
<td><strong>AsSQLTimeStamp</strong></td>
<td>Used to specify the value of the parameter when it represents a SQL timestamp field.</td>
</tr>
<tr>
<td><strong>AsString</strong></td>
<td>Used to assign the string value to the parameter.</td>
</tr>
<tr>
<td><strong>AsWideString</strong></td>
<td>Used to assign the Unicode string value to the parameter.</td>
</tr>
<tr>
<td><strong>IsNull</strong></td>
<td>Used to indicate whether the value assigned to a parameter is NULL.</td>
</tr>
</tbody>
</table>

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataType</strong></td>
<td>Indicates the data type of the parameter.</td>
</tr>
<tr>
<td><strong>ParamType</strong></td>
<td>Used to indicate the type of use for a parameter.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Specifies the size of a string type parameter.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Used to represent the value of the parameter as Variant.</td>
</tr>
</tbody>
</table>

### See Also
- TDAParam Class
- TDAParam Class Members
5.11.1.17.2.1 AsBlob Property

Used to set and read the value of the BLOB parameter as string.

Class

TDAParam

Syntax

```pascal
property AsBlob: TBlobData;
```

Remarks

Use the AsBlob property to set and read the value of the BLOB parameter as string. Setting AsBlob will set the DataType property to ftBlob. AsBlob is the value of the parameter when it represents the value of the LONG RAW type.

5.11.1.17.2.2 AsBlobRef Property

Used to set and read the value of the BLOB parameter as a TBlob object.

Class

TDAParam

Syntax

```pascal
property AsBlobRef: TBlob;
```

Remarks

Use the AsBlobRef property to set and read the value of the BLOB parameter as a TBlob object. Setting AsBlobRef will set the DataType property to ftBlob. Specifies the value of the parameter when it represents the value of the LONG RAW type.
5.11.1.17.2.3 AsFloat Property

Used to assign the value for a float field to a parameter.

Class
TDAParam

Syntax

```property AsFloat: double;```

Remarks

Use the AsFloat property to assign the value for a float field to the parameter. Setting AsFloat will set the DataType property to dtFloat.

Read the AsFloat property to determine the value that was assigned to an output parameter, represented as Double. The value of the parameter will be converted to the Double value if possible.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.17.2.4 AsInteger Property

Used to assign the value for an integer field to the parameter.

Class
TDAParam

Syntax

```property AsInteger: LongInt;```

Remarks

Use the AsInteger property to assign the value for an integer field to the parameter. Setting AsInteger will set the DataType property to dtInteger.

Read the AsInteger property to determine the value that was assigned to an output parameter, represented as a 32-bit integer. The value of the parameter will be converted to the Integer value if possible.

© 1997-2021 Devart. All Rights Reserved.
5.11.1.17.2.5  AsLargeInt Property

Used to assign the value for a LargeInteger field to the parameter.

Class
TDAParam

Syntax

```pascal
property AsLargeInt: Int64;
```

Remarks
Set the AsLargeInt property to assign the value for an Int64 field to the parameter. Setting AsLargeInt will set the DataType property to dtLargeint.

Read the AsLargeInt property to determine the value that was assigned to an output parameter, represented as a 64-bit integer. The value of the parameter will be converted to the Int64 value if possible.

5.11.1.17.2.6  AsMemo Property

Used to assign the value for a memo field to the parameter.

Class
TDAParam

Syntax

```pascal
property AsMemo: string;
```

Remarks
Use the AsMemo property to assign the value for a memo field to the parameter. Setting AsMemo will set the DataType property to ftMemo. AsMemo is the value of the parameter when it represents the value of the LONG type.
5.11.1.17.2.7 AsMemoRef Property

Used to set and read the value of the memo parameter as a TBlob object.

Class

TDAParam

Syntax

```property`` AsMemoRef: TBlob;
```

Remarks

Use the AsMemoRef property to set and read the value of the memo parameter as a TBlob object. Setting AsMemoRef will set the DataType property to ftMemo. Specifies the value of the parameter when it represents the value of the LONG type.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.17.2.8 AsSQLTimeStamp Property

Used to specify the value of the parameter when it represents a SQL timestamp field.

Class

TDAParam

Syntax

```property`` AsSQLTimeStamp: TSQLTimeStamp;
```

Remarks

Set the AsSQLTimeStamp property to assign the value for a SQL timestamp field to the parameter. Setting AsSQLTimeStamp sets the DataType property to ftTimeStamp.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.17.2.9 AsString Property

Used to assign the string value to the parameter.

Class
TDAParam

Syntax

```property AsString: string;```

Remarks

Use the AsString property to assign the string value to the parameter. Setting AsString will set the DataType property to ftString.

Read the AsString property to determine the value that was assigned to an output parameter represented as a string. The value of the parameter will be converted to a string.

© 1997-2021 Devart. All Rights Reserved.

Class

TDAParam

Syntax

```property AsWideString: string;```

Remarks

Set AsWideString to assign the Unicode string value to the parameter. Setting AsWideString will set the DataType property to ftWideString.

Read the AsWideString property to determine the value that was assigned to an output parameter, represented as a Unicode string. The value of the parameter will be converted to a Unicode string.

© 1997-2021 Devart. All Rights Reserved.
5.11.1.17.2.11  DataType Property

 Indicates the data type of the parameter.

Class
TDAParam

Syntax

```pascal
property DataType: TFieldType stored IsDataTypeStored;
```

Remarks

DataType is set automatically when a value is assigned to a parameter. Do not set DataType for bound fields, as this may cause the assigned value to be misinterpreted.

Read DataType to learn the type of data that was assigned to the parameter. Every possible value of DataType corresponds to the type of a database field.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.17.2.12  IsNull Property

 Used to indicate whether the value assigned to a parameter is NULL.

Class
TDAParam

Syntax

```pascal
property IsNull: boolean;
```

Remarks

Use the IsNull property to indicate whether the value assigned to a parameter is NULL.

© 1997-2021
Devart. All Rights Reserved.
5.11.1.17.2.13  ParamType Property

Used to indicate the type of use for a parameter.

Class

TDAParam

Syntax

<table>
<thead>
<tr>
<th>property</th>
<th>ParamType</th>
<th>default</th>
<th>DB . ptUnknown;</th>
</tr>
</thead>
</table>

Remarks

Objects that use TDAParam objects to represent field parameters set ParamType to indicate the type of use for a parameter.

To learn the description of TParamType refer to Delphi Help.

**Note:** The value of ParamType is important for the LONG, LONG RAW, BLOB and CLOB parameters. To write data to database, set ptInput to ParamType, to read data from database, set ptOutput to ParamType.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.17.2.14  Size Property

Specifies the size of a string type parameter.

Class

TDAParam

Syntax

<table>
<thead>
<tr>
<th>property</th>
<th>Size: integer</th>
<th>default</th>
<th>0;</th>
</tr>
</thead>
</table>

Remarks

Use the Size property to indicate the maximum number of characters the parameter may contain. Use the Size property only for Output parameters of the ftString, ftFixedChar, ftBytes, ftVarBytes, or ft WideString type.

© 1997-2021 Devart. All Rights Reserved.
5.11.1.17.2.15 Value Property

Used to represent the value of the parameter as Variant.

Class

TDAParam

Syntax

property Value: variant stored IsValueStored;

Remarks

The Value property represents the value of the parameter as Variant.

Use Value in generic code that manipulates the values of parameters without the need to know the field type the parameter represent.

Methods

Methods of the TDAParam class.

For a complete list of the TDAParam class members, see the TDAParam Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignField</td>
<td>Assigns field name and field value to a param.</td>
</tr>
<tr>
<td>AssignFieldValue</td>
<td>Assigns the specified field properties and value to a parameter.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>Places the content of a specified file into a TDAParam object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Places the content from a stream into a TDAParam object.</td>
</tr>
<tr>
<td>SetBlobData</td>
<td>Overloaded. Writes the data from a specified buffer to BLOB.</td>
</tr>
</tbody>
</table>
5.11.1.17.3.1 AssignField Method

Assigns field name and field value to a param.

Class

TDAParam

Syntax

procedure AssignField(Field: TField);

Parameters

Field

Holds the field which name and value should be assigned to the param.

Remarks

Call the AssignField method to assign field name and field value to a param.

5.11.1.17.3.2 AssignFieldValue Method

Assigns the specified field properties and value to a parameter.

Class

TDAParam

Syntax

procedure AssignFieldValue(Field: TField; const Value: Variant); virtual;

Parameters

Field

© 1997-2021 Devart. All Rights Reserved.

Request Support   DAC Forum   Provide Feedback
Holds the field the properties of which will be assigned to the parameter.

*Value*
Holds the value for the parameter.

**Remarks**

Call the AssignFieldValue method to assign the specified field properties and value to a parameter.

```
5.11.1.17.3.3  LoadFromFile Method
```

Places the content of a specified file into a TDAParam object.

**Class**

_TDAParam_

**Syntax**

```
procedure LoadFromFile(const FileName: string; BlobType: TBlobType);
```

**Parameters**

*FileName*
Holds the name of the file.

*BlobType*
Holds a value that modifies the DataType property so that this TDAParam object now holds the BLOB value.

**Remarks**

Use the LoadFromFile method to place the content of a file specified by FileName into a TDAParam object. The BlobType value modifies the DataType property so that this TDAParam object now holds the BLOB value.

**See Also**

- LoadFromStream
5.11.1.17.3.4  LoadFromStream Method

Places the content from a stream into a TDAParam object.

Class

TDAParam

Syntax

procedure LoadFromStream(Stream: TStream; BlobType: TBlobType);
virtual;

Parameters

Stream
  Holds the stream to copy content from.

BlobType
  Holds a value that modifies the DataType property so that this TDAParam object now holds
  the BLOB value.

Remarks

Call the LoadFromStream method to place the content from a stream into a TDAParam
object. The BlobType value modifies the DataType property so that this TDAParam object
now holds the BLOB value.

See Also

• LoadFromFile

© 1997-2021
Devart. All Rights Reserved.

5.11.1.17.3.5  SetBlobData Method

Writes the data from a specified buffer to BLOB.

Class

TDAParam

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetBlobData(Buffer: TValueBuffer)</td>
<td>Writes the data from a specified buffer to BLOB.</td>
</tr>
</tbody>
</table>
SetBlobData(Buffer: IntPtr; Size: Integer)  Writes the data from a specified buffer to BLOB.

Writes the data from a specified buffer to BLOB.

Class
TDAParam

Syntax

procedure SetBlobData(Buffer: TValueBuffer); overload;

Parameters

Buffer
Holds the pointer to the data.

Size
Holds the number of bytes to read from the buffer.

Remarks

Call the SetBlobData method to write data from a specified buffer to BLOB.
5.11.1.18 TDAParams Class

This class is used to manage a list of TDAParam objects for an object that uses field parameters.

For a list of all members of this type, see TDAParams members.

Unit

DBAccess

Syntax

TDAParams = class(TParams);

Remarks

Use TDAParams to manage a list of TDAParam objects for an object that uses field parameters. For example, TCustomDADataset objects and TCustomDASQL objects use TDAParams objects to create and access their parameters.

See Also

- TCustomDADataset.Params
- TCustomDASQL.Params
- TDAParam

© 1997-2021
Devar. All Rights Reserved.

5.11.1.18.1 Members

TDAParams class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to iterate through all parameters.</td>
</tr>
</tbody>
</table>

Methods

| Name | Description |
FindParam
Searches for a parameter with the specified name.

ParamByName
Searches for a parameter with the specified name.

Properties of the TDAParams class.

For a complete list of the TDAParams class members, see the TDAParams Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to iterate through all parameters.</td>
</tr>
</tbody>
</table>

See Also
- TDAParams Class
- TDAParams Class Members

Items Property(Indexer)

Used to iterate through all parameters.

Class
TDAParams

Syntax

```property Items[Index: integer]: TDAParam; default;```

Parameters

Index
Holds an index in the range 0..Count - 1.
Remarks

Use the Items property to iterate through all parameters. Index identifies the index in the range 0..Count - 1. Items can reference a particular parameter by its index, but the ParamByName method is preferred in order to avoid depending on the order of the parameters.

### 5.11.18.3.1 FindParam Method

Searches for a parameter with the specified name.

**Class**

**TDAParams**

**Syntax**

```
function FindParam(const Value: string): TDAParam;
```

**Parameters**

- **Value**
Holds the parameter name.

**Return Value**

a parameter, if a match was found. Nil otherwise.

**Remarks**

Use the FindParam method to find a parameter with the name passed in Value. If a match is found, FindParam returns the parameter. Otherwise, it returns nil. Use this method rather than a direct reference to the Items property to avoid depending on the order of the entries.

To locate more than one parameter at a time by name, use the GetParamList method instead. To get only the value of a named parameter, use the ParamValues property.
5.11.1.19 TDATransaction Class

A base class that implements functionality for controlling transactions.

For a list of all members of this type, see TDATransaction members.

Unit

DBAccess

Syntax

TDATransaction = class(TComponent);

Remarks

TDATransaction is a base class for components implementing functionality for managing transactions.

Do not create instances of TDATransaction. Use descendants of the TDATransaction class instead.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.19.1 Members

TDATransaction class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to determine if the transaction is active.</td>
</tr>
<tr>
<td>DefaultCloseAction</td>
<td>Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.</td>
</tr>
</tbody>
</table>

Methods
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>Commits the current transaction.</td>
</tr>
<tr>
<td>Rollback</td>
<td>Discards all modifications of data associated with the current transaction and ends the transaction.</td>
</tr>
<tr>
<td>StartTransaction</td>
<td>Begins a new transaction.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnCommit</td>
<td>Occurs after the transaction has been successfully committed.</td>
</tr>
<tr>
<td>OnCommitRetaining</td>
<td>Occurs after CommitRetaining has been executed.</td>
</tr>
<tr>
<td>OnError</td>
<td>Used to process errors that occur during executing a transaction.</td>
</tr>
<tr>
<td>OnRollback</td>
<td>Occurs after the transaction has been successfully rolled back.</td>
</tr>
<tr>
<td>OnRollbackRetaining</td>
<td>Occurs after RollbackRetaining has been executed.</td>
</tr>
</tbody>
</table>

Properties of the `TDATransaction` class.

For a complete list of the `TDATransaction` class members, see the `TDATransaction Members` topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to determine if the</td>
</tr>
</tbody>
</table>
5.11.1.19.2.1  Active Property

Used to determine if the transaction is active.

Class
TDATransaction

Syntax

property Active: boolean;

Remarks
Indicates whether the transaction is active. This property is read-only.

5.11.1.19.2.2  DefaultCloseAction Property

Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

Class
TDATransaction

Syntax
property DefaultCloseAction: TCRTransactionAction default taRollback;

Remarks

Use DefaultCloseAction to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>Commits the current transaction.</td>
</tr>
<tr>
<td>Rollback</td>
<td>Discards all modifications of data associated with the current transaction and ends the transaction.</td>
</tr>
<tr>
<td>StartTransaction</td>
<td>Begins a new transaction.</td>
</tr>
</tbody>
</table>

See Also

- TDATransaction Class
- TDATransaction Class Members
Syntax

```plaintext
procedure Commit; virtual;
```

Remarks
Call the Commit method to commit the current transaction. On commit server writes permanently all pending data updates associated with the current transaction to the database, and then finishes the transaction.

See Also
- `Rollback`
- `StartTransaction`

Class
`TDATransaction`

Syntax

```plaintext
procedure Rollback; virtual;
```

Remarks
Call Rollback to cancel all data modifications made within the current transaction to the database server, and finish the transaction.

See Also
- `Commit`
- `StartTransaction`
5.11.1.19.3.3  StartTransaction Method

Starts a new transaction.

Class

TDATransaction

Syntax

```pascal
procedure StartTransaction; virtual;
```

Remarks

Call the StartTransaction method to begin a new transaction against the database server. Before calling StartTransaction, an application should check the `Active` property. If `TDATransaction.Active` is True, indicating that a transaction is already in progress, a subsequent call to StartTransaction will raise EDatabaseError. An active transaction must be finished by call to `Commit` or `Rollback` before call to StartTransaction. Call to StartTransaction when connection is closed also will raise EDatabaseError.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until the application calls `Commit` to save the changes, or `Rollback` to cancel them.

See Also

- `Commit`
- `Rollback`

© 1997-2021 Devart. All Rights Reserved.

5.11.1.19.4  Events

Events of the `TDATransaction` class.

For a complete list of the `TDATransaction` class members, see the `TDATransaction Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnCommit</code></td>
<td>Occurs after the transaction has been successfully committed.</td>
</tr>
</tbody>
</table>
### OnCommit Event

Occurs after the transaction has been successfully committed.

**Class**  
**TDATransaction**

**Syntax**

```plaintext
property OnCommit: TNotifyEvent;
```

**Remarks**

The OnCommit event fires when the M:Devart.Dac.TDATransaction.Commit method is executed, just after the transaction is successfully committed. In order to respond to the M:Devart.Odac.TOraTransaction.CommitRetaining() method execution, the **OnCommitRetaining** event is used. When an error occurs during commit, the **OnError** event fires.

**See Also**

- **Commit**
- **OnError**

© 1997-2021 Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback
5.11.1.19.4.2  OnCommitRetaining Event

Occurs after CommitRetaining has been executed.

Class
TDATransaction

Syntax

property OnCommitRetaining: TNotifyEvent;

Remarks
The OnCommitRetaining event fires when the CommitRetaining method is executed, just after the transaction is successfully committed. In order to respond to the M:Devart.Dac.TDATransaction.Commit method execution, the OnCommit event is used. When an error occurs during commit, the OnError event fired.

See Also
- Commit
- OnCommit
- OnError

5.11.1.19.4.3  OnError Event

Used to process errors that occur during executing a transaction.

Class
TDATransaction

Syntax

property OnError: TDATransactionErrorEvent;

Remarks
Add a handler to the OnError event to process errors that occur during executing a transaction and save point control statements such as Commit, Rollback, TOraTransaction.Savepoint, TOraTransaction.RollbackToSavepoint, and others. Check the E parameter to get the error code.

See Also
- Commit
- Rollback
- StartTransaction

5.11.1.19.4.4 OnRollback Event

Occurs after the transaction has been successfully rolled back.

Class
TDATransaction

Syntax

```pascal
property OnRollback: TNotifyEvent;
```

Remarks
The OnRollback event fires when the M:Devart.Dac.TDATransaction.Rollback method is executed, just after the transaction is successfully rolled back. In order to respond to the M:Devart.Odac.TOraTransaction.RollbackRetaining() method execution, the OnRollbackRetaining event is used.

When an error occurs during rollback, the OnError event fired.

See Also
- Rollback
- OnError
5.11.1.19.4.5 OnRollbackRetaining Event

Occurs after RollbackRetaining has been executed.

Class
TDATransaction

Syntax

property OnRollbackRetaining: TNotifyEvent;

Remarks
The OnRollbackRetaining event fires when the RollbackRetaining method is executed, just after the transaction is successfully rolled back. In order to respond to the Rollback method execution, the OnRollback event is used. When an error occurs during rollback, the OnError event fired.

See Also
• Rollback
• OnRollback
• OnError

© 1997-2021
Devart. All Rights Reserved.

5.11.1.20 TMacro Class

Object that represents the value of a macro.

For a list of all members of this type, see TMacro members.

Unit
DBAccess

Syntax

TMacro = class(TCollectionItem);

Remarks
TMacro object represents the value of a macro. Macro is a variable that holds string value. You just insert &MacroName in a SQL query text and change the value of macro by the
Macro property editor at design time or the Value property at run time. At the time of opening query macro is replaced by its value.

If by any reason it is not convenient for you to use the ' & ' symbol as a character of macro replacement, change the value of the MacroChar variable.

See Also
- TMacros

© 1997-2021
Devart. All Rights Reserved.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to determine if the macro should be expanded.</td>
</tr>
<tr>
<td>AsDateTime</td>
<td>Used to set the TDateTime value to a macro.</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Used to set the float value to a macro.</td>
</tr>
<tr>
<td>AsInteger</td>
<td>Used to set the integer value to a macro.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to assign the string value to a macro.</td>
</tr>
<tr>
<td>Name</td>
<td>Used to identify a particular macro.</td>
</tr>
<tr>
<td>Value</td>
<td>Used to set the value to a macro.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

Properties of the TMacro class.

For a complete list of the TMacro class members, see the TMacro Members topic.
Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsDateTime</td>
<td>Used to set the TDateTime value to a macro.</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Used to set the float value to a macro.</td>
</tr>
<tr>
<td>AsInteger</td>
<td>Used to set the integer value to a macro.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to assign the string value to a macro.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to determine if the macro should be expanded.</td>
</tr>
<tr>
<td>Name</td>
<td>Used to identify a particular macro.</td>
</tr>
<tr>
<td>Value</td>
<td>Used to set the value to a macro.</td>
</tr>
</tbody>
</table>

See Also
- TMacro Class
- TMacro Class Members

5.11.1.20.2.1  Active Property

Used to determine if the macro should be expanded.

Class

TMacro

Syntax

```property Active: boolean default True;```

Remarks
When set to True, the macro will be expanded, otherwise macro definition is replaced by null string. You can use the Active property to modify the SQL property.

The default value is True.

Example

OraQuery.SQL.Text := 'SELECT * FROM Dept WHERE DeptNo > 20 &Cond1';
OraQuery.Macros[0].Value := 'and DName is NULL';
OraQuery.Macros[0].Active := False;

5.11.1.20.2.2  AsDateTime Property

Used to set the TDateTime value to a macro.

Class

TMacro

Syntax

property AsDateTime: TDateTime;

Remarks

Use the AsDateTime property to set the TDateTime value to a macro.

5.11.1.20.2.3  AsFloat Property

Used to set the float value to a macro.

Class

TMacro

Syntax

property AsFloat: double;

Remarks
5.11.1.20.2.4 AsInteger Property

Used to set the integer value to a macro.

Class

TMacro

Syntax

```pascal
property AsInteger: integer;
```

Remarks

Use the AsInteger property to set the integer value to a macro.

5.11.1.20.2.5 AsString Property

Used to assign the string value to a macro.

Class

TMacro

Syntax

```pascal
property AsString: string;
```

Remarks

Use the AsString property to assign the string value to a macro. Read the AsString property to determine the value of macro represented as a string.
5.11.1.20.2.6 Name Property

Used to identify a particular macro.

Class

TMacro

Syntax

property Name: string;

Remarks

Use the Name property to identify a particular macro.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.20.2.7 Value Property

Used to set the value to a macro.

Class

TMacro

Syntax

property Value: string;

Remarks

Use the Value property to set the value to a macro.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.21 TMacros Class

Controls a list of TMacro objects for the TCustomDASQL.Macros or TCustomDADataset components.

For a list of all members of this type, see TMacros members.

Unit
**DBAccess**

**Syntax**

```pascal
TMacros = class(TCollection);
```

**Remarks**

Use TMacros to manage a list of TMacro objects for the TCustomDASQL or TCustomDADataset components.

**See Also**

- **TMacro**

© 1997-2021 Devart. All Rights Reserved.

---

**TMacros** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to iterate through all the macros parameters.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignValues</td>
<td>Copies the macros values and properties from the specified source.</td>
</tr>
<tr>
<td>Expand</td>
<td>Changes the macros in the passed SQL statement to their values.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares itself with another TMacro object.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Used to search for a macro with the specified name.</td>
</tr>
<tr>
<td>Scan</td>
<td>Creates a macros from the</td>
</tr>
</tbody>
</table>
5.11.1.21.2 Properties

Properties of the TMacros class.

For a complete list of the TMacros class members, see the TMacros Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to iterate through all the macros parameters.</td>
</tr>
</tbody>
</table>

See Also
- TMacros Class
- TMacros Class Members

5.11.1.21.2.1 Items Property(Indexer)

Used to iterate through all the macros parameters.

Class

TMacros

Syntax

```property Items[Index: integer]: TMacro; default;```

Parameters

Index

Holds the index in the range 0..Count - 1.

Remarks

Use the Items property to iterate through all macros parameters. Index identifies the index in the range 0..Count - 1.
5.11.1.21.3 Methods

Methods of the TMacros class.

For a complete list of the TMacros class members, see the TMacros Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignValues</td>
<td>Copies the macros values and properties from the specified source.</td>
</tr>
<tr>
<td>Expand</td>
<td>Changes the macros in the passed SQL statement to their values.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>IsEqual</td>
<td>Compares itself with another TMacro object.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Used to search for a macro with the specified name.</td>
</tr>
<tr>
<td>Scan</td>
<td>Creates a macros from the passed SQL statement.</td>
</tr>
</tbody>
</table>

See Also
- TMacros Class
- TMacros Class Members

5.11.1.21.3.1 AssignValues Method

Copies the macros values and properties from the specified source.

Class
TMacros

Syntax
procedure AssignValues(Value: TMacros);

Parameters

Value
Holds the source to copy the macros values and properties from.

Remarks

The Assign method copies the macros values and properties from the specified source. Macros are not recreated. Only the values of macros with matching names are assigned.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.21.3.2 Expand Method

Changes the macros in the passed SQL statement to their values.

Class

TMacros

Syntax

procedure Expand(var SQL: string);

Parameters

SQL
Holds the passed SQL statement.

Remarks

Call the Expand method to change the macros in the passed SQL statement to their values.

© 1997-2021
Devart. All Rights Reserved.

5.11.1.21.3.3 FindMacro Method

Finds a macro with the specified name.

Class

TMacros

Syntax
function FindMacro(const Value: string): TMacro;

Parameters

Value
Holds the value of a macro to search for.

Return Value
TMacro object if a match is found, nil otherwise.

Remarks
Call the FindMacro method to find a macro with the specified name. If a match is found, FindMacro returns the macro. Otherwise, it returns nil. Use this method instead of a direct reference to the Items property to avoid depending on the order of the items.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.21.3.4 IsEqual Method

Compares itself with another TMacro object.

Class
TMacros

Syntax

function IsEqual(Value: TMacros): boolean;

Parameters

Value
Holds the values of TMacro objects.

Return Value
True, if the number of TMacro objects and the values of all TMacro objects are equal.

Remarks
Call the IsEqual method to compare itself with another TMacro object. Returns True if the number of TMacro objects and the values of all TMacro objects are equal.

© 1997-2021 Devart. All Rights Reserved.
5.11.1.21.3.5  MacroByName Method

Used to search for a macro with the specified name.

Class

TMacros

Syntax

function MacroByName(const Value: string): TMacro;

Parameters

Value

Holds a name of the macro to search for.

Return Value

TMacro object, if a macro with specified name was found.

Remarks

Call the MacroByName method to find a Macro with the name passed in Value. If a match is found, MacroByName returns the Macro. Otherwise, an exception is raised. Use this method instead of a direct reference to the Items property to avoid depending on the order of the items.

To locate a macro by name without raising an exception if the parameter is not found, use the FindMacro method.

To set a value to a macro, use the TMacro.Value property.

5.11.1.21.3.6  Scan Method

Creates a macros from the passed SQL statement.

Class

TMacros

Syntax

procedure Scan(const SQL: string);
SQL
Holds the passed SQL statement.

Remarks
Call the Scan method to create a macros from the passed SQL statement. On that all existing TMacro objects are cleared.

5.11.1.22 TPoolingOptions Class

This class allows setting up the behaviour of the connection pool.

For a list of all members of this type, see TPoolingOptions members.

Unit
DBAccess

Syntax

TPoolingOptions = class(TPersistent);

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionLifetime</td>
<td>Used to specify the maximum time during which an opened connection can be used by connection pool.</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Used to specify the maximum number of connections that can be opened in connection pool.</td>
</tr>
<tr>
<td>MinPoolSize</td>
<td>Used to specify the minimum number of</td>
</tr>
</tbody>
</table>
Properties of the `TPoolingOptions` class.

For a complete list of the `TPoolingOptions` class members, see the `TPoolingOptions Members` topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConnectionLifetime</strong></td>
<td>Used to specify the maximum time during which an opened connection can be used by connection pool.</td>
</tr>
<tr>
<td><strong>MaxPoolSize</strong></td>
<td>Used to specify the maximum number of connections that can be opened in connection pool.</td>
</tr>
<tr>
<td><strong>MinPoolSize</strong></td>
<td>Used to specify the minimum number of connections that can be opened in the connection pool.</td>
</tr>
<tr>
<td><strong>Validate</strong></td>
<td>Used for a connection to be validated when it is returned from the pool.</td>
</tr>
</tbody>
</table>

See Also
- `TPoolingOptions Class`
- `TPoolingOptions Class Members`
5.11.1.22.2.1 ConnectionLifetime Property

Used to specify the maximum time during which an opened connection can be used by connection pool.

Class
TPoolingOptions

Syntax

```
property ConnectionLifetime: integer default DefValConnectionLifetime;
```

Remarks
Use the ConnectionLifeTime property to specify the maximum time during which an opened connection can be used by connection pool. Measured in milliseconds. Pool deletes connections with exceeded connection lifetime when TCustomDACConnection is about to close. If the ConnectionLifetime property is set to 0 (by default), then the lifetime of connection is infinity. ConnectionLifetime concerns only inactive connections in the pool.

© 1997-2021 Devart. All Rights Reserved.

5.11.1.22.2.2 MaxPoolSize Property

Used to specify the maximum number of connections that can be opened in connection pool.

Class
TPoolingOptions

Syntax

```
property MaxPoolSize: integer default DefValMaxPoolSize;
```

Remarks
Specifies the maximum number of connections that can be opened in connection pool. Once this value is reached, no more connections are opened. The valid values are 1 and higher.

© 1997-2021 Devart. All Rights Reserved.
5.11.1.22.2.3  MinPoolSize Property

Used to specify the minimum number of connections that can be opened in the connection pool.

Class
TPoolingOptions

Syntax

```plaintext
property MinPoolSize: integer default DefValMinPoolSize;
```

Remarks

Use the MinPoolSize property to specify the minimum number of connections that can be opened in the connection pool.

5.11.1.22.2.4  Validate Property

Used for a connection to be validated when it is returned from the pool.

Class
TPoolingOptions

Syntax

```plaintext
property Validate: boolean default DefValValidate;
```

Remarks

If the Validate property is set to True, connection will be validated when it is returned from the pool. By default this option is set to False and pool does not validate connection when it is returned to be used by a TCustomDACConnection component.
5.11.1.23 TSmartFetchOptions Class

Smart fetch options are used to set up the behavior of the SmartFetch mode.

For a list of all members of this type, see TSmartFetchOptions members.

Unit

DBAccess

Syntax

```plaintext
TSmartFetchOptions = class(TPersistent);
```

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Sets SmartFetch mode enabled or not.</td>
</tr>
<tr>
<td>LiveBlock</td>
<td>Used to minimize memory consumption.</td>
</tr>
<tr>
<td>PrefetchedFields</td>
<td>List of fields additional to key fields that will be read from the database on dataset open.</td>
</tr>
<tr>
<td>SQLGetKeyValues</td>
<td>SQL query for the read key and prefetched fields from the database.</td>
</tr>
</tbody>
</table>

For a complete list of the TSmartFetchOptions class members, see the TSmartFetchOptions Members topic.
**5.11.1.23.2.1  Enabled Property**

Sets SmartFetch mode enabled or not.

**Class**

**TSmartFetchOptions**

**Syntax**

```pascal
property Enabled: Boolean default False;
```

**5.11.1.23.2.2  LiveBlock Property**

Used to minimize memory consumption.

**Class**

**TSmartFetchOptions**

© 2021 Devart
5.11.1.23.2.3  PrefetchedFields Property

List of fields additional to key fields that will be read from the database on dataset open.

Class

TSmartFetchOptions

Syntax

property PrefetchedFields: string;

Remarks

If you are going to use locate, filter or sort by some fields, then these fields should be added to the prefetched fields list to avoid excessive reading from the database.

© 1997-2021
Devar. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback

5.11.1.23.4  SQLGetKeyValues Property

SQL query for the read key and prefetched fields from the database.

Class

TSmartFetchOptions

© 1997-2021 Devart. All Rights Reserved.
Request Support  
DAC Forum  
Provide Feedback

© 1997-2021 Devart. All Rights Reserved.
Request Support  
DAC Forum  
Provide Feedback
Syntax

```pascal
property SQLGetKeyValues: TStrings;
```

Remarks

SQLGetKeyValues is used when the basic SQL query is complex and the query for reading the key and prefetched fields can't be generated automatically.

5.11.2 Types

Types in the **DBAccess** unit.

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAfterExecuteEvent</td>
<td>This type is used for the TCustomDADataset.AfterExecute and</td>
</tr>
<tr>
<td></td>
<td>TCustomDASQL.AfterExecute events.</td>
</tr>
<tr>
<td>TAfterFetchEvent</td>
<td>This type is used for the TCustomDADataset.AfterFetch event.</td>
</tr>
<tr>
<td>TBeforeFetchEvent</td>
<td>This type is used for the TCustomDADataset.Before Fetch event.</td>
</tr>
<tr>
<td>TConnectionLostEvent</td>
<td>This type is used for the TCustomDACConnection.OnConnectionLost event.</td>
</tr>
<tr>
<td>TDACConnectionString</td>
<td>This type is used for the TCustomDACConnectionString.</td>
</tr>
<tr>
<td>TDACConnectionErrorEvent</td>
<td>This type is used for the TCustomDACConnection.OnError event.</td>
</tr>
<tr>
<td>TDATransactionErrorEvent</td>
<td>This type is used for the TDATransaction.OnError event.</td>
</tr>
<tr>
<td>TRefreshOptions</td>
<td>Represents the set of TRefreshOption.</td>
</tr>
<tr>
<td>TUpdateExecuteEvent</td>
<td>This type is used for the TCustomDADataset.AfterUpdateExecute and</td>
</tr>
</tbody>
</table>
5.11.2.1 **TAfterExecuteEvent Procedure Reference**

This type is used for the `TCustomDADataset.AfterExecute` and `TCustomDASQL.AfterExecute` events.

**Unit**

`DBAccess`

**Syntax**

```plaintext
TAfterExecuteEvent = procedure (Sender: TObject; Result: boolean)
of object;
```

**Parameters**

- **Sender**
  - An object that raised the event.
- **Result**
  - The result is True if SQL statement is executed successfully. False otherwise.

5.11.2.2 **TAfterFetchEvent Procedure Reference**

This type is used for the `TCustomDADataset.AfterFetch` event.

**Unit**

`DBAccess`

**Syntax**

```plaintext
TAfterFetchEvent = procedure (DataSet: TCustomDADataset)
of object;
```

**Parameters**

- **DataSet**
  - Holds the TCustomDADataset descendant to synchronize the record position with.
5.11.2.3 TBeforeFetchEvent Procedure Reference

This type is used for the TCustomDADataSet.BeforeFetch event.

Unit

DBAccess

Syntax

TBeforeFetchEvent = procedure (DataSet: TCustomDADataSet; var Cancel: boolean) of object;

Parameters

DataSet

Holds the TCustomDADataSet descendant to synchronize the record position with.

Cancel

True, if the current fetch operation should be aborted.

5.11.2.4 TConnectionLostEvent Procedure Reference

This type is used for the TCustomDACollection.ConnectionLost event.

Unit

DBAccess

Syntax

TConnectionLostEvent = procedure (Sender: TObject; Component: TComponent; ConnLostCause: TConnLostCause; var RetryMode: TRetryMode) of object;

Parameters

Sender

An object that raised the event.

Component

ConnLostCause

The reason of the connection loss.
RetryMode

The application behavior when connection is lost.

5.11.2.5  TDAConnectionErrorEvent Procedure Reference

This type is used for the TCustomDACConnection.OnError event.

Unit

DBAccess

Syntax

TDAConnectionErrorEvent = procedure (Sender: TObject; E: EDAError; var Fail: boolean) of object;

Parameters

Sender

An object that raised the event.

E

The error information.

Fail

False, if an error dialog should be prevented from being displayed and EAbort exception should be raised to cancel current operation.

5.11.2.6  TDATransactionErrorEvent Procedure Reference

This type is used for the TDATransaction.OnError event.

Unit

DBAccess

Syntax

TDATransactionErrorEvent = procedure (Sender: TObject; E: EDAError; var Fail: boolean) of object;

Parameters

Sender
An object that raised the event.

\( E \)

The error code.

\( \text{Fail} \)

False, if an error dialog should be prevented from being displayed and EAbort exception to cancel the current operation should be raised.

### 5.11.2.7 TRefreshOptions Set

Represents the set of TRefreshOption.

**Unit**

`DBAccess`

**Syntax**

\[
\text{TRefreshOptions} = \textit{set of} \; \text{TRefreshOption};
\]

### 5.11.2.8 TUpdateExecuteEvent Procedure Reference

This type is used for the TCustomDADataSet.AfterUpdateExecute and TCustomDADataSet.BeforeUpdateExecute events.

**Unit**

`DBAccess`

**Syntax**

\[
\text{TUpdateExecuteEvent} = \textit{procedure} \; (\text{Sender: TDataSet}; \; \text{StatementTypes: TStatementTypes}; \; \text{Params: TDParams}) \; \text{of object};
\]

**Parameters**

\( \text{Sender} \)

An object that raised the event.

\( \text{StatementTypes} \)

Holds the type of the SQL statement being executed.

\( \text{Params} \)
Holds the parameters with which the SQL statement will be executed.

Enumerations

Enumerations in the **DBAccess** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TCheckMode</strong></td>
<td>Specifies the action to take when another user makes modifications to a record.</td>
</tr>
<tr>
<td><strong>TLabelSet</strong></td>
<td>Sets the language of labels in the connect dialog.</td>
</tr>
<tr>
<td><strong>TLockMode</strong></td>
<td>Specifies the lock mode.</td>
</tr>
<tr>
<td><strong>TRefreshOption</strong></td>
<td>Indicates when the editing record will be refreshed.</td>
</tr>
<tr>
<td><strong>TRetryMode</strong></td>
<td>Specifies the application behavior when connection is lost.</td>
</tr>
</tbody>
</table>

**5.11.3** **TCheckMode** Enumeration

Specifies the action to take when another user makes modifications to a record.

Unit

**DBAccess**

Syntax

```
TCheckMode = (cmNone, cmException, cmRefresh);
```

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmException</td>
<td>If a record was changed, TOraDataSet raises an exception.</td>
</tr>
<tr>
<td>cmNone</td>
<td>No check is performed. The default value.</td>
</tr>
</tbody>
</table>
5.11.3.2 TLabelSet Enumeration

Sets the language of labels in the connect dialog.

Unit

DBAccess

Syntax

TLabelSet = (lsCustom, lsEnglish, lsFrench, lsGerman, lsItalian,
lsPolish, lsPortuguese, lsRussian, lsSpanish);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsCustom</td>
<td>Set the language of labels in the connect dialog manually.</td>
</tr>
<tr>
<td>lsEnglish</td>
<td>Set English as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsFrench</td>
<td>Set French as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsGerman</td>
<td>Set German as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsItalian</td>
<td>Set Italian as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsPolish</td>
<td>Set Polish as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsPortuguese</td>
<td>Set Portuguese as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsRussian</td>
<td>Set Russian as the language of labels in the connect dialog.</td>
</tr>
<tr>
<td>lsSpanish</td>
<td>Set Spanish as the language of labels in the connect dialog.</td>
</tr>
</tbody>
</table>

5.11.3.3 TLockMode Enumeration

Specifies the lock mode.

Unit

DBAccess

Syntax
TLockMode = (lmNone);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmLockDelayed</td>
<td>Locking occurs when the user posts an edited record, then the lock is released. Locking is done by the RefreshRecord method. Corresponds to optimistic locking.</td>
</tr>
<tr>
<td>lmLockImmediate</td>
<td>Locking occurs when the user starts editing a record. The lock is released after the user has posted or canceled the changes. Corresponds to pessimistic locking.</td>
</tr>
<tr>
<td>lmNone</td>
<td>No locking occurs. This mode should only be used in single user applications. The default value.</td>
</tr>
</tbody>
</table>

5.11.3.4 TRefreshOption Enumeration

Indicates when the editing record will be refreshed.

Unit

DBAccess

Syntax

TRefreshOption = (roAfterInsert, roAfterUpdate, roBeforeEdit);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>roAfterInsert</td>
<td>Refresh is performed after inserting.</td>
</tr>
<tr>
<td>roAfterUpdate</td>
<td>Refresh is performed after updating.</td>
</tr>
<tr>
<td>roBeforeEdit</td>
<td>Refresh is performed by Edit method.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.  
Request Support       DAC Forum       Provide Feedback
5.11.3.5 TRetryMode Enumeration

Specifies the application behavior when connection is lost.

Unit

**DBAccess**

Syntax

```plaintext
TRetryMode = (rmRaise, rmReconnect, rmReconnectExecute);
```

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmRaise</td>
<td>An exception is raised.</td>
</tr>
<tr>
<td>rmReconnect</td>
<td>Reconnect is performed and then exception is raised.</td>
</tr>
<tr>
<td>rmReconnectExecute</td>
<td>Reconnect is performed and abortive operation is reexecuted. Exception is not raised.</td>
</tr>
</tbody>
</table>

5.11.4 Variables

Variables in the **DBAccess** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQLOldBehavior</td>
<td>After assigning SQL text and modifying it by <strong>AddWhere</strong>, <strong>DeleteWhere</strong>, and <strong>SetOrderBy</strong>, all subsequent changes of the SQL property will not be reflected in the BaseSQL property.</td>
</tr>
<tr>
<td>ChangeCursor</td>
<td>When set to True allows data access components to change screen cursor for the execution time.</td>
</tr>
<tr>
<td>SQLGeneratorCompatibility</td>
<td>The value of the <strong>TCustomDADataset.BaseSQL</strong> property is used to</td>
</tr>
</tbody>
</table>
### 5.11.4.1 BaseSQLOldBehavior Variable

After assigning SQL text and modifying it by AddWhere, DeleteWhere, and SetOrderBy, all subsequent changes of the SQL property will not be reflected in the BaseSQL property.

**Unit**

*DBAccess*

**Syntax**

```
BaseSQLOldBehavior: boolean = False;
```

**Remarks**

The **BaseSQL** property is similar to the SQL property, but it does not store changes made by the AddWhere, DeleteWhere, and SetOrderBy methods. After assigning SQL text and modifying it by one of these methods, all subsequent changes of the SQL property will not be reflected in the BaseSQL property. This behavior was changed in ODAC 5.55.1.26. To restore old behavior, set the BaseSQLOldBehavior variable to True.

### 5.11.4.2 ChangeCursor Variable

When set to True allows data access components to change screen cursor for the execution time.

**Unit**

*DBAccess*

**Syntax**
5.11.4.3 SQLGeneratorCompatibility Variable

The value of the TCustomDADataSet.BaseSQL property is used to complete the refresh SQL statement, if the manually assigned TCustomDAUpdateSQL.RefreshSQL property contains only WHERE clause.

Unit

DBAccess

Syntax

SQLGeneratorCompatibility: boolean = False;

Remarks

If the manually assigned TCustomDAUpdateSQL.RefreshSQL property contains only WHERE clause, ODAC uses the value of the TCustomDADataSet.BaseSQL property to complete the refresh SQL statement. In this situation all modifications applied to the SELECT query by functions TCustomDADataSet.AddWhere, TCustomDADataSet.DeleteWhere are not taken into account. This behavior was changed in ODAC 6.00.0.4. To restore the old behavior, set the BaseSQLOldBehavior variable to True.

5.12 MemData

This unit contains classes for storing data in memory.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAttribute</td>
<td>Holds the description of object attributes.</td>
</tr>
<tr>
<td>TBlob</td>
<td>Holds large object value for field and parameter dtBlob, dtMemo data types.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCompressedBlob</td>
<td>Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.</td>
</tr>
<tr>
<td>TDBObject</td>
<td>A base class for classes that work with user-defined data types that have attributes.</td>
</tr>
<tr>
<td>TMemData</td>
<td>Implements in-memory database.</td>
</tr>
<tr>
<td>TObjectType</td>
<td>Holds description object type and its attributes.</td>
</tr>
<tr>
<td>TSharedObject</td>
<td>A base class that allows to simplify memory management for object referenced by several other objects.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLocateExOptions</td>
<td>Represents the set of TLocateExOption.</td>
</tr>
<tr>
<td>TUpdateRecKinds</td>
<td>Represents the set of TUpdateRecKind.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCompressBlobMode</td>
<td>Specifies when the values should be compressed and the way they should be stored.</td>
</tr>
<tr>
<td>TConnLostCause</td>
<td>Specifies the cause of the connection loss.</td>
</tr>
<tr>
<td>TDANumericType</td>
<td>Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.</td>
</tr>
<tr>
<td>TLocateExOption</td>
<td>Allows to set additional search parameters which will be used by the LocateEx method.</td>
</tr>
<tr>
<td>TSortType</td>
<td>Specifies a sort type for...</td>
</tr>
</tbody>
</table>
**5.12.1 Classes**

Classes in the **MemData** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TAttribute</strong></td>
<td>Holds the description of object attributes.</td>
</tr>
<tr>
<td><strong>TBlob</strong></td>
<td>Holds large object value for field and parameter dtBlob, dtMemo data types.</td>
</tr>
<tr>
<td><strong>TCompressedBlob</strong></td>
<td>Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.</td>
</tr>
<tr>
<td><strong>TDBObject</strong></td>
<td>A base class for classes that work with user-defined data types that have attributes.</td>
</tr>
<tr>
<td><strong>TMemData</strong></td>
<td>Implements in-memory database.</td>
</tr>
<tr>
<td><strong>TObjectType</strong></td>
<td>Holds description object type and its attributes.</td>
</tr>
<tr>
<td><strong>TSharedObject</strong></td>
<td>A base class that allows to simplify memory management for object referenced by several other objects.</td>
</tr>
</tbody>
</table>
5.12.1.1 **TAttribute Class**

Holds the description of object attributes.

For a list of all members of this type, see TAttribute members.

**Unit**

MemData

**Syntax**

```pascal
TAttribute = class(System.TObject);
```

**Remarks**

The TAttribute class holds the description of object attributes. You can use TObjectType.Attributes to access individual attributes. To create TAttribute objects call the TOraType.Describe method. It is called implicitly when ODAC fetches Oracle objects.

**See Also**

- TObjectType
- TOraType

© 1997-2021 Devart. All Rights Reserved.

### 5.12.1.1.1 Members

**TAttribute** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeNo</td>
<td>Returns an attribute's ordinal position in object.</td>
</tr>
<tr>
<td>DataSize</td>
<td>Returns the size of an attribute value in internal representation.</td>
</tr>
<tr>
<td>DataType</td>
<td>Returns the type of data that was assigned to the Attribute.</td>
</tr>
<tr>
<td>Length</td>
<td>Returns the length of the string for dtString attribute</td>
</tr>
</tbody>
</table>
5.12.1.1.2 Properties

Properties of the `TAttribute` class.

For a complete list of the `TAttribute` class members, see the `TAttribute Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeNo</td>
<td>Returns an attribute’s ordinal position in object.</td>
</tr>
<tr>
<td>DataSize</td>
<td>Returns the size of an attribute value in internal representation.</td>
</tr>
<tr>
<td>DataType</td>
<td>Returns the type of data that was assigned to the Attribute.</td>
</tr>
<tr>
<td>Length</td>
<td>Returns the length of the string for dtString attribute and precision for dtInteger and dtFloat attribute.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>Returns a TObjectType object for an object attribute.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Offset</strong></td>
<td>Returns an offset of the attribute value in internal representation.</td>
</tr>
<tr>
<td><strong>Owner</strong></td>
<td>Indicates TObjectType that uses the attribute to represent one of its attributes.</td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td>Returns the scale of dtFloat and dtInteger attributes.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Returns the size of an attribute value in external representation.</td>
</tr>
</tbody>
</table>

**See Also**
- [TAttribute Class](#)
- [TAttribute Class Members](#)

5.12.1.2.1 AttributeNo Property

Returns an attribute's ordinal position in object.

**Class**

[TAttribute](#)

**Syntax**

```delphi
property AttributeNo: Word;
```

**Remarks**

Use the AttributeNo property to learn an attribute's ordinal position in object, where 1 is the first field.

**See Also**
- [TObjectType.Attributes](#)
5.12.1.2.2 DataSize Property

Returns the size of an attribute value in internal representation.

Class
TAttribute

Syntax

```pascal
property DataSize: Integer;
```

Remarks
Use the DataSize property to learn the size of an attribute value in internal representation.

For example:

<table>
<thead>
<tr>
<th>Type</th>
<th>Size (OCIDate)</th>
<th>Size (OCINumber)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtDate</td>
<td>17 (sizeof(OCIDate))</td>
<td></td>
</tr>
<tr>
<td>dtFloat</td>
<td>22 (sizeof(OCINumber))</td>
<td></td>
</tr>
<tr>
<td>dtInteger</td>
<td>22 (sizeof(OCINumber))</td>
<td></td>
</tr>
</tbody>
</table>

See Also
- TOraObject.Instance
- Offset

---

5.12.1.2.3 DataType Property

Returns the type of data that was assigned to the Attribute.

Class
TAttribute

Syntax

```pascal
property DataType: Word;
```

Remarks
Use the DataType property to discover the type of data that was assigned to the Attribute.

Possible values: dtDate, dtFloat, dtInteger, dtString, dtObject.

© 1997-2021 Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback
5.12.1.2.4 Length Property

Returns the length of the string for `dtString` attribute and precision for `dtInteger` and `dtFloat` attribute.

Class

`TAtribute`

Syntax

```pascal
property Length: Word;
```

Remarks

Use the `Length` property to learn the length of the string for `dtString` attribute and precision for `dtInteger` and `dtFloat` attribute.

See Also

- `Scale`

5.12.1.2.5 ObjectType Property

Returns a `TObjectType` object for an object attribute.

Class

`TAtribute`

Syntax

```pascal
property ObjectType: TObjectType;
```

Remarks

Use the `ObjectType` property to return a `TObjectType` object for an object attribute.
5.12.1.1.2.6 Offset Property

Returns an offset of the attribute value in internal representation.

Class

TAttribute

Syntax

property Offset: Integer;

Remarks

Use the DataSize property to learn an offset of the attribute value in internal representation.

See Also

• TOraObject.Instance
• DataSize

5.12.1.1.2.7 Owner Property

Indicates TObjectType that uses the attribute to represent one of its attributes.

Class

TAttribute

Syntax

property Owner: TObjectType;

Remarks

Check the value of the Owner property to determine TObjectType that uses the attribute to represent one of its attributes. Applications should not assign the Owner property directly. It is assigned automatically when attribute is created from TOraType.Describe.
5.12.1.1.2.8 Scale Property

Returns the scale of dtFloat and dtInteger attributes.

Class

TAttribute

Syntax

```plaintext
property Scale: Word;
```

Remarks

Use the Scale property to learn the scale of dtFloat and dtInteger attributes.

See Also

• Length

5.12.1.1.2.9 Size Property

Returns the size of an attribute value in external representation.

Class

TAttribute

Syntax

```plaintext
property Size: Integer;
```

Remarks

Read Size to learn the size of an attribute value in external representation.

For example:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dtDate</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>(sizeof(TDateTime))</td>
</tr>
<tr>
<td>dtFloat</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>(sizeof(Double))</td>
</tr>
<tr>
<td>dtInteger</td>
<td>4</td>
</tr>
</tbody>
</table>
5.12.1.2 TBlob Class

Holds large object value for field and parameter dtBlob, dtMemo data types.

For a list of all members of this type, see TBlob members.

Unit

MemData

Syntax

TBlob = class(TSharedObject);

Remarks

Object TBlob holds large object value for the field and parameter dtBlob, dtMemo, dtWideMemo data types.

Inheritance Hierarchy

TSharedObject

TBlob
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsString</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td>As WideString</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td>IsUnicode</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
<tr>
<td>RefCount</td>
<td>(inherited from TSharedObject) Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>Size</td>
<td>Used to learn the size of the TBlob value in bytes.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>(inherited from TSharedObject) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Assign</td>
<td>Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td>Clear</td>
<td>Deletes the current value in TBlob object.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td>Read</td>
<td>Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
<tr>
<td>Release</td>
<td>(inherited from TSharedObject) Decrements the reference count.</td>
</tr>
<tr>
<td>SaveToFile</td>
<td>Saves the contents of the TBlob object to a file.</td>
</tr>
<tr>
<td>SaveToStream</td>
<td>Copies the contents of a TBlob object to a stream.</td>
</tr>
<tr>
<td>Truncate</td>
<td>Sets new TBlob size and discards all data over it.</td>
</tr>
<tr>
<td>Write</td>
<td>Stores a raw sequence of bytes into a TBlob object.</td>
</tr>
</tbody>
</table>
5.12.1.2.2 Properties

Properties of the **TBlob** class.

For a complete list of the **TBlob** class members, see the [TBlob Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsString</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td>AsWideString</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td>IsUnicode</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
<tr>
<td>RefCount</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>Size</td>
<td>Used to learn the size of the TBlob value in bytes.</td>
</tr>
</tbody>
</table>

**See Also**
- [TBlob Class](#)
- [TBlob Class Members](#)

### 5.12.1.2.2.1 AsString Property

Used to manipulate BLOB value as string.

**Class**

**TBlob**

**Syntax**
**property** AsString: string;

**Remarks**
Use the AsString property to manipulate BLOB value as string.

**See Also**
- Assign
- AsWideString

© 1997-2021 Devart. All Rights Reserved.

5.12.1.2.2.2 AsWideString Property

Used to manipulate BLOB value as Unicode string.

**Class**
**TBlob**

**Syntax**
**property** AsWideString: string;

**Remarks**
Use the AsWideString property to manipulate BLOB value as Unicode string.

**See Also**
- Assign
- AsString

© 1997-2021 Devart. All Rights Reserved.

5.12.1.2.2.3 IsUnicode Property

Gives choice of making TBlob store and process data in Unicode format or not.

**Class**
**TBlob**
Syntax

```pascal
property IsUnicode: boolean;
```

Remarks

Set IsUnicode to True if you want TBlob to store and process data in Unicode format.

**Note:** changing this property raises an exception if TBlob is not empty.

```pascal
property Size: Cardinal;
```

Remarks

Use the Size property to find out the size of the TBlob value in bytes.

5.12.1.2.3 Methods

Methods of the **TBlob** class.

For a complete list of the **TBlob** class members, see the [TBlob Members](#) topic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>Assign</td>
<td>Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td>Clear</td>
<td>Deletes the current value in TBlob object.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td>Read</td>
<td>Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
<tr>
<td>Release</td>
<td>(inherited from TSharedObject) Decrements the reference count.</td>
</tr>
<tr>
<td>SaveToFile</td>
<td>Saves the contents of the TBlob object to a file.</td>
</tr>
<tr>
<td>SaveToStream</td>
<td>Copies the contents of a TBlob object to a stream.</td>
</tr>
<tr>
<td>Truncate</td>
<td>Sets new TBlob size and discards all data over it.</td>
</tr>
<tr>
<td>Write</td>
<td>Stores a raw sequence of bytes into a TBlob object.</td>
</tr>
</tbody>
</table>

### See Also
- TBlob Class
- TBlob Class Members

© 1997-2021 Devart. All Rights Reserved.

5.12.1.2.3.1  Assign Method

Sets BLOB value from another TBlob object.

### Class

**TBlob**

### Syntax

```pascal
procedure Assign(Source: TBlob);
```

### Parameters

**Source**

Holds the BLOB from which the value to the current object will be assigned.
Remarks
Call the Assign method to set BLOB value from another TBlob object.

See Also
- LoadFromStream
- AsString
- AsWideString

5.12.1.2.3.2 Clear Method

Deletes the current value in TBlob object.

Class
TBlob

Syntax

```
procedure Clear; virtual;
```

Remarks
Call the Clear method to delete the current value in TBlob object.

5.12.1.2.3.3 LoadFromFile Method

Loads the contents of a file into a TBlob object.

Class
TBlob

Syntax

```
procedure LoadFromFile(const FileName: string);
```

Parameters
FileName
Holds the name of the file from which the TBlob value is loaded.

Remarks
Call the LoadFromFile method to load the contents of a file into a TBlob object. Specify the name of the file to load into the field as the value of the FileName parameter.

See Also
- SaveToFile

5.12.1.2.3.4 LoadFromStream Method
Copies the contents of a stream into the TBlob object.

Class
TBlob

Syntax

```pascal
procedure LoadFromStream(Stream: TStream); virtual;
```

Parameters

- Stream
  - Holds the specified stream from which the field's value is copied.

Remarks
Call the LoadFromStream method to copy the contents of a stream into the TBlob object. Specify the stream from which the field's value is copied as the value of the Stream parameter.

See Also
- SaveToStream
5.12.1.2.3.5 Read Method

Acquires a raw sequence of bytes from the data stored in TBlob.

**Class**

TBlob

**Syntax**

```pascal
function Read(Position: Cardinal; Count: Cardinal; Dest: IntPtr): Cardinal; virtual;
```

**Parameters**

- **Position**
  Holds the starting point of the byte sequence.

- **Count**
  Holds the size of the sequence in bytes.

- **Dest**
  Holds a pointer to the memory area where to store the sequence.

**Return Value**

Actually read byte count if the sequence crosses object size limit.

**Remarks**

Call the Read method to acquire a raw sequence of bytes from the data stored in TBlob.

The Position parameter is the starting point of byte sequence which lasts Count number of bytes. The Dest parameter is a pointer to the memory area where to store the sequence.

If the sequence crosses object size limit, function will return actually read byte count.

**See Also**

- Write

© 1997-2021
Devart. All Rights Reserved.

5.12.1.2.3.6 SaveToFile Method

Saves the contents of the TBlob object to a file.

**Class**

TBlob
**Syntax**

```plaintext
procedure SaveToFile(const FileName: string);
```

**Parameters**

*FileName*

Holds a string that contains the name of the file.

**Remarks**

Call the `SaveToFile` method to save the contents of the TBlob object to a file. Specify the name of the file as the value of the `FileName` parameter.

**See Also**

- [LoadFromFile](#)

---

5.12.1.2.3.7  SaveToStream Method

Copies the contents of a TBlob object to a stream.

**Class**

*TBlob*

**Syntax**

```plaintext
procedure SaveToStream(Stream: TStream); virtual;
```

**Parameters**

*Stream*

Holds the name of the stream.

**Remarks**

Call the `SaveToStream` method to copy the contents of a TBlob object to a stream. Specify the name of the stream to which the field's value is saved as the value of the `Stream` parameter.

**See Also**

- [LoadFromStream](#)
5.12.1.2.3.8  Truncate Method

Sets new TBlob size and discards all data over it.

Class
TBlob

Syntax

```pascal
procedure Truncate(NewSize: Cardinal); virtual;
```

Parameters

NewSize
Holds the new size of TBlob.

Remarks

Call the Truncate method to set new TBlob size and discard all data over it. If NewSize is greater or equal TBlob.Size, it does nothing.

© 1997-2021
Devart. All Rights Reserved.

5.12.1.2.3.9  Write Method

Stores a raw sequence of bytes into a TBlob object.

Class
TBlob

Syntax

```pascal
procedure Write(Position: Cardinal; Count: Cardinal; Source: IntPtr); virtual;
```

Parameters

Position
Holds the starting point of the byte sequence.

Count
Holds the size of the sequence in bytes.

Source

Holds a pointer to a source memory area.

Remarks

Call the Write method to store a raw sequence of bytes into a TBlob object.

The Position parameter is the starting point of byte sequence which lasts Count number of bytes. The Source parameter is a pointer to a source memory area.

If the value of the Position parameter crosses current size limit of TBlob object, source data will be appended to the object data.

See Also

- Read

TCompressedBlob Class

Holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data.

For a list of all members of this type, see TCompressedBlob members.

Unit

MemData

Syntax

TCompressedBlob = class(TBlob);

Remarks

TCompressedBlob is a descendant of the TBlob class. It holds large object value for field and parameter dtBlob, dtMemo data types and can compress its data. For more information about using BLOB compression see TCustomDADataSet.Options.

Note: Internal compression functions are available in CodeGear Delphi 2007 for Win32, Borland Developer Studio 2006, Borland Delphi 2005, and Borland Delphi 7. To use BLOB compression under Borland Delphi 6 and Borland C++ Builder you should use your own compression functions. To use them set the CompressProc and UncompressProc variables declared in the MemUtils unit.
Example

```delphi
type
  TCompressProc = function(dest: IntPtr; destLen: IntPtr; const source: IntPtr; sourceLen: longint): longint;
  TUncompressProc = function(dest: IntPtr; destlen: IntPtr; source: IntPtr; sourceLne: longint): longint;

var
  CompressProc: TCompressProc;
  UncompressProc: TUncompressProc;
```

Inheritance Hierarchy

- TSharedObject
  - TBlob
    - TCompressedBlob

See Also

- TBlob
- TMemDataSet.GetBlob
- TCustomDADataSet.Options

© 1997-2021 Devart. All Rights Reserved.

5.12.1.3.1 Members

**TCompressedBlob** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsString</code> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td><code>AsWideString</code> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td><code>Compressed</code></td>
<td>Used to indicate if the Blob is compressed.</td>
</tr>
<tr>
<td><code>CompressedSize</code></td>
<td>Used to indicate compressed size of the Blob data.</td>
</tr>
<tr>
<td><code>IsUnicode</code> (inherited from <strong>TBlob</strong>)</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefCount</td>
<td>(inherited from <code>TSharedObject</code>) Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>Size</td>
<td>(inherited from <code>TBlob</code>) Used to learn the size of the TBlob value in bytes.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>(inherited from <code>TSharedObject</code>) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Assign</td>
<td>(inherited from <code>TBlob</code>) Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td>Clear</td>
<td>(inherited from <code>TBlob</code>) Deletes the current value in TBlob object.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>(inherited from <code>TBlob</code>) Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>(inherited from <code>TBlob</code>) Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td>Read</td>
<td>(inherited from <code>TBlob</code>) Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
<tr>
<td>Release</td>
<td>(inherited from <code>TSharedObject</code>) Decrements the reference count.</td>
</tr>
<tr>
<td>SaveToFile</td>
<td>(inherited from <code>TBlob</code>) Saves the contents of the TBlob object to a file.</td>
</tr>
<tr>
<td>SaveToStream</td>
<td>(inherited from <code>TBlob</code>) Copies the contents of a TBlob object to a stream.</td>
</tr>
<tr>
<td>Truncate</td>
<td>(inherited from <code>TBlob</code>) Sets new TBlob size and discards all data over it.</td>
</tr>
<tr>
<td>Write</td>
<td>(inherited from <code>TBlob</code>) Stores a raw sequence of bytes into a TBlob object.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devar. All Rights Reserved.
Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsString</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td><strong>AsWideString</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td><strong>Compressed</strong></td>
<td>Used to indicate if the Blob is compressed.</td>
</tr>
<tr>
<td><strong>CompressedSize</strong></td>
<td>Used to indicate compressed size of the Blob data.</td>
</tr>
<tr>
<td><strong>IsUnicode</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
<tr>
<td><strong>RefCount</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td><strong>Size</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to learn the size of the TBlob value in bytes.</td>
</tr>
</tbody>
</table>

See Also
- **TCompressedBlob Class**
- **TCompressedBlob Class Members**

Used to indicate if the Blob is compressed.

Class

**TCompressedBlob**

Syntax

```
property Compressed: boolean;
```
Remarks

Indicates whether the Blob is compressed. Set this property to True or False to compress or decompress the Blob.

5.12.1.3.2.2 CompressedSize Property

Used to indicate compressed size of the Blob data.

Class

TCompressedBlob

Syntax

property CompressedSize: Cardinal;

Remarks

Indicates compressed size of the Blob data.

5.12.1.4 TDBObject Class

A base class for classes that work with user-defined data types that have attributes.

For a list of all members of this type, see TDBObject members.

Unit

MemData

Syntax

TDBObject = class(TSharedObject);

Remarks

TDBObject is a base class for classes that work with user-defined data types that have attributes.
Inheritance Hierarchy

**TSharedObject**
**TDBObject**

© 1997-2021 Devart. All Rights Reserved.

### 5.12.1.4.1 Members

**TDBObject** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefCount</td>
<td>(inherited from <strong>TSharedObject</strong>)</td>
</tr>
<tr>
<td></td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>(inherited from <strong>TSharedObject</strong>)</td>
</tr>
<tr>
<td></td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Release</td>
<td>(inherited from <strong>TSharedObject</strong>)</td>
</tr>
<tr>
<td></td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

### 5.12.1.5 TMemData Class

Implements in-memory database.

For a list of all members of this type, see **TMemData** members.

Unit

**MemData**

Syntax
TMemData = class(TData);

Inheritance Hierarchy
TDdata
    TMemData

© 1997-2021 Devart. All Rights Reserved.

5.12.1.5.1 Members

TMemData class overview.

© 1997-2021 Devart. All Rights Reserved.

5.12.1.6 TObjectType Class

Holds description object type and its attributes.

For a list of all members of this type, see TObjectType members.

Unit
MemData

Syntax
TObjectType = class(TSharedObject);

Remarks
TObjectType holds description object type and its attributes. TObjectType is an ancestor for TOraType.

Inheritance Hierarchy
TSharedObject
    TObjectType

See Also
• TOraType
5.12.1.6.1 Members

**TObjectType** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeCount</td>
<td>Used to indicate the number of attributes of type.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Used to access separate attributes.</td>
</tr>
<tr>
<td>DataType</td>
<td>Used to indicate the type of object dtObject, dtArray or dtTable.</td>
</tr>
<tr>
<td>RefCount</td>
<td>(inherited from <strong>TSharedObject</strong>) Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>Size</td>
<td>Used to learn the size of an object instance.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>(inherited from <strong>TSharedObject</strong>) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>FindAttribute</td>
<td>Indicates whether a specified Attribute component is referenced in the TAttributes object.</td>
</tr>
<tr>
<td>Release</td>
<td>(inherited from <strong>TSharedObject</strong>) Decrements the reference count.</td>
</tr>
</tbody>
</table>

5.12.1.6.2 Properties

Properties of the **TObjectType** class.

For a complete list of the **TObjectType** class members, see the **TObjectType Members**
AttributeCount Property

Used to indicate the number of attributes of type.

Class

TObjectType

Syntax

```pascal
property AttributeCount: Integer;
```

Remarks

Use the AttributeCount property to determine the number of attributes of type.
5.12.1.6.2.2 Attributes Property(Indexer)

Used to access separate attributes.

Class

TObjectType

Syntax

```pascal
property Attributes[Index: integer]: TAttribute;
```

Parameters

Index

Holds the attribute's ordinal position.

Remarks

Use the Attributes property to access individual attributes. The value of the Index parameter corresponds to the AttributeNo property of TAttribute.

See Also

- TAttribute
- FindAttribute

© 1997-2021 Devart. All Rights Reserved.

5.12.1.6.2.3 DataType Property

Used to indicate the type of object dtObject, dtArray or dtTable.

Class

TObjectType

Syntax

```pascal
property DataType: Word;
```

Remarks

Use the DataType property to determine the type of object dtObject, dtArray or dtTable.
Reserved.

5.12.1.6.3 Methods

Methods of the TObjectType class.

For a complete list of the TObjectType class members, see the TObjectType Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>(inherited from TSharedObject) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>FindAttribute</td>
<td>Indicates whether a specified Attribute component is referenced in the TAttributes object.</td>
</tr>
<tr>
<td>Release</td>
<td>(inherited from TSharedObject) Decrements the reference count.</td>
</tr>
</tbody>
</table>
5.12.1.6.3.1 FindAttribute Method

Indicates whether a specified Attribute component is referenced in the TAttributes object.

Class

TObjectType

Syntax

function FindAttribute(const Name: string): TAttribute; virtual;

Parameters

Name
  Holds the name of the attribute to search for.

Return Value
  TAttribute, if an attribute with a matching name was found. Nil Otherwise.

Remarks

Call FindAttribute to determine if a specified Attribute component is referenced in the
TAttributes object. Name is the name of the Attribute for which to search. If FindAttribute finds
an Attribute with a matching name, it returns the TAttribute. Otherwise it returns nil.

See Also

- TAttribute
- Attributes

5.12.1.7 TSharedObject Class

A base class that allows to simplify memory management for object referenced by several other objects.
For a list of all members of this type, see `TSharedObject` members.

Unit

`MemData`

Syntax

```cpp
TSharedObject = class(System.TObject);
```

Remarks

`TSharedObject` allows to simplify memory management for object referenced by several other objects. TSharedObject holds a count of references to itself. When any object (referer object) is going to use TSharedObject, it calls the `TSharedObject.AddRef` method. Referer object has to call the `TSharedObject.Release` method after using `TSharedObject`.

See Also

- `TBlob`
- `TObjectType`

© 1997-2021
Devar. All Rights Reserved.

Request Support DAC Forum Provide Feedback

`TSharedObject` class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RefCount</code></td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRef</code></td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
</tbody>
</table>
Properties of the `TSharedObject` class.

For a complete list of the `TSharedObject` class members, see the [TSharedObject Members](#) topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RefCount</code></td>
<td>Used to return the count of reference to a <code>TSharedObject</code> object.</td>
</tr>
</tbody>
</table>

See Also

- [TSharedObject Class](#)
- [TSharedObject Class Members](#)

Remarks

Returns the count of reference to a `TSharedObject` object.

© 1997-2021 Devart. All Rights Reserved.

Request Support  DAC Forum  Provide Feedback
Methods of the `TSharedObject` class.

For a complete list of the `TSharedObject` class members, see the `TSharedObject Members` topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>Increments the reference count for the number of references dependent on the <code>TSharedObject</code> object.</td>
</tr>
<tr>
<td>Release</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

### See Also
- `TSharedObject Class`
- `TSharedObject Class Members`

© 1997-2021 Devart. All Rights Reserved.

© 2021 Devart
See Also
• Release

5.12.1.7.3.2 Release Method

Decrements the reference count.

Class
TSharedObject

Syntax

\[
\text{procedure Release;}
\]

Remarks
Call the Release method to decrement the reference count. When RefCount is 1, TSharedObject is deleted from memory.

See Also
• AddRef

5.12.2 Types

Types in the \texttt{MemData} unit.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLocateExOptions</td>
<td>Represents the set of TLocateExOption.</td>
</tr>
<tr>
<td>TUpdateRecKinds</td>
<td>Represents the set of TUpdateRecKind.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
5.12.2.1 TLocateExOptions Set

Represents the set of TLocateExOption.

Unit

MemData

Syntax

\[
\text{TLocateExOptions} = \text{set of } \text{TLocateExOption};
\]

5.12.2.2 TUpdateRecKinds Set

Represents the set of TUpdateRecKind.

Unit

MemData

Syntax

\[
\text{TUpdateRecKinds} = \text{set of } \text{TUpdateRecKind};
\]

5.12.3 Enumerations

Enumerations in the MemData unit.

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCompressBlobMode</td>
<td>Specifies when the values should be compressed and the way they should be stored.</td>
</tr>
<tr>
<td>TConnLostCause</td>
<td>Specifies the cause of the connection loss.</td>
</tr>
<tr>
<td>TDANumericType</td>
<td>Specifies the format of storing and representing of</td>
</tr>
</tbody>
</table>
### TCompressBlobMode Enumeration

Specifies when the values should be compressed and the way they should be stored.

**Unit**

**MemData**

**Syntax**

\[ \text{TCompressBlobMode} = (\text{cbNone}, \text{cbClient}, \text{cbServer}, \text{cbClientServer}); \]

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>cbClient</td>
<td>Values are compressed and stored as compressed data at the client side. Before posting data to the server decompression is performed and data at the server side stored in the original form. Allows to reduce used client memory due to increase access time to field values. The time spent on the opening DataSet and executing Post increases.</td>
</tr>
<tr>
<td>cbClientServer</td>
<td>Values are compressed and stored in compressed form. Allows to decrease the volume of used memory at client and server sides. Access time to the field values increases as for cbClient. The time spent on opening DataSet and executing Post decreases. <strong>Note:</strong> On using cbServer or cbClientServer data on the server is stored as compressed. Other applications can add records in uncompressed format but can’t read and write already compressed data. If compressed BLOB is partially changed by</td>
</tr>
</tbody>
</table>
another application (if signature was not changed), DAC will consider its value as NULL. Blob compression is not applied to Memo fields because of possible cutting.

**cbNone**
Values not compressed. The default value.

**cbServer**
Values are compressed before passing to the server and store at the server in compressed form. Allows to decrease database size on the server. Access time to the field values does not change. The time spent on opening DataSet and executing Post usually decreases.

5.12.3.2 **TConnLostCause Enumeration**

Specifies the cause of the connection loss.

**Unit**
MemData

**Syntax**

```
TConnLostCause = (clUnknown, clExecute, clOpen, clRefresh, clApply, clServiceQuery, clTransStart, clConnectionApply, clConnect);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>clApply</td>
<td>Connection loss detected during DataSet.ApplyUpdates (Reconnect/Reexecute possible).</td>
</tr>
<tr>
<td>clConnect</td>
<td>Connection loss detected during connection establishing (Reconnect possible).</td>
</tr>
<tr>
<td>clConnectionApply</td>
<td>Connection loss detected during Connection.ApplyUpdates (Reconnect/Reexecute possible).</td>
</tr>
<tr>
<td>clExecute</td>
<td>Connection loss detected during SQL execution (Reconnect with exception is possible).</td>
</tr>
<tr>
<td>clOpen</td>
<td>Connection loss detected during execution of a SELECT statement (Reconnect with exception possible).</td>
</tr>
<tr>
<td>clRefresh</td>
<td>Connection loss detected during query opening (Reconnect/Reexecute possible).</td>
</tr>
<tr>
<td>clServiceQuery</td>
<td>Connection loss detected during service information request (Reconnect/Reexecute possible).</td>
</tr>
<tr>
<td>clTransStart</td>
<td>Connection loss detected during transaction start (Reconnect/Reexecute possible).</td>
</tr>
</tbody>
</table>
5.12.3.3  TDANumericType Enumeration

Specifies the format of storing and representing of the NUMERIC (DECIMAL) fields.

Unit

MemData

Syntax

```
TDANumericType = (ntFloat, ntBCD, ntFmtBCD);
```

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntBCD</td>
<td>Data is stored on the client side as currency and represented as TBCDField. This format allows storing data with precision up to 0.0001.</td>
</tr>
<tr>
<td>ntFloat</td>
<td>Data stored on the client side is in double format and represented as TFloatField. The default value.</td>
</tr>
<tr>
<td>ntFmtBCD</td>
<td>Data is represented as TFMTBCDField. TFMTBCDField gives greater precision and accuracy than TBCDField, but it is slower.</td>
</tr>
</tbody>
</table>

5.12.3.4  TLocateExOption Enumeration

Allows to set additional search parameters which will be used by the LocateEx method.

Unit

MemData

Syntax

```
TLocateExOption = (lxCaseInsensitive, lxPartialKey, lxNearest,
```
Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>lxCaseInsensitive</td>
<td>Similar to loCaseInsensitive. Key fields and key values are matched without regard to the case.</td>
</tr>
<tr>
<td>lxNearest</td>
<td>LocateEx moves the cursor to a specific record in a dataset or to the first record in the dataset that is greater than the values specified in the KeyValues parameter. For this option to work correctly dataset should be sorted by the fields the search is performed in. If dataset is not sorted, the function may return a line that is not connected with the search condition.</td>
</tr>
<tr>
<td>lxNext</td>
<td>LocateEx searches from the current record.</td>
</tr>
<tr>
<td>lxPartialCompare</td>
<td>Similar to lxPartialKey, but the difference is that it can process value entries in any position. For example, 'HAM' would match both 'HAMM', 'HAMMER', and also 'MR HAMMER'.</td>
</tr>
<tr>
<td>lxPartialKey</td>
<td>Similar to loPartialKey. Key values can include only a part of the matching key field value. For example, 'HAM' would match both 'HAMM' and 'HAMMER', but not 'MR HAMMER'.</td>
</tr>
<tr>
<td>lxUp</td>
<td>LocateEx searches from the current record to the first record.</td>
</tr>
</tbody>
</table>

5.12.3.5 TSortType Enumeration

Specifies a sort type for string fields.

Unit

MemData

Syntax

\[
TSortType = (stCaseSensitive, stCaseInsensitive, stBinary);
\]

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>stBinary</td>
<td>Sorting by character ordinal values (this comparison is also case sensitive).</td>
</tr>
<tr>
<td>stCaseInsensitive</td>
<td>Sorting without case sensitivity.</td>
</tr>
</tbody>
</table>
5.12.3.6 TUpdateRecKind Enumeration

Indicates records for which the ApplyUpdates method will be performed.

Unit
MemData

Syntax

TUpdateRecKind = (ukDelete, ukInsert, ukUpdate);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ukDelete</td>
<td>ApplyUpdates will be performed for deleted records.</td>
</tr>
<tr>
<td>ukInsert</td>
<td>ApplyUpdates will be performed for inserted records.</td>
</tr>
<tr>
<td>ukUpdate</td>
<td>ApplyUpdates will be performed for updated records.</td>
</tr>
</tbody>
</table>

5.13 MemDS

This unit contains implementation of the TMemDataSet class.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMemDataSet</td>
<td>A base class for working with data and manipulating data in memory.</td>
</tr>
</tbody>
</table>

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoNotRaiseExcetionOnUaFail</td>
<td>An exception will be raised if</td>
</tr>
</tbody>
</table>
the value of the UpdateAction parameter is uaFail.

SendDataSetChangeEventAfterOpen

The DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.

© 1997-2021 Devart. All Rights Reserved.

5.13.1 Classes

Classes in the MemDS unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMemDataSet</td>
<td>A base class for working with data and manipulating data in memory.</td>
</tr>
</tbody>
</table>

5.13.1.1 TMemDataSet Class

A base class for working with data and manipulating data in memory.

For a list of all members of this type, see TMemDataSet members.

Unit

MemDS

Syntax

TMemDataSet = class(TDataSet);

Remarks

TMemDataSet derives from the TDataSet database-engine independent set of properties,
events, and methods for working with data and introduces additional techniques to store and manipulate data in memory.

© 1997-2021
Devart. All Rights Reserved.

5.13.1.1.1 Members

**TMemDataSet** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>Prepared</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td>Ranged</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>UpdatesPending</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

### Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyRange</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>CancelRange</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>DeferredPost</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td>GetBlob</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td>Locate</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Prepare</td>
<td>Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>Cancels changes made to</td>
</tr>
</tbody>
</table>
the current record when cached updates are enabled.

**SaveToXML**
- Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

**SetRange**
- Sets the starting and ending values of a range, and applies it.

**SetRangeEnd**
- Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

**SetRangeStart**
- Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

**UnPrepare**
- Frees the resources allocated for a previously prepared query on the server and client sides.

**UpdateResult**
- Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

**UpdateStatus**
- Indicates the current update status for the dataset when cached updates are enabled.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnUpdateError</strong></td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td><strong>OnUpdateRecord</strong></td>
<td>Occurs when a single update component can not</td>
</tr>
</tbody>
</table>
### Properties of the `TMemDataSet` class.

For a complete list of the `TMemDataSet` class members, see the [TMemDataSet Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>Prepared</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td>Ranged</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>UpdatesPending</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>
5.13.1.2.1 CachedUpdates Property

Used to enable or disable the use of cached updates for a dataset.

Class

TMemDataSet

Syntax

property CachedUpdates: boolean default False;

Remarks

Use the CachedUpdates property to enable or disable the use of cached updates for a dataset. Setting CachedUpdates to True enables updates to a dataset (such as posting changes, inserting new records, or deleting records) to be stored in an internal cache on the client side instead of being written directly to the dataset's underlying database tables. When changes are completed, an application writes all cached changes to the database in the context of a single transaction.

Cached updates are especially useful for client applications working with remote database servers. Enabling cached updates brings up the following benefits:

- Fewer transactions and shorter transaction times.
- Minimized network traffic.

The potential drawbacks of enabling cached updates are:

- Other applications can access and change the actual data on the server while users are editing local copies of data, resulting in an update conflict when cached updates are applied to the database.
- Other applications cannot access data changes made by an application until its cached updates are applied to the database.

The default value is False.

Note: When establishing master/detail relationship the CachedUpdates property of detail
dataset works properly only when `TDADatasetOptions.LocalMasterDetail` is set to True.

See Also
- `UpdatesPending`
- `TMemDataSet.ApplyUpdates`
- `RestoreUpdates`
- `CommitUpdates`
- `CancelUpdates`
- `UpdateStatus`
- `TCustomDADataset.Options`

5.13.1.1.2.2 `IndexFieldNames` Property

Used to get or set the list of fields on which the recordset is sorted.

Class

`TMemDataSet`

Syntax

```
property IndexFieldNames: string;
```

Remarks

Use the `IndexFieldNames` property to get or set the list of fields on which the recordset is sorted. Specify the name of each column in `IndexFieldNames` to use as an index for a table. Ordering of column names is significant. Separate names with semicolon. The specified columns don't need to be indexed. Set `IndexFieldNames` to an empty string to reset the recordset to the sort order originally used when the recordset's data was first retrieved.

Each field may optionally be followed by the keyword ASC / DESC or CIS / CS / BIN.

Use ASC, DESC keywords to specify a sort direction for the field. If one of these keywords is not used, the default sort direction for the field is ascending.

Use CIS, CS or BIN keywords to specify a sort type for string fields:

- CIS - compare without case sensitivity;
- CS - compare with case sensitivity;
BIN - compare by character ordinal values (this comparison is also case sensitive).

If a dataset uses a TCustomDAConnection component, the default value of sort type depends on the TCustomDAConnection.Options option of the connection. If a dataset does not use a connection (TVirtualTable dataset), the default is CS.

Read IndexFieldNames to determine the field (or fields) on which the recordset is sorted.

Ordering is processed locally.

**Note**: You cannot process ordering by BLOB fields. IndexFieldNames cannot be set to True when TCustomDADataset.UniDirectional=True.

**Example**

The following procedure illustrates how to set IndexFieldNames in response to a button click:

```pascal
DataSet1.IndexFieldNames := 'LastName ASC CIS; DateDue DESC';
```

5.13.1.1.2.3 KeyExclusive Property

Specifies the upper and lower boundaries for a range.

**Class**

TMemDataSet

**Syntax**

```
property KeyExclusive: Boolean;
```

**Remarks**

Use KeyExclusive to specify whether a range includes or excludes the records that match its specified starting and ending values.

By default, KeyExclusive is False, meaning that matching values are included.

To restrict a range to those records that are greater than the specified starting value and less than the specified ending value, set KeyExclusive to True.

**See Also**

- SetRange
### 5.13.1.1.2.4 LocalConstraints Property

Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

**Class**

`TMemDataSet`

**Syntax**

```
property LocalConstraints: boolean default True;
```

**Remarks**

Use the LocalConstraints property to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet. When LocalConstraints is True, TMemDataSet ignores NOT NULL server constraints. It is useful for tables that have fields updated by triggers.

LocalConstraints is obsolete, and is only included for backward compatibility.

The default value is True.

### 5.13.1.1.2.5 LocalUpdate Property

Used to prevent implicit update of rows on database server.

**Class**

`TMemDataSet`

**Syntax**

```
property LocalUpdate: boolean default False;
```

**Remarks**

Used to prevent implicit update of rows on database server.
Set the LocalUpdate property to True to prevent implicit update of rows on database server. Data changes are cached locally in client memory.

5.13.1.2.6 Prepared Property

Determined whether a query is prepared for execution or not.

Class

TMemDataSet

Syntax

```property
Prepared: boolean;
```

Remarks

Check the Prepared property to determine if a query is already prepared for execution. Prepared is True if the query has already been prepared. While queries don't need to be prepared before execution, performance is often boosted if queries are prepared beforehand, particularly if there are parameterized queries that are executed more than once using the same parameter values.

See Also

- Prepare

5.13.1.2.7 Ranged Property

Indicates whether a range is applied to a dataset.

Class

TMemDataSet

Syntax

```property
Ranged: Boolean;
```
Remarks
Use the Ranged property to detect whether a range is applied to a dataset.

See Also
- SetRange
- SetRangeEnd
- SetRangeStart

© 1997-2021
Devart. All Rights Reserved.

5.13.1.1.2.8 UpdateRecordTypes Property

Used to indicate the update status for the current record when cached updates are enabled.

Class
TMemDataSet

Syntax

```pascal
property UpdateRecordTypes: TUpdateRecordTypes default
[rtModified, rtInserted, rtUnmodified];
```

Remarks
Use the UpdateRecordTypes property to determine the update status for the current record when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateRecordTypes offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of records.

See Also
- CachedUpdates

© 1997-2021
Devart. All Rights Reserved.
5.13.1.2.9 UpdatesPending Property

Used to check the status of the cached updates buffer.

Class

TMemDataSet

Syntax

property UpdatesPending: boolean;

Remarks

Use the UpdatesPending property to check the status of the cached updates buffer. If UpdatesPending is True, then there are edited, deleted, or inserted records remaining in local cache and not yet applied to the database. If UpdatesPending is False, there are no such records in the cache.

See Also

• CachedUpdates

© 1997-2021 Devart. All Rights Reserved.

5.13.1.3 Methods

Methods of the TMemDataSet class.

For a complete list of the TMemDataSet class members, see the TMemDataSet Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyRange</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>CancelRange</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates</td>
<td>Clears all pending cached</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CommitUpdates</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>DeferredPost</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td>GetBlob</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td>Locate</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Prepare</td>
<td>Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetRange</td>
<td>Sets the starting and ending values of a range, and</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SetRangeStart</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>UpdateResult</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

**See Also**
- TMemDataSet Class
- TMemDataSet Class Members

**Syntax**

```plaintext
procedure ApplyRange;
```
Remarks

Call ApplyRange to cause a range established with SetRangeStart and SetRangeEnd, or EditRangeStart and EditRangeEnd, to take effect.

When a range is in effect, only those records that fall within the range are available to the application for viewing and editing.

After a call to ApplyRange, the cursor is left on the first record in the range.

See Also

- CancelRange
- EditRangeEnd
- EditRangeStart
- IndexFieldNames
- SetRange
- SetRangeEnd
- SetRangeStart

© 1997-2021
Devart. All Rights Reserved.

5.13.1.1.3.2 ApplyUpdates Method

Writes dataset's pending cached updates to a database.

Class

TMemDataSet

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyUpdates</td>
<td>Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>ApplyUpdates(const UpdateRecKinds: TUpdateRecKinds)</td>
<td>Writes dataset's pending cached updates of specified records to a database.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.
Writes dataset's pending cached updates to a database.

Class

TMemDataSet

Syntax

procedure ApplyUpdates; overload; virtual;

Remarks

Call the ApplyUpdates method to write a dataset's pending cached updates to a database. This method passes cached data to the database, but the changes are not committed to the database if there is an active transaction. An application must explicitly call the database component's Commit method to commit the changes to the database if the write is successful, or call the database's Rollback method to undo the changes if there is an error.

Following a successful write to the database, and following a successful call to a connection's Commit method, an application should call the CommitUpdates method to clear the cached update buffer.

Note: The preferred method for updating datasets is to call a connection component's ApplyUpdates method rather than to call each individual dataset's ApplyUpdates method. The connection component's ApplyUpdates method takes care of committing and rolling back transactions and clearing the cache when the operation is successful.

Example

The following procedure illustrates how to apply a dataset's cached updates to a database in response to a button click:

```pascal
procedure ApplyButtonClick(Sender: TObject);
begin
  with MyQuery do
  begin
    Session.StartTransaction;
    try
      ... <Modify data>
      ApplyUpdates; <try to write the updates to the database>
      Session.Commit; <on success, commit the changes>
    except
      RestoreUpdates; <restore update result for applied records>
      Session.Rollback; <on failure, undo the changes>
      raise; <raise the exception to prevent a call to CommitUpdates!>
    end;
    CommitUpdates; <on success, clear the cache>
  end
end;
```
end;

See Also
- TMemDataSet.CachedUpdates
- TMemDataSet.CancelUpdates
- TMemDataSet.CommitUpdates
- TMemDataSet.UpdateStatus

© 1997-2021 Devart. All Rights Reserved.

Request Support    DAC Forum    Provide Feedback

Writes dataset's pending cached updates of specified records to a database.

Class
TMemDataSet

Syntax

procedure ApplyUpdates(const UpdateRecKinds: TUpdateRecKinds);

overload; virtual:

Parameters

UpdateRecKinds
Indicates records for which the ApplyUpdates method will be performed.

Remarks

Call the ApplyUpdates method to write a dataset's pending cached updates of specified records to a database. This method passes cached data to the database, but the changes are not committed to the database if there is an active transaction. An application must explicitly call the database component's Commit method to commit the changes to the database if the write is successful, or call the database's Rollback method to undo the changes if there is an error.

Following a successful write to the database, and following a successful call to a connection's Commit method, an application should call the CommitUpdates method to clear the cached update buffer.

Note: The preferred method for updating datasets is to call a connection component's ApplyUpdates method rather than to call each individual dataset's ApplyUpdates method. The connection component's ApplyUpdates method takes care of committing and rolling back transactions and clearing the cache when the operation is successful.
5.13.1.3.3 CancelRange Method

Removes any ranges currently in effect for a dataset.

Class

TMemDataSet

Syntax

procedure CancelRange;

Remarks

Call CancelRange to remove a range currently applied to a dataset. Canceling a range reenables access to all records in the dataset.

See Also

- ApplyRange
- EditRangeEnd
- EditRangeStart
- IndexFieldNames
- SetRange
- SetRangeEnd
- SetRangeStart

5.13.1.3.4 CancelUpdates Method

Clears all pending cached updates from cache and restores dataset in its prior state.

Class

TMemDataSet

Syntax

procedure CancelUpdates;
Remarks
Call the CancelUpdates method to clear all pending cached updates from cache and restore dataset in its prior state.

It restores the dataset to the state it was in when the table was opened, cached updates were last enabled, or updates were last successfully applied to the database.

When a dataset is closed, or the CachedUpdates property is set to False, CancelUpdates is called automatically.

See Also
- CachedUpdates
- TMemDataSet.ApplyUpdates
- UpdateStatus

Clears the cached updates buffer.

Class
TMemDataSet

Syntax

```pascal
procedure CommitUpdates;
```

Remarks
Call the CommitUpdates method to clear the cached updates buffer after both a successful call to ApplyUpdates and a database component's Commit method. Clearing the cache after applying updates ensures that the cache is empty except for records that could not be processed and were skipped by the OnUpdateRecord or OnUpdateError event handlers. An application can attempt to modify the records still in cache.

CommitUpdates also checks whether there are pending updates in dataset. And if there are, it calls ApplyUpdates.

Record modifications made after a call to CommitUpdates repopulate the cached update buffer and require a subsequent call to ApplyUpdates to move them to the database.
5.13.1.1.3.6  DeferredPost Method

Makes permanent changes to the database server.

Class

TMemDataSet

Syntax

procedure DeferredPost;

Remarks

Call DeferredPost to make permanent changes to the database server while retaining dataset in its state whether it is dsEdit or dsInsert.

Explicit call to the Cancel method after DeferredPost has been applied does not abandon modifications to a dataset already fixed in database.

5.13.1.1.3.7  EditRangeEnd Method

Enables changing the ending value for an existing range.

Class

TMemDataSet

Syntax

procedure EditRangeEnd;

Remarks
Call EditRangeEnd to change the ending value for an existing range.

To specify an end range value, call FieldByName after calling EditRangeEnd.

After assigning a new ending value, call ApplyRange to activate the modified range.

See Also

- ApplyRange
- CancelRange
- EditRangeStart
- IndexFieldNames
- SetRange
- SetRangeEnd
- SetRangeStart

5.13.1.3.8 EditRangeStart Method

Enables changing the starting value for an existing range.

Class

TMemDataSet

Syntax

procedure EditRangeStart;

Remarks

Call EditRangeStart to change the starting value for an existing range.

To specify a start range value, call FieldByName after calling EditRangeStart.

After assigning a new ending value, call ApplyRange to activate the modified range.

See Also

- ApplyRange
- CancelRange
- EditRangeEnd
- IndexFieldNames
5.13.1.1.3.9 GetBlob Method

Retrieves TBlob object for a field or current record when only its name or the field itself is known.

Class

TMemDataSet

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBlob(Field: TField)</td>
<td>Retrieves TBlob object for a field or current record when the field itself is known.</td>
</tr>
<tr>
<td>GetBlob(const FieldName: string)</td>
<td>Retrieves TBlob object for a field or current record when its name is known.</td>
</tr>
</tbody>
</table>

Syntax

```
function GetBlob(Field: TField): TBlob; overload;
```

Parameters

- **Field**
  - Holds an existing TField object.

Return Value

TBlob object that was retrieved.
Remarks

Call the GetBlob method to retrieve TBlob object for a field or current record when only its name or the field itself is known. FieldName is the name of an existing field. The field should have MEMO or BLOB type.

Retrieves TBlob object for a field or current record when its name is known.

Class

TMemDataSet

Syntax

function GetBlob(const FieldName: string): TBlob; overload;

Parameters

FieldName

Holds the name of an existing field.

Return Value

TBlob object that was retrieved.

Example

OraQuery1.GetBlob('Comment').SaveToFile('Comment.txt');

See Also

• TBlob

Locate Method

Searches a dataset for a specific record and positions the cursor on it.

Class

TMemDataSet

Overload List
Searches a dataset by the specified fields for a specific record and positions cursor on it.

**Class**

TMemDataSet

**Syntax**

```pascal
function Locate(const KeyFields: array of TField; const KeyValues: variant; Options: TLocateOptions): boolean;
```

**Parameters**

- `KeyFields`
  - Holds TField objects in which to search.
- `KeyValues`
  - Holds the variant that specifies the values to match in the key fields.
- `Options`
  - Holds additional search latitude when searching in string fields.

**Return Value**

- True if it finds a matching record, and makes this record the current one. Otherwise it returns False.

Searches a dataset by the fields specified by name for a specific record and positions the cursor on it.

**Class**

TMemDataSet
Syntax

```go
function Locate(const KeyFields: string; const KeyValues: variant; Options: TLocateOptions): boolean; overload; override;
```

**Parameters**

- **KeyFields**
  Holds a semicolon-delimited list of field names in which to search.

- **KeyValues**
  Holds the variant that specifies the values to match in the key fields.

- **Options**
  Holds additional search latitude when searching in string fields.

**Return Value**

True if it finds a matching record, and makes this record the current one. Otherwise it returns False.

**Remarks**

Call the Locate method to search a dataset for a specific record and position cursor on it.

KeyFields is a string containing a semicolon-delimited list of field names on which to search.

KeyValues is a variant that specifies the values to match in the key fields. If KeyFields lists a single field, KeyValues specifies the value for that field on the desired record. To specify multiple search values, pass a variant array as KeyValues, or construct a variant array on the fly using the VarArrayOf routine. An example is provided below.

Options is a set that optionally specifies additional search latitude when searching in string fields. If Options contains the loCaseInsensitive setting, then Locate ignores case when matching fields. If Options contains the loPartialKey setting, then Locate allows partial-string matching on strings in KeyValues. If Options is an empty set, or if KeyFields does not include any string fields, Options is ignored.

Locate returns True if it finds a matching record, and makes this record the current one. Otherwise it returns False.

The Locate function works faster when dataset is locally sorted on the KeyFields fields. Local dataset sorting can be set with the `TMemDataSet.IndexFieldNames` property.

**Example**

An example of specifying multiple search values:

```go
with CustTable do
  Locate('Company;Contact;Phone', VarArrayOf(['Sight Diver', 'P',
```
5.13.1.1.3.11 LocateEx Method

Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.

Class

TMemDataSet

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LocateEx(const KeyFields: array of TField; const KeyValues: variant; Options: TLocateExOptions)</td>
<td>Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet by the specified fields.</td>
</tr>
<tr>
<td>LocateEx(const KeyFields: string; const KeyValues: variant; Options: TLocateExOptions)</td>
<td>Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet by the specified field names.</td>
</tr>
</tbody>
</table>

Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet by the specified fields.

Class

TMemDataSet

Syntax

```pascal
function LocateEx(const KeyFields: array of TField; const KeyValues: variant; Options: TLocateExOptions): boolean; overload;
```
Parameters

KeyFields
Holds TField objects to search in.

KeyValue
Holds the values of the fields to search for.

Options
Holds additional search parameters which will be used by the LocateEx method.

Return Value
True, if a matching record was found. Otherwise returns False.

Class
TMemDataSet

Syntax

```pascal
function LocateEx(const KeyFields: string; const KeyValue: variant; Options: TLocateExOptions): boolean; overload;
```

Parameters

KeyFields
Holds the fields to search in.

KeyValue
Holds the values of the fields to search for.

Options
Holds additional search parameters which will be used by the LocateEx method.

Return Value
True, if a matching record was found. Otherwise returns False.

Remarks

Call the LocateEx method when you need some features not to be included to the TMemDataSet.Locate method of TDataSet.

LocateEx returns True if it finds a matching record, and makes that record the current one. Otherwise LocateEx returns False.
The LocateEx function works faster when dataset is locally sorted on the KeyFields fields. Local dataset sorting can be set with the `TMemDataSet.IndexFieldNames` property.

**Note:** Please add the MemData unit to the "uses" list to use the TLocalExOption enumeration.

### See Also
- `TMemDataSet.IndexFieldNames`
- `TMemDataSet.Locate`

## 5.13.1.3.12 Prepare Method

Allocates resources and creates field components for a dataset.

### Class
- `TMemDataSet`

### Syntax

```
procedure Prepare; virtual;
```

### Remarks

Call the Prepare method to allocate resources and create field components for a dataset. The Prepare method is called automatically by the Open method if dataset is not prepared. To learn whether dataset is prepared or not use the Prepared property.

The UnPrepare method unprepares a query.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

The Prepare method is called automatically by the Open method if dataset is not prepared.

### See Also
- `Prepared`
- `UnPrepare`
5.13.1.3.13  RestoreUpdates Method

Marks all records in the cache of updates as unapplied.

Class

TMemDataSet

Syntax

procedure RestoreUpdates;

Remarks

Call the RestoreUpdates method to return the cache of updates to its state before calling ApplyUpdates. RestoreUpdates marks all records in the cache of updates as unapplied. It is useful when ApplyUpdates fails.

See Also

- CachedUpdates
- TMemDataSet.ApplyUpdates
- CancelUpdates
- UpdateStatus

© 1997-2021 Devart. All Rights Reserved.

5.13.1.3.14  RevertRecord Method

Cancels changes made to the current record when cached updates are enabled.

Class

TMemDataSet

Syntax

procedure RevertRecord;

Remarks

Call the RevertRecord method to undo changes made to the current record when cached updates are enabled.
5.13.1.3.15 SaveToXML Method

Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

Class

TMemDataSet

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SaveToXML(Destination: TStream)</td>
<td>Saves the current dataset data to a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SaveToXML(const FileName: string)</td>
<td>Saves the current dataset data to a file in the XML format compatible with ADO format.</td>
</tr>
</tbody>
</table>

Saves the current dataset data to a stream in the XML format compatible with ADO format.

Class

TMemDataSet

Syntax

```
procedure SaveToXML(Destination: TStream); overload;
```

Parameters

Destination

Holds a TStream object.
Remarks

Call the SaveToXML method to save the current dataset data to a file or a stream in the XML format compatible with ADO format.

If the destination file already exists, it is overwritten. It remains open from the first call to SaveToXML until the dataset is closed. This file can be read by other applications while it is opened, but they cannot write to the file.

When saving data to a stream, a TStream object must be created and its position must be set in a preferable value.

See Also

- TVirtualTable.LoadFromFile
- TVirtualTable.LoadFromStream

Saves the current dataset data to a file in the XML format compatible with ADO format.

Class

TMemDataSet

Syntax

```delphi
procedure SaveToXML(const FileName: string); overload;
```

Parameters

- **FileName**
  
  Holds the name of a destination file.

See Also

- TVirtualTable.LoadFromFile
- TVirtualTable.LoadFromStream
5.13.1.3.16  SetRange Method

Sets the starting and ending values of a range, and applies it.

Class

TMemDataSet

Syntax

```
procedure SetRange(const StartValues: array of System.TVarRec;
const EndValues: array of System.TVarRec; StartExclusive: Boolean = False;
EndExclusive: Boolean = False);
```

Parameters

StartValues

Indicates the field values that designate the first record in the range. In C++,
StartValues_Size is the index of the last value in the StartValues array.

EndValues

Indicates the field values that designate the last record in the range. In C++,
EndValues_Size is the index of the last value in the EndValues array.

StartExclusive

Indicates the upper and lower boundaries of the start range.

EndExclusive

Indicates the upper and lower boundaries of the end range.

Remarks

Call SetRange to specify a range and apply it to the dataset. The new range replaces the
currently specified range, if any.

SetRange combines the functionality of SetRangeStart, SetRangeEnd, and ApplyRange in a
single procedure call. SetRange performs the following functions:

1. Puts the dataset into dsSetKey state.
2. Erases any previously specified starting range values and ending range values.
3. Sets the start and end range values.
4. Applies the range to the dataset.

After a call to SetRange, the cursor is left on the first record in the range.

If either StartValues or EndValues has fewer elements than the number of fields in the current
index, then the remaining entries are ignored when performing a search.

See Also
5.13.1.1.3.17  SetRangeEnd Method

Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

Class

TMemDataSet

Syntax

procedure SetRangeEnd;

Remarks

Call SetRangeEnd to put the dataset into dsSetKey state, erase any previous end range values, and set them to NULL.

Subsequent field assignments made with FieldByName specify the actual set of ending values for a range.

After assigning end-range values, call ApplyRange to activate the modified range.

See Also

- ApplyRange
- CancelRange
- EditRangeEnd
- EditRangeStart
- IndexFieldNames
- KeyExclusive
- SetRange
- SetRangeEnd
- SetRangeStart
5.13.1.3.18 SetRangeStart Method

Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

Class

TMemDataSet

Syntax

`procedure SetRangeStart;`

Remarks

Call SetRangeStart to put the dataset into dsSetKey state, erase any previous start range values, and set them to NULL.

Subsequent field assignments to FieldByName specify the actual set of starting values for a range.

After assigning start-range values, call ApplyRange to activate the modified range.

See Also

- ApplyRange
- CancelRange
- EditRangeStart
- IndexFieldNames
- SetRange
- SetRangeEnd

5.13.1.3.19 UnPrepare Method

Frees the resources allocated for a previously prepared query on the server and client sides.

Class
**TMemDataSet**

**Syntax**

```plaintext
procedure UnPrepare; virtual;
```

**Remarks**

Call the UnPrepare method to free the resources allocated for a previously prepared query on the server and client sides.

**Note:** When you change the text of a query at runtime, the query is automatically closed and unprepared.

**See Also**

- [Prepare](#)

---

**5.13.1.1.3.20** **UpdateResult Method**

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

**Class**

**TMemDataSet**

**Syntax**

```plaintext
function UpdateResult: TUpdateAction;
```

**Return Value**

A value of the TUpdateAction enumeration.

**Remarks**

Call the UpdateResult method to read the status of the latest call to the ApplyUpdates method while cached updates are enabled. UpdateResult reflects updates made on the records that have been edited, inserted, or deleted.

UpdateResult works on the record by record basis and is applicable to the current record only.
5.13.1.1.3.21 UpdateStatus Method

Indicates the current update status for the dataset when cached updates are enabled.

Class

**TMemDataSet**

Syntax

```objc
function UpdateStatus: TUpdateStatus; override;
```

Return Value

A value of the TUpdateStatus enumeration.

Remarks

Call the UpdateStatus method to determine the current update status for the dataset when cached updates are enabled. Update status can change frequently as records are edited, inserted, or deleted. UpdateStatus offers a convenient method for applications to assess the current status before undertaking or completing operations that depend on the update status of the dataset.

See Also

- **CachedUpdates**

© 1997-2021

Devart. All Rights Reserved.

Request Support    DAC Forum    Provide Feedback
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnUpdateError</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td>OnUpdateRecord</td>
<td>Occurs when a single update component can not handle the updates.</td>
</tr>
</tbody>
</table>

**See Also**
- [TMemDataSet Class](#)
- [TMemDataSet Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

5.13.1.1.4.1 OnUpdateError Event

Occurred when an exception is generated while cached updates are applied to a database.

**Class**

[TMemDataSet](#)

**Syntax**

```property
property OnUpdateError: TUpdateErrorEvent;
```

**Remarks**

Write the OnUpdateError event handler to respond to exceptions generated when cached updates are applied to a database.

E is a pointer to an EDatabaseError object from which application can extract an error message and the actual cause of the error condition. The OnUpdateError handler can use this information to determine how to respond to the error condition.

UpdateKind describes the type of update that generated the error.

UpdateAction indicates the action to take when the OnUpdateError handler exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateError can handle or correct the error, set UpdateAction to uaRetry before exiting the error handler.

The error handler can use the TField.OldValue and TField.NewValue properties to evaluate
error conditions and set TField.NewValue to a new value to reapply. In this case, set UpdateAction to uaRetry before exiting.

**Note:** If a call to ApplyUpdates raises an exception and ApplyUpdates is not called within the context of a try...except block, an error message is displayed. If the OnUpdateError handler cannot correct the error condition and leaves UpdateAction set to uaFail, the error message is displayed twice. To prevent redisplay, set UpdateAction to uaAbort in the error handler.

See Also
- [CachedUpdates](#)

### 5.13.1.1.4.2 OnUpdateRecord Event

Occurs when a single update component can not handle the updates.

**Class**
- `TMemDataSet`

**Syntax**

```
property OnUpdateRecord: TUpdateRecordEvent;
```

**Remarks**

Write the OnUpdateRecord event handler to process updates that cannot be handled by a single update component, such as implementation of cascading updates, insertions, or deletions. This handler is also useful for applications that require additional control over parameter substitution in update components.

UpdateKind describes the type of update to perform.

UpdateAction indicates the action taken by the OnUpdateRecord handler before it exits. On entry into the handler, UpdateAction is always set to uaFail. If OnUpdateRecord is successful, it should set UpdateAction to uaApplied before exiting.

See Also
- [CachedUpdates](#)

© 1997-2021

Devart. All Rights Reserved.
5.13.2 Variables

Variables in the **MemDS** unit.

### Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DoNotRaiseExcetionOnUaFail</strong></td>
<td>An exception will be raised if the value of the UpdateAction parameter is uaFail.</td>
</tr>
<tr>
<td><strong>SendDataSetChangeEventAfterOpen</strong></td>
<td>The DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.</td>
</tr>
</tbody>
</table>

#### 5.13.2.1 **DoNotRaiseExcetionOnUaFail** Variable

An exception will be raised if the value of the UpdateAction parameter is uaFail.

**Unit**

**MemDS**

**Syntax**

```plaintext
DoNotRaiseExcetionOnUaFail: boolean = False;
```

**Remarks**

Starting with ODAC 6.20.0.12, if the **OnUpdateRecord** event handler sets the UpdateAction parameter to uaFail, an exception is raised. The default value of UpdateAction is uaFail. So, the exception will be raised when the value of this parameter is left unchanged.

To restore the old behaviour, set DoNotRaiseExcetionOnUaFail to True.
5.13.2.2 SendDataSetChangeEventAfterOpen Variable

The DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids.

Unit

MemDS

Syntax

SendDataSetChangeEventAfterOpen: boolean = True;

Remarks

Starting with ODAC 6.20.0.11, the DataSetChange event is sent after a dataset gets open. It was necessary to fix a problem with disappeared vertical scrollbar in some types of DB-aware grids. This problem appears only under Windows XP when visual styles are enabled.

To disable sending this event, change the value of this variable to False.

© 1997-2021 Devart. All Rights Reserved.

5.14 OdacVcl

This unit contains the visual constituent of ODAC.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TConnectDialog</td>
<td>A component providing a dialog box for a user to supply login information.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.14.1 Classes

Classes in the OdacVcl unit.

Classes
**Name**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TConnectDialog</td>
<td>A component providing a dialog box for a user to supply login information.</td>
</tr>
</tbody>
</table>

**5.14.1.1 TConnectDialog Class**

A component providing a dialog box for a user to supply login information.

For a list of all members of this type, see TConnectDialog members.

**Unit**

`OdacVcl`

**Syntax**

```delphi
tConnectDialog = class(TCustomConnectDialog);
```

**Remarks**

TConnectDialog component is a direct descendent of TCustomConnectDialog class. Use TConnectDialog to provide a dialog box for a user to supply username, password and server name. You may want to customize appearance of the dialog box using the properties of this class.

**Inheritance Hierarchy**

TCustomConnectDialog

TConnectDialog

**See Also**

- TCustomDACConnection.ConnectDialog

© 1997-2021

Devart. All Rights Reserved.

Request Support  DAC Forum  Provide Feedback
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CancelButton</td>
<td>(inherited from TCustomConnectDialog) Used to specify the label for the Cancel button.</td>
</tr>
<tr>
<td>Caption</td>
<td>(inherited from TCustomConnectDialog) Used to set the caption of dialog box.</td>
</tr>
<tr>
<td>ConnectButton</td>
<td>(inherited from TCustomConnectDialog) Used to specify the label for the Connect button.</td>
</tr>
<tr>
<td>DialogClass</td>
<td>(inherited from TCustomConnectDialog) Used to specify the class of the form that will be displayed to enter login information.</td>
</tr>
<tr>
<td>LabelSet</td>
<td>(inherited from TCustomConnectDialog) Used to set the language of buttons and labels captions.</td>
</tr>
<tr>
<td>PasswordLabel</td>
<td>(inherited from TCustomConnectDialog) Used to specify a prompt for password edit.</td>
</tr>
<tr>
<td>ReadAliases</td>
<td>Used to specify where the TConnectDialog object should acquire the names of database instances.</td>
</tr>
<tr>
<td>Retries</td>
<td>(inherited from TCustomConnectDialog) Used to indicate the number of retries of failed connections.</td>
</tr>
<tr>
<td>SavePassword</td>
<td>(inherited from TCustomConnectDialog) Used for the password to be displayed in ConnectDialog in asterisks.</td>
</tr>
<tr>
<td>ServerLabel</td>
<td>(inherited from TCustomConnectDialog) Used to specify a prompt for the server name edit.</td>
</tr>
<tr>
<td>Session</td>
<td>Shows what TOraSession component uses TConnectDialog object.</td>
</tr>
<tr>
<td>StoreLogInfo</td>
<td>(inherited from TCustomConnectDialog) Used to specify whether the login information should be kept in system registry after a connection was established.</td>
</tr>
<tr>
<td>UsernameLabel</td>
<td>(inherited from TCustomConnectDialog) Used to specify a prompt for username edit.</td>
</tr>
</tbody>
</table>

## Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execute</strong></td>
<td>Displays the connect dialog and calls the connection's Connect method when user clicks the Connect button.</td>
</tr>
<tr>
<td><strong>GetServerList</strong></td>
<td>Retrieves a list of available server names.</td>
</tr>
</tbody>
</table>

5.14.1.1.2 Properties

Properties of the **TConnectDialog** class.

For a complete list of the **TConnectDialog** class members, see the **TConnectDialog Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CancelButton</strong></td>
<td>Used to specify the label for the Cancel button.</td>
</tr>
<tr>
<td><strong>Caption</strong></td>
<td>Used to set the caption of dialog box.</td>
</tr>
<tr>
<td><strong>ConnectButton</strong></td>
<td>Used to specify the label for the Connect button.</td>
</tr>
<tr>
<td><strong>DialogClass</strong></td>
<td>Used to specify the class of the form that will be displayed to enter login information.</td>
</tr>
<tr>
<td><strong>LabelSet</strong></td>
<td>Used to set the language of buttons and labels captions.</td>
</tr>
<tr>
<td><strong>PasswordLabel</strong></td>
<td>Used to specify a prompt for password edit.</td>
</tr>
<tr>
<td><strong>Retries</strong></td>
<td>Used to indicate the number of retries of failed connections.</td>
</tr>
<tr>
<td><strong>SavePassword</strong></td>
<td>Used for the password to be displayed in ConnectDialog in asterisks.</td>
</tr>
<tr>
<td><strong>ServerLabel</strong></td>
<td>Used to specify a prompt for the server name edit.</td>
</tr>
</tbody>
</table>
### ReadAliases Property

Used to specify where the TConnectDialog object should acquire the names of database instances.

**Class**

TConnectDialog

**Syntax**

```prop
property ReadAliases: boolean default False;
```

**Remarks**

Use ReadAliases property to specify whether the TConnectDialog object should acquire the
names of database instances from TNSNAMES.ORA configuration file found in the Oracle home directory or read them from Odac registry entries.

Set this property to False to make ODAC read aliases from registry.

The default value is False.

**Note:** ODAC relies on valid local Oracle home directory structure if ReadAliases property is set to True.

### Description

`T:Devart.Odac.Units.OdacVcl`

© 1997-2021

Devart. All Rights Reserved.

5.14.1.1.2.2 Session Property

Shows what TOraSession component uses TConnectDialog object.

### Class

`TConnectDialog`

### Syntax

```
property Session: TOraSession;
```

### Remarks

Read Session property to learn what TOraSession component uses TConnectDialog object. This property is read-only.

### See Also

- `TCustomDACConnection.ConnectDialog`

© 1997-2021

Devart. All Rights Reserved.

5.15 Ora

This unit contains main components of ODAC.

### Classes
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBFileField</td>
<td>A class representing BFile field in dataset.</td>
</tr>
<tr>
<td>TCursorField</td>
<td>A class representing REF CURSOR field in dataset.</td>
</tr>
<tr>
<td>TCustomOraQuery</td>
<td>A base class defining functionality for descendent classes which access database using SQL statements.</td>
</tr>
<tr>
<td>TOraChangeNotification</td>
<td>A component for keeping information in local dataset up-to-date through receiving notifications.</td>
</tr>
<tr>
<td>TOraChangeNotificationOptions</td>
<td>Used to specify how query notifications will be generated.</td>
</tr>
<tr>
<td>TOraConnectionSSLOptions</td>
<td>A class for setting up the SSL options.</td>
</tr>
<tr>
<td>TOraDataSet</td>
<td>A class defining the Oracle functionality for a dataset.</td>
</tr>
<tr>
<td>TOraDataSetField</td>
<td>A class providing access to Oracle nested datasets.</td>
</tr>
<tr>
<td>TOraDataSetOptions</td>
<td>This class allows setting up the behaviour of the TOraDataSet class.</td>
</tr>
<tr>
<td>TOraDataSetOptionsDS</td>
<td>This class allows setting up the behaviour of the TOraDataSet class (this property is obsolete).</td>
</tr>
<tr>
<td>TOraDataSource</td>
<td>TOraDataSource provides an interface between an ODAC dataset components and data-aware controls on a form.</td>
</tr>
<tr>
<td>TOraEncryptor</td>
<td>The class that performs encrypting and decrypting of data.</td>
</tr>
<tr>
<td>TOraIntervalField</td>
<td>A class providing access to the Oracle interval fields.</td>
</tr>
<tr>
<td>TOraMetaData</td>
<td>A component for obtaining metainformation about database objects from the server.</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOraNestedTable</td>
<td>A component for controlling nested table data.</td>
</tr>
<tr>
<td>TOraNumberField</td>
<td>A class providing access to the Oracle number fields.</td>
</tr>
<tr>
<td>TOraParam</td>
<td>A class that is used to set the values of individual parameters passed with queries or stored procedures.</td>
</tr>
<tr>
<td>TOraParams</td>
<td>Used to control TOraParam objects.</td>
</tr>
<tr>
<td>TOraPoolingOptions</td>
<td>This class allows setting up the behaviour of the connection pool.</td>
</tr>
<tr>
<td>TOraQuery</td>
<td>A component for executing queries and operating record sets. It also provides flexible way to update data.</td>
</tr>
<tr>
<td>TOraReferenceField</td>
<td>A class representing an Oracle REF field in a dataset.</td>
</tr>
<tr>
<td>TOraSession</td>
<td>A component for maintaining connection to an Oracle database.</td>
</tr>
<tr>
<td>TOraSessionOptions</td>
<td>This class allows setting up the behaviour of the TOraSession class.</td>
</tr>
<tr>
<td>TOraSQL</td>
<td>A component for executing SQL statements and calling stored procedures on the database server.</td>
</tr>
<tr>
<td>TOraStoredProc</td>
<td>A component for accessing and executing stored procedures and functions.</td>
</tr>
<tr>
<td>TOraTimeStampField</td>
<td>A class providing access to the Oracle timestamp fields.</td>
</tr>
<tr>
<td>TOraTrace</td>
<td>A component allowing starting and stopping a SQL trace for a specified session. This component provides access to the DBMS_TRACE package.</td>
</tr>
<tr>
<td>TOraUpdateSQL</td>
<td>A component for tuning update operations for the</td>
</tr>
</tbody>
</table>
### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TConnectChangeEvent</td>
<td>This type is used for the TOraSession.OnConnectChangeEvent event.</td>
</tr>
<tr>
<td>TFailoverEvent</td>
<td>This Type is used for the TOraSession.OnFailover event.</td>
</tr>
<tr>
<td>TOraChangeNotificationEvent</td>
<td>This type is used for the TOraChangeNotification.OnChange event.</td>
</tr>
<tr>
<td>TPSqlTraceMode</td>
<td>Specifies the level of PL/SQL trace.</td>
</tr>
<tr>
<td>TSqITraceMode</td>
<td>Specifies the level of SQL trace statistics level.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFailoverState</td>
<td>Indicates the failover state.</td>
</tr>
<tr>
<td>TFailoverType</td>
<td>Specifies the failover type.</td>
</tr>
<tr>
<td>TOraIsolationLevel</td>
<td>Specifies the way the transactions containing database modifications are handled.</td>
</tr>
<tr>
<td>TRefreshMode</td>
<td>Defines when to refresh an editing record.</td>
</tr>
<tr>
<td>TSequenceMode</td>
<td>Specifies the method used internally to generate sequenced field.</td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefSession</td>
<td>Read this variable to get</td>
</tr>
</tbody>
</table>
pointer to default session object. Same as DefaultSession function.

**OraQueryCompatibilityMode**

All TOraQuery components in project become editable, and can be modified by the end users.

**Sessions**

Holds pointers to all TOraSession objects of an application.

**UseDefSession**

When set to True enables TOraDataSet and TOraSQL components to use default session if they are not attached to any session.

## 5.15.1 Classes

Classes in the Ora unit.

### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBFileField</td>
<td>A class representing BFile field in dataset.</td>
</tr>
<tr>
<td>TCursorField</td>
<td>A class representing REF CURSOR field in dataset.</td>
</tr>
<tr>
<td>TCustomOraQuery</td>
<td>A base class defining functionality for descendent classes which access database using SQL statements.</td>
</tr>
<tr>
<td>TOraChangeNotification</td>
<td>A component for keeping information in local dataset up-to-date through receiving notifications.</td>
</tr>
<tr>
<td>TOraChangeNotificationOptions</td>
<td>Used to specify how query notifications will be generated.</td>
</tr>
<tr>
<td>TOraConnectionSSLOptions</td>
<td>A class for setting up the SSL options.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TOraDataSet</td>
<td>A class defining the Oracle functionality for a dataset.</td>
</tr>
<tr>
<td>TOraDataSetField</td>
<td>A class providing access to Oracle nested datasets.</td>
</tr>
<tr>
<td>TOraDataSetOptions</td>
<td>This class allows setting up the behaviour of the TOraDataSet class.</td>
</tr>
<tr>
<td>TOraDataSetOptionsDS</td>
<td>This class allows setting up the behaviour of the TOraDataSet class (this</td>
</tr>
<tr>
<td></td>
<td>property is obsolete).</td>
</tr>
<tr>
<td>TOraDataSource</td>
<td>TOraDataSource provides an interface between an ODAC dataset components and</td>
</tr>
<tr>
<td></td>
<td>data-aware controls on a form.</td>
</tr>
<tr>
<td>TOraEncryptor</td>
<td>The class that performs encrypting and decrypting of data.</td>
</tr>
<tr>
<td>TOraIntervalField</td>
<td>A class providing access to the Oracle interval fields.</td>
</tr>
<tr>
<td>TOraMetaData</td>
<td>A component for obtaining metainformation about database objects from the</td>
</tr>
<tr>
<td></td>
<td>server.</td>
</tr>
<tr>
<td>TOraNestedTable</td>
<td>A component for controlling nested table data.</td>
</tr>
<tr>
<td>TOraNumberField</td>
<td>A class providing access to the Oracle number fields.</td>
</tr>
<tr>
<td>TOraParam</td>
<td>A class that is used to set the values of individual parameters passed with</td>
</tr>
<tr>
<td></td>
<td>queries or stored procedures.</td>
</tr>
<tr>
<td>TOraParams</td>
<td>Used to control TOraParam objects.</td>
</tr>
<tr>
<td>TOraPoolingOptions</td>
<td>This class allows setting up the behaviour of the connection pool.</td>
</tr>
<tr>
<td>TOraQuery</td>
<td>A component for executing queries and operating record sets. It also provides</td>
</tr>
<tr>
<td></td>
<td>flexible way to update data.</td>
</tr>
</tbody>
</table>
| TOraReferenceField                         | A class representing an
<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraSession</td>
<td>A component for maintaining connection to an Oracle database.</td>
</tr>
<tr>
<td>TOraSessionOptions</td>
<td>This class allows setting up the behaviour of the TOraSession class.</td>
</tr>
<tr>
<td>TOraSQL</td>
<td>A component for executing SQL statements and calling stored procedures on the database server.</td>
</tr>
<tr>
<td>TOraStoredProc</td>
<td>A component for accessing and executing stored procedures and functions.</td>
</tr>
<tr>
<td>TOraTimeStampField</td>
<td>A class providing access to the Oracle timestamp fields.</td>
</tr>
<tr>
<td>TOraTrace</td>
<td>A component allowing starting and stopping a SQL trace for a specified session. This component provides access to the DBMS_TRACE package.</td>
</tr>
<tr>
<td>TOraUpdateSQL</td>
<td>A component for tuning update operations for the DataSet component.</td>
</tr>
<tr>
<td>TOraXMLField</td>
<td>A class providing access to the Oracle SYS.XMLTYPE objects.</td>
</tr>
</tbody>
</table>

5.15.1.1 TBFileField Class

A class representing BFile field in dataset.

For a list of all members of this type, see TBFileField members.

Unit
ora

Syntax
TBFileField = class(TBlobField);

Remarks

TBFileField object represents BFile field in dataset.

The BFile datatype provides access to file LOBs that are stored in file systems outside an Oracle database. Oracle 8 currently supports access to binary files, or BFILEs. The BFILE datatype allows read-only support of large binary files; you cannot modify a file through Oracle.

TBFileField holds a TOraFile object. To get it use the AsFile property.

As a descendent of TField, TBFileField inherits many properties, methods, and events that are useful for managing the value and properties of a field in the database.

See Also
• TOraFile

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsFile</td>
<td>Returns a TOraFile object.</td>
</tr>
<tr>
<td>AutoRefresh</td>
<td>Used to refresh content of BFile when FileDir or FileName is being changed.</td>
</tr>
<tr>
<td>BlobType</td>
<td>Indicates the type of BLOB field.</td>
</tr>
<tr>
<td>Exists</td>
<td>Returns True when a file associated with BFile exists, False otherwise.</td>
</tr>
<tr>
<td>FileDir</td>
<td>Used to indicate the directory alias where BFile is stored.</td>
</tr>
<tr>
<td>FileName</td>
<td>Indicates the name of a file associated with BFile.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Reloads BFile.</td>
</tr>
</tbody>
</table>

## Properties of the TBFileField class.

For a complete list of the TBFileField class members, see the [TBFileField Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsFile</td>
<td>Returns a TOraFile object.</td>
</tr>
<tr>
<td>Exists</td>
<td>Returns True when a file associated with BFile exists, False otherwise.</td>
</tr>
<tr>
<td>FileDir</td>
<td>Used to indicate the directory alias where BFile is stored.</td>
</tr>
<tr>
<td>FileName</td>
<td>Indicates the name of a file associated with BFile.</td>
</tr>
</tbody>
</table>

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoRefresh</td>
<td>Used to refresh content of BFile when FileDir or FileName is being changed.</td>
</tr>
<tr>
<td>BlobType</td>
<td>Indicates the type of BLOB field.</td>
</tr>
</tbody>
</table>

See Also
- TBFileField Class
- TBFileField Class Members
5.15.1.1.2.1 AsFile Property

Returns a T0raFile object.

Class
TBFileField

Syntax

property AsFile: TOraFile;

Remarks
Returns a T0raFile object. Later you can open T0raDataSet once.

See Also
• T0raFile

5.15.1.1.2.2 AutoRefresh Property

Used to refresh content of BFile when FileDir or FileName is being changed.

Class
TBFileField

Syntax

property AutoRefresh: boolean default True;

Remarks
When AutoRefresh is True, TBFileField will refresh content of BFile when FileDir or FileName is changed.

The default value is True.
5.15.1.1.2.3 BlobType Property

Indicates the type of BLOB field.

Class

TBFileField

Syntax

```
property BlobType: TBlobType;
```

Remarks

Indicates the type of BLOB field. For TBFileField is always equal to ftBlob.

```
© 1997-2021
Devart. All Rights Reserved.
```

5.15.1.1.2.4 Exists Property

Returns True when a file associated with BFile exists, False otherwise.

Class

TBFileField

Syntax

```
property Exists: boolean;
```

Remarks

Exists returns True when a file associated with BFile exists, False otherwise.

Example

```
if not TBFileField(DataSet.FieldByName('Value')).Exists then
  St:= St + ' (NoExist)';
```

© 1997-2021
Devart. All Rights Reserved.
5.15.1.1.2.5  FileDir Property

Used to indicate the directory alias where BFile is stored.

Class

TBFileField

Syntax

```delphi
class TBFileField;


dtProperty FileDir: string;
```

Remarks

Use the FileDir property to determine the directory alias where BFile is stored.

To create a directory alias use CREATE DIRECTORY.

Example

```delphi
edFileDir.Text := TBFileField(OraQuery.FieldByName('Value')).FileDir;
```

5.15.1.1.2.6  FileName Property

Indicates the name of a file associated with BFile.

Class

TBFileField

Syntax

```delphi
class TBFileField;


dtProperty FileName: string;
```

Remarks

Use the FileName property to determine the name of a file associated with BFile.

Example

```delphi
edFileName.Text := BFileField(OraQuery.FieldByName('Value')).FileName;
```
5.15.1.3 Methods

Methods of the **TBFileField** class.

For a complete list of the **TBFileField** class members, see the [TBFileField Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Reloads BFile.</td>
</tr>
</tbody>
</table>

See Also
- [TBFileField Class](#)
- [TBFileField Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

5.15.1.3.1 Refresh Method

Reloads BFile.

**Class**

**TBFileField**

**Syntax**

```pascal
procedure Refresh;
```

**Remarks**

Call the Refresh procedure to reload BFile.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.2 **TCursorField Class**

A class representing REF CURSOR field in dataset.

For a list of all members of this type, see [TCursorField members](#).
Ora

Syntax

```plaintext
TCursorField = class(TDACursorField);
```

Remarks

A TCursorField object represents REF CURSOR field in dataset.

TCursorField holds a TOraCursor object. To get it use the AsCursor property.

As a descendent of TField, TCursorField inherits many properties, methods, and events that are useful for managing the value and properties of a field in a database.

Inheritance Hierarchy

TDACursorField
    TCursorField

See Also

- TOraCursor
- TOraDataSet.Cursor

© 1997-2021 Devart. All Rights Reserved.
5.15.1.2.2 Properties

Properties of the TCursorField class.

For a complete list of the TCursorField class members, see the TCursorField Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsCursor</td>
<td>Returns a TOraCursor object which you can assign to the Cursor property of TOraDataSet.</td>
</tr>
</tbody>
</table>

See Also

- TCursorField Class
- TCursorField Class Members

© 1997-2021 Devart. All Rights Reserved.
5.15.1.3 TCustomOraQuery Class

A base class defining functionality for descendent classes which access database using SQL statements.

For a list of all members of this type, see TCustomOraQuery members.

Unit
ora

Syntax

TCustomOraQuery = class(TOraDataSet);

Remarks

TCustomOraQuery is a base class that defines functionality for descendent classes which access database using SQL statements. Applications never use TCustomOraQuery objects directly. Instead they use descendants of TCustomOraQuery, such as TOraQuery, TSmartQuery, TOraStoredProc and TOraTable.

TCustomOraQuery implements functionality for an insertion, deletion, and update of a record by dynamically generated SQL statements. It offers such features as automatic blocking of records, checking records before edit, refreshing records after post.

To modify records of TCustomOraQuery SELECT statement in the SQL property should retrieve ROWID of updating table. When the KeyFields property is modified, TCustomOraQuery is updated too. TCustomOraQuery can update only one Oracle table. Updating table is defined by the UpdatingTable property or used by the first table in the FROM clause.

SQLInsert, SQLDelete, SQLUpdate, SQLLock, SQLRefresh properties support automatic binding of parameters which have names identical to the fields captions. To retrieve the value of a field as it was before operation, use the field name with 'OLD_'. This is particularly useful when doing field comparisons in the WHERE clause of a statement. Use the TCustomDADataset.BeforeUpdateExecute event to assign the value to additional parameters and the TCustomDADataset.AfterUpdateExecute event for reading them.

TCustomOraQuery is read-only when none of the SQLInsert, SQLDelete, SQLUpdate properties are defined.

Inheritance Hierarchy
### TCustomOraQuery class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>CheckMode</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td>CommandTimeout</td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Used to add WHERE</td>
</tr>
<tr>
<td>Member</td>
<td>Inheritance</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------</td>
</tr>
<tr>
<td><strong>Connection</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>FetchAll</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td><strong>FetchRows</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>IsPLSQL</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td><strong>IsQuery</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>KeyFields</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td><strong>KeySequence</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>LockMode</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to define when to perform the locking of an editing record.</td>
</tr>
<tr>
<td><strong>MacroCount</strong></td>
<td>(inherited from <strong>TCustomDADataSet</strong>) Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td><strong>Macros</strong></td>
<td>(inherited from <strong>TCustomDADataSet</strong>) Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td><strong>MasterFields</strong></td>
<td>(inherited from <strong>TCustomDADataSet</strong>) Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>MasterSource</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Specified in <strong>MasterSource</strong>. Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td><strong>NonBlocking</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to execute a SQL statement and fetch rows by a separate thread.</td>
</tr>
<tr>
<td><strong>Options</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the behaviour of <strong>TOraDataSetObject</strong>.</td>
</tr>
<tr>
<td><strong>OptionsDS</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the behaviour of <strong>TOraDataSetObject</strong>.</td>
</tr>
<tr>
<td><strong>ParamCheck</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify whether parameters for the <strong>Params</strong> property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to indicate how many parameters are there in the <strong>Params</strong> property.</td>
</tr>
<tr>
<td><strong>Params</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Contains the parameters for a query's SQL statement.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>ReadOnly</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td><strong>RefreshMode</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify when to refresh an editing record.</td>
</tr>
<tr>
<td><strong>RefreshOptions</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>RowsProcessed</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the number of rows</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>(inherited from TOraDataSet) Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td>Session</td>
<td>(inherited from TOraDataSet) Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td>SmartFetch</td>
<td>(inherited from TOraDataSet) The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td>SQL</td>
<td>(inherited from TCustomDADataset) Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>(inherited from TCustomDADataset) Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td>SQLLock</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>(inherited from TCustomDADataset) Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>(inherited from TOraDataSet) Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>StrictUpdate</strong></td>
<td>Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td><strong>UniDirectional</strong></td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td><strong>UpdateObject</strong></td>
<td>Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong></td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong></td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddWhere</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td><strong>ApplyRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateBlobStream</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td>CreateProcCall</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td>DeferredPost</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>DeleteWhere</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td>Returns the parse error offset.</td>
</tr>
<tr>
<td>Execute</td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td>Executing</td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td>Fetched</td>
<td>Used to find out whether TCustomDADataset has fetched all rows.</td>
</tr>
<tr>
<td>Fetching</td>
<td>Used to learn whether TCustomDADataset is still fetching rows.</td>
</tr>
<tr>
<td>FetchingAll</td>
<td>Used to learn whether TCustomDADataset is fetching all rows to the end.</td>
</tr>
<tr>
<td>FindKey</td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>FindNearest</td>
<td>Moves the cursor to a specific record or to the first record in the dataset that</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>FindParam</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraArray object for a field when only its name is known.</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetDataType</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td><strong>GetErrorPos</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Returns a multireference shared object from field.</td>
</tr>
<tr>
<td><strong>GetFieldPrecision</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td><strong>GetFieldScale</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td><strong>GetFile</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetInterval</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetKeyFieldNames</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td><strong>GetKeyList</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the list of table primary key fields.</td>
</tr>
<tr>
<td><strong>GetLob</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetLobLocator</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetObject</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetOrderBy</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves an ORDER BY</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>GetRef</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetTable</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetTime Stamp</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Locate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LocateEx</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Lock</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>MacroByName</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>OpenNext</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>ParamByName</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Prepare</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RefreshRecord</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RestoreSQL</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>cached updates are enabled</code></td>
<td></td>
</tr>
<tr>
<td><code>SaveSQL</code> (inherited from <code>TCustomDADataSet</code>)</td>
<td>Saves the SQL property value to <code>BaseSQL</code>.</td>
</tr>
<tr>
<td><code>SaveToXML</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><code>SetOrderBy</code> (inherited from <code>TCustomDADataSet</code>)</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
<tr>
<td><code>SetRange</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><code>SetRangeEnd</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><code>SetRangeStart</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><code>SQLSaved</code> (inherited from <code>TCustomDADataSet</code>)</td>
<td>Determines if the SQL property value was saved to the <code>BaseSQL</code> property.</td>
</tr>
<tr>
<td><code>Unlock</code> (inherited from <code>TCustomDADataSet</code>)</td>
<td>Releases a record lock.</td>
</tr>
<tr>
<td><code>UnPrepare</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><code>UpdateResult</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td><code>UpdateStatus</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

**Events**
### Name | Description
--- | ---
**AfterExecute** (inherited from `TCustomDADataset`) | Occurs after a component has executed a query to database.

**AfterFetch** (inherited from `TCustomDADataset`) | Occurs after dataset finishes fetching data from server.

**AfterUpdateExecute** (inherited from `TCustomDADataset`) | Occurs after executing insert, delete, update, lock and refresh operations.

**BeforeFetch** (inherited from `TCustomDADataset`) | Occurs before dataset is going to fetch block of records from the server.

**BeforeUpdateExecute** (inherited from `TCustomDADataset`) | Occurs before executing insert, delete, update, lock, and refresh operations.

**OnUpdateError** (inherited from `TMemDataSet`) | Occurs when an exception is generated while cached updates are applied to a database.

**OnUpdateRecord** (inherited from `TMemDataSet`) | Occurs when a single update component can not handle the updates.

© 1997-2021 Devart. All Rights Reserved.

### 5.15.1.4 `TOraChangeNotification` Class

A component for keeping information in local dataset up-to-date through receiving notifications.

For a list of all members of this type, see `TOraChangeNotification` members.

**Unit**

`Ora`

**Syntax**

```pascal
TOraChangeNotification = class(TComponent);
```

**Remarks**

The `TOraChangeNotification` component is used to register queries with the database and receive notifications in response to DML or DDL changes on the objects associated with...
queries. The notifications are published by database when the DML or DDL transaction commits.

See Also
- **TOraChangeNotification Component**

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Indicates whether there is an active dataset connected with the TOraChangeNotification component.</td>
</tr>
<tr>
<td>Enabled</td>
<td>Used to enable or disable using change notification.</td>
</tr>
<tr>
<td>Operations</td>
<td>Used to be notified of the particular operations execution.</td>
</tr>
<tr>
<td>Persistent</td>
<td>Used to store notifications in a database persistently.</td>
</tr>
<tr>
<td>TimeOut</td>
<td>Indicates the interval for a notification to remain active.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveRegistration</td>
<td>Removes the change notification registration for all open datasets, connected with the TOraChangeNotification component.</td>
</tr>
</tbody>
</table>

### Events
5.15.1.4.2 Properties

Properties of the `TOraChangeNotification` class.

For a complete list of the `TOraChangeNotification` class members, see the `TOraChangeNotification Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Indicates whether there is an active dataset connected with the <code>TOraChangeNotification</code> component.</td>
</tr>
<tr>
<td>Persistent</td>
<td>Used to store notifications in a database persistently.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Used to enable or disable using change notification.</td>
</tr>
<tr>
<td>Operations</td>
<td>Used to be notified of the particular operations execution.</td>
</tr>
<tr>
<td>TimeOut</td>
<td>Indicates the interval for a notification to remain active.</td>
</tr>
</tbody>
</table>

See Also
- `TOraChangeNotification Class`
- `TOraChangeNotification Class Members`
5.15.1.4.2.1 Active Property

Indicates whether there is an active dataset connected with the TOraChangeNotification component.

Class

TOraChangeNotification

Syntax

```property
Active: boolean;
```

Remarks

Set the Enabled property to False to disable change notification for all datasets connected to the TOraChangeNotification component. Setting this property to True allows datasets, connected to the TOraChangeNotification component, to use change notification.

5.15.1.4.2.2 Enabled Property

Used to enable or disable using change notification.

Class

TOraChangeNotification

Syntax

```property
Enabled: boolean default True;
```

Remarks

Set the Enabled property to False to disable change notification for all datasets connected to the TOraChangeNotification component. Setting this property to True allows datasets, connected to the TOraChangeNotification component, to use change notification.

5.15.1.4.2.3 Operations Property

Used to be notified of the particular operations execution.

Class

TOraChangeNotification
Syntax

```property` Operations: TChangeNotifyDMLOperations `default`
[cnoInsert, cnoUpdate, cnoDelete];```

Remarks

Set the Operations property to provide a notification when particular operations are being executed.

Class

`TOraChangeNotification`

Syntax

```property` Persistent: boolean;```

Remarks

If True, notifications will be stored persistently in a database and would not be lost if server instance crashes after generating notifications, and they would be sent after Oracle server restart.

Class

`TOraChangeNotification`

Syntax

```property` TimeOut: integer `default` 0;```
Remarks
Set the TimeOut property to determine time interval in seconds, after which the notification registration will expire.

The minimum value is 1 second, maximum is $2^{31}-1$ seconds.

Methods of the **TOraChangeNotification** class.

For a complete list of the **TOraChangeNotification** class members, see the **TOraChangeNotification Members** topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RemoveRegistration</td>
<td>Removes the change notification registration for all open datasets, connected with the TOraChangeNotification component.</td>
</tr>
</tbody>
</table>

See Also
- **TOraChangeNotification Class**
- **TOraChangeNotification Class Members**

Removes the change notification registration for all open datasets, connected with the TOraChangeNotification component.

Class
**TOraChangeNotification**
## Syntax

```plaintext
procedure RemoveRegistration(Session: TOraSession);
```

### Parameters

- **Session**

### Remarks

Use the `RemoveRegistration` method to remove the change notification registration for all open datasets, connected with the `TOraChangeNotification` component. To register the change notification again, reopen all required datasets.

---

### Published

#### Events of the `TOraChangeNotification` class.

Events of the `TOraChangeNotification` class.

For a complete list of the `TOraChangeNotification` class members, see the `TOraChangeNotification Members` topic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnChange</td>
<td>Occurs when data in one of the associated datasets was changed on the server.</td>
</tr>
</tbody>
</table>

#### See Also

- `TOraChangeNotification Class`
- `TOraChangeNotification Class Members`

---

© 1997-2021 Devart. All Rights Reserved.

Request Support   DAC Forum   Provide Feedback

---

© 2021 Devart

5.15.1.4.1 OnChange Event

Occurs when data in one of the associated datasets was changed on the server.

Class
**TOraChangeNotification**

**Syntax**

```property```
OnChange: TOraChangeNotificationEvent;
```property```

**Remarks**

The OnChange event occurs when data in one of the associated datasets has been changed on the server. To receive change notifications the **Enabled** property must be set to True. The NotifyType parameter contains the type of the event occurred. The TableChanges parameter contains information on all table changes.

**See Also**

- **Enabled**
- **TOraChangeNotification Component**

© 1997-2021 Devart. All Rights Reserved.

5.15.1.5 **TOraChangeNotificationOptions Class**

Used to specify how query notifications will be generated.

For a list of all members of this type, see **TOraChangeNotificationOptions** members.

**Unit**

*Ora*

**Syntax**

```TOraChangeNotificationOptions = class (TPersistent);```
### QueryResultOnly Property

If the property is set to True, query level granularity is required. Notification should be only generated if the query result set changes.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QueryResultOnly</td>
<td>If the property is set to True, query level granularity is required. Notification should be only generated if the query result set changes.</td>
</tr>
</tbody>
</table>

See Also

- TOraChangeNotificationOptions Class
- TOraChangeNotificationOptions Class Members

© 1997-2021
Devart. All Rights Reserved.

© 2021 Devart
property QueryResultOnly: boolean default false;

5.15.1.6 TораConnectionSSLOptions Class

A class for setting up the SSL options.

For a list of all members of this type, see TораConnectionSSLOptions members.

Unit

Ora

Syntax

TораConnectionSSLOptions = class (TDAConnectionSSLOptions);

Inheritance Hierarchy

TDAConnectionSSLOptions
   TораConnectionSSLOptions

5.15.1.6.1 Members

TораConnectionSSLOptions class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACert</td>
<td>Holds the path to the certificate authority file.</td>
</tr>
<tr>
<td>Cert</td>
<td>Holds the path to the client certificate.</td>
</tr>
<tr>
<td>CipherList</td>
<td>Holds the list of allowed SSL ciphers.</td>
</tr>
<tr>
<td>Key</td>
<td>Holds the path to the private client key.</td>
</tr>
<tr>
<td>ServerCertDN</td>
<td>Used to specify the server’s distinguished name (DN).</td>
</tr>
</tbody>
</table>
Properties of the `TOraConnectionSSLOptions` class.

For a complete list of the `TOraConnectionSSLOptions` class members, see the `TOraConnectionSSLOptions Members` topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CACert (inherited from <code>TDACConnectionSSLOptions</code>)</td>
<td>Holds the path to the certificate authority file.</td>
</tr>
<tr>
<td>Cert (inherited from <code>TDACConnectionSSLOptions</code>)</td>
<td>Holds the path to the client certificate.</td>
</tr>
<tr>
<td>CipherList (inherited from <code>TDACConnectionSSLOptions</code>)</td>
<td>Holds the list of allowed SSL ciphers.</td>
</tr>
<tr>
<td>Key (inherited from <code>TDACConnectionSSLOptions</code>)</td>
<td>Holds the path to the private client key.</td>
</tr>
<tr>
<td>ServerCertDN</td>
<td>Used to specify the server's distinguished name (DN).</td>
</tr>
</tbody>
</table>

See Also
- `TOraConnectionSSLOptions Class`
- `TOraConnectionSSLOptions Class Members`

ServerCertDN Property

Used to specify the server's distinguished name (DN).

Class

`TOraConnectionSSLOptions`

Syntax

```pascal
property ServerCertDN: string;
```
Remarks

Use the ServerCertDN property so specify the server's distinguished name (DN) to enable server DN matching. It checks whether the server is genuine by matching the server's global database name against the DN from the server certificate.

5.15.1.7 TOraDataSet Class

A class defining the Oracle functionality for a dataset.

For a list of all members of this type, see TOraDataSet members.

Unit

Ora

Syntax

TOraDataSet = class(TCustomDADataset);

Remarks

TOraDataSet is a component that defines the Oracle functionality for a dataset. TOraDataSet can execute queries, fetch rows and control Oracle specific data types. Applications never use TOraDataSet objects directly. Instead they use descendants of TOraDataSet, such as TOraQuery, TSmartQuery, TOraStoredProc, and TOraTable, which inherit its database-related properties and methods.

Inheritance Hierarchy

TMemDataSet
  TCustomDADataset
  TOraDataSet

See Also

- TOraQuery
- TSmartQuery
- TOraStoredProc
- TOraTable
### TOraDataSet class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>CheckMode</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td>CommandTimeout</td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td>Conditions (inherited from TCustomDADataSet)</td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td>Connection (inherited from TCustomDADataSet)</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td>Cursor</td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td>DataTypeMap (inherited from TCustomDADataSet)</td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td>Debug (inherited from TCustomDADataSet)</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td>DetailFields (inherited from TCustomDADataSet)</td>
<td>Used to specify the fields</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Disconnected</td>
<td>(inherited from TCustomDADataSet) Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td>DMLRefresh</td>
<td>Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td>Encryption</td>
<td>Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td>FetchAll</td>
<td>Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td>FetchRows</td>
<td>(inherited from TCustomDADataSet) Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td>FilterSQL</td>
<td>(inherited from TCustomDADataSet) Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td>FinalSQL</td>
<td>(inherited from TCustomDADataSet) Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>(inherited from TMemDataSet) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>IsPLSQL</td>
<td>Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td>IsQuery</td>
<td>Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>(inherited from TMemDataSet) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>KeyFields</td>
<td>Used to build SQL statements for the that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KeySequence</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>(inherited from <code>TMemDataSet</code>) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>(inherited from <code>TMemDataSet</code>) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>LockMode</td>
<td>Used to define when to perform the locking of an editing record.</td>
</tr>
<tr>
<td>MacroCount</td>
<td>(inherited from <code>TCustomDADataset</code>) Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros</td>
<td>(inherited from <code>TCustomDADataset</code>) Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>MasterFields</td>
<td>(inherited from <code>TCustomDADataset</code>) Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td>MasterSource</td>
<td>(inherited from <code>TCustomDADataset</code>) Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>Used to execute a SQL statement and fetch rows by a separate thread.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td>OptionsDS</td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ParamCheck</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to indicate how many parameters are there in the Params property.</td>
</tr>
<tr>
<td><strong>Params</strong></td>
<td>Contains the parameters for a query's SQL statement.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>ReadOnly</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td><strong>RefreshMode</strong></td>
<td>Used to specify when to refresh an editing record.</td>
</tr>
<tr>
<td><strong>RefreshOptions</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong></td>
<td>Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>RowsProcessed</strong></td>
<td>Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td><strong>SequenceMode</strong></td>
<td>Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td><strong>SmartFetch</strong></td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
</tbody>
</table>
| **SQL** (inherited from **TCustomDADataSet**) | Used to provide a SQL statement that a query
<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLDelete</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td>SQLLock</td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td>UpdateObject</td>
<td>Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from TMemDataSet)</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong> (inherited from TMemDataSet)</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddWhere</strong> (inherited from TCustomDADataSet)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td><strong>ApplyRange</strong> (inherited from TMemDataSet)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from TMemDataSet)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong> (inherited from TCustomDADataSet)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from TMemDataSet)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from TMemDataSet)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from TMemDataSet)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong> (inherited from TCustomDADataSet)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong></td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong> (inherited from TMemDataSet)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>DeleteWhere</strong> (inherited from TCustomDADataSet)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong></td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>ErrorOffset</strong></td>
<td>Returns the parse error offset.</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>Used to find out whether TCustomDADataSet has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>Used to learn whether TCustomDADataSet is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>Used to learn whether TCustomDADataSet is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong></td>
<td>Retrieves a TOraArray object for a field when only its name is known.</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetDataType</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td><strong>GetErrorPos</strong></td>
<td>Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Returns a multireference shared object from field.</td>
</tr>
<tr>
<td><strong>GetFieldPrecision</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td><strong>GetFieldScale</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td><strong>GetFile</strong></td>
<td>Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetInterval</strong></td>
<td>Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetKeyFieldNames</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td><strong>GetKeyList</strong></td>
<td>Returns the list of table primary key fields.</td>
</tr>
<tr>
<td><strong>GetLob</strong></td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetLobLocator</strong></td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetObject</strong></td>
<td>Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetOrderBy</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td><strong>GetRef</strong></td>
<td>Retrieves a TOraRef object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetTable</strong></td>
<td>Retrieve a TOraNestTable object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetTimeStamp</strong></td>
<td>Retrieves a TOraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td><strong>GotoCurrent</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Sets the current record in this dataset similar to the current record in another</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Locate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LocateEx</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Lock</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>MacroByName</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>OpenNext</td>
<td></td>
</tr>
<tr>
<td>ParamByName</td>
<td></td>
</tr>
<tr>
<td>Prepare</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RefreshRecord</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RestoreSQL</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>SaveSQL</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>SetOrderBy</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>SetRange</td>
<td>TMemDataSet</td>
</tr>
</tbody>
</table>
applies it.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetRangeEnd (inherited from TMemDataSet)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SetRangeStart (inherited from TMemDataSet)</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SQLSaved (inherited from TCustomDADataset)</td>
<td>Determines if the SQL property value was saved to the BaseSQL property.</td>
</tr>
<tr>
<td>UnLock (inherited from TCustomDADataset)</td>
<td>Releases a record lock.</td>
</tr>
<tr>
<td>UnPrepare (inherited from TMemDataSet)</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>UpdateResult (inherited from TMemDataSet)</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus (inherited from TMemDataSet)</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute (inherited from TCustomDADataset)</td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td>AfterFetch (inherited from TCustomDADataset)</td>
<td>Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td>AfterUpdateExecute (inherited from TCustomDADataset)</td>
<td>Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
<tr>
<td>BeforeFetch (inherited from TCustomDADataset)</td>
<td>Occurs before dataset is going to fetch block of records from the server.</td>
</tr>
</tbody>
</table>
5.15.1.7.2 Properties

Properties of the TOracleDataSet class.

For a complete list of the TOracleDataSet class members, see the TOracleDataSet Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>CheckMode</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td>CommandTimeout</td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>Symbol</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>Conditions</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td><strong>Connection</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong></td>
<td>Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>FetchAll</strong></td>
<td>Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td><strong>FetchRows</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>IsPLSQL</td>
<td>Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td>IsQuery</td>
<td>Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>KeyFields</td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td>KeySequence</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>LockMode</td>
<td>Used to define when to perform the locking of an editing record.</td>
</tr>
<tr>
<td>MacroCount</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>MasterFields</td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>MasterSource</strong></td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td><strong>NonBlocking</strong></td>
<td>Used to execute a SQL statement and fetch rows by a separate thread.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td><strong>OptionsDS</strong></td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td><strong>ParamCheck</strong></td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong></td>
<td>Used to indicate how many parameters are there in the Params property.</td>
</tr>
<tr>
<td><strong>Params</strong></td>
<td>Contains the parameters for a query's SQL statement.</td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong></td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>ReadOnly</strong></td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td><strong>RefreshMode</strong></td>
<td>Used to specify when to refresh an editing record.</td>
</tr>
<tr>
<td><strong>RefreshOptions</strong></td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong></td>
<td>Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong></td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td>SmartFetch</td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td>SQLLock</td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>Used to specify a SQL statement.</td>
</tr>
</tbody>
</table>
### ChangeNotification Property

Used to receive database change notification messages to refresh dataset when required.

<table>
<thead>
<tr>
<th><strong>Class</strong></th>
<th><strong>Syntax</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOraDataSet</strong></td>
<td><code>property ChangeNotification: TOraChangeNotification;</code></td>
</tr>
</tbody>
</table>

### Remarks

**StrictUpdate**
- Statement that will be used when applying an update to a dataset.
- Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.

**UniDirectional** (inherited from [TCustomDADataset](#))
- Used if an application does not need bidirectional access to records in the result set.

**UpdateObject**
- Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

**UpdateRecordTypes** (inherited from [TMemDataSet](#))
- Used to indicate the update status for the current record when cached updates are enabled.

**UpdatesPending** (inherited from [TMemDataSet](#))
- Used to check the status of the cached updates buffer.
Use the ChangeNotification property to associate the component with the TOraChangeNotification component to receive database change notification messages to refresh dataset when required.

See Also
- TOraChangeNotification
- TOraChangeNotification Component
- Options

5.15.1.7.2.2 CheckMode Property

Used to define the check mode before editing a record.

Class
TOraDataSet

Syntax

```
property CheckMode: TCheckMode default cmNone;
```

Remarks

Use the CheckMode property to define the check mode before editing a record. Checking records is useful in creating concurrent multi-user applications. Set CheckMode to specify what action to take when another user makes modifications to a record. TOraDataSet first refetches record values and compares them with those of a client.

5.15.1.7.2.3 CommandTimeout Property

Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.

Class
TOraDataSet
### Cursor Property

**Syntax**

```
property CommandTimeout: integer default 0;
```

**Remarks**

The wait time is specified in seconds to wait for the command to execute. The default value is 0. The value of 0 indicates there are no time limits (an attempt to execute a command will wait indefinitely).

---

**Class**

**TOraDataSet**

**Syntax**

```
property Cursor: TOraCursor;
```

**Remarks**

Use the Cursor property to fetch data from the cursor parameter and cursor field in Oracle 8. You can assign the value of TOraParam.AsCursor or TCursorField.AsCursor to the Cursor property. After assigning you can open the dataset once.

**Example**

```
OraQuery1.Cursor := OraSQL1.ParamByName('Cur').AsCursor;
OraQuery1.Open;
```

**See Also**

- **TOraCursor**
- **TOraParam.AsCursor**
5.15.1.7.2.5 DMLRefresh Property

Used to refresh record by RETURNING clause when insert or update is performed.

Class

T0raDataSet

Syntax

property DMLRefresh: boolean;

Remarks

Use the DMLRefresh property to refresh record by RETURNING clause when insert or update is performed. This feature is only for Oracle 8.

The default value is False.

Note: When the DMLRefresh property is set to True, the value of TCustomDADataset.RefreshOptions is ignored to avoid refetching field values from the server.

5.15.1.7.2.6 Encryption Property

Used to specify encryption options in a dataset.

Class

T0raDataSet

Syntax

property Encryption: T0raEncryption;

Remarks

Set the Encryption options for using encryption in a dataset.
5.15.1.7.2.7 FetchAll Property

Used to request all records of the query from database server when a dataset is being opened.

Class

T0raDataSet

Syntax

property FetchAll: boolean;

Remarks

When set to True, all records of the query are requested from database server when a dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to the TMemDataSet.Locate and TMemDataSet.LocateEx methods may take a lot of time to retrieve additional records to the client side.

5.15.1.7.2.8 IsPLSQL Property

Indicates whether a SQL statement is a PL/SQL block.

Class

T0raDataSet

Syntax

property IsPLSQL: boolean;

Remarks

Use the IsPLSQL property to check whether a SQL statement is a PL/SQL block. T0raDataSet must be prepared beforehand.

IsPLSQL is a read-only property.
### 5.15.1.7.2.9 IsQuery Property

Indicates whether SQL statement returns rows or not.

**Class**

**TOraDataSet**

**Syntax**

```plaintext
property IsQuery: boolean;
```

**Remarks**

When the TOraDataSet component is prepared, it returns True. If SQL statement is SELECT or PL/SQL block it returns the REF CURSOR parameter.

Use the IsQuery property to check whether SQL statement returns rows or not. TOraDataSet returns rows when SQL statement is SELECT or PL/SQL block with the REF CURSOR parameter. TOraDataSet must be prepared beforehand.

IsQuery is a read-only property.

**See Also**

- IsPLSQL

---

### 5.15.1.7.10 KeyFields Property

Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.

**Class**

**TOraDataSet**
Syntax

```property`` KeyFields: `string`;
```

Remarks

Assign the KeyFields property to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database. To exploit this feature KeyFields may hold a list of semicolon-delimited field names. If KeyFields was not defined before opening the dataset, ROWID pseudo fields are used.

Besides, the KeyFields property may hold the name of a field which will be later assigned with Oracle sequenced values. Beforehand Oracle sequence must be created and its name passed to the `KeySequence` property.

Sequences are generated when either TDataSet.Insert or TDataSet.Post method is called. Which of these two methods is used to modify the database is determined by the `SequenceMode` property.

**Note:** Although keys may be created across a number of table fields, sequence is generated only for the first field found in the KeyFields property.

See Also

- `KeySequence`
- `SequenceMode`
- `TCustomDADataset.SQLDelete`
- `TCustomDADataset.SQLInsert`
- `TCustomDADataset.SQLRefresh`
- `TCustomDADataset.SQLUpdate`

© 1997-2021 Devart. All Rights Reserved.

5.15.1.7.2.11 KeySequence Property

Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.

Class

`TOraDataSet`
Syntax

```plaintext
property KeySequence: string;
```

Remarks

Use the KeySequence property to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.

**Note:** KeySequence is used by TOraDataSet only if the `KeyFields` property is assigned.

Example

Here is an example of PL/SQL block generated by TOraDataSet:

```sql
begin
    SELECT DEPT_SEQ.NEXTVAL
    INTO :DEPTNO
    FROM Dual;
    INSERT INTO DEPT
    (DEPTNO, DNAME)
    VALUES
    (:DEPTNO, :DNAME);
end;
```

See Also

- `SequenceMode`

Used to define when to perform the locking of an editing record.

Class

`TOraDataSet`

Syntax

```plaintext
property LockMode: TLockMode;
```

Remarks

Use the LockMode property to define when to perform the locking of an editing record. Locking a record is useful when creating multi-user applications. It prevents the possibility of several
users modifying a record at the same time. Locking is realized through the execution of SELECT FOR UPDATE NOWAIT statement.

Locking is performed by the RefreshRecord method.

To set pessimistic locking use LockMode = lmLockImmediate, CheckMode = cmException.
To set optimistic locking use LockMode = lmLockDelayed, CheckMode = cmException.

The default value is lmNone.

See Also
- TCustomDADataset.Lock
- TCustomDADataset.SQLLock
- CheckMode

5.15.1.7.2.13 NonBlocking Property

Used to execute a SQL statement and fetch rows by a separate thread.

Class

T0raDataSet

Syntax

```
property NonBlocking: boolean;
```

Remarks

Set the NonBlocking property to True to execute SQL statement and fetch rows by a separate thread.

5.15.1.7.2.14 Options Property

Used to specify the behaviour of T0raDataSetObject.

Class

T0raDataSet
Syntax

```objectivec
property Options: TOraDataSetOptions;
```

Remarks

Set the properties of Options to specify the behaviour of a TOraDataSet object.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoClose</td>
<td>Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td>CacheLobs</td>
<td>Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td>DefaultValues</td>
<td>Used for TOraDataSet to fill the DefaultExpression property of TField objects by appropriate value.</td>
</tr>
<tr>
<td>DeferredLobRead</td>
<td>Used to fetch all Oracle 8 Lob values when they are explicitly requested.</td>
</tr>
<tr>
<td>EnableBCD</td>
<td>Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td>EnableFMTBCD</td>
<td>Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td>ExtendedFieldsInfo</td>
<td>Used to perform an additional query to get information about returned fields and the tables they belong to.</td>
</tr>
<tr>
<td>FieldsAsString</td>
<td>Used to treat all non-BLOB fields as being of string datatype.</td>
</tr>
<tr>
<td>FullRefresh</td>
<td>Used to refresh fields of all tables by the RefreshRecord method.</td>
</tr>
<tr>
<td>PrefetchLobSize</td>
<td>Used to retrieve the LOB length and the LOB data beginning during regular fetch.</td>
</tr>
<tr>
<td>PrefetchRows</td>
<td>Used to get or set the number of rows that OCI prefetches when executing a query.</td>
</tr>
<tr>
<td>PrepareUpdateSQL</td>
<td>Used to automatically prepare update queries before execution.</td>
</tr>
<tr>
<td>ProcNamedParams</td>
<td>Used to specify a notation method of passing parameter values to the stored PL/SQL object.</td>
</tr>
<tr>
<td>RawAsString</td>
<td>Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td>ReflectChangeNotify</td>
<td>Used for a dataset component to refresh its</td>
</tr>
</tbody>
</table>

© 2021 Devart
Used to specify the behaviour of TораDataSetObject.

Class

TораDataSet

Syntax

```
property OptionsDS: TораDataSetOptionsDS stored False;
```

Remarks

Set the properties of OptionsDS to specify the behaviour of a TораDataSet object.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoClose</td>
<td>Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td><strong>CacheLobs</strong></td>
<td>Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DefaultValues</strong></td>
<td>Used for TOraDataSet to fill the DefaultExpression property of TField objects with the appropriate value.</td>
</tr>
<tr>
<td><strong>DeferredLobRead</strong></td>
<td>Used fetch all Oracle 8 Lob values only when they are explicitly requested.</td>
</tr>
<tr>
<td><strong>FieldsAsString</strong></td>
<td>Used to treat all non-BLOB fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>HideRowId</strong></td>
<td>Used to display the ROWID column.</td>
</tr>
<tr>
<td><strong>KeepPrepared</strong></td>
<td>Used to keep TOraDataSet prepared after closing.</td>
</tr>
<tr>
<td><strong>RawAsString</strong></td>
<td>Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>ScrollableCursor</strong></td>
<td>Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.</td>
</tr>
</tbody>
</table>

**See Also**

- TCustomDADataSet.Options

Contains the parameters for a query's SQL statement.

**Class**

TOraDataSet

**Syntax**

```
property Params: TOraParams stored False;
```

**Remarks**

The Params parameter contains the parameters for a query's SQL statement.

Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.
An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

See Also
- **TOraParam**
- **ParamByName**

© 1997-2021 Devart. All Rights Reserved.

5.15.1.7.2.17 RefreshMode Property

Used to specify when to refresh an editing record.

Class
**TOraDataSet**

Syntax

```
property RefreshMode: TRefreshMode stored False;
```

Remarks

Use the RefreshMode property to define when to perform a refresh of editing record. Refreshing a record is useful when a table has triggers or fields of a table have default value.

Refresh is performed by the RefreshRecord method.

The default value is rmNone.

**Note:** RefreshMode is obsolete, and only included for backward compatibility. Use `TCustomDADataSet.RefreshOptions` instead.

See Also
- **TCustomDADataSet.RefreshRecord**
- **TCustomDADataSet.SQLRefresh**
5.15.1.7.2.18 ReturnParams Property

Used to return a new fields value to dataset after insert or update.

Class

T0raDataSet

Syntax

```
property ReturnParams: boolean stored False;
```

Remarks

Use the ReturnParams property to return the new fields value to dataset after insert or update. The actual value of a field after insert or update may be different from the value stored in local memory if a table has a trigger.

When ReturnParams is True, OUT parameters of SQLInsert and SQLUpdate statements are assigned to the corresponding fields.

OUT parameters can have PL/SQL block or DML statements with the RETURNING clause (for Oracle 8 only).

The default value is False.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.7.2.19 RowsProcessed Property

Returns the number of rows processed by a query.

Class

T0raDataSet

Syntax

```
property RowsProcessed: integer;
```

Remarks

Use the RowsProcessed property to return the number of rows processed by a query. Useful for SELECT, UPDATE and DELETE statements. In case of SELECT statement, RowsProcessed increments by FetchRows.
5.15.1.7.2.20  SequenceMode Property

Used to specify the methods used internally to generate a sequenced field.

Class

TOraDataSet

Syntax

```
property SequenceMode: TSequenceMode default smPost;
```

Remarks

Set the SequenceMode property to specify which method is used internally to generate a sequenced field.

See Also

- KeyFields
- KeySequence

5.15.1.7.2.21  Session Property

Used to specify the session in which dataset will be executed.

Class

TOraDataSet

Syntax

```
property Session: TOraSession;
```

Remarks
Use the Session property to specify the session in which dataset will be executed. If Session is not connected, the Open method calls Session.Connect.

See Also
- TOraSession

5.15.1.7.2.22 SmartFetch Property

The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.

Class
TOraDataSet

Syntax

```pascal
property SmartFetch: TSmartFetchOptions;
```

See Also
- TSmartFetchOptions

5.15.1.7.2.23 SQLType Property

Used to get the typecode of the SQL statement being processed by Oracle database server.

Class
TOraDataSet

Syntax

```pascal
property SQLType: integer;
```

Remarks
Read the SQLType property to get the typecode of the SQL statement being processed by Oracle database server.
5.15.1.7.2.24 StrictUpdate Property

Used for T0raDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.

Class

T0raDataSet

Syntax

```plaintext
property StrictUpdate: boolean stored False;
```

Remarks

When True, T0raDataSet raises the 'Update failed' exception when the number of updated or deleted records are not equal 1. The exception does not occur when you use a PL/SQL block. The default value is True.

Note: StrictUpdate is obsolete, and included for backward compatibility only. Use TCustomDADataset.Options instead.

5.15.1.7.2.25 UpdateObject Property

Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

Class

T0raDataSet

Syntax

```plaintext
property UpdateObject: T0raUpdateSQL;
```

Remarks

The UpdateObject property specifies an update object component which provides SQL
Methods of the `TOraDataSet` class.

For a complete list of the `TOraDataSet` class members, see the [TOraDataSet Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddWhere</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td><strong>ApplyRange</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong></td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Makes permanent changes</td>
</tr>
<tr>
<td>Method</td>
<td>Base Class</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>DeleteWhere</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td></td>
</tr>
<tr>
<td>Execute</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Executing</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Fetched</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Fetching</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>FetchingAll</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>FindKey</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>FindMacro</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>FindNearest</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>FindParam</td>
<td></td>
</tr>
<tr>
<td>GetArray</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>(inherited from)</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
</tr>
<tr>
<td>GetBlob</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>GetDataType</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>GetErrorPos</td>
<td></td>
</tr>
<tr>
<td>GetFieldObject</td>
<td></td>
</tr>
<tr>
<td>GetFieldPrecision</td>
<td></td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>GetFile</td>
<td></td>
</tr>
<tr>
<td>GetInterval</td>
<td></td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>GetKeyList</td>
<td></td>
</tr>
<tr>
<td>GetLob</td>
<td></td>
</tr>
<tr>
<td>GetLobLocator</td>
<td></td>
</tr>
<tr>
<td>GetObject</td>
<td></td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>GetRef</td>
<td></td>
</tr>
<tr>
<td>GetTable</td>
<td></td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Returns/Details</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GotoCurrent</strong></td>
<td>ToraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td></td>
<td>Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td><strong>Locate</strong></td>
<td>Overloaded. Searches a dataset for a specific record and positions the specific record on it.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TMemDataSet)</td>
</tr>
<tr>
<td><strong>LocateEx</strong></td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TMemDataSet)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Locks the current record.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>MacroByName</strong></td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>OpenNext</strong></td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>ParamByName</strong></td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>Prepare</strong></td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>RefreshRecord</strong></td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>RestoreSQL</strong></td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong></td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TMemDataSet)</td>
</tr>
<tr>
<td><strong>RevertRecord</strong></td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TMemDataSet)</td>
</tr>
<tr>
<td><strong>SaveSQL</strong></td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td></td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td><strong>SaveToXML</strong></td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO</td>
</tr>
<tr>
<td></td>
<td>(inherited from TMemDataSet)</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td><code>SetOrderBy</code></td>
<td><code>TCustomDADataSet</code></td>
</tr>
<tr>
<td><code>SetRange</code></td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td><code>SetRangeEnd</code></td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td><code>SetRangeStart</code></td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td><code>SQLSaved</code></td>
<td><code>TCustomDADataSet</code></td>
</tr>
<tr>
<td><code>UnLock</code></td>
<td><code>TCustomDADataSet</code></td>
</tr>
<tr>
<td><code>UnPrepare</code></td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td><code>UpdateResult</code></td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td><code>UpdateStatus</code></td>
<td><code>TMemDataSet</code></td>
</tr>
</tbody>
</table>

See Also
- `TOraDataSet Class`
- `TOraDataSet Class Members`
5.15.1.7.3.1 CreateProcCall Method

Generates the stored procedure call.

Class

T0raDataSet

Syntax

procedure CreateProcCall(Name: string; Overload: integer = 0);

Parameters

Name

Holds the name of the stored procedure.

Overload

Holds the number of the overloaded procedure.

Remarks

Call the CreateProcCall method to assign a PL/SQL block that calls stored procedure specified by Name to the SQL property. Overload parameter must contain the number of the overloaded procedure. Retrieves the information about the procedure parameters from Oracle. After calling CreateProcCall you can execute stored procedure by the Execute method.

See Also

- TCustomDADataSet.Execute
- TCustomDAConnection.ExecProc
- T0raStoredProc

© 1997-2021
Devart. All Rights Reserved.
Request Support DAC Forum Provide Feedback

5.15.1.7.3.2 ErrorOffset Method

Returns the parse error offset.

Class

T0raDataSet

Syntax
function ErrorOffset: integer;

Return Value
the parse error offset.

Remarks
Call the ErrorOffset method to return the parse error offset for a SQL statement. Check ErrorOffset after TOraDataSet raises an exception.

See Also
• GetErrorPos

Determines whether a parameter with the specified name exists in a dataset.

Class
TOraDataSet

Syntax
function FindParam(const Value: string): TParam;

Parameters
Value
Holds the name of the param to search for.

Return Value
a TParam object for the specified Name.

Remarks
Call the FindParam method to determine if parameter with the specified name exists in a dataset. Name is the name of the param for which to search. If FindParam finds a param with a matching name, it returns a TParam object for the specified Name. Otherwise it returns nil.

See Also
• Params
• ParamByName
5.15.1.7.3.4  GetArray Method

Retrieves a T OrionArray object for a field when only its name is known.

Class
T OrionDataSet

Syntax

```pascal
function GetArray(const FieldName: string): T OrionArray;
```

Parameters

FieldName
Holds the name of an existing field.

Return Value

a T OrionArray object for a field with known name.

Remarks

Call the GetArray method to retrieve a T OrionArray object for a field when only its name is known. FieldName is the name of an existing field. The field should have the ftArray type.

See Also

- T OrionArray
- T OrionDataSet.GetData Type
- T OrionParam.AsArray

5.15.1.7.3.5  GetErrorPos Method

Returns a row and column of parse error for a SQL statement.

Class
T OrionDataSet

Syntax
**procedure** `getErrorPos(var Row: integer; var Col: integer);`  

**Parameters**

*Row*
Holds the row number.

*Col*
Holds the column number.

**Remarks**
Call the `getErrorPos` method to return a row and column of parse error for a SQL statement. Use `getErrorPos` after `TData` raises an exception.

**See Also**
- `ErrorOffset`

5.15.1.7.3.6 GetFile Method

Retrieves a `TData` object for a field with known name.

**Class**
`TData`

**Syntax**

```delphi
function GetFile(const FieldName: string): TData;```

**Parameters**

*FieldName*
Holds the name of an existing field.

**Return Value**

A `TData` object for a field with known name.

**Remarks**
Call the `GetFile` method to retrieve a `TData` object for a field when only its name is known. `FieldName` is the name of an existing field. The field should have `ftBFile`.

**See Also**
- `TData`
5.15.1.7.3.7  GetInterval Method

Retrieves a TOraInterval object for a field with known name.

Class

TOraDataSet

Syntax

function GetInterval(const FieldName: string): TOraInterval;

Parameters

FieldName
Holds the name of an existing field.

Return Value
a TOraInterval object for a field with known name.

Remarks

Call the GetInterval method to retrieve a TOraInterval object for a field when only its name is known. FieldName is the name of an existing field. The field should have ftIntervalYM or ftIntervalDS.

See Also

• TOraInterval
• TOraParam.AsInterval
**TOraDataSet**

**Syntax**

```pascal
function GetKeyList(TableName: string; List: TStrings): string;
```

**Parameters**

- `TableName`:
  - Holds the table name.

- `List`:
  - Holds the list of table primary key fields.

**Return Value**

- the list of table primary key fields.

**Remarks**

Call the `GetKeyList` method to get the list of table primary key fields.

---

**5.15.1.7.3.9 GetLob Method**

- Retrieves a `TOraLob` object for a field with known name.

**Class**

`TOraDataSet`

**Syntax**

```pascal
function GetLob(const FieldName: string): TOraLob;
```

**Parameters**

- `FieldName`:
  - Holds the name of an existing field.

**Return Value**

- a `TOraLob` object for a field with known name.

**Remarks**

Call the `GetLob` method to retrieve a `TOraLob` object for a field when only its name is known. `FieldName` holds the name of an existing field. The field should have the `ftOraClob` or `ftOraBlob` type.
5.15.1.7.3.10  GetLobLocator Method

Retrieves a TOraLob object for a field with known name.

Class

TOraDataSet

Syntax

function GetLobLocator(const FieldName: string): TOraLob;

Parameters

FieldName
Holds the name of an existing field.

Return Value

a TOraLob object for a field with known name.

Remarks

Call the GetLobLocator method to retrieve a TOraLob object for a field when only its name is known. FieldName is the name of an existing field. The field should have the ftOraClob or ftOraBlob type.

Note: GetLobLocator is an obsolete method. In newer projects call GetLob instead.

See Also

- GetLob
- TOraLob
- TCustomDADataset.GetDataType
- TOraParam.AsBLOBLocator
5.15.1.7.3.11 GetObject Method

Retrieves a TOraObject object for a field with known name.

Class

TOraDataSet

Syntax

function GetObject(const FieldName: string): TOraObject;

Parameters

FieldName
Holds the name of an existing field.

Return Value
a TOraObject object for a field with known name.

Remarks

Call the GetObject method to retrieve a TOraObject object for a field when only its name is known. FieldName is the name of an existing field. The field should have the ftObject, ftReference, ftArray or ftTable type.

See Also
- TOraObject
- TCustomDADataset.GetDataType
- TOraParam.AsObject

© 1997-2021 Devart. All Rights Reserved.

5.15.1.7.3.12 GetRef Method

Retrieves a TOraRef object for a field with known name.

Class

TOraDataSet

Syntax

function GetRef(const FieldName: string): TOraRef;

Parameters
**FieldName**
Holds the name of an existing field.

**Return Value**
a TOraRef object for a field with known name.

**Remarks**
Call the GetRef method to retrieve a TOraRef object for a field when only its name is known. FieldName is the name of an existing field. The field should have the ftReference type.

**See Also**
- TOraRef
- TCustomDADataset.GetDataType
- TOraNestedTable.Ref
- TOraParam.AsRef

© 1997-2021 Devart. All Rights Reserved.
5.15.1.7.3.14  GetTimeStamp Method

Retrieves a TOraTimeStamp object for a field with known name.

Class

TOraDataSet

Syntax

function GetTimeStamp(const FieldName: string): TOraTimeStamp;

Parameters

FieldName

Holds the name of an existing field.

Return Value

a TOraTimeStamp object for a field with known name.

Remarks

Call the GetTimeStamp method to retrieve a TOraTimeStamp object for a field when only its name is known. FieldName is the name of an existing field. The field should have ftTimeStamp.

See Also

• TOraTimeStamp
• TOraParam.AsTimeStamp
5.15.1.7.3.15 OpenNext Method

Opens next cursor or rowset in the statement.

Class

TOraDataSet

Syntax

function OpenNext: boolean;

Return Value

True, if DataSet opens.

Remarks

Call the OpenNext method to get the second and other Cursors or ResultSets while executing a query. If DataSet opens, it returns True. If there are no cursors or record sets to be represented, it will return False, and the current record set will be closed.

Example for working with cursors:

```pascal
OraQuery1.SQL.Text := 'BEGIN ' +
  '  OPEN :Cur1 FOR SELECT * FROM Dept; ' +
  '  OPEN :Cur2 FOR SELECT * FROM Emp; ' +
  'END;';
OraQuery1.Params[0].DataType := ftCursor;
OraQuery1.Params[0].ParamType := ptOutput;
OraQuery1.Params[1].DataType := ftCursor;
OraQuery1.Params[1].ParamType := ptOutput;
OraQuery1.Open;     // open first cursor
OraQuery1.OpenNext; // open next cursor
```

Example for working with Implicit Result Sets in Oracle 12c and higher:

```pascal
OraQuery1.SQL.Text := 'DECLARE ' +
  '  dept_cur SYS_REFCURSOR; ' +
  '  emp_cur SYS_REFCURSOR; ' +
  'BEGIN ' +
  '  OPEN dept_cur FOR SELECT * FROM Dept; ' +
  '  DBMS_SQL.RETURN_RESULT(dept_cur); ' +
  '  OPEN emp_cur FOR SELECT * FROM Emp; ' +
  '  DBMS_SQL.RETURN_RESULT(emp_cur); ' +
  'END;';
OraQuery1.Open;     // open first result set
OraQuery1.OpenNext; // open next result set
```
5.15.1.7.3.16  ParamByName Method

Sets or uses parameter information for a specific parameter based on its name.

Class

T0raDataSet

Syntax

function ParamByName(const Value: string): T0raParam;

Parameters

Value
holds the name of the parameter to retrieve information for.

Return Value

a object.

Remarks

Call the ParamByName method to set or use parameter information for a specific parameter based on its name. Name is the name of the parameter for which to retrieve information. ParamByName is used to set a parameter's value at runtime and returns a T0raParam object.

Example

The following statement retrieves the current value of a parameter called "Contact" into an edit box:

EditText := Query1.ParamsByName('Contact').AsString;

See Also

- T0raParam
- Params
- TCustomDADataset.FindParam
5.15.1.8  **TOraDataSetField Class**

A class providing access to Oracle nested datasets.

For a list of all members of this type, see [TOraDataSetField members](#).

**Unit**

**Ora**

**Syntax**

```plaintext
TOraDataSetField = class (TDataSetField);
```

**Remarks**

TOraDataSetField provides access to Oracle nested datasets.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.8.1  **Members**

**TOraDataSetField** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
<td>Indicates whether tafield was modified.</td>
</tr>
</tbody>
</table>

5.15.1.8.2  **Properties**

Properties of the **TOraDataSetField** class.

For a complete list of the **TOraDataSetField** class members, see the [TOraDataSetField Members](#) topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
<td>Indicates whether tafield was</td>
</tr>
</tbody>
</table>
5.15.1.8.2.1 Modified Property

Indicates whether the field was modified.

Class

TOraDataSetField

Syntax

```
property Modified: boolean;
```

Remarks

The Modified property indicates whether a field was modified. The property is writable.

5.15.1.9 TOraDataSetOptions Class

This class allows setting up the behaviour of the TOraDataSet class.

For a list of all members of this type, see TOraDataSetOptions members.

Unit

Ora

Syntax

```
TOraDataSetOptions = class(TOraDataSetOptionsDS);
```

Remarks

Cached Gets a value indicating whether Oracle resources associated with the current
statement will be cached inside a session. If you execute many different SELECT statements this option may significantly increase the performance of your application. But using this property you may easily step over maximum open cursors on Oracle server. And thus you must be attentive when using the Cached option. This option is only available with Oracle 9.2i and higher. It will work only if **TOraSession.Options** is set to True.

**Inheritance Hierarchy**

TDADatasetOptions  
  TOraDataSetOptionsDS  
  TOraDataSetOptions

© 1997-2021 Devart. All Rights Reserved.

Request Support  
DAC Forum  
Provide Feedback

5.15.1.9.1 Members

**TOraDataSetOptions** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoClose</strong></td>
<td>Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td><strong>AutoPrepare</strong></td>
<td>(inherited from TDADatasetOptions) Used to execute automatic TCustomDADataSet.Prepare on the query execution.</td>
</tr>
<tr>
<td><strong>CacheCalcFields</strong></td>
<td>(inherited from TDADatasetOptions) Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td><strong>CacheLobs</strong></td>
<td>Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td><strong>CompressBlobMode</strong></td>
<td>(inherited from TDADatasetOptions) Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td><strong>DefaultValues</strong></td>
<td>Used for TOraDataSet to fill the DefaultExpression property of TField objects by appropriate value.</td>
</tr>
<tr>
<td><strong>DeferredLobRead</strong></td>
<td>Used to fetch all Oracle 8 Lob values when they are explicitly requested.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DetailDelay</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td><strong>EnableBCD</strong></td>
<td>Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td><strong>EnableFMTBCD</strong></td>
<td>Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td><strong>ExtendedFieldsInfo</strong></td>
<td>Used to perform an additional query to get information about returned fields and the tables they belong to.</td>
</tr>
<tr>
<td><strong>FieldsAsString</strong></td>
<td>Used to treat all non-BLOB fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>FieldsOrigin</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.</td>
</tr>
<tr>
<td><strong>FlatBuffers</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to control how a dataset treats data of the ftString and ftVarBytes fields.</td>
</tr>
<tr>
<td><strong>FullRefresh</strong></td>
<td>Used to refresh fields of all tables by the RefreshRecord method.</td>
</tr>
<tr>
<td><strong>HideRowId</strong> (inherited from <strong>TOraDataSetOptionsDS</strong>)</td>
<td>Used to display the ROWID column.</td>
</tr>
<tr>
<td><strong>InsertAllSetFields</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to include all set dataset fields in the generated INSERT statement.</td>
</tr>
<tr>
<td><strong>KeepPrepared</strong> (inherited from <strong>TOraDataSetOptionsDS</strong>)</td>
<td>Used to keep TOraDataSet prepared after closing.</td>
</tr>
<tr>
<td><strong>LocalMasterDetail</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used for TCustomDADataset to use local filtering to establish master/detail relationship for</td>
</tr>
<tr>
<td>Reference 777</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>© 2021 Devart</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>LongStrings</strong> (inherited from <strong>TDADatasetOptions</strong>)</th>
<th>detail dataset and does not refer to the server.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MasterFieldsNullable</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to represent string fields with the length that is greater than 255 as T StringField.</td>
</tr>
<tr>
<td><strong>NumberRange</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).</td>
</tr>
<tr>
<td><strong>PrefetchLobSize</strong></td>
<td>Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.</td>
</tr>
<tr>
<td><strong>PrefetchRows</strong></td>
<td>Used to retrieve the LOB length and the LOB data beginning during regular fetch.</td>
</tr>
<tr>
<td><strong>PrepareUpdateSQL</strong></td>
<td>Used to get or set the number of rows that OCI prefetches when executing a query.</td>
</tr>
<tr>
<td><strong>ProcNamedParams</strong></td>
<td>Used to automatically prepare update queries before execution.</td>
</tr>
<tr>
<td><strong>QueryRecCount</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to specify a notation method of passing parameter values to the stored PL/SQL object.</td>
</tr>
<tr>
<td><strong>QuoteNames</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used for TCustomDADataset to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td><strong>PrepareUpdateSQL</strong></td>
<td>Used for TCustomDADataset to quote all database object names in autogenerated</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RawAsString</td>
<td>SQL statements such as update SQL. Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td>ReflectChangeNotify</td>
<td>Used for a dataset component to refresh its data when it gets database change notification messages in response to DML or DDL changes on the objects associated with the dataset query.</td>
</tr>
<tr>
<td>RemoveOnRefresh</td>
<td>(inherited from TDADatasetOptions) Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td>RequiredFields</td>
<td>(inherited from TDADatasetOptions) Used for TCustomDADataSet to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>(inherited from TDADatasetOptions) Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td>ScrollableCursor</td>
<td>Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.</td>
</tr>
<tr>
<td>SetFieldsReadOnly</td>
<td>(inherited from TDADatasetOptions) Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td>StatementCache</td>
<td>Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>(inherited from TDADatasetOptions) Used for TCustomDADataSet to raise an exception when the number of updated or deleted records is not equal 1.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TemporaryLobUpdate</strong></td>
<td>Temporary LOBs are used to write input and input/output LOB parameters into database when executing dataset's SQL statements.</td>
</tr>
<tr>
<td><strong>TrimFixedChar</strong>    (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Specifies whether to discard all trailing spaces in the string fields of a dataset.</td>
</tr>
<tr>
<td><strong>UpdateAllFields</strong>      (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to include all dataset fields in the generated UPDATE and INSERT statements.</td>
</tr>
<tr>
<td><strong>UpdateBatchSize</strong>  (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.</td>
</tr>
</tbody>
</table>

For a complete list of the **TOraDataSetOptions** class members, see the [**TOraDataSetOptions Members** topic](#).

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoPrepare</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to execute automatic [<strong>TCustomDADDataSet.Prepare</strong> on the query execution.</td>
</tr>
<tr>
<td><strong>CacheCalcFields</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td><strong>CompressBlobMode</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td><strong>DetailDelay</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset.</td>
</tr>
<tr>
<td>Field Name</td>
<td>Inherited From</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>FieldsOrigin</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>FlatBuffers</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>InsertAllSetFields</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>LocalMasterDetail</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>LongStrings</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>MasterFieldsNullable</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>NumberRange</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td>QueryRecCount</td>
<td>TDADDataSetOptions</td>
</tr>
<tr>
<td><strong>Property</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>QuoteNames</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for <strong>TCustomDADataset</strong> to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td><strong>RemoveOnRefresh</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td><strong>RequiredFields</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for <strong>TCustomDADataset</strong> to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>SetFieldsReadOnly</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td><strong>StrictUpdate</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for <strong>TCustomDADataset</strong> to raise an exception when the number of updated or deleted records is not equal 1.</td>
</tr>
<tr>
<td><strong>TrimFixedChar</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Specifies whether to discard all trailing spaces in the string fields of a dataset.</td>
</tr>
<tr>
<td><strong>UpdateAllFields</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to include all dataset fields in the generated UPDATE and INSERT statements.</td>
</tr>
<tr>
<td><strong>UpdateBatchSize</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutoClose</td>
<td>Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td>CacheLobs</td>
<td>Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td>DefaultValues</td>
<td>Used for TOraDataSet to fill the DefaultExpression property of TField objects by appropriate value.</td>
</tr>
<tr>
<td>DeferredLobRead</td>
<td>Used to fetch all Oracle 8 Lob values when they are explicitly requested.</td>
</tr>
<tr>
<td>EnableBCD</td>
<td>Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td>EnableFMTBCD</td>
<td>Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td>ExtendedFieldsInfo</td>
<td>Used to perform an additional query to get information about returned fields and the tables they belong to.</td>
</tr>
<tr>
<td>FieldsAsString</td>
<td>Used to treat all non-BLOB fields as being of string datatype.</td>
</tr>
<tr>
<td>FullRefresh</td>
<td>Used to refresh fields of all tables by the RefreshRecord method.</td>
</tr>
<tr>
<td>HideRowId (inherited from TOraDataSetOptionsDS)</td>
<td>Used to display the ROWID column.</td>
</tr>
<tr>
<td>KeepPrepared (inherited from TOraDataSetOptionsDS)</td>
<td>Used to keep TOraDataSet prepared after closing.</td>
</tr>
<tr>
<td>PrefetchLobSize</td>
<td>Used to retrieve the LOB length and the LOB data beginning during regular fetch.</td>
</tr>
<tr>
<td>PrefetchRows</td>
<td>Used to get or set the number of rows that OCI prefetches when executing a query.</td>
</tr>
<tr>
<td>PrepareUpdateSQL</td>
<td>Used to automatically</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>prepare update queries before execution.</td>
<td></td>
</tr>
<tr>
<td><strong>ProcNamedParams</strong></td>
<td>Used to specify a notation method of passing parameter values to the stored PL/SQL object.</td>
</tr>
<tr>
<td><strong>RawAsString</strong></td>
<td>Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>ReflectChangeNotify</strong></td>
<td>Used for a dataset component to refresh its data when it gets database change notification messages in response to DML or DDL changes on the objects associated with the dataset query.</td>
</tr>
<tr>
<td><strong>ScrollableCursor</strong></td>
<td>Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.</td>
</tr>
<tr>
<td><strong>StatementCache</strong></td>
<td>Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.</td>
</tr>
<tr>
<td><strong>TemporaryLobUpdate</strong></td>
<td>Temporary LOBs are used to write input and input/output LOB parameters into database when executing dataset's SQL statements.</td>
</tr>
</tbody>
</table>

See Also

- [TOraDataSetOptions Class](#)
- [TOraDataSetOptions Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

5.15.1.9.2.1  AutoClose Property

Used to close OCI cursor after fetching all rows.

Class
**T0raDataSetOptions**

Syntax

```property``

```AutoClose: boolean stored True;```

Remarks

Use the AutoClose property for T0raDataSet to close OCI cursor after fetching all rows. Allows reducing the number of opened cursors on the server.

**CacheLobs Property**

Used to allocate local memory buffer to hold a copy of the Lob content.

Class

**T0raDataSetOptions**

Syntax

```property``

```CacheLobs: boolean stored True;```

Remarks

If True, (the default value) then local memory buffer is allocated to hold a copy of the Lob content. If this option is set to False, it is highly recommended to set the DeferredLobRead option to True. Otherwise, LOB values are fetched to the dataset, and it can result in performance loss.

**Note:** The CacheLobs option controls the way Lob objects are handled while an application fetches records from the database. Setting CacheLobs to False may bring up the following benefits for time-critical applications: reduced traffic over the network since Lob objects are only transferred on demand; less memory is needed on the client side because returned record sets do not hold contents of the Lob fields. Actual value for the Lob field is passed to the client only when a data-aware control requests it.

© 1997-2021 Devart. All Rights Reserved.  
Request Support  DAC Forum  Provide Feedback
5.15.1.9.2.3 DefaultValues Property

Used for TOraDataSet to fill the DefaultExpression property of TField objects by appropriate value.

Class

TOraDataSetOptions

Syntax

```plaintext
property DefaultValues: boolean stored True;
```

Remarks

If True, TOraDataSet fills the DefaultExpression property of TField objects by appropriate value.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.9.2.4 DeferredLobRead Property

Used to fetch all Oracle 8 Lob values when they are explicitly requested.

Class

TOraDataSetOptions

Syntax

```plaintext
property DeferredLobRead: boolean stored True;
```

Remarks

If True, all Oracle 8 Lob values are only fetched when they are explicitly requested. Otherwise entire record set with any Lob values is returned when dataset is opened. Whether Lob values are cached locally to be reused later or not is controlled by CacheLobs option.

© 1997-2021 Devart. All Rights Reserved.
5.15.1.9.2.5 EnableBCD Property

Used to enable currency type. Default value of this option is False.

Class
T0raDataSetOptions

Syntax

```delphi
property EnableBCD: boolean;
```

5.15.1.9.2.6 EnableFMTBCD Property

Used to enable using FMTBCD instead of float for large integer numbers to keep precision.

Class
T0raDataSetOptions

Syntax

```delphi
property EnableFMTBCD: boolean;
```

5.15.1.9.2.7 ExtendedFieldsInfo Property

Used to perform an additional query to get information about returned fields and the tables they belong to.

Class
T0raDataSetOptions

Syntax

```delphi
property ExtendedFieldsInfo: boolean;
```

Remarks

If True, an additional query is performed to get information about returned fields and the tables
they belong to. True by default for TSmartQuery, and False by default for other TOraDataSet descendants.

5.15.1.9.2.8 FieldsAsString Property

Used to treat all non-BLOB fields as being of string datatype.

Class

TOraDataSetOptions

Syntax

```plaintext
property FieldsAsString: boolean stored True;
```

Remarks

If True, all non-BLOB fields are treated as being of string datatype.

5.15.1.9.2.9 FullRefresh Property

Used to refresh fields of all tables by the RefreshRecord method.

Class

TOraDataSetOptions

Syntax

```plaintext
property FullRefresh: boolean;
```

Remarks

Set the FullRefresh property to True to refresh fields of all tables by the RefreshRecord method. To perform full refreshing TCustomSmartQuery executes modified SELECT statement defined by the SQL property. When FullRefresh is False TCustomSmartQuery performs refreshing fields of the updating table only. The default value is False.
5.15.1.9.2.10 PrefetchLobSize Property

Used to retrieve the LOB length and the LOB data beginning during regular fetch.

Class
T0raDataSetOptions

Syntax

| property          | PrefetchLobSize: Integer default 0; |

Remarks

Use the PrefetchLobSize option to retrieve the LOB length and the chunk size as well as the beginning of the LOB data along with the locator during regular fetch. The PrefetchLobSize property specifies the size of LOB data that will be prefetched. If the total LOB size is less or equals to PrefetchLobSize, then all LOB data will be fetched during regular fetch without additional round trips that can improve performance greatly.

Note: Prefetching LOB data is available in Oracle 11 and higher.

5.15.1.9.2.11 PrefetchRows Property

Used to get or set the number of rows that OCI prefetched when executing a query.

Class
T0raDataSetOptions

Syntax

| property          | PrefetchRows: integer default 0; |

Remarks

Use the PrefetchRows property to get or set the number of rows that OCI prefetched when executing a query. Setting the property to a value greater than 0 reduces the server round trip count, which increases the performance of the application. The default value is 0 - the number of prefetched rows is determined automatically. To disable row prefetching, set the
property to -1.

**Note:** Some queries (for example, query `SELECT * FROM DUAL CONNECT BY LEVEL <= 5` returns 1 row when prefetching is enabled, and 5 rows when it is disabled) can return invalid rows count when prefetching is enabled.

---

### 5.15.1.9.2.12 PrepareUpdateSQL Property

Used to automatically prepare update queries before execution.

**Class**

**TOraDataSetOptions**

**Syntax**

```plaintext
property PrepareUpdateSQL: boolean;
```

**Remarks**

If True, update queries are automatically prepared before executing.

---

### 5.15.1.9.2.13 ProcNamedParams Property

Used to specify a notation method of passing parameter values to the stored PL/SQL object.

**Class**

**TOraDataSetOptions**

**Syntax**

```plaintext
property ProcNamedParams: boolean default False;
```

**Remarks**

Positional Notation is used if `OraStoredProc.Options.ProcNamedParams = False` (default value). If True, Named Notation is used.

Named Notation allows passing parameter values in any order regardless of the position.
Example

Sample of stored proc call with Named Notation:

```sql
credit_acct(amount => amt, acct_no => acct);
```

5.15.1.9.2.14 RawAsString Property

Used to treat all RAW fields as being of string datatype.

Class

`T0raDataSetOptions`

Syntax

```
property RawAsString: boolean stored True;
```

Remarks

If True, all RAW fields are treated as being of string datatype, i.e. represented as hexadecimal string.

5.15.1.9.2.15 ReflectChangeNotify Property

Used for a dataset component to refresh its data when it gets database change notification messages in response to DML or DDL changes on the objects associated with the dataset query.

Class

`T0raDataSetOptions`

Syntax

```
property ReflectChangeNotify: boolean default False;
```

Remarks

If True and the `T0raDataSet.ChangeNotification` property is set, dataset component refreshes
its data when it gets database change notification messages in response to DML or DDL changes on the objects associated with the dataset query.

5.15.1.9.2.16 ScrollableCursor Property

Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.

Class
TOraDataSetOptions

Syntax
property ScrollableCursor: boolean stored True;

Remarks
If True, TOraDataSet does not cache data on the client side but uses scrollable server cursor (available since Oracle 9 only). This option can be used to reduce memory usage, because dataset stores only current fetched block. Unlike the TCustomDADataSet UniDirectional option ScrollableCursor allows bidirectional dataset navigation. Note that scrollable cursor is read-only by its nature.

5.15.1.9.2.17 StatementCache Property

Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.

Class
TOraDataSetOptions

Syntax
property StatementCache: boolean default False;

Remarks
OCI statement cache is enabled when you set `TOraSessionOptions.StatementCacheSize` in `TOraSession.Options` to a positive value. Set `TOraSessionOptions.StatementCacheSize` to 0 (default) or `TOraSession.Options.StatementCache` to false if you don't want the statements to be cached.

Temporary LOBs are used to write input and input/output LOB parameters into database when executing dataset's SQL statements.

**Class**

`TOraDataSetOptions`

**Syntax**

```
property TemporaryLobUpdate: boolean default False;
```

**Remarks**

Set the TemporaryLobUpdate property to True to use temporary LOBs to write input and input/output LOB parameters into database when executing dataset's SQL statements.

**Note:** CacheLobs option controls the way Lob objects are handled while the application fetches records from the database. Setting CacheLobs to False may bring up the following benefits for time-critical applications: reduced traffic over the network since Lob objects are only transferred on demand; less memory is needed on the client side because returned record sets do not hold contents of Lob fields. Actual value for the Lob field is passed to the client only when a data-aware control requests it.
Syntax

```csharp
TOraDataSetOptionsDS = class(TDADatasetOptions);
```

Remarks

**Note:** The OptionsDS property is obsolete. It is provided for backward compatibility only.

Inheritance Hierarchy

```
TDADatasetOptions
  TOraDataSetOptionsDS
```

5.15.1.10.1 Members

**TOraDataSetOptionsDS** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoClose</td>
<td>Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td>AutoPrepare</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to execute automatic TCustomDADataset.Prepare on the query execution.</td>
</tr>
<tr>
<td>CacheCalcFields</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td>CacheLobs</td>
<td>Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td>CompressBlobMode</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td>DefaultValues</td>
<td>Used for TOraDataSet to fill the DefaultExpression property of TField objects with the appropriate value.</td>
</tr>
<tr>
<td>DeferredLobRead</td>
<td>Used fetch all Oracle 8 Lob values only when they are</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>DetailDelay</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td><strong>FieldsAsString</strong></td>
<td>Used to treat all non-BLOB fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>FieldsOrigin</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used for TCustomDADataset to fill the Origin property of the TField objects by appropriate value when opening a dataset.</td>
</tr>
<tr>
<td><strong>FlatBuffers</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to control how a dataset treats data of the ftString and ftVarBytes fields.</td>
</tr>
<tr>
<td><strong>HideRowId</strong></td>
<td>Used to display the ROWID column.</td>
</tr>
<tr>
<td><strong>InsertAllSetFields</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to include all set dataset fields in the generated INSERT statement</td>
</tr>
<tr>
<td><strong>KeepPrepared</strong></td>
<td>Used to keep TORaDataSet prepared after closing.</td>
</tr>
<tr>
<td><strong>LocalMasterDetail</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used for TCustomDADataset to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.</td>
</tr>
<tr>
<td><strong>LongStrings</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to represent string fields with the length that is greater than 255 as TStringField.</td>
</tr>
<tr>
<td><strong>MasterFieldsNullable</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>NumberRange</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.</td>
</tr>
<tr>
<td><strong>QueryRecCount</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used for TCustomDADataSet to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td><strong>QuoteNames</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used for TCustomDADataSet to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td><strong>RawAsString</strong></td>
<td>Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>RemoveOnRefresh</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td><strong>RequiredFields</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used for TCustomDADataSet to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>ScrollableCursor</strong></td>
<td>Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.</td>
</tr>
<tr>
<td><strong>SetFieldsReadOnly</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td><strong>StrictEqual</strong> (inherited from <strong>TDADatasetOptions</strong>)**</td>
<td>Used for TCustomDADataSet to raise an exception when the</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AutoPrepare</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to execute automatic TCustomDADataset.Prepare on the query execution.</td>
</tr>
<tr>
<td><strong>CacheCalcFields</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td><strong>CompressBlobMode</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td><strong>DetailDelay</strong> (inherited from <strong>TDADatasetOptions</strong>)</td>
<td>Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FieldsOrigin</td>
<td>Used for TCustomDADataSet to fill the Origin property of the TField objects by appropriate value when opening a dataset.</td>
</tr>
<tr>
<td>FlatBuffers</td>
<td>Used to control how a dataset treats data of the ftString and ftVarBytes fields.</td>
</tr>
<tr>
<td>InsertAllSetFields</td>
<td>Used to include all set dataset fields in the generated INSERT statement.</td>
</tr>
<tr>
<td>LocalMasterDetail</td>
<td>Used for TCustomDADataSet to use local filtering to establish master/detail relationship for detail dataset and does not refer to the server.</td>
</tr>
<tr>
<td>LongStrings</td>
<td>Used to represent string fields with the length that is greater than 255 as TStringField.</td>
</tr>
<tr>
<td>MasterFieldsNullable</td>
<td>Allows to use NULL values in the fields by which the relation is built, when generating the query for the Detail tables (when this option is enabled, the performance can get worse).</td>
</tr>
<tr>
<td>NumberRange</td>
<td>Used to set the MaxValue and MinValue properties of TIntegerField and TFloatField to appropriate values.</td>
</tr>
<tr>
<td>QueryRecCount</td>
<td>Used for TCustomDADataSet to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>QuoteNames</td>
<td>Used for TCustomDADataSet to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td>RemoveOnRefresh</td>
<td>Used for a dataset to locally remove a record that cannot be found on the server.</td>
</tr>
<tr>
<td>RequiredFields</td>
<td>Used for TCustomDADataSet to set the Required property of the TField objects for the NOT NULL fields.</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td>SetFieldsReadOnly</td>
<td>Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>Used for TCustomDADataSet to raise an exception when the number of updated or deleted records is not equal 1.</td>
</tr>
<tr>
<td>TrimFixedChar</td>
<td>Specifies whether to discard all trailing spaces in the string fields of a dataset.</td>
</tr>
<tr>
<td>UpdateAllFields</td>
<td>Used to include all dataset fields in the generated UPDATE and INSERT statements.</td>
</tr>
<tr>
<td>UpdateBatchSize</td>
<td>Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutoClose</td>
<td>Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td>CacheLobs</td>
<td>Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td>DefaultValues</td>
<td>Used for TOraDataSet to fill the DefaultExpression property of TField objects with the appropriate value.</td>
</tr>
<tr>
<td>DeferredLobRead</td>
<td>Used to fetch all Oracle 8 Lob values only when they are explicitly requested.</td>
</tr>
<tr>
<td>FieldsAsString</td>
<td>Used to treat all non-BLOB fields as being of string datatype.</td>
</tr>
<tr>
<td>HideRowId</td>
<td>Used to display the ROWID column.</td>
</tr>
<tr>
<td>KeepPrepared</td>
<td>Used to keep TOraDataSet prepared after closing.</td>
</tr>
<tr>
<td>RawAsString</td>
<td>Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td>ScrollableCursor</td>
<td>Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.</td>
</tr>
</tbody>
</table>

See Also
- TOraDataSetOptionsDS Class
- TOraDataSetOptionsDS Class Members
Syntax

```objectivec
property AutoClose: boolean stored False default False;
```

Remarks

Use the AutoClose property for TOraDataSet to close OCI cursor after fetching all rows. Allows to reduce the number of opened cursors on the server.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.10.2.2 CacheLobs Property

Used to allocate local memory buffer to hold a copy of the Lob content.

Class

`TOraDataSetOptionsDS`

Syntax

```objectivec
property CacheLobs: boolean stored False default True;
```

Remarks

If True, (the default value) then local memory buffer is allocated to hold a copy of the Lob content. See the notes below for further details.

**Note:** CacheLobs option controls the way Lob objects are handled while the application fetches records from the database. Setting CacheLobs to False may bring up the following benefits for time-critical applications: reduced traffic over the network since Lob objects are only transferred on demand; less memory is needed on the client side because returned record sets do not hold contents of Lob fields. Actual value for the Lob field is passed to the client only when a data-aware control requests it.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.10.2.3 DefaultValues Property

Used for TOraDataSet to fill the DefaultExpression property of TField objects with the appropriate value.
Class

**TOraDataSetOptionsDS**

Syntax

```property
DefaultValues: boolean stored False;
```

Remarks

If True, TOraDataSet fills the DefaultExpression property of TField objects with the appropriate value.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.10.2.4 DeferredLobRead Property

Used fetch all Oracle 8 Lob values only when they are explicitly requested.

Class

**TOraDataSetOptionsDS**

Syntax

```property
DeferredLobRead: boolean stored False default False;
```

Remarks

If True, all Oracle 8 Lob values are only fetched when they are explicitly requested. Otherwise entire record set with any Lob values is returned when dataset is opened. Whether Lob values are cached locally to be reused later or not is controlled by CacheLobs option.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.10.2.5 FieldsAsString Property

Used to treat all non-BLOB fields as being of string datatype.

Class

**TOraDataSetOptionsDS**

© 2021 Devart
Syntax

```
property FieldsAsString: boolean stored False default False;
```

Remarks

If True, all non-BLOB fields are treated as being of string datatype.

5.15.1.10.2.6  HideRowId Property

Used to display the ROWID column.

Class

```
T0raDataSetOptionsDS
```

Syntax

```
property HideRowId: boolean default True;
```

Remarks

Used to display the RowId service field. By default, the Visible property for this field is set to False.

5.15.1.10.2.7  KeepPrepared Property

Used to keep TOraDataSet prepared after closing.

Class

```
T0raDataSetOptionsDS
```

Syntax

```
property KeepPrepared: boolean stored False;
```

Remarks

If True, TOraDataSet remains prepared after closing. It allows to avoid needless reopening
cursor on the server.

5.15.1.10.2.8 RawAsString Property

Used to treat all RAW fields as being of string datatype.

Class
TOraDataSetOptionsDS

Syntax

```plaintext
property RawAsString: boolean stored False default False;
```

Remarks

If True, all RAW fields are treated as being of string datatype, i.e. represented as hexadecimal string.

5.15.1.10.2.9 ScrollableCursor Property

Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.

Class
TOraDataSetOptionsDS

Syntax

```plaintext
property ScrollableCursor: boolean stored False default False;
```

Remarks

If True, TOraDataSet does not cache data on the client side but uses scrollable server cursor (available since Oracle 9 only). This option can be used to reduce memory usage, because dataset stores only current fetched block. Unlike the `TCustomDADataset.UniDirectional` option ScrollableCursor allows bidirectional dataset navigation. Note that scrollable cursor is read-only by its nature.
5.15.1.11 TOraDataSource Class

TOraDataSource provides an interface between an ODAC dataset components and data-aware controls on a form.

For a list of all members of this type, see TOraDataSource members.

Unit
ora

Syntax

```
TOraDataSource = class(TCRDataSource);
```

Remarks

TOraDataSource provides an interface between an ODAC dataset components and data-aware controls on a form.

TOraDataSource inherits its functionality directly from the TDataSource component.

At design-time assign individual data-aware components' DataSource properties from their drop-down listboxes.

Inheritance Hierarchy

TCRDataSource
  TOraDataSource
5.15.1.12 TOraEncryptor Class

The class that performs encrypting and decrypting of data.

For a list of all members of this type, see `TOraEncryptor` members.

Unit
ora

Syntax

```pascal
TOraEncryptor = class(TCREncryptor);
```

Inheritance Hierarchy

- `TCREncryptor`
- `TOraEncryptor`

5.15.1.12.1 Members

`TOraEncryptor` class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataHeader (inherited from <code>TCREncryptor</code>)</td>
<td>Specifies whether the additional information is stored with the encrypted data.</td>
</tr>
<tr>
<td>EncryptionAlgorithm (inherited from <code>TCREncryptor</code>)</td>
<td>Specifies the algorithm of data encryption.</td>
</tr>
<tr>
<td>HashAlgorithm (inherited from <code>TCREncryptor</code>)</td>
<td>Specifies the algorithm of generating hash data.</td>
</tr>
<tr>
<td>InvalidHashAction (inherited from <code>TCREncryptor</code>)</td>
<td>Specifies the action to perform on data fetching when hash data is invalid.</td>
</tr>
<tr>
<td>Password (inherited from <code>TCREncryptor</code>)</td>
<td>Used to set a password that is used to generate a key for encryption.</td>
</tr>
</tbody>
</table>
5.15.1.13 `TOraIntervalField` Class

A class providing access to the Oracle interval fields.

For a list of all members of this type, see `TOraIntervalField` members.

**Unit**

`ora`

**Syntax**

```delphi
TOraIntervalField = class(TField);
```

**Remarks**

`TOraIntervalField` provides access to the Oracle interval fields. Unlike other `TField` descendants the `TOraIntervalField.DataType` property has two valid values `ftIntervalYM` and `ftIntervalDS` depending on the type of the interval.

You can access actual interval value using properties `AsString` and `AsOraInterval` properties.

**See Also**

- `TOraInterval`

© 1997-2021

Devart. All Rights Reserved.

Provide Feedback

© 2021 Devart
### FracPrecision

Used to get or set the number of digits used to represent fractional seconds when getting interval value as string.

### LeadPrecision

Used to get or set the number of digits that are used to represent the leading interval part when getting the interval value as string.

---

5.15.1.13.2 Properties

Properties of the `TOraIntervalField` class.

For a complete list of the `TOraIntervalField` class members, see the [TOraIntervalField Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsInterval</td>
<td>Used to provide access to a <code>TOraInterval</code> object.</td>
</tr>
</tbody>
</table>

#### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FracPrecision</td>
<td>Used to get or set the number of digits used to represent fractional seconds when getting interval value as string.</td>
</tr>
<tr>
<td>LeadPrecision</td>
<td>Used to get or set the number of digits that are used to represent the leading interval part when getting the interval value as string.</td>
</tr>
</tbody>
</table>
5.15.1.13.2.1  AsInterval Property

Used to provide access to a TOraInterval object.

Class

TOraIntervalField

Syntax

| property AsInterval: TOraInterval; |

Remarks

Use the AsInterval property to provide access to a TOraInterval object you can use for manipulations with the interval value.

See Also

• TOraInterval

5.15.1.13.2.2  FracPrecision Property

Used to get or set the number of digits used to represent fractional seconds when getting interval value as string.

Class

TOraIntervalField

Syntax

| property FracPrecision: integer default 6; |

Remarks
Use the FracPrecision property to get or set the number of digits used to represent fractional seconds when getting interval value as string. This property affects only INTERVAL DAY TO SECOND (ftIntervalDS). The default value of the property is 6.

See Also
- `TORalInterval.FracPrecision`
- `TORalInterval.AsString`

5.15.1.13.2.3 LeadPrecision Property

Used to get or set the number of digits that are used to represent the leading interval part when getting the interval value as string.

Class
- `TORaIntervalField`

Syntax

```pascal
property LeadPrecision: integer default 2;
```

Remarks

Use the LeadPrecision property to get or set the number of digits that are used to represent the leading interval part when getting the interval value as string. The default value of the property is 2.

See Also
- `TORalInterval.LeadPrecision`
- `TORalInterval.AsString`

5.15.1.14 TOraMetaData Class

A component for obtaining metainformation about database objects from the server.

For a list of all members of this type, see `TOraMetaData` members.
Unit
ora

Syntax

```pascal
TOraMetaData = class(TDAMetaData);
```

Remarks

The TOraMetaData component is used to obtain metainformation from the server about objects in the database, such as tables, table columns, stored procedures, etc.

Inheritance Hierarchy

```
TMemDataSet
   TDAMetaData
   TOraMetaData
```

See Also

- TCustomDADataset.Debug
- TCustomDASQL.Debug
- DBMonitor

© 1997-2021
Devart. All Rights Reserved.

5.15.1.14.1

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>(inherited from TMemDataSet) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>Connection</td>
<td>(inherited from TDAMetaData) Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>(inherited from TMemDataSet) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>MetaDataKind</strong> (inherited from <strong>TDAMetaData</strong>)</td>
<td>Used to specify which kind of metainformation to show.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>Restrictions</strong> (inherited from <strong>TDAMetaData</strong>)</td>
<td>Used to provide one or more conditions restricting the list of objects to be described.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>CommitUpdates</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>DeferredPost</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>GetBlob</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>GetMetaDataKinds</td>
<td>TDAMetaData</td>
</tr>
<tr>
<td>GetRestrictions</td>
<td>TDAMetaData</td>
</tr>
<tr>
<td>Locate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LocateEx</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Prepare</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SetRange</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SetRangeStart</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>UpdateResult</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnUpdateError</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td>OnUpdateRecord</td>
<td>Occurs when a single update component can not handle the updates.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.
5.15.1.15 TOraNestedTable Class

A component for controlling nested table data.

For a list of all members of this type, see TOraNestedTable members.

Unit

Ora

Syntax

TOraNestedTable = class(TMemDataSet);

Remarks

Nested table is a dataset component that encapsulates a database table that is nested as a field within another table. Use TOraNestedTable to access data contained in a nested dataset. A nested table provides much of the functionality of a table component, with the difference that the data it accesses is stored in a nested table.

TOraNestedTable is derived from the TMemDataSet component.

Inheritance Hierarchy

TMemDataSet
  TOraNestedTable

See Also

- TOraNestTable
- TOraRef

© 1997-2021 Devart. All Rights Reserved.  Request Support   DAC Forum   Provide Feedback

5.15.1.15.1 Members

TOraNestedTable class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
</tbody>
</table>
### IndexFieldNames (inherited from `TMemDataSet`)
Used to get or set the list of fields on which the recordset is sorted.

### KeyExclusive (inherited from `TMemDataSet`)
Specifies the upper and lower boundaries for a range.

### LocalConstraints (inherited from `TMemDataSet`)
Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

### LocalUpdate (inherited from `TMemDataSet`)
Used to prevent implicit update of rows on database server.

### Prepared (inherited from `TMemDataSet`)
Determines whether a query is prepared for execution or not.

### Ranged (inherited from `TMemDataSet`)
Indicates whether a range is applied to a dataset.

### Ref
Used to assign reference data to TOraNestedTable.

### Table
Used to assign nested table data to TOraNestedTable.

### UpdateRecordTypes (inherited from `TMemDataSet`)
Used to indicate the update status for the current record when cached updates are enabled.

### UpdatesPending (inherited from `TMemDataSet`)
Used to check the status of the cached updates buffer.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyRange</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>CancelRange</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>CommitUpdates</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>DeferredPost</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>GetBlob</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Locate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LocateEx</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Prepare</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>SetRange</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SetRangeStart</strong> (inherited from TMemDataSet)</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><strong>UnPrepare</strong> (inherited from TMemDataSet)</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><strong>UpdateResult</strong> (inherited from TMemDataSet)</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdateStatus</strong> (inherited from TMemDataSet)</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnUpdateError</strong> (inherited from TMemDataSet)</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td><strong>OnUpdateRecord</strong> (inherited from TMemDataSet)</td>
<td>Occurs when a single update component can not handle the updates.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback

5.15.1.15.2 Properties

Properties of the TOraNestedTable class.

For a complete list of the TOraNestedTable class members, see the TOraNestedTable Members topic.
Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CachedUpdates</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>Ref</strong></td>
<td>Used to assign reference data to TOraNestedTable.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Used to assign nested table data to TOraNestedTable.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

See Also
- [TOraNestedTable Class](#)
- [TOraNestedTable Class Members](#)
5.15.1.15.2.1 Ref Property

Used to assign reference data to TOraNestedTable.

Class
TOraNestedTable

Syntax

```property`` Ref: TOraRef;
```

Remarks
Use the Ref property to assign reference data to TOraNestedTable. After assigning you can call the Open method to browse the reference data.

See Also
- TOraRef
- TOraParam.AsRef
- TOraDataSet.GetRef

© 1997-2021 Devart. All Rights Reserved.

5.15.1.15.2.2 Table Property

Used to assign nested table data to TOraNestedTable.

Class
TOraNestedTable

Syntax

```property`` Table: TOraNestTable;
```

Remarks
Use the Table property to assign nested table data to TOraNestedTable. After assigning you can call the Open method to browse the nested table data.

Example

```OraNestedTable1.Table := OraSQL1.ParamByName('Content').AsTable;
OraNestedTable1.Open;```

© 2021 Devart
5.15.1.16 TOraNumberField Class

A class providing access to the Oracle number fields.

For a list of all members of this type, see `TOraNumberField` members.

**Unit**

`Ora`

**Syntax**

```
TOraNumberField = class(TNumericField);
```

**Remarks**

TOraNumberField provides access to Oracle number fields. The `TOraNumberField.DataType` property values equals to `ftNumber`.

You can access actual number value using `AsString`, `AsInteger` and `AsFloat` properties.

**See Also**

- `TOraNumber`
Properties of the **TOraNumberField** class.

For a complete list of the **TOraNumberField** class members, see the [**TOraNumberField Members** topic](#).

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsNumber</strong></td>
<td>Used to provide access to a TOraNumber object.</td>
</tr>
</tbody>
</table>

### See Also

- [**TOraNumberField Class**](#)
- [**TOraNumberField Class Members**](#)

### AsNumber Property

Used to provide access to a TOraNumber object.

**Class**

**TOraNumberField**

**Syntax**

```property
AsNumber: TOraNumber;
```

**Remarks**

Use the AsNumber property to provide access to a TOraNumber object that you can use for manipulations with the number value.

**See Also**
5.15.1.17 TOraParam Class

A class that is used to set the values of individual parameters passed with queries or stored procedures.

For a list of all members of this type, see TOraParam members.

Unit
ora

Syntax

TOraParam = class(TDAParam);

Remarks

Use the properties of TOraParam to set the value of a parameter. Objects that use parameters create TOraParam objects to represent these parameters. For example, TOraParam objects are used by TOraSQL, TCustomOraDataSet.

TOraParam shares many properties with TField, as both describe the value of a field in a dataset. However, a TField object has several properties to describe the field binding, and how the field is displayed, edited, or calculated that are not needed in a TOraParam object. Conversely, TOraParam includes properties that indicate how the field value is passed as a parameter.

Inheritance Hierarchy

TDAParam
  TOraParam

See Also

- TOraDataSet
- TOraSQL
- TOraParams
### Members

**TOraParam** class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsArray</td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle array type.</td>
</tr>
<tr>
<td>AsBFile</td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle BFile type.</td>
</tr>
<tr>
<td>AsBlob</td>
<td>Used to set and read the value of the BLOB parameter as string.</td>
</tr>
<tr>
<td>AsBlobLocator</td>
<td>Used to specify the value of a parameter when it represents the value of the BLOB type.</td>
</tr>
<tr>
<td>AsBlobRef</td>
<td>Used to set and read the value of the BLOB parameter as a TBlob object.</td>
</tr>
<tr>
<td>AsCLOBLocator</td>
<td>Used to specify the value of the parameter when it represents the value of CLOB type.</td>
</tr>
<tr>
<td>AsCursor</td>
<td>Used to specify the value of the parameter when it represents the value of the PL/SQL REF CURSOR type.</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Used to assign the value for a float field to a parameter.</td>
</tr>
<tr>
<td>AsInteger</td>
<td>Used to assign the value for an integer field to the parameter.</td>
</tr>
<tr>
<td>AsInterval</td>
<td>Used to specify the parameter value when it represents Oracle 9 interval type.</td>
</tr>
<tr>
<td><strong>AsLargeInt</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to assign the value for a LargeInteger field to the parameter.</td>
</tr>
<tr>
<td><strong>AsMemo</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to assign the value for a memo field to the parameter.</td>
</tr>
<tr>
<td><strong>AsMemoRef</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to set and read the value of the memo parameter as a TBlob object.</td>
</tr>
<tr>
<td><strong>AsNumber</strong></td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle internal number type.</td>
</tr>
<tr>
<td><strong>AsObject</strong></td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle object type.</td>
</tr>
<tr>
<td><strong>AsOraBlob</strong></td>
<td>Used to specify the value of a parameter when it represents the value of the BLOB type.</td>
</tr>
<tr>
<td><strong>AsOraClob</strong></td>
<td>Used to specify the value of the parameter when it represents the value of CLOB type.</td>
</tr>
<tr>
<td><strong>AsRef</strong></td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle reference type.</td>
</tr>
<tr>
<td><strong>AsSQLTimeStamp</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to specify the value of the parameter when it represents a SQL timestamp field.</td>
</tr>
<tr>
<td><strong>AsString</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to assign the string value to the parameter.</td>
</tr>
<tr>
<td><strong>AsTable</strong></td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle nested table type.</td>
</tr>
<tr>
<td><strong>AsTimeStamp</strong></td>
<td>Used to specify parameter value when it represents Oracle 9 timestamp type.</td>
</tr>
<tr>
<td><strong>AsWideString</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to assign the Unicode value to the parameter.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AsXML</strong></td>
<td>Used to specify the value of the parameter when it represents the value of Oracle SYS.XMLTYPE.</td>
</tr>
<tr>
<td><strong>DataType</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Indicates the data type of the parameter.</td>
</tr>
<tr>
<td><strong>IsNull</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to indicate whether the value assigned to a parameter is NULL.</td>
</tr>
<tr>
<td><strong>ItemAsDateTime</strong></td>
<td>Used to access a PL/SQL table item as TDateTime by Index.</td>
</tr>
<tr>
<td><strong>ItemAsFloat</strong></td>
<td>Used to access a PL/SQL table item as Double by Index.</td>
</tr>
<tr>
<td><strong>ItemAsInteger</strong></td>
<td>Used to access a PL/SQL table item as Integer by Index.</td>
</tr>
<tr>
<td><strong>ItemAsInterval</strong></td>
<td>Used to access a PL/SQL table item as TOraInterval by Index.</td>
</tr>
<tr>
<td><strong>ItemAsString</strong></td>
<td>Used to access a PL/SQL table item as String by Index.</td>
</tr>
<tr>
<td><strong>ItemAsTimeStamp</strong></td>
<td>Used to access a PL/SQL table item as TOraTimeStamp by Index.</td>
</tr>
<tr>
<td><strong>ItemIsNull</strong></td>
<td>Used to indicate if an item value is Null.</td>
</tr>
<tr>
<td><strong>ItemText</strong></td>
<td>Allows to modify data without changing its type.</td>
</tr>
<tr>
<td><strong>ItemValue</strong></td>
<td>Used to access the item value as variant.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>Used to determine the maximum length of a PL/SQL table parameter.</td>
</tr>
<tr>
<td><strong>National</strong></td>
<td>Used to determine whether the parameter is in National charset.</td>
</tr>
<tr>
<td><strong>ParamType</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Used to indicate the type of use for a parameter.</td>
</tr>
<tr>
<td><strong>Size</strong> (inherited from <strong>TDAParam</strong>)</td>
<td>Specifies the size of a string type parameter.</td>
</tr>
</tbody>
</table>
### Properties

Properties of the `TOraParam` class.

For a complete list of the `TOraParam` class members, see the [TOraParam Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsArray</td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle array type.</td>
</tr>
<tr>
<td>AsBFile</td>
<td>Used to specify the value of the parameter when it</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AsBlob (inherited from TDAParam)</td>
<td>Used to set and read the value of the BLOB parameter as string.</td>
</tr>
<tr>
<td>AsBLOBLocator</td>
<td>Used to specify the value of a parameter when it represents the value of the BLOB type.</td>
</tr>
<tr>
<td>AsBlobRef (inherited from TDAParam)</td>
<td>Used to set and read the value of the BLOB parameter as a TBlob object.</td>
</tr>
<tr>
<td>AsCLOBLocator</td>
<td>Used to specify the value of the parameter when it represents the value of CLOB type.</td>
</tr>
<tr>
<td>AsCursor</td>
<td>Used to specify the value of the parameter when it represents the value of the PL/SQL REF CURSOR type.</td>
</tr>
<tr>
<td>AsFloat (inherited from TDAParam)</td>
<td>Used to assign the value for a float field to a parameter.</td>
</tr>
<tr>
<td>AsInteger (inherited from TDAParam)</td>
<td>Used to assign the value for an integer field to the parameter.</td>
</tr>
<tr>
<td>AsInterval</td>
<td>Used to specify the parameter value when it represents Oracle 9 interval type.</td>
</tr>
<tr>
<td>AsLargeInt (inherited from TDAParam)</td>
<td>Used to assign the value for a LargeInteger field to the parameter.</td>
</tr>
<tr>
<td>AsMemo (inherited from TDAParam)</td>
<td>Used to assign the value for a memo field to the parameter.</td>
</tr>
<tr>
<td>AsMemoRef (inherited from TDAParam)</td>
<td>Used to set and read the value of the memo parameter as a TBlob object.</td>
</tr>
<tr>
<td>AsNumber</td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle BFile type.</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AsObject</td>
<td>Oracle internal number type. Used to specify the value of the parameter when it represents the value of the Oracle object type.</td>
</tr>
<tr>
<td>AsOraBlob</td>
<td>Used to specify the value of a parameter when it represents the value of the BLOB type.</td>
</tr>
<tr>
<td>AsOraClob</td>
<td>Used to specify the value of the parameter when it represents the value of the CLOB type.</td>
</tr>
<tr>
<td>AsRef</td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle reference type.</td>
</tr>
<tr>
<td>AsSQLTimeStamp</td>
<td>Used to specify the value of the parameter when it represents a SQL timestamp field.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to assign the string value to the parameter.</td>
</tr>
<tr>
<td>AsTable</td>
<td>Used to specify the value of the parameter when it represents the value of the Oracle nested table type.</td>
</tr>
<tr>
<td>AsTimeStamp</td>
<td>Used to specify parameter value when it represents Oracle 9 timestamp type.</td>
</tr>
<tr>
<td>As WideString</td>
<td>Used to assign the Unicode string value to the parameter.</td>
</tr>
<tr>
<td>AsXML</td>
<td>Used to specify the value of the parameter when it represents the value of Oracle SYS.XMLTYPE.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to indicate whether the value assigned to a parameter is NULL.</td>
</tr>
<tr>
<td>ItemAsDateTime</td>
<td>Used to access a PL/SQL table item as TDateTime by Index.</td>
</tr>
</tbody>
</table>
| ItemAsFloat          | Used to access a PL/SQL
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemAsInteger</td>
<td>Used to access a PL/SQL table item as Integer by Index.</td>
</tr>
<tr>
<td>ItemAsInterval</td>
<td>Used to access a PL/SQL table item as TOraInterval by Index.</td>
</tr>
<tr>
<td>ItemAsString</td>
<td>Used to access a PL/SQL table item as String by Index.</td>
</tr>
<tr>
<td>ItemAsTimeStamp</td>
<td>Used to access a PL/SQL table item as TOraTimeStamp by Index.</td>
</tr>
<tr>
<td>ItemIsNull</td>
<td>Used to indicate if an item value is Null.</td>
</tr>
<tr>
<td>ItemText</td>
<td>Allows to modify data without changing its type.</td>
</tr>
<tr>
<td>ItemValue</td>
<td>Used to access the item value as variant.</td>
</tr>
</tbody>
</table>

**Published**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataType</strong></td>
<td>Indicates the data type of the parameter.</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>Used to determine the maximum length of a PL/SQL table parameter.</td>
</tr>
<tr>
<td><strong>National</strong></td>
<td>Used to determine whether the parameter is in National charset.</td>
</tr>
<tr>
<td><strong>ParamType</strong></td>
<td>Used to indicate the type of use for a parameter.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Specifies the size of a string type parameter.</td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td>Used to determine if the parameter is a PL/SQL table.</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Used to represent the value of the parameter as Variant.</td>
</tr>
</tbody>
</table>

**See Also**
5.15.1.17.2.1 AsArray Property

Used to specify the value of the parameter when it represents the value of the Oracle array type.

Class

TOraParam

Syntax

```
property AsArray: TOraArray;
```

Remarks

Use the AsArray property to specify the value of the parameter when it represents the value of the Oracle array type.

Setting AsArray will set the DataType property to ftArray.

5.15.1.17.2.2 AsBFile Property

Used to specify the value of the parameter when it represents the value of the Oracle BFile type.

Class

TOraParam

Syntax

```
property AsBFile: TOraFile;
```

Remarks

Use the AsBFile property to specify the value of the parameter when it represents the value of the Oracle BFile type.
Setting AsBFile will set the DataType property to ftBFile.

5.15.1.17.2.3 AsBLOBLocator Property

Used to specify the value of a parameter when it represents the value of the BLOB type.

Class

TOraParam

Syntax

property AsBLOBLocator: TOraLob;

Remarks

Use the AsBLOBLocator property to specify the value of a parameter when it represents the value of the BLOB type.

Setting AsBlobLocator will set the DataType property to ftOraBlob.

Note: This property is obsolete, use AsOraBlob instead.

See Also

• AsOraBlob

5.15.1.17.2.4 AsCLOBLocator Property

Used to specify the value of the parameter when it represents the value of CLOB type.

Class

TOraParam

Syntax

property AsCLOBLocator: TOraLob;

Remarks
Use the AsCLOBLocator property to specify the value of the parameter when it represents the value of CLOB type.

Setting AsClobLocator will set the DataType property to ftOraClob.

**Note:** This property is obsolete, use **AsOraClob** instead.

**See Also**
- **AsOraClob**

5.15.1.17.2.5  AsCursor Property

Used to specify the value of the parameter when it represents the value of the PL/SQL REF CURSOR type.

**Class**

**TOraParam**

**Syntax**

```
property AsCursor: T0raCursor;
```

**Remarks**

Use the AsCursor property to specify the value of the parameter when it represents the value of the PL/SQL REF CURSOR type.

Setting AsCursor will set the DataType property to ftCursor.

5.15.1.17.2.6  AsInterval Property

Used to specify the parameter value when it represents Oracle 9 interval type.

**Class**

**TOraParam**

**Syntax**
### property AsInterval: TOraInterval;

**Remarks**

Use the AsInterval property to specify the parameter value when it represents Oracle 9 interval type.

Setting AsInterval will set the DataType property to ftIntervalYM or ftIntervalDS depending on the `TOraInterval.DescriptorType` property value.

**See Also**

- `TOraInterval.DescriptorType`

### 5.15.1.17.2.7 AsNumber Property

Used to specify the value of the parameter when it represents the value of the Oracle internal number type.

**Class**

`TOraParam`

**Syntax**

```
property AsNumber: TOraNumber;
```

**Remarks**

Use the AsNumber property to specify the value of the parameter when it represents the value of the Oracle internal number type.

Setting AsNumber will set the DataType property to ftNumber.

© 1997-2021 Devart. All Rights Reserved.

Request Support DAC Forum Provide Feedback

5.15.1.17.2.8 AsObject Property

Used to specify the value of the parameter when it represents the value of the Oracle object type.

**Class**
T0raParam

Syntax

```
property AsObject: T0raObject;
```

Remarks

Use the AsObject property to specify the value of the parameter when it represents the value of the Oracle object type.

Setting AsObject will set the DataType property to ftObject.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.17.2.9 AsOraBlob Property

Used to specify the value of a parameter when it represents the value of the BLOB type.

Class

T0raParam

Syntax

```
property AsOraBlob: T0raLob;
```

Remarks

Use the AsOraBlob property to specify the value of the parameter when it represents the value of BLOB type.

Setting AsOraBlob will set the DataType property to ftOraBlob.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.17.2.10 AsOraClob Property

Used to specify the value of the parameter when it represents the value of CLOB type.

Class

T0raParam
Syntax

```plaintext
property AsOraClob: TOraLob;
```

Remarks

Use the AsOraClob property to specify the value of the parameter when it represents the value of the CLOB type.

Setting AsOraClob will set the DataType property to ftOraClob.

Class

**TOraParam**

Syntax

```plaintext
property AsRef: TOraRef;
```

Remarks

Use the AsRef property to specify the value of the parameter when it represents the value of the Oracle reference type.

Setting AsRef will set the DataType property to ftReference.

Class

**TOraParam**

Syntax

```plaintext
property AsTable: TOraTab;
```

Remarks

Used to specify the value of the parameter when it represents the value of the Oracle nested table type.

Class

**TOraParam**
**5.15.1.17.2.13  AsTable Property**

Used to specify the value of the parameter when it represents the value of the Oracle nested table type.

**Syntax**

```object
property AsTable: TOraNestTable;
```

**Remarks**

Use the AsTable property to specify the value of the parameter when it represents the value of the Oracle nested table type.

Setting AsTable will set the DataType property to ftTable.

`© 1997-2021 Devart. All Rights Reserved.  
Request Support  DAC Forum  Provide Feedback`

**Class**

`TOraParam`

**5.15.1.17.2.14  AsTimeStamp Property**

Used to specify the value of the parameter when it represents Oracle 9 timestamp type.

**Syntax**

```object
property AsTimeStamp: TOraTimeStamp;
```

**Remarks**

Specifies parameter value when it represents Oracle 9 timestamp type.

Setting AsTimeStamp will set the DataType property to ftTimestamp, ftTimestampTZ or ftTimestampLTZ depending on the `TOraTimeStamp.DescriptorType` property value.

**See Also**

- `TOraTimeStamp.DescriptorType`

`© 1997-2021 Devart. All Rights Reserved.  
Request Support  DAC Forum  Provide Feedback`

**5.15.1.17.2.14  AsXML Property**

Used to specify the value of the parameter when it represents the value of Oracle SYS.XMLTYPE.

**Class**
**T0raParam**

Syntax

```
property AsXML: T0raXML;
```

Remarks

Use the AsXML property to specify the value of the parameter when it represents the value of Oracle SYS.XMLTYPE.

Setting AsXML will set the DataType property to ftXML.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.17.2.15 ItemAsDateTime Property(Indexer)

Used to access a PL/SQL table item as TDateTime by Index.

Class

**T0raParam**

Syntax

```
property ItemAsDateTime[Index: integer]: TDateTime;
```

Parameters

*Index*

Holds the index of a PL/SQL table item.

Remarks

Use the ItemAsDateTime property to access a PL/SQL table item by Index. Returns the table item as a TDateTime value.

Index starts with 1. The index value cannot be greater than the Length value.

See Also

* Table
5.15.1.17.2.16  *ItemAsFloat Property (Indexer)*

Used to access a PL/SQL table item as Double by Index.

**Class**

*T0raParam*

**Syntax**

```
property ItemAsFloat[Index: integer]: double;
```

**Parameters**

*Index*

Holds the index of a PL/SQL table item.

**Remarks**

Use the *ItemAsFloat* property to access a PL/SQL table item by Index. Returns the table item as a Double value.

Index starts with 1. The index value cannot be greater than the Length value.

**See Also**

- *Table*

---

5.15.1.17.2.17  *ItemAsInteger Property (Indexer)*

Used to access a PL/SQL table item as Integer by Index.

**Class**

*T0raParam*

**Syntax**

```
property ItemAsInteger[Index: integer]: integer;
```

**Parameters**

*Index*

Holds the index of a PL/SQL table item.

**Remarks**

...
Use the `ItemAsInteger` property to access a PL/SQL table item by index. Returns the table item as an Integer value.

Index starts with 1. The index value cannot be greater than the Length value.

**See Also**
- `Table`

5.15.1.17.2.19  `ItemAsString` Property (Indexer)

Used to access a PL/SQL table item as `TOraInterval` by index.

**Class**

**TOraParam**

**Syntax**

```property ItemAsInterval[Index: integer]: TOraInterval;```

**Parameters**

- `Index`
  
  Holds the index of a PL/SQL table item.

**Remarks**

Use the `ItemAsInterval` property to access a PL/SQL table item by index. Returns the table item as a `TOraInterval` value.

Index starts with 1. The index value cannot be greater than the Length value.

**See Also**

- `Table`
**T0raParam**

Syntax

```pascal
property ItemAsString[Index: integer]: string;
```

**Parameters**

`Index`

Holds the index of a PL/SQL table item.

**Remarks**

Use the `ItemAsString` property to access a PL/SQL table item by Index. Returns the table item as a String value.

Index starts with 1. The index value cannot be greater than the Length value.

**See Also**

- `Table`

---

5.15.1.17.2.20  ItemAsTimeStamp Property(Indexer)

Used to access a PL/SQL table item as T0raTime佘mp by Index.

**Class**

**T0raParam**

Syntax

```pascal
property ItemAsTimeStamp[Index: integer]: T0raTime佘mp;
```

**Parameters**

`Index`

Holds the index of a PL/SQL table item.

**Remarks**

Use the `ItemAsTimeStamp` property to access a PL/SQL table item by Index. Returns the table item as a T0raTime佘mp value.

Index starts with 1. The index value cannot be greater than the Length value.
5.15.1.17.2.21 ItemIsNull Property (Indexer)

Used to indicate if an item value is Null.

Class

**T0raParam**

Syntax

```plaintext
property ItemIsNull[Index: integer]: boolean;
```

Parameters

*Index*

Holds the index of a PL/SQL table item.

Remarks

Use the ItemIsNull property to access PL/SQL table item by Index. Returns True, if the item value is Null.

Index starts with 1. The index value cannot be greater than Length value.

See Also

• **Table**

5.15.1.17.2.22 ItemText Property (Indexer)

Allows to modify data without changing its type.

Class

**T0raParam**

Syntax
**property** ItemText[Index: integer]: string;

**Parameters**

*Index*

Holds the index of a PL/SQL table item.

**Remarks**

Use the ItemText property to access PL/SQL table item by Index.

Index starts with 1 and lasts till **Length**.

Modifying Items using ItemText doesn't affect DataType property.

Class

T0raParam

**Syntax**

**property** ItemValue[Index: integer]: variant;

**Parameters**

*Index*

Holds the index of a PL/SQL table item.

**Remarks**

Use the ItemValue property to access PL/SQL table item by Index. Returns the item value.

Index starts with 1. The index value cannot be greater than Length value.

**See Also**

- Table
5.15.1.17.2.24 Length Property

Used to determine the maximum length of a PL/SQL table parameter.

Class
TOraParam

Syntax

```
property Length: integer default 1;
```

Remarks

Use the Length property to determine the maximum length of a PL/SQL table parameter. For a scalar parameter Length is always 1.

Remember that you can use ItemAsInteger, ItemAsString and other similar properties for PL/SQL tables only.

See Also

- Table

© 1997-2021 Devart. All Rights Reserved.

5.15.1.17.2.25 National Property

Used to determine whether the parameter is in National charset.

Class
TOraParam

Syntax

```
property National: Boolean default False;
```

Remarks

Use the National property to determine or set whether a parameter is in National character set. When National is True, the parameter is in National character set, otherwise the parameter is in default character set.

The parameter datatype must be one of the following: ftString, ftFixedChar, ftWideString, ftMemo or ftFixedWideChar.
5.15.1.17.2.26 Table Property

Used to determine if the parameter is a PL/SQL table.

**Class**

TOraParam

**Syntax**

```plaintext
property Table: boolean default False;
```

**Remarks**

Set the Table property to True when parameter is a PL/SQL table. The maximum length of the table can be set by the Length property. Table may be fString, fInteger, fFloat or fDate type only. To access an item of a table use the ItemAsString, ItemAsInteger, ItemAsFloat, ItemAsDateTime and ItemIsNull properties.

**See Also**

- Length

---

5.15.1.17.3 Methods

Methods of the TOraParam class.

For a complete list of the TOraParam class members, see the TOraParam Members topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignField</td>
<td>(inherited from TDAParam) Assigns field name and field value to a param.</td>
</tr>
<tr>
<td>AssignFieldValue</td>
<td>(inherited from TDAParam) Assigns the specified field properties and value to a parameter.</td>
</tr>
</tbody>
</table>
### ItemClear Method

Assigns a NULL value to a table parameter item.

**Class**

TOraParam

**Syntax**

```pascal
procedure ItemClear(Index: integer);
```

**Parameters**

*Index*

Holds the item index.

**Remarks**

Call the `ItemClear` method to assign a NULL value to a table parameter item.

Index starts with 1 and lasts till `Length`.

© 1997-2021

Devart. All Rights Reserved.

Request Support    DAC Forum    Provide Feedback
5.15.1.17.3.2 SetBlobData Method

Sets the parameter value from the memory buffer.

Class

TOraParam

Syntax

procedure SetBlobData(Buffer: IntPtr; Size: integer);

Parameters

Buffer

Holds the pointer to the data.

Size

Holds the buffer size.

Remarks

Use the SetBlobData method to set the parameter value from the memory buffer. After this procedure call DataType property is assigned to #ftBlob#.

© 1997-2021
Devart. All Rights Reserved.

5.15.1.18 TOraParams Class

Used to control TOraParam objects.

For a list of all members of this type, see TOraParams members.

Unit

Ora

Syntax

TOraParams = class(TDAParams);

Remarks

Use TOraParams to manage a list of TOraParam objects for an object that uses field parameters. For example, TOraStoredProc objects and TOraQuery objects use TOraParams objects to create and access their parameters.
Inheritance Hierarchy

**TDAParams**

**TOraParams**

See Also

- **TOraParam**
- **TCustomDASQL.Params**
- **TCustomDADataSet.Params**
- **TOraDataSet.Params**
- **TOraSQL.Params**

© 1997-2021 Devart. All Rights Reserved.

5.15.1.18.1 **Members**

**TOraParams** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Items</strong></td>
<td>Used to iterate through all field parameters.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FindParam</strong> (inherited from <strong>TDAParams</strong>)</td>
<td>Searches for a parameter with the specified name.</td>
</tr>
<tr>
<td><strong>ParamByName</strong> (inherited from <strong>TDAParams</strong>)</td>
<td>Searches for a parameter with the specified name.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.15.1.18.2 **Properties**

Properties of the **TOraParams** class.

For a complete list of the **TOraParams** class members, see the [**TOraParams Members**](#).
Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to iterate through all field parameters.</td>
</tr>
</tbody>
</table>

See Also
- TOraParams Class
- TOraParams Class Members

Parameters

- **Index**
  - Holds the index in the range 0..Count - 1.

Remarks

Use the Items property to iterate through all field parameters. Index identifies the index in the range 0..Count - 1. Items can refer to a particular parameter by its index, but the TDAParams.ParamByName method is preferred to avoid depending on the order of the parameters.
5.15.1.19 TOraPoolingOptions Class

This class allows setting up the behaviour of the connection pool.

For a list of all members of this type, see `TOraPoolingOptions` members.

Unit
ora

Syntax

```pascal
TOraPoolingOptions = class(TPoolingOptions);
```

Inheritance Hierarchy

- `TPoolingOptions`
- `TOraPoolingOptions`

5.15.1.19.1 Members

`TOraPoolingOptions` class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectionLifetime</td>
<td>Used to specify the maximum time during which an opened connection can be used by connection pool. (inherited from <code>TPoolingOptions</code>)</td>
</tr>
<tr>
<td>MaxPoolSize</td>
<td>Used to specify the maximum number of connections that can be opened in connection pool. (inherited from <code>TPoolingOptions</code>)</td>
</tr>
<tr>
<td>MinPoolSize</td>
<td>Used to specify the minimum number of connections that can be opened in the connection pool. (inherited from <code>TPoolingOptions</code>)</td>
</tr>
<tr>
<td>PoolType</td>
<td>Used to specify the pool type.</td>
</tr>
<tr>
<td>ProxyPassword</td>
<td>Used to specify a password</td>
</tr>
</tbody>
</table>
Properties of the `TOraPoolingOptions` class.

For a complete list of the `TOraPoolingOptions` class members, see the `TOraPoolingOptions Members` topic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ConnectionLifetime</strong> (inherited from <code>TPoolingOptions</code>)</td>
<td>Used to specify the maximum time during which an opened connection can be used by connection pool.</td>
</tr>
<tr>
<td><strong>MaxPoolSize</strong> (inherited from <code>TPoolingOptions</code>)</td>
<td>Used to specify the maximum number of connections that can be opened in connection pool.</td>
</tr>
<tr>
<td><strong>MinPoolSize</strong> (inherited from <code>TPoolingOptions</code>)</td>
<td>Used to specify the minimum number of connections that can be opened in the connection pool.</td>
</tr>
<tr>
<td><strong>PoolType</strong></td>
<td>Used to specify the pool type.</td>
</tr>
<tr>
<td><strong>ProxyPassword</strong></td>
<td>Used to specify a password for proxy pooling.</td>
</tr>
<tr>
<td><strong>ProxyUsername</strong></td>
<td>Used to specify a user name for proxy pooling.</td>
</tr>
<tr>
<td><strong>Validate</strong> (inherited from <code>TPoolingOptions</code>)</td>
<td>Used for a connection to be validated when it is returned from the pool.</td>
</tr>
</tbody>
</table>
See Also
- TOraPoolingOptions Class
- TOraPoolingOptions Class Members

5.15.1.19.2.1  PoolType Property

Used to specify the pool type.

Class
TOraPoolingOptions

Syntax

| property PoolType: TOraPoolingType default optLocal; |

Remarks

Use the PoolType property to specify the pool type. Note that you should explicitly include T:Devart.Odac.Units.OraConnectionPool unit to "uses" list to use the PoolType option at runtime.

5.15.1.19.2.2  ProxyPassword Property

Used to specify a password for proxy pooling.

Class
TOraPoolingOptions

Syntax

| property ProxyPassword: string; |

Remarks

Use the ProxyPassword property to specify a password for proxy pooling.
5.15.1.19.2.3 ProxyUsername Property

Used to specify a user name for proxy pooling.

Class

**TOraPoolingOptions**

Syntax

```property
ProxyUsername: string;
```

Remarks

Use the ProxyUsername to specify a user name for the proxy pooling. Pool connections are stored with the same Username/Password properties. When giving connection from the pool, a connection under another user is created, based on one of these connections. Thus, connections under various users can be get from the pool.

5.15.1.20 TOraQuery Class

A component for executing queries and operating record sets. It also provides flexible way to update data.

For a list of all members of this type, see **TOraQuery** members.

Unit

**Ora**

Syntax

```TOraQuery = class(TCustomOraQuery);
```

Remarks

TOraQuery is a direct descendant of the **TOraDataSet** component. It publishes most of its inherited properties and events so that they can be manipulated at design-time.

Use TOraQuery to perform fetching, insertion, deletion and update of record by dynamically
generated SQL statements. TOraQuery provides automatic blocking of records, their checking before edit and refreshing after post. Set SQL, SQLInsert, SQLDelete, SQLRefresh, and SQLUpdate properties to define SQL statements for subsequent accesses to the database server. There is no restriction to their syntax, so any SQL statement is allowed. Usually you need to use INSERT, DELETE, and UPDATE statements but you also may use stored procedures in more diverse cases.

To modify records of TOraQuery SELECT statement in SQL, property should retrieve ROWID of updating table. To modify records, you can specify KeyFields. If they are not specified, TOraQuery will retrieve primary keys for UpdatingTable from metadata. TOraQuery can automatically update only one table. Updating table is defined by the UpdatingTable property if this property is set. Otherwise, the table a field of which is the first field in the field list in the SELECT clause is used as an updating table.

The SQLInsert, SQLDelete, SQLUpdate, SQLRefresh properties support automatic binding of parameters which have identical names to fields captions. To retrieve the value of a field as it was before the operation use the field name with the 'OLD_' prefix. This is especially useful when doing field comparisons in the WHERE clause of the statement. Use the TCustomDADataset.BeforeUpdateExecute event to assign the value to additional parameters and the TCustomDADataset.AfterUpdateExecute event to read them.

TOraQuery performs read-only access if none of SQLInsert, SQLDelete, SQLUpdate properties is defined.

Inheritance Hierarchy

TMemDataSet
  TCustomDADataset
    TOraDataSet
      TCustomOraQuery
        TOraQuery

See Also
- Updating Data with ODAC Dataset Components
- Master/Detail Relationships
- TOraStoredProc
- TOraTable

© 1997-2021 Devart. All Rights Reserved.
The **TOraQuery** class overview.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BaseSQL</strong></td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td><strong>CachedUpdates</strong></td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>ChangeNotification</strong></td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td><strong>CheckMode</strong></td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td><strong>CommandTimeout</strong></td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Disconnected</td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td>DMLRefresh</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>Encryption</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>FetchAll</td>
<td></td>
</tr>
<tr>
<td>FetchRows</td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td>FilterSQL</td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td>FinalSQL</td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td>IsPLSQL</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>IsQuery</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>KeyExclusive</td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td>KeyFields</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KeySequence</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>LockMode</td>
<td>Used to specify what kind of lock will be performed when editing a record.</td>
</tr>
<tr>
<td>MacroCount</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>MasterFields</td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td>MasterSource</td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>Used to execute a SQL statement and fetch rows by a separate thread.</td>
</tr>
<tr>
<td>Options</td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td>OptionsDS</td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL</td>
</tr>
<tr>
<td>Property</td>
<td>Inheritance</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>ParamCount</td>
<td>inherited from TCustomDADataset</td>
</tr>
<tr>
<td>Params</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>Prepared</td>
<td>inherited from TMemDataSet</td>
</tr>
<tr>
<td>Ranged</td>
<td>inherited from TMemDataSet</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>inherited from TCustomDADataset</td>
</tr>
<tr>
<td>RefreshMode</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>inherited from TCustomDADataset</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>RowsAffected</td>
<td>inherited from TCustomDADataset</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>Session</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>SmartFetch</td>
<td>inherited from TOraDataSet</td>
</tr>
<tr>
<td>SQL</td>
<td>inherited from TCustomDADataset</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>inherited from TCustomDADataset</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td>SQLLock</td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataSet.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td>UpdateObject</td>
<td>Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
</tbody>
</table>
UpdatesPending (inherited from TMemDataSet)  Used to check the status of the cached updates buffer.

UpdatingTable  Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere (inherited from TCustomDADataset)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange (inherited from TMemDataSet)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates (inherited from TMemDataSet)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>BreakExec (inherited from TCustomDADataset)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td>CancelRange (inherited from TMemDataSet)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates (inherited from TMemDataSet)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates (inherited from TMemDataSet)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>CreateBlobStream (inherited from TCustomDADataset)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td>CreateProcCall (inherited from TOraDataSet)</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td>DeferredPost (inherited from TMemDataSet)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>DeleteWhere (inherited from TCustomDADataset)</td>
<td>Removes WHERE clause from the SQL property and</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>ErrorOffset</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Returns the parse error offset.</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to find out whether TCustomDADataset has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to learn whether TCustomDADataset is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to learn whether TCustomDADataset is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValue parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Retrieves a TOraArray object for a field when only its name is known.</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Overloaded. Retrieves TBlob object for a field or</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>GetDataType</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>GetErrorPos</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetFieldObject</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>GetFieldPrecision</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>GetFile</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetInterval</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>GetKeyList</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetLob</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetLobLocator</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetObject</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>GetRef</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetTable</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Locate (inherited from TMemDataSet)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx (inherited from TMemDataSet)</td>
<td>Overloaded. Excludes features that don’t need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Lock (inherited from TCustomDADataset)</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td>MacroByName (inherited from TCustomDADataset)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>OpenNext (inherited from TOraDataSet)</td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td>ParamByName (inherited from TOraDataSet)</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td>Prepare (inherited from TCustomDADataset)</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>RefreshRecord (inherited from TCustomDADataset)</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td>RestoreSQL (inherited from TCustomDADataset)</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td>RestoreUpdates (inherited from TMemDataSet)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord (inherited from TMemDataSet)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveSQL (inherited from TCustomDADataset)</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td>SaveToXML (inherited from TMemDataSet)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetOrderBy (inherited from TCustomDADataset)</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
</tbody>
</table>
### SetRange (inherited from TMemDataSet)
Sets the starting and ending values of a range, and applies it.

### SetRangeEnd (inherited from TMemDataSet)
Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

### SetRangeStart (inherited from TMemDataSet)
Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

### SQLSaved (inherited from TCustomDADataset)
Determines if the SQL property value was saved to the BaseSQL property.

### UnLock (inherited from TCustomDADataset)
Releases a record lock.

### UnPrepare (inherited from TMemDataSet)
Frees the resources allocated for a previously prepared query on the server and client sides.

### UpdateResult (inherited from TMemDataSet)
Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

### UpdateStatus (inherited from TMemDataSet)
Indicates the current update status for the dataset when cached updates are enabled.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td>AfterFetch</td>
<td>Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td>AfterUpdateExecute</td>
<td>Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
<tr>
<td>BeforeFetch</td>
<td>Occurs before dataset is</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>BaseSQL</strong></td>
<td>(inherited from <strong>TCustomDADataSet</strong>) Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td><strong>CachedUpdates</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>ChangeNotification</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td><strong>CheckMode</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td><strong>CommandTimeout</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>Conditions</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td><strong>Connection</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>FetchRows</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsPLSQL (inherited from TораDataSet)</td>
<td>Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td>IsQuery (inherited from TораDataSet)</td>
<td>Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td>KeyExclusive (inherited from TMemDataSet)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>KeyFields (inherited from TораDataSet)</td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td>KeySequence (inherited from TораDataSet)</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints (inherited from TMemDataSet)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate (inherited from TMemDataSet)</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>MacroCount (inherited from TCustomDADataset)</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Macros (inherited from TCustomDADataset)</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>MasterFields (inherited from TCustomDADataset)</td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td>MasterSource (inherited from TCustomDADataset)</td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Options</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>OptionsDS</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>ParamCount</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Params</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Prepared</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Ranged</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RefreshMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>RowsAffected</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Session</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td><strong>Component</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>SmartFetch</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td><strong>SQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td><strong>SQLDelete</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td><strong>SQLInsert</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td><strong>SQLLock</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td><strong>SQLRecCount</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td><strong>SQLRefresh</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the <strong>TCustomDADataset.RefreshRecord</strong> procedure.</td>
</tr>
<tr>
<td><strong>SQLType</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td><strong>SQLUpdate</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td><strong>StrictUpdate</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
</tbody>
</table>
**UniDirectional** (inherited from **TCustomDADataset**)

Used if an application does not need bidirectional access to records in the result set.

**UpdateObject** (inherited from **TOraDataSet**)

Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

**UpdateRecordTypes** (inherited from **TMemDataSet**)

Used to indicate the update status for the current record when cached updates are enabled.

**UpdatesPending** (inherited from **TMemDataSet**)

Used to check the status of the cached updates buffer.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FetchAll</td>
<td>Defines whether to request all records of the query from database server when the dataset is being opened.</td>
</tr>
<tr>
<td>LockMode</td>
<td>Used to specify what kind of lock will be performed when editing a record.</td>
</tr>
<tr>
<td>UpdatingTable</td>
<td>Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.</td>
</tr>
</tbody>
</table>

### See Also

- [TOraQuery Class](#)
- [TOraQuery Class Members](#)

© 1997-2021

Devart. All Rights Reserved.
5.15.1.20.2.1  FetchAll Property

Defines whether to request all records of the query from database server when the dataset is being opened.

Class
T0raQuery

Syntax
```
property FetchAll: boolean;
```

Remarks
When set to True, all records of the query are requested from database server when the dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to `TMemDataSet.Locate` and `TMemDataSet.LocateEx` methods may take a lot of time to retrieve additional records to the client side.

5.15.1.20.2.2  LockMode Property

Used to specify what kind of lock will be performed when editing a record.

Class
T0raQuery

Syntax
```
property LockMode: TLockMode default lmNone;
```

Remarks
Use the LockMode property to define what kind of lock will be performed when editing a record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time.
Locking is performed by the RefreshRecord method.

The default value is lmNone.

To set pessimistic locking use LockMode = lmLockImmediate, $TOraDataSet.CheckMode = cmException. To set optimistic locking use LockMode = lmLockDelayed, CheckMode = cmException.

See Also
- $TOraStoredProc.LockMode
- $TOraDataTable.LockMode

5.15.1.20.2.3 UpdatingTable Property

Used to specify which table in a query is assumed to be the target for subsequent data-modification queries as a result of user incentive to insert, update or delete records.

Class
$TOraQuery

Syntax

```property UpdatingTable: string:
```

Remarks

Use the UpdatingTable property to specify which table in a query is assumed to be the target for the subsequent data-modification queries as a result of user incentive to insert, update or delete records.

This property is used on Insert, Update, Delete or RefreshRecord (see also $TOraDataSet.Options) if appropriate SQL (SQLInsert, SQLUpdate or SQLDelete) is not provided.

If UpdatingTable is not set then the first table used in a query is assumed to be the target.
5.15.1.21 TOraReferenceField Class

A class representing an Oracle REF field in a dataset.

For a list of all members of this type, see TOraReferenceField members.

Unit
Ora

Syntax
TOraReferenceField = class(TReferenceField);

Remarks
TOraReferenceField represents an Oracle REF field in a dataset.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified</td>
<td>Used to indicate whether a field was modified.</td>
</tr>
</tbody>
</table>

For a complete list of the TOraReferenceField class members, see the TOraReferenceField Members topic.
5.15.1.21.2.1 Modified Property

Used to indicate whether a field was modified.

Class

TOraReferenceField

Syntax

```pascal
property Modified: boolean;
```

Remarks

Use the Modify property to indicate whether a field was modified. The property is writable.

5.15.1.22 TOraSession Class

A component for maintaining connection to an Oracle database.

For a list of all members of this type, see TOraSession members.

Unit

ora

Syntax

```pascal
TOraSession = class(TCustomDAConnection);
```

Remarks

The TOraSession component is used to maintain connection to an Oracle database. After
setting the Username, Password and Server properties, you can establish a connection to the
database by calling the Connect method or setting the Connected property to True. There are
also many properties at the session level that affect the default behavior of the queries
executed within this session. Furthermore, you can control transactions using methods from
this class.

All components that are dedicated to perform data access, such as TOraQuery, TOraSQL,
TOraScript, must have their Session property assigned with one of the TOraSession
instances.

Inheritance Hierarchy

TCustomDACConnection

TOraSession

See Also

• TOraDataSet.Session
• TOraSQL.Session

5.15.1.22.1 Members

TOraSession class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCommit</td>
<td>Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.</td>
</tr>
<tr>
<td>ConnectDialog</td>
<td>(inherited from TCustomDACConnection) Allows to link a TCustomConnectDialog component.</td>
</tr>
<tr>
<td>Connected</td>
<td>Used to indicate if the database connection is active.</td>
</tr>
<tr>
<td>ConnectMode</td>
<td>Used to specify the system privileges to use when a</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ConnectPrompt</td>
<td>Used to supply a prompt for a name and password.</td>
</tr>
<tr>
<td>ConnectString</td>
<td>Used to specify the connection information, such as: UserName, Password, Server, etc.</td>
</tr>
<tr>
<td>ConvertEOL</td>
<td>Allows customizing line breaks in string fields and parameters.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display SQL statements being executed with their parameter values and data types.</td>
</tr>
<tr>
<td>Home</td>
<td>Used to specify the Oracle client to be used in the application.</td>
</tr>
<tr>
<td>HomeName</td>
<td>Used to select the Oracle client to use with the application.</td>
</tr>
<tr>
<td>HttpOptions</td>
<td>Used to set up the HTTP options.</td>
</tr>
<tr>
<td>InternalName</td>
<td>Used to get or set the client database name that will be recorded when performing global transactions.</td>
</tr>
<tr>
<td>InTransaction</td>
<td>Indicates whether the transaction is active.</td>
</tr>
<tr>
<td>LastError</td>
<td>Used to get an error code which resulted from previous call to the OCI interface function.</td>
</tr>
<tr>
<td>LDA</td>
<td>Provides a pointer to Oracle 7 login data area of the current connection.</td>
</tr>
<tr>
<td>LoginPrompt</td>
<td>Specifies whether a login dialog appears immediately before opening a new connection.</td>
</tr>
<tr>
<td>OCICallStyle</td>
<td>Indicates the set of OCI routines used.</td>
</tr>
<tr>
<td>OCISvcCtx</td>
<td>Used to return Oracle 8 service context handle of the current connection.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td>Used to specify the behaviour of a TOraSession object.</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td><strong>OracleVersion</strong></td>
<td>Used to get Oracle server version number as string.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Used to specify a password for a connection.</td>
</tr>
<tr>
<td><strong>Pooling</strong></td>
<td>Enables or disables using connection pool.</td>
</tr>
<tr>
<td>(inherited from <strong>TCustomDAConnection</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>PoolingOptions</strong></td>
<td>Used to specify the behaviour of connection pool.</td>
</tr>
<tr>
<td><strong>ProxySession</strong></td>
<td>Used to enable multiple user sessions within a single database session.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td>Used to change the current schema of the session to the specified schema.</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>Contains the server name.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Uses embedded TOraSQL object to execute any SQL statement.</td>
</tr>
<tr>
<td><strong>SSLOptions</strong></td>
<td>Used to set up the SSL options.</td>
</tr>
<tr>
<td><strong>ThreadSafety</strong></td>
<td>Used to allow the usage of the OCI in multi-threaded environment.</td>
</tr>
<tr>
<td><strong>Username</strong></td>
<td>Contains username.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TCustomDAConnection</strong>)</td>
<td>Overloaded. Applies changes in datasets.</td>
</tr>
<tr>
<td><strong>AssignConnect</strong></td>
<td>Shares database connection between the TOraSession components.</td>
</tr>
<tr>
<td><strong>AssignLDA</strong></td>
<td>The method is used to assign LDA handle.</td>
</tr>
<tr>
<td><strong>AssignSvcCtx</strong></td>
<td>Overloaded. The method is used to assign service context handle.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ChangePassword</td>
<td>Changes the password of an account to the new password.</td>
</tr>
<tr>
<td>Commit (inherited from TCustomDACo\nc</td>
<td>Commits current transaction.</td>
</tr>
<tr>
<td>Connect (inherited from TCustomDACo\nc</td>
<td>Establishes a connection to the server.</td>
</tr>
<tr>
<td>CreateSQL (inherited from TCustomDACo\nc</td>
<td>Creates a component for queries execution.</td>
</tr>
<tr>
<td>Disconnect (inherited from TCustomDACo\nc</td>
<td>Performs disconnect.</td>
</tr>
<tr>
<td>ExecProc (inherited from TCustomDACo\nc</td>
<td>Allows to execute stored procedure or function providing its name and parameters.</td>
</tr>
<tr>
<td>ExecProcEx (inherited from TCustomDACo\nc</td>
<td>Allows to execute a stored procedure or function.</td>
</tr>
<tr>
<td>ExecSQL (inherited from TCustomDACo\nc</td>
<td>Executes a SQL statement with parameters.</td>
</tr>
<tr>
<td>ExecSQLEx (inherited from TCustomDACo\nc</td>
<td>Executes any SQL statement outside the TQuery or TSQL components.</td>
</tr>
<tr>
<td>GetDatabaseNames (inherited from TCustomDACo\nc</td>
<td>Returns a database list from the server.</td>
</tr>
<tr>
<td>GetKeyFieldNames (inherited from TCustomDACo\nc</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetSequenceNames</td>
<td>Provides the names of available sequences.</td>
</tr>
<tr>
<td>GetStoredProcNames (inherited from TCustomDACo\nc</td>
<td>Returns a list of stored procedures from the server.</td>
</tr>
<tr>
<td>GetTableNames (inherited from TCustomDACo\nc</td>
<td>Provides a list of available tables names.</td>
</tr>
<tr>
<td>MonitorMessage (inherited from TCustomDACo\nc</td>
<td>Sends a specified message through the TCustomDASQLMonitor component.</td>
</tr>
<tr>
<td>ParamByName</td>
<td>Provides access to OUT parameters and their values after processing SQL state\nm with ExecSQL or stored procedure with</td>
</tr>
</tbody>
</table>
Ping (inherited from TCustomDACConnection) Used to check state of connection to the server.
RemoveFromPool (inherited from TCustomDACConnection) Marks the connection that should not be returned to the pool after disconnect.
Rollback (inherited from TCustomDACConnection) Discards all current data changes and ends transaction.
RollbackToSavepoint Cancels all updates for the current transaction.
Savepoint Defines a point in the transaction to which you can roll back later.
StartTransaction Overloaded. Begins a new user transaction against the database server.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnConnectChange</td>
<td>Occurs after Connected property was changed.</td>
</tr>
<tr>
<td>OnConnectionLost</td>
<td>This event occurs when connection was lost.</td>
</tr>
<tr>
<td>OnError</td>
<td>This event occurs when an error has arisen in the connection.</td>
</tr>
<tr>
<td>OnFailover</td>
<td>Occurs when Transparent Application Failover (TAF) seamlessly attempts to failover to another Oracle instance.</td>
</tr>
<tr>
<td>OnInfoMessage</td>
<td>The event occurs if the server returned the OCI_SUCCESS_WITH_INFO error.</td>
</tr>
</tbody>
</table>
5.15.1.22.2 Properties

Properties of the **TOraSession** class.

For a complete list of the **TOraSession** class members, see the **TOraSession Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConnectDialog</td>
<td>Allows to link a <strong>TCustomConnectDialog</strong> component.</td>
</tr>
<tr>
<td>ConnectString</td>
<td>Used to specify the connection information, such as: UserName, Password, Server, etc.</td>
</tr>
<tr>
<td>ConvertEOL</td>
<td>Allows customizing line breaks in string fields and parameters.</td>
</tr>
<tr>
<td>InternalName</td>
<td>Used to get or set the client database name that will be recorded when performing global transactions.</td>
</tr>
<tr>
<td>InTransaction</td>
<td>Indicates whether the transaction is active.</td>
</tr>
<tr>
<td>LastError</td>
<td>Used to get an error code which resulted from previous call to the OCI interface function.</td>
</tr>
<tr>
<td>LDA</td>
<td>Provides a pointer to Oracle 7 login data area of the current connection.</td>
</tr>
<tr>
<td>LoginPrompt</td>
<td>Specifies whether a login dialog appears immediately before opening a new connection.</td>
</tr>
<tr>
<td>OCICallStyle</td>
<td>Indicates the set of OCI routines used.</td>
</tr>
<tr>
<td>OCISvcCtx</td>
<td>Used to return Oracle 8 service context handle of the current connection.</td>
</tr>
<tr>
<td>OracleVersion</td>
<td>Used to get Oracle server version number as string.</td>
</tr>
</tbody>
</table>
### Pooling (inherited from TCustomDACConnection)

Enables or disables using connection pool.

### ProxySession

Used to enable multiple user sessions within a single database session.

### SQL

Uses embedded TOraSQL object to execute any SQL statement.

#### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AutoCommit</strong></td>
<td>Used to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server.</td>
</tr>
<tr>
<td><strong>Connected</strong></td>
<td>Used to indicate if the database connection is active.</td>
</tr>
<tr>
<td><strong>ConnectMode</strong></td>
<td>Used to specify the system privileges to use when a user connects to the server.</td>
</tr>
<tr>
<td><strong>ConnectPrompt</strong></td>
<td>Used to supply a prompt for a name and password.</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used to display SQL statements being executed with their parameter values and data types.</td>
</tr>
<tr>
<td><strong>Home</strong></td>
<td>Used to specify the Oracle client to be used in the application.</td>
</tr>
<tr>
<td><strong>HomeName</strong></td>
<td>Used to select the Oracle client to use with the application.</td>
</tr>
<tr>
<td><strong>HttpOptions</strong></td>
<td>Used to set up the HTTP options.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td>Used to specify the behaviour of a TOraSession object.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>Used to specify a password for a connection.</td>
</tr>
</tbody>
</table>
PoolingOptions | Used to specify the behaviour of connection pool.

Schema | Used to change the current schema of the session to the specified schema.

Server | Contains the server name.

SSLOptions | Used to set up the SSL options.

ThreadSafety | Used to allow the usage of the OCI in multi-threaded environment.

Username | Contains username.

See Also
- TOraSession Class
- TOraSession Class Members

Class
TOraSession

Syntax

```plaintext
property AutoCommit: boolean;
```

Remarks
Use the AutoCommit property to permit or prevent permanent updates, insertions, and deletions of data associated with the current transaction against the database server without explicit calls to the Commit or Rollback methods.

Set AutoCommit to True to permit implicit call to Commit method after every database access.
AutoCommit property in TOraSession has higher precedence over the same properties in dataset components. Its default value is True.

**Note:** The AutoCommit property in TOraSession globally specifies whether all queries to modify a database are implicitly committed or not. Components which descend from the TCustomDADataset and TCustomDASQL classes inherit their AutoCommit properties. This allows them to specify their implicit transaction selectively committing the behavior after each data modifying access.

This is an example of procedure that removes all records from Dept table and makes this change permanent.

**Example**

```delphi
type
  OraSession = class;

procedure TForm1.DeleteClick(Sender: TObject);
begin
  OraSQL.Session := OraSession;
  OraSession.AutoCommit := True;
  OraSQL.AutoCommit := False;
  OraSQL.SQL := 'DELETE FROM Dept';
  OraSQL.Execute; // delete all records, commit is not performed
  OraSession.Rollback;// restore deleted records
  OraSession.AutoCommit := False;
  OraSQL.AutoCommit := True;
  OraSQL.SQL := 'DELETE FROM Dept';
  OraSQL.Execute; // delete all records, commit is not performed
  OraSession.Rollback; // restore deleted records
  OraSession.AutoCommit := True;
  OraSQL.SQL := 'DELETE FROM Dept';
  OraSQL.Execute; // delete all records, commit is performed
  OraSession.Rollback; // couldn't restore deleted records
end;
```

**See Also**
- [TCustomDACConnection.Commit](#)

© 1997-2021
Devart. All Rights Reserved.

5.15.1.22.2.2 Connected Property

Used to indicate if the database connection is active.

**Class**

TOraSession
**Syntax**

```plaintext
property Connected stored IsConnectedStored;
```

**Remarks**

Use the Connected property to indicate whether the database connection is active. Setting this property is equivalent to calling the `TCustomDAConnection.Connect` or `TCustomDAConnection.Disconnect` methods at runtime.

`OnConnectChange` event occurs after the Connected property has been changed.

**See Also**

- `TCustomDAConnection.Connect`
- `TCustomDAConnection.Disconnect`
- `OnConnectChange`

---

**5.15.1.22.2.3  ConnectMode Property**

Used to specify the system privileges to use when a user connects to the server.

**Class**

`T0raSession`

**Syntax**

```plaintext
property ConnectMode: TConnectMode default DefValConnectMode;
```

**Remarks**

Use the ConnectMode property to specify which system privileges to use when a user connects to the server.

**Note:** User must have SYSOPER, SYSDBA or both these roles granted before he connects to the server and wishes to use either of these roles. ConnectMode is not supported for OCI 7.

**See Also**

- `Password`
5.15.1.22.2.4 ConnectPrompt Property

Used to supply a prompt for a name and password.

Class

T0raSession

Syntax

```
property ConnectPrompt: boolean stored False default True;
```

Remarks

Set the ConnectPrompt property to True to provide login support when establishing a connection. When ConnectPrompt is True, a dialog appears to prompt a user for a name and a password.

When ConnectPrompt is False, an application must supply user name and password values programmatically.

**Warning:** Storing a hard-coded user name and password entries as property values or in code for an OnLogin event handler can compromise server security.

See Also

- Password
- Server
- Username
- TCustomDACConnection.ConnectionString

5.15.1.22.2.5 Debug Property

Used to display SQL statements being executed with their parameter values and data types.

Class
TOraSession

Syntax

```pascal
property Debug: boolean;
```

Remarks

Set the Debug property to True to display SQL statements being executed with their parameter values and data types.

**Note**: To use this property you should explicitly include OdacVcl (OdacClx under Linux) unit to your project.

If TOraSQLMonitor is used in the project and the TOraSQLMonitor.Active property is set to False, the debug window is not displayed.

© 1997-2021

© 1997-2021

© 1997-2021

© 1997-2021

Request Support

5.15.1.22.2.6 Home Property

Used to specify the Oracle client to be used in the application.

Class

TOraSession

Syntax

```pascal
property Home: TOracleHome stored False;
```

Remarks

Set the Home property to select which Oracle client will be used in your application. Use this property in cases when there is a number of Oracle clients on the machine. ODAC searches all available homes in the HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ALL_HOMES registry folder.

See Also

- TCustomDACConnection.Connect
- TOracleHome
Reserved.

5.15.1.22.2.7 HomeName Property

Used to select the Oracle client to use with the application.

Class

TOraSession

Syntax

```
property HomeName: string;
```

Remarks

Use the HomeName property to select which Oracle client will be used in your application. Use this property in cases when there is a number of Oracle clients on the machine. ODAC searches all available homes in HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE registry folder. If HomeName property is set to '', ODAC uses first directory from the list of homes encountered in environment PATH variable as default Oracle home.

See Also

- TCustomDACConnection.Connect
- TOracleHome

© 1997-2021 Devart. All Rights Reserved.

5.15.1.22.2.8 HttpOptions Property

Used to set up the HTTP options.

Class

TOraSession

Syntax

```
property HttpOptions: THttpOptions;
```

Remarks

Use the HttpOptions property to set up the HTTP options.
5.15.1.22.2.9 InternalName Property

Used to get or set the client database name that will be recorded when performing global transactions.

Class

TOraSession

Syntax

```property InternalName: string;```

Remarks

Use the InternalName property to get or set the client database name that will be recorded when performing global transactions. While there is no actual global transaction support, setting this property to a non-empty string can give performance gains on SQL statement execution. But there is one undesirable effect: you cannot commit or rollback transaction from PL/SQL block. You should call TCustomDAConnection.Commit or TCustomDAConnection.Rollback explicitly.

See Also

- TCustomDAConnection.Commit
- TCustomDAConnection.Rollback

5.15.1.22.2.10 LastError Property

Used to get an error code which resulted from previous call to the OCI interface function.

Class

TOraSession

Syntax

```property LastError: integer;```
Remarks

Use the LastError property to get an error code which resulted from previous call to the OCI interface function.

5.15.1.22.2.11  LDA Property

Provides a pointer to Oracle 7 login data area of the current connection.

Class

T0raSession

Syntax

```
property LDA: PLDA;
```

Remarks

Call the LDA method to get a pointer to Oracle 7 login data area of the current connection.

LDA structure is relevant mainly to OCI 7 call interface.

See Also

- OCISvcCtx
- AssignConnect

5.15.1.22.2.12  OCICallStyle Property

Indicates the set of OCI routines used.

Class

T0raSession

Syntax

```
property OCICallStyle: TOCICallStyle;
```
Remarks

Use the OCICallStyle property to check what set of OCI routines is used. Write OCICallStyle before connection to specify that either OCI 7.3 or OCI 8.0 routines will be used. TOraSession initializes this property on behalf of the default OCI client found in the system at the time when OCI library is being loaded.

See Also
- Options

5.15.1.22.2.13 OCISvcCtx Property

Used to return Oracle 8 service context handle of the current connection.

Class

TOraSession

Syntax

```plaintext
property OCISvcCtx: TOCISvcCtx;
```

Remarks

Use the OCISvcCtx property to return Oracle 8 service context handle of the current connection.

See Also
- LDA
- AssignConnect

5.15.1.22.2.14 Options Property

Used to specify the behaviour of a TOraSession object.

Class
**TOraSession**

**Syntax**
```
property Options: TOraSessionOptions;
```

**Remarks**
Set the properties of Options to specify the behaviour of a TOraSession object.

Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CharLength</td>
<td>Used to specify the size of a single character in bytes.</td>
</tr>
<tr>
<td>Charset</td>
<td>Used to set the character set that ODAC uses to read and write character data.</td>
</tr>
<tr>
<td>ClientIdentifier</td>
<td>Used to determine the client identifier in the session.</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Used to specify the time to wait for a connection to open before raising an exception.</td>
</tr>
<tr>
<td>ConvertEOL</td>
<td>Affects the line break behavior in string fields and parameters.</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Used to specify the default date format used when Oracle makes conversions from internal date format into string values and vice versa.</td>
</tr>
<tr>
<td>DateLanguage</td>
<td>Used to specify the default language used when Oracle parses internal date format into string values and vice versa.</td>
</tr>
<tr>
<td>Direct</td>
<td>Used for ODAC to connect directly over TCP/IP (in Direct mode) and without requiring Oracle software on the client side.</td>
</tr>
<tr>
<td>EnableBCD</td>
<td>Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td>EnableFMTBCD</td>
<td>Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td>EnableIntegers</td>
<td>Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.</td>
</tr>
<tr>
<td>EnableLargeint</td>
<td>Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.</td>
</tr>
<tr>
<td>EnableNumbers</td>
<td>Used for ODAC to map Oracle numbers with precision larger than 15 to TIntegerField.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EnableOraTimestamp</td>
<td>Used to create TOraTimeStampField for columns of TIMESTAMP data type.</td>
</tr>
<tr>
<td>IPVersion</td>
<td>Used to specify Internet Protocol Version.</td>
</tr>
<tr>
<td>OptimizerMode</td>
<td>Used to get or set the default optimizer mode for connection.</td>
</tr>
<tr>
<td>StatementCache</td>
<td>Used for ODAC to cache statement handles.</td>
</tr>
<tr>
<td>StatementCacheSize</td>
<td>Used to specify the statement handle cache size.</td>
</tr>
<tr>
<td>SubscriptionPort</td>
<td>Sets the client port used to receive notifications.</td>
</tr>
<tr>
<td>UnicodeEnvironment</td>
<td>Enables or disables using OCI Unicode Environment.</td>
</tr>
<tr>
<td>UseOCI7</td>
<td>Used to force TOraSession use OCI 7 call style only.</td>
</tr>
<tr>
<td>UseUnicode</td>
<td>Used to enable or disable Unicode support.</td>
</tr>
</tbody>
</table>

**See Also**
- Connecting in Direct Mode
- Unicode Character Data
- TOraNumberField
- TOraDataSet.Options
- TOraSQL.StatementCache

Used to get Oracle server version number as string.

**Class**

TOraSession

**Syntax**

```plaintext
property OracleVersion: string;
```

**Remarks**
Use the OracleVersion property to get Oracle server version number as string, for an example '7.3.2.3'

Works only when TOraSession instance is connected.

### 5.15.1.22.2.16 Password Property

Used to specify a password for a connection.

**Class**

**TOraSession**

**Syntax**

```
property Password: string;
```

**Remarks**

Use the Password property to specify a password for a connection. TOraSession uses Password to build connect string in the form `Username/Password@Server`.

When property is being changed TOraSession calls Disconnect method.

**See Also**

- [Username](#)
- [Server](#)
- [TCustomDACConnection.Connect](#)

### 5.15.1.22.2.17 PoolingOptions Property

Used to specify the behaviour of connection pool.

**Class**

**TOraSession**

**Syntax**
property PoolingOptions: TOraPoolingOptions;

Remarks
Set the properties of PoolingOptions to specify the behaviour of the connection pool.
Descriptions of all options are in the table below.

<table>
<thead>
<tr>
<th>Option Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PoolType</td>
<td>Used to specify the pool type.</td>
</tr>
<tr>
<td>ProxyPassword</td>
<td>Used to specify a password for proxy pooling.</td>
</tr>
<tr>
<td>ProxyUsername</td>
<td>Used to specify a user name for proxy pooling.</td>
</tr>
</tbody>
</table>

See Also
- TCustomDACConnection.Pooling

5.15.1.22.2.18 ProxySession Property

Used to enable multiple user sessions within a single database session.

Class
TOraSession

Syntax

property ProxySession: TOraSession;

Remarks
Applications can have multiple user sessions within a single database session. These "lightweight sessions" allow each user to be authenticated, preserving the identity of the real user through the middle tier.

The application server creates a proxy session for itself once it connects to a server. It authenticates itself to the database in a normal way creating the application server trust zone. The application server identity is now well known and trusted to the data server. Application server verifies the identity of a client. After that it can create TOraSession component and establish session for each client without authentication on Oracle server. That will reduce
time for connection.

For each client session you must refer the TOraSession.ProxySession property to proxy TOraSession object. TOraSession.Password may by empty for client session. To use this feature Oracle users must have CONNECT THROUGH privilege.

**Note:** ProxySession property in TOraSession is supported with OCI connection only when Options.Direct=False

5.15.1.22.2.19 Schema Property

Used to change the current schema of the session to the specified schema.

**Class**

TOraSession

**Syntax**

```
property Schema: string stored IsSchemaStored;
```

**Remarks**

Use the Schema property to change the current schema of the session to the specified schema. This setting offers a convenient way to perform operations on objects in a schema other than that of the current user without having to qualify the objects with the schema name. This setting changes the current schema, but it neither changes the session user or the current user, nor gives you any additional system or object privileges for the session.

If **Connected** = True read this property to receive the name of the current schema.

© 1997-2021
Devart. All Rights Reserved.

5.15.1.22.2.20 Server Property

Contains the server name.

**Class**

TOraSession
Syntax

```plaintext
property Server: string;
```

Remarks

Use the Server property to supply server name to handle server's request for a login.

Formatting of this property is different and depends on the value of Options.Direct property:
- If Options. Direct is set to False, Server assumes **TNS** alias name for the requested database or an **EZCONNECT** connection string.
- If Options. Direct is True, Server accepts a string holding three fields separated by a colon. If Oracle servers has SID is equal to Service Name then string is the following: "Host:Port:SID". If Oracle Server has SID differ from Service Name then string is the following: "Host:Port:sid=SID" for connection using SID and "Host:Port:sn=Service Name" for connection using Service Name. Here Host is an IP address of the server that hosts the database, Port is a port number that server listens, SID is a system identifier that specifies an Oracle database instance name, and Service Name is a system alias to an Oracle database instance (or many instances).

**Note:** EZCONNECT connection string format is supported in **Direct Mode** as well.

**Note:** If prefixes sid= or sn= aren't set, then connection will be established using SID.

See Also

- **Username**
- **Password**
- **TCustomDACConnection.Connect**
- **HomeName**
- **Connecting in Direct Mode**

© 1997-2021 Devart. All Rights Reserved.
**5.15.1.22.2.22 SSLOptions Property**

Used to set up the SSL options.

**Class**

*TOraSession*

**Syntax**

```property**
SSLOptions: *TOraConnectionSSLOptions*;
```

**Remarks**

Use the SSLOptions property to set up the SSL options.

---

**5.15.1.22.2.23 ThreadSafety Property**

Used to allow the usage of the OCI in multi-threaded environment.

**Class**

*TOraSession*

**Syntax**

```property**
```
property ThreadSafety: boolean default True;

Remarks
Use the ThreadSafety property to enable the usage of the OCI in multi-threaded environment. The ThreadSafety property must be True before any non-blocking fetch of rows or SQL statement execution takes place.

See Also
- TOraDataSet.NonBlocking
- TOraSQL.NonBlocking

Contains username.

Class
TOraSession

Syntax
property Username: string;

Remarks
Use the Username property to supply user name to handle server's request for a login.

TOraSession uses the Username, Password and Server properties to build connect string in the format Username/Password@Server.

When property is being changed TOraSession calls Disconnect method

See Also
- Password
- Server
- TCustomDAConnection.Connect
Methods of the **TOraSession** class.

For a complete list of the **TOraSession** class members, see the [TOraSession Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Overloaded. Applies changes in datasets.</td>
</tr>
<tr>
<td><strong>AssignConnect</strong></td>
<td>Shares database connection between the TOraSession components.</td>
</tr>
<tr>
<td><strong>AssignLDA</strong></td>
<td>The method is used to assign LDA handle.</td>
</tr>
<tr>
<td><strong>AssignSvcCtx</strong></td>
<td>Overloaded. The method is used to assign service context handle.</td>
</tr>
<tr>
<td><strong>ChangePassword</strong></td>
<td>Changes the password of an account to the new password.</td>
</tr>
<tr>
<td><strong>Commit</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Commits current transaction.</td>
</tr>
<tr>
<td><strong>Connect</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Establishes a connection to the server.</td>
</tr>
<tr>
<td><strong>CreateSQL</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Creates a component for queries execution.</td>
</tr>
<tr>
<td><strong>Disconnect</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Performs disconnect.</td>
</tr>
<tr>
<td><strong>ExecProc</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Allows to execute stored procedure or function providing its name and parameters.</td>
</tr>
<tr>
<td><strong>ExecProcEx</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Allows to execute a stored procedure or function.</td>
</tr>
<tr>
<td><strong>ExecSQL</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Executes a SQL statement with parameters.</td>
</tr>
<tr>
<td><strong>ExecSQLEX</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Executes any SQL statement outside the TQuery or TSQL components.</td>
</tr>
<tr>
<td><strong>GetDatabaseNames</strong> (inherited from <strong>TCustomDACConnection</strong>)</td>
<td>Returns a database list from the server.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetKeyFieldNames (inherited from TCustomDACConnection)</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetSequenceNames</td>
<td>Provides the names of available sequences.</td>
</tr>
<tr>
<td>GetStoredProcNames (inherited from TCustomDACConnection)</td>
<td>Returns a list of stored procedures from the server.</td>
</tr>
<tr>
<td>GetTableNames (inherited from TCustomDACConnection)</td>
<td>Provides a list of available tables names.</td>
</tr>
<tr>
<td>MonitorMessage (inherited from TCustomDACConnection)</td>
<td>Sends a specified message through the TCustomDASQLMonitor component.</td>
</tr>
<tr>
<td>ParamByName</td>
<td>Provides access to OUT parameters and their values after processing SQL statement with ExecSQL or stored procedure with ExecProc.</td>
</tr>
<tr>
<td>Ping (inherited from TCustomDACConnection)</td>
<td>Used to check state of connection to the server.</td>
</tr>
<tr>
<td>RemoveFromPool (inherited from TCustomDACConnection)</td>
<td>Marks the connection that should not be returned to the pool after disconnect.</td>
</tr>
<tr>
<td>Rollback (inherited from TCustomDACConnection)</td>
<td>Discards all current data changes and ends transaction.</td>
</tr>
<tr>
<td>RollbackToSavepoint</td>
<td>Cancels all updates for the current transaction.</td>
</tr>
<tr>
<td>Savepoint</td>
<td>Defines a point in the transaction to which you can roll back later.</td>
</tr>
<tr>
<td>StartTransaction</td>
<td>Overloaded. Begins a new user transaction against the database server.</td>
</tr>
</tbody>
</table>

See Also
- TOraSession Class
- TOraSession Class Members
5.15.1.22.3.1 AssignConnect Method

Shares database connection between the TOraSession components.

Class
TOraSession

Syntax

```procedure AssignConnect(Source: TOraSession); overload;
```

Parameters

Source
Points to a preconnected session and sets Connected property to True for this instance of TOraSession.

Remarks

Use the AssignConnect method to share database connection between the TOraSession components.

AssignConnect assumes that the Source parameter points to a preconnected session and sets Connected property to True for this instance of TOraSession. Note that AssignConnect doesn't make any references to the Source session. So before disconnecting parent session call AssignConnect(Nil) or Disconnect method for all assigned sessions.

Example

```pascal
OraSession1.Connect;
OraSession2.AssignConnect(OraSession1);                 // OraSession2.Connected is True
OraSQL.Session := OraSession2;
OraSQL.Execute;
OraSession2.AssignConnect(nil);                        // OraSession2.Connected is False
OraSession1.Disconnect;
```

See Also

- LDA
- OCISvcCtx
- TCustomDAConnection.Connect

© 1997-2021 Devart. All Rights Reserved.
5.15.1.22.3.2 AssignLDA Method

The method is used to assign LDA handle.

Class

T0raSession

Syntax

```
procedure AssignLDA(LDA: pLDA);
```

Parameters

LDA
Specifications the LDA handle.

5.15.1.22.3.3 AssignSvcCtx Method

The method is used to assign service context handle.

Class

T0raSession

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignSvcCtx(hOCISvcCtx: pOCISvcCtx)</td>
<td>The method is used to assign service context handle.</td>
</tr>
<tr>
<td>AssignSvcCtx(hOCISvcCtx: pOCISvcCtx; hOCIEnv: pOCIEnv)</td>
<td>The method is used to assign service context handle.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.  Request Support  DAC Forum  Provide Feedback
procedure AssignSvcCtx(hOCISvcCtx: pOCISvcCtx); overload;
deprecated;

Parameters
hOCISvcCtx
  Specifies the service context handle.

The method is used to assign service context handle.

Class
TOraSession

Syntax
procedure AssignSvcCtx(hOCISvcCtx: pOCISvcCtx; hOCIEnv: pOCIEnv);
overload;

Parameters
hOCISvcCtx
  Specifies the service context handle.

hOCIEnv
  Specifies the environment handle.

ChangePassword Method

Changes the password of an account to the new password.

Class
TOraSession

Syntax
procedure ChangePassword(NewPassword: string);

Parameters
NewPassword
  Takes the new password.
Remarks
Call the ChangePassword method to replace the current password of an account with the new password.

The previous values must be provided for the Password and UserName properties before calling ChangePassword.

The ChangePassword method is used mainly when logging in to the user account fails due to an expired password or any other reason accompanied by an exception with ORA-2800 Oracle error code family (see Oracle Error Messages).

See Also
- Username
- Password

5.15.1.22.3.5 GetSequenceNames Method

Provides the names of available sequences.

Class
T0raSession

Syntax

procedure GetSequenceNames(List: TStrings; AllSequences: boolean = False);

Parameters
List
Holds the list of available sequences names.

AllSequences
If True, method returns sequences from all schemas.

Remarks
Call the GetSequenceNames method to get the names of available sequences.

See Also
- TCustomDACConnection.GetTableNames
5.15.1.22.3.6  ParamByName Method

Provides access to OUT parameters and their values after processing SQL statement with ExecSQL or stored procedure with ExecProc.

Class

TOraSession

Syntax

function ParamByName(Name: string): TOraParam;

Parameters

Name
  Holds the parameter name.

Return Value
  a TOraParam object.

Remarks

Use the ParamByName method to get access to OUT parameters and their values after processing SQL statement with ExecSQL or stored procedure with ExecProc. Name should be equal to the parameter name as it occurred in SQL statement.

Implicitely calls ParamByName function of TOraSQL.

See Also

• SQL
• TCustomDACConnection.ExecSQL
• TCustomDACConnection.ExecSQLEx
5.15.1.22.3.7 RollbackToSavepoint Method

Cancels all updates for the current transaction.

Class

TOraSession

Syntax

```
procedure RollbackToSavepoint(const Name: string);
```

Parameters

- **Name**

Remarks

Call the RollbackToSavepoint method to cancel all updates for the current transaction and restore its state up to the moment of the last defined savepoint.

See Also

- Savepoint
- TCustomDAConnection.Rollback

5.15.1.22.3.8 Savepoint Method

Defines a point in the transaction to which you can roll back later.

Class

TOraSession

Syntax

```
procedure Savepoint(const Name: string);
```

Parameters

- **Name**

Remarks

Call the Savepoint method to define a point in the transaction to which you can roll back later. As the parameter, you can pass any valid name to identify the savepoint.
To roll back to the last savepoint call `RollbackToSavepoint`.

See Also
- `RollbackToSavepoint`

Begin a new transaction against the database server.

Class
`TOraSession`

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StartTransaction</code></td>
<td>Begins a new user transaction against the database server.</td>
</tr>
<tr>
<td><code>StartTransaction(IsolationLevel: TOraIsolationLevel; const RollbackSegment: string; const Name: string)</code></td>
<td>Begins a new user transaction against the database server.</td>
</tr>
</tbody>
</table>

Remarks

StartTransaction is an overload method for `TCustomDAConnection.StartTransaction`. Call the StartTransaction method to begin a new user transaction against the database server. Before
calling StartTransaction, an application should check the status of the InTransaction property. If InTransaction is True, it indicates that a transaction is already in progress, a subsequent call to StartTransaction without first calling TCustomDACConnection.Commit or TCustomDACConnection.Rollback to end the current transaction raises EDatabaseError. Calling StartTransaction when connection is closed also raises EDatabaseError.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until an application calls Commit to save the changes or Rollback to cancel them.

See Also
- TCustomDACConnection.Commit
- TCustomDACConnection.Rollback
- TCustomDACConnection.InTransaction
- TCustomDACConnection.StartTransaction

Begins a new user transaction against the database server.

Class
TOraSession

Syntax

```pascal
procedure StartTransaction(IsolationLevel: TOraIsolationLevel;
const RollbackSegment: string = ''; const Name: string = '');
reintroduce; overload;
```

Parameters

- **IsolationLevel**
  Specifies how the transactions containing database modifications are handled.
- **RollbackSegment**
  Holds the rollback segment to assign the current transaction to.
- **Name**
  Holds the current transaction name.

Remarks

Specify the RollbackSegment parameter to assign the current transaction to the specified rollback segment. This clause also implicitly establishes the transaction as a read/write
transaction.

The Name parameter is useful in distributed database environments when you must identify and resolve in-doubt transactions. The text string is limited to 255 bytes.

5.15.1.22.4 Events

Events of the TOraSession class.

For a complete list of the TOraSession class members, see the TOraSession Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnConnectionLost (inherited from TCustomDAConnection)</td>
<td>This event occurs when connection was lost.</td>
</tr>
<tr>
<td>OnError (inherited from TCustomDAConnection)</td>
<td>This event occurs when an error has arisen in the connection.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnConnectChange</td>
<td>Occurs after Connected property was changed.</td>
</tr>
<tr>
<td>OnFailover</td>
<td>Occurs when Transparent Application Failover (TAF) seamlessly attempts to failover to another Oracle instance.</td>
</tr>
<tr>
<td>OnInfoMessage</td>
<td>The event occurs if the server returned the OCI_SUCCESS_WITH_INFO error.</td>
</tr>
</tbody>
</table>

See Also
- TOraSession Class
- TOraSession Class Members
5.15.1.22.4.1  OnConnectChange Event

Occurs after Connected property was changed.

Class
T0raSession

Syntax

```
property OnConnectChange: TConnectChangeEvent;
```

Remarks

Occurs after the Connected property was changed. Connected parameter indicates whether the connection is active or not.

**Note:** This event is obsolete. Use AfterConnect and AfterDisconnect event handlers instead.

See Also

- TCustomDAConnection.Connect
- TCustomDAConnection.Disconnect

5.15.1.22.4.2  OnFailover Event

Occurs when Transparent Application Failover (TAF) seamlessly attempts to failover to another Oracle instance.

Class
T0raSession

Syntax

```
property OnFailover: TFailoverEvent;
```

Remarks

Occurs when Transparent Application Failover (TAF) seamlessly attempts to failover to
another Oracle instance.

FailoverType parameter specifies the type of failover. This allows the event to know what type of failover the client has requested.

See Also
- Transparent Application Failover Support

5.15.1.22.4.3 OnInfoMessage Event

The event occurs if the server returned the OCI_SUCCESS_WITH_INFO error.

Class

TOraSession

Syntax

```pascal
property OnInfoMessage: TInfoMessageEvent;
```

5.15.1.23 TOraSessionOptions Class

This class allows setting up the behaviour of the TOraSession class.

For a list of all members of this type, see TOraSessionOptions members.

Unit

ora

Syntax

```pascal
TOraSessionOptions = class(TDAConnectionOptions);
```

Inheritance Hierarchy

TDAConnectionOptions

TOraSessionOptions
### TOraSessionOptions class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowImplicitConnect</td>
<td>Specifies whether to allow or not implicit connection opening.</td>
</tr>
<tr>
<td>CharLength</td>
<td>Used to specify the size of a single character in bytes.</td>
</tr>
<tr>
<td>Charset</td>
<td>Used to set the character set that ODAC uses to read and write character data.</td>
</tr>
<tr>
<td>ClientIdentifier</td>
<td>Used to determine the client identifier in the session.</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Used to specify the time to wait for a connection to open before raising an exception.</td>
</tr>
<tr>
<td>ConvertEOL</td>
<td>Affects the line break behavior in string fields and parameters.</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Used to specify the default date format used when Oracle makes conversions from internal date format into string values and vice versa.</td>
</tr>
<tr>
<td>DateLanguage</td>
<td>Used to specify the default language used when Oracle parses internal date format into string values and vice versa.</td>
</tr>
<tr>
<td>DefaultSortType</td>
<td>Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the TMemDataSet.IndexFieldNames property of a dataset.</td>
</tr>
<tr>
<td>Direct</td>
<td>Used for ODAC to connect</td>
</tr>
<tr>
<td><strong>DisconnectedMode</strong> (inherited from <strong>TDAConnectionOptions</strong>)</td>
<td>Used to open a connection only when needed for performing a server call and closes after performing the operation.</td>
</tr>
<tr>
<td><strong>EnableBCD</strong></td>
<td>Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td><strong>EnableFMTBCD</strong></td>
<td>Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td><strong>EnableIntegers</strong></td>
<td>Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.</td>
</tr>
<tr>
<td><strong>EnableLargeint</strong></td>
<td>Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.</td>
</tr>
<tr>
<td><strong>EnableNumbers</strong></td>
<td>Used for ODAC to map Oracle numbers with precision larger than 15 to TOraNumberField.</td>
</tr>
<tr>
<td><strong>EnableOraTimestamp</strong></td>
<td>Used to create TOraTimeStampField for columns of TIMESTAMP data type.</td>
</tr>
<tr>
<td><strong>IPVersion</strong></td>
<td>Used to specify Internet Protocol Version.</td>
</tr>
<tr>
<td><strong>KeepDesignConnected</strong> (inherited from <strong>TDAConnectionOptions</strong>)</td>
<td>Used to prevent an application from establishing a connection at the time of startup.</td>
</tr>
<tr>
<td><strong>LocalFailover</strong> (inherited from <strong>TDAConnectionOptions</strong>)</td>
<td>If True, the <strong>TCustomDACConnection.OnConnectionLost</strong> event occurs and a failover operation can be performed after connection breaks.</td>
</tr>
</tbody>
</table>
### Properties of the **TOraSessionOptions** class.

For a complete list of the **TOraSessionOptions** class members, see the **TOraSessionOptions Members** topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DefaultSortType</strong></td>
<td>(inherited from <strong>TDACConnectionOptions</strong>) Used to determine the default type of local sorting for string fields. It is used when a sort type is not specified explicitly after the field name in the <strong>TMemDataSet.IndexFieldNames</strong> property of a dataset.</td>
</tr>
<tr>
<td><strong>DisconnectedMode</strong></td>
<td>(inherited from <strong>TDACConnectionOptions</strong>) Used to open a connection only when needed for performing a server call and closes after performing the operation.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AllowImplicitConnect</td>
<td>Specifies whether to allow or not implicit connection opening.</td>
</tr>
<tr>
<td>CharLength</td>
<td>Used to specify the size of a single character in bytes.</td>
</tr>
<tr>
<td>CharSet</td>
<td>Used to set the character set that ODAC uses to read and write character data.</td>
</tr>
<tr>
<td>ClientIdentifier</td>
<td>Used to determine the client identifier in the session.</td>
</tr>
<tr>
<td>ConnectionTimeout</td>
<td>Used to specify the time to wait for a connection to open before raising an exception.</td>
</tr>
<tr>
<td>ConvertEOL</td>
<td>Affects the line break behavior in string fields and parameters.</td>
</tr>
<tr>
<td>DateFormat</td>
<td>Used to specify the default date format used when Oracle makes conversions from internal date format into string values and vice versa.</td>
</tr>
<tr>
<td>DateLanguage</td>
<td>Used to specify the default language used when Oracle parses internal date format into string values and vice versa.</td>
</tr>
<tr>
<td>Direct</td>
<td>Used for ODAC to connect directly over TCP/IP (in Direct mode) and without requiring Oracle software on</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EnableBCD</td>
<td>Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td>EnableFMTBCD</td>
<td>Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td>EnableIntegers</td>
<td>Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.</td>
</tr>
<tr>
<td>EnableLargeint</td>
<td>Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.</td>
</tr>
<tr>
<td>EnableNumbers</td>
<td>Used for ODAC to map Oracle numbers with precision larger than 15 to TOraNumberField.</td>
</tr>
<tr>
<td>EnableOraTimestamp</td>
<td>Used to create TOraTimeStampField for columns of TIMESTAMP data type.</td>
</tr>
<tr>
<td>IPVersion</td>
<td>Used to specify Internet Protocol Version.</td>
</tr>
<tr>
<td>OptimizerMode</td>
<td>Used to get or set the default optimizer mode for connection.</td>
</tr>
<tr>
<td>StatementCache</td>
<td>Used for ODAC to cache statement handles.</td>
</tr>
<tr>
<td>StatementCacheSize</td>
<td>Used to specify the statement handle cache size.</td>
</tr>
<tr>
<td>SubscriptionPort</td>
<td>Sets the client port used to receive notifications.</td>
</tr>
<tr>
<td>UnicodeEnvironment</td>
<td>Enables or disables using OCI Unicode Environment.</td>
</tr>
<tr>
<td>UseOCI7</td>
<td>Used to force TOraSession use OCI 7 call style only.</td>
</tr>
<tr>
<td>UseUnicode</td>
<td>Used to enable or disable Unicode support.</td>
</tr>
</tbody>
</table>

See Also
5.15.1.23.2.1 CharLength Property

Used to specify the size of a single character in bytes.

Class

TOraSessionOptions

Syntax

```delphi
property CharLength: TCharLength default 0;
```

Remarks

Use the CharLength property to specify the size of a single character in bytes. Set this option with the number in range [0..6] to reflect Oracle support for the national languages. Setting CharLength to zero will instruct TOraSession to interrogate Oracle server for the actual character length. The default value is 1.

5.15.1.23.2.2 Charset Property

Used to set the character set that ODAC uses to read and write character data.

Class

TOraSessionOptions

Syntax

```delphi
property Charset: string;
```

Remarks

Use the Charset property to set the character set that ODAC uses to read and write character data. Supported with Oracle 8 client or later.
5.15.1.23.2.3 ClientIdentifier Property

Used to determine the client identifier in the session.

Class

TOraSessionOptions

Syntax

```
property ClientIdentifier: string;
```

Remarks

Use the ClientIdentifier property to determine the client identifier in the session. The client identifier can be set in the session handle at any time during the session. Then, on the next request to the server, the information is propagated and stored in the server session. The first character of the ClientIdentifier should not be ':', because an exception will be raised. This property has no effect if you use the version server lower than Oracle 9.

© 1997-2021
Devart. All Rights Reserved.

5.15.1.23.2.4 ConnectionTimeout Property

Used to specify the time to wait for a connection to open before raising an exception.

Class

TOraSessionOptions

Syntax

```
property ConnectionTimeout: integer default DefValOraConnectionTimeout;
```

Remarks

Use the ConnectionTimeout property to specify the time to wait for a connection to open before raising an exception. Works only when Direct mode is set to True.
5.15.1.23.2.5 ConvertEOL Property

Affects the line break behavior in string fields and parameters.

Class

TOraSessionOptions

Syntax

```plaintext
property ConvertEOL: boolean default False;
```

Remarks

Affects the line break behavior in string fields and parameters. When fetching strings (including CLOBs and LONGs) with ConvertEOL = True dataset converts their line breaks from LF to CRLF form. And when posting strings to server with ConvertEOL turned on their line breaks converted from CRLF to LF form. By default, strings are not converted.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.23.2.6 DateFormat Property

Used to specify the default date format used when Oracle makes conversions from internal date format into string values and vice versa.

Class

TOraSessionOptions

Syntax

```plaintext
property DateFormat: string stored FIsDateFormat Stored;
```

Remarks

Use the DateFormat property to specify the default date format used when Oracle makes conversions from internal date format into string values and vice versa. An example of valid expression for this property could be "MM/DD/YYYY".

© 1997-2021 Devart. All Rights Reserved.
5.15.1.23.2.7 DateLanguage Property

Used to specify the default language used when Oracle parses internal date format into string values and vice versa.

Class

TOraSessionOptions

Syntax

```
property DateLanguage: string stored FIsDateLanguageStored;
```

Remarks

Use the DateLanguage property to specify the default language used when Oracle parses internal date format into string values and vice versa. Examples of valid expressions for this property could be "French", "German" etc.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.23.2.8 Direct Property

Used for ODAC to connect directly over TCP/IP (in Direct mode) and without requiring Oracle software on the client side.

Class

TOraSessionOptions

Syntax

```
property Direct: boolean default DefValDirect;
```

Remarks

If the Direct property is set to True, ODAC connects directly over TCP/IP (in Direct mode) and does not require Oracle software on the client side. Otherwise, ODAC connects in Client mode. Supported by ODAC Professional and Professional with source code editions.

© 1997-2021 Devart. All Rights Reserved.
5.15.1.23.2.9  EnableBCD Property

Used to enable currency type. Default value of this option is False.

Class

TOraSessionOptions

Syntax

property EnableBCD: boolean;

5.15.1.23.2.10  EnableFMTBCD Property

Used to enable using FMTBCD instead of float for large integer numbers to keep precision.

Class

TOraSessionOptions

Syntax

property EnableFMTBCD: boolean;

Remarks

Use the EnableFMTBCD property to enable using FMTBCD instead of float for large integer numbers to keep precision.

5.15.1.23.2.11  EnableIntegers Property

Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.

Class

TOraSessionOptions

Syntax

property EnableIntegers: boolean default True;
Remarks

When the EnableIntegers property is set to True ODAC maps Oracle numbers with precision less than 10 to TIntegerField. If EnableIntegers is set to False numbers are mapped to TFloatField or TораNumberField.

5.15.1.23.2.12 EnableLargeint Property

Used for ODAC to map Oracle numbers with precision less than 10 to TIntegerField.

Class

TOraSessionOptions

Syntax

```
property EnableLargeint: boolean default False;
```

Remarks

When the EnableIntegers property is set to True, ODAC maps Oracle numbers with precision less than 10 to TIntegerField. If EnableIntegers is set to False, numbers are mapped to TFloatField or TораNumberField.

5.15.1.23.2.13 EnableNumbers Property

Used for ODAC to map Oracle numbers with precision larger than 15 to TораNumberField.

Class

TOraSessionOptions

Syntax

```
property EnableNumbers: boolean default False;
```

Remarks

When the EnableNumbers property is set to True ODAC maps Oracle numbers with
5.15.1.23.2.14 EnableOraTimestamp Property

Used to create TOraTimeStampField for columns of TIMESTAMP data type.

Class

TOraSessionOptions

Syntax

```
property EnableOraTimestamp: boolean default True;
```

Remarks

When the EnableOraTimestamp property is set to True, TOraTimeStampField is created for columns of TIMESTAMP data type.

When False, standard TSQLTimeStampField is created for columns of TIMESTAMP data type.

5.15.1.23.2.15 IPVersion Property

Used to specify Internet Protocol Version.

Class

TOraSessionOptions

Syntax

```
property IPVersion: TIPVersion default DefValIPVersion;
```

Remarks

Use the IPVersion property to specify Internet Protocol Version.

Supported values:
- ivIPBoth - specifies that either Internet Protocol Version 6 (IPv6) or Version 4 (IPv4) will be
used;
- ivIPv4 (default) - specifies that Internet Protocol Version 4 (IPv4) will be used;
- ivIPv6 - specifies that Internet Protocol Version 6 (IPv6) will be used.

Note: When the TIPVersion property is set to ivIPBoth, a connection attempt will be made via IPv6 if it is enabled on the operating system. If the connection attempt fails, a new connection attempt will be made via IPv4.

5.15.1.23.2.16 OptimizerMode Property

Used to get or set the default optimizer mode for connection.

Class
TOraSessionOptions

Syntax

```property OptimizerMode: TOptimizerMode default omDefault;```

Remarks

Use the OptimizerMode property to get or set the default optimizer mode for connection.

5.15.1.23.2.17 StatementCache Property

Used for ODAC to cache statement handles.

Class
TOraSessionOptions

Syntax

```property StatementCache: boolean default False;```

Remarks

When the StatementCache property is set to True, ODAC caches statement handles.
5.15.1.23.2.18  StatementCacheSize Property

Used to specify the statement handle cache size.

Class

TOraSessionOptions

Syntax

```property`` statementCacheSize: integer `default 20``

Remarks

Use the StatementCacheSize property to specify the statement handle cache size.

**Note:** If StatementCache property is set, you can use TOraDataSet.Options and TOraSQL.StatementCache to adjust performance of dataset and SQL components using this session.

5.15.1.23.2.19  SubscriptionPort Property

Sets the client port used to receive notifications.

Class

TOraSessionOptions

Syntax

```property`` SubscriptionPort: Integer `default 0``

Remarks

Use the SubscriptionPort property to set the port number for receiving notifications (for example, for clients on a computer behind a firewall). This option applies only to OCI clients when there is an exact correspondence between the client and the server versions.
5.15.1.23.2.20 UnicodeEnvironment Property

Enables or disables using OCI Unicode Environment.

**Class**

**TOraSessionOptions**

**Syntax**

```plaintext
property UnicodeEnvironment: boolean default False;
```

**Remarks**

When this option is enabled, Unicode characters can be used in SQL statements. Disable this option if you have encountered some problems with Unicode Environment.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.23.2.21 UseOCI7 Property

Used to force TOraSession use OCI 7 call style only.

**Class**

**TOraSessionOptions**

**Syntax**

```plaintext
property UseOCI7: boolean default False;
```

**Remarks**

Use the UseOCI7 property to force TOraSession use OCI 7 call style only.

© 1997-2021 Devart. All Rights Reserved.

5.15.1.23.2.22 UseUnicode Property

Used to enable or disable Unicode support.

**Class**
**TOraSessionOptions**

**Syntax**

```plaintext
property UseUnicode: boolean default DefValUseUnicode;
```

**Remarks**

Use the UseUnicode property to enable or disable Unicode support. Affects character data fetched from the server. When set to True all character data stored as WideStrings and TStringField is replaced with TWideStringFiled. Supported starting with Oracle 8.

---

**5.15.1.24 TOraSQL Class**

A component for executing SQL statements and calling stored procedures on the database server.

For a list of all members of this type, see TOraSQL members.

**Unit**

`Ora`

**Syntax**

```plaintext
TOraSQL = class(TCustomDASQL);
```

**Remarks**

The TOraSQL component is a direct descendant of the TCustomDASQL class.

Use The TOraSQL component when a client application must execute SQL statement or the PL/SQL block, and call stored procedure on the database server. The SQL statement should not retrieve rows from the database.

**Inheritance Hierarchy**

- TCustomDASQL
- TOraSQL

**See Also**

- TOraQuery
**TOraScript**

© 1997-2021 Devart. All Rights Reserved.

5.15.1.24.1 Members

**TOraSQL** class overview.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ArrayLength</strong></td>
<td>Used to set the Length property to all parameters.</td>
</tr>
<tr>
<td><strong>ChangeCursor</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Enables or disables changing screen cursor when executing commands in the NonBlocking mode.</td>
</tr>
<tr>
<td><strong>CommandTimeout</strong></td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Used to return a SQL statement with expanded macros.</td>
</tr>
<tr>
<td><strong>MacroCount</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td><strong>Macros</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td><strong>NonBlocking</strong></td>
<td>Used to execute a SQL statement by a separate thread.</td>
</tr>
</tbody>
</table>
| **ParamCheck**    | (inherited from **TCustomDASQL**) Used to specify whether parameters for the Params property are implicitly generated when the SQL
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ParamCount (inherited from TCustomDASQL)</td>
<td>Indicates the number of parameters in the Params property.</td>
</tr>
<tr>
<td>Params</td>
<td>Contains the parameters for a SQL statement.</td>
</tr>
<tr>
<td>ParamValues (inherited from TCustomDASQL)</td>
<td>Used to get or set the values of individual field parameters that are identified by name.</td>
</tr>
<tr>
<td>Prepared (inherited from TCustomDASQL)</td>
<td>Used to indicate whether a query is prepared for execution.</td>
</tr>
<tr>
<td>RowsAffected (inherited from TCustomDASQL)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>Used to return the number of rows processed by a SQL statement.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to define the session to execute SQL in.</td>
</tr>
<tr>
<td>SQL (inherited from TCustomDASQL)</td>
<td>Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.</td>
</tr>
<tr>
<td>SQLType</td>
<td>Used to get the typecode of the SQL statement being processed by an Oracle database server.</td>
</tr>
<tr>
<td>StatementCache</td>
<td>Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.</td>
</tr>
<tr>
<td>TemporaryLobUpdate</td>
<td>Specifies whether to use temporary LOBs for writing input and input/output LOB parameters into database when executing SQL statements.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BreakExec</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Breaks execution of an SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong></td>
<td>Assigns a PL/SQL block that calls stored procedure</td>
</tr>
<tr>
<td><strong>ErrorOffset</strong></td>
<td>Obtains zero-based starting byte position of a parse error detected by an Oracle server while processing SQL statement.</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Checks whether TCustomDASQL still executes a SQL statement.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>Searches for and returns a parameter with the specified name.</td>
</tr>
<tr>
<td><strong>MacroByName</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>ParamByName</strong></td>
<td>Searches for and returns a parameter with the specified name.</td>
</tr>
<tr>
<td><strong>Prepare</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td><strong>UnPrepare</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><strong>WaitExecuting</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Waits until TCustomDASQL executes a SQL statement.</td>
</tr>
</tbody>
</table>

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AfterExecute</strong></td>
<td>(inherited from <strong>TCustomDASQL</strong>) Occurs after a SQL statement has been executed.</td>
</tr>
</tbody>
</table>
Properties of the `TOracleSQL` class.

For a complete list of the `TOracleSQL` class members, see the [TOracleSQL Members](#) topic.

## Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ArrayLength</strong></td>
<td>Used to set the Length property to all parameters.</td>
</tr>
<tr>
<td><strong>ChangeCursor</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Enables or disables changing screen cursor when executing commands in the NonBlocking mode.</td>
</tr>
<tr>
<td><strong>Connection</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Debug</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Used to return a SQL statement with expanded macros.</td>
</tr>
<tr>
<td><strong>MacroCount</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td><strong>Macros</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td><strong>ParamCheck</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Used to specify whether parameters for the Params property are implicitly generated when the SQL property is being changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Indicates the number of parameters in the Params property.</td>
</tr>
<tr>
<td><strong>ParamValues</strong> (inherited from <code>TCustomDASQL</code>)</td>
<td>Used to get or set the values of individual field parameters that are</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TCustomDASQL</strong>)</td>
<td>Used to indicate whether a query is prepared for execution.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong> (inherited from <strong>TCustomDASQL</strong>)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>RowsProcessed</strong></td>
<td>Used to return the number of rows processed by a SQL statement.</td>
</tr>
<tr>
<td><strong>SQL</strong> (inherited from <strong>TCustomDASQL</strong>)</td>
<td>Used to provide a SQL statement that a TCustomDASQL component executes when the Execute method is called.</td>
</tr>
<tr>
<td><strong>SQLType</strong></td>
<td>Used to get the typecode of the SQL statement being processed by an Oracle database server.</td>
</tr>
</tbody>
</table>

**Published**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CommandTimeout</strong></td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>NonBlocking</strong></td>
<td>Used to execute a SQL statement by a separate thread.</td>
</tr>
<tr>
<td><strong>Params</strong></td>
<td>Contains the parameters for a SQL statement.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>Used to define the session to execute SQL in.</td>
</tr>
<tr>
<td><strong>StatementCache</strong></td>
<td>Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.</td>
</tr>
<tr>
<td><strong>TemporaryLobUpdate</strong></td>
<td>Specifies whether to use temporary LOBs for writing</td>
</tr>
</tbody>
</table>
See Also
- **TOraSQL Class**
- **TOraSQL Class Members**

### 5.15.1.24.2.1 ArrayLength Property

Used to set the Length property to all parameters.

**Class**

TOraSQL

**Syntax**

```plaintext
property ArrayLength: integer;
```

**Remarks**

Use the ArrayLength property to set the Length property to all parameters. It is useful for DML array operations.

**See Also**
- **TOraParam.Length**

### 5.15.1.24.2.2 CommandTimeout Property

Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.

**Class**

TOraSQL
Syntax

```
property CommandTimeout: integer default 0;
```

Remarks
The wait time is specified in seconds to wait for the command to execute. The default value is 0. The value of 0 indicates there are no time limits (an attempt to execute a command will wait indefinitely).

---

5.15.1.24.2.3 NonBlocking Property

Used to execute a SQL statement by a separate thread.

Class

```
T0raSQL
```

Syntax

```
property NonBlocking: boolean;
```

Remarks
Set the NonBlocking property to True to execute a SQL statement by a separate thread.

---

5.15.1.24.2.4 Params Property

Contains the parameters for a SQL statement.

Class

```
T0raSQL
```

Syntax

```
property Params: T0raParams stored False;
```

Remarks
Contains the parameters for a SQL statement.

Access Params at runtime to view and set parameter names, values, and data types dynamically (at design time use the Parameters editor to set parameter information). Params is a zero-based array of parameter records. Index specifies the array element to access.

An easier way to set and retrieve parameter values when the name of each parameter is known is to call ParamByName.

Example
Setting parameters in runtime.

```pascal
procedure TForm1.Button1Click(Sender: TObject);
begin
  with OraSQL do
  begin
    SQL.Clear;
    SQL.Add('INSERT INTO Temp_Table(Id, Name)');
    SQL.Add('VALUES (:id, :Name)');
    ParamByName('Id').AsInteger := 55;
    Params[1].AsString := 'Green';
    Execute;
  end;
end;
```

See Also
- TOraParam
- FindParam

5.15.1.24.2.5  RowsProcessed Property

Used to return the number of rows processed by a SQL statement.

Class
- T0raSQL

Syntax

```pascal
property RowsProcessed: integer;
```

Remarks
Use the RowsProcessed property to return the number of rows processed by a SQL statement. Useful for inserting, updating and deleting statements.

**5.15.1.24.2.6 Session Property**

Used to define the session to execute SQL in.

**Class**

**TOraSQL**

**Syntax**

```
property Session: TOraSession;
```

**Remarks**

Use the Session property to specify the session in which SQL will be executed. If Session is not connected, the Execute method calls Session.Connect.

**See Also**

- **TOraSession**

**5.15.1.24.2.7 SQLType Property**

Used to get the typecode of the SQL statement being processed by an Oracle database server.

**Class**

**TOraSQL**

**Syntax**

```
property SQLType: integer;
```

**Remarks**

Use the SQLType property to get the typecode of the SQL statement being processed by an
5.15.1.24.2.8 StatementCache Property

Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.

Class
TOraSQL

Syntax

```
property StatementCache: boolean default False;
```

Remarks

OCI statement cache is enabled when you set TOraSessionOptions.StatementCacheSize in TOraSession.Options to a positive value. Set TOraSessionOptions.StatementCacheSize to 0 (default) or TOraSession.Options.StatementCache to false if you don't want the statements to be cached.

5.15.1.24.2.9 TemporaryLobUpdate Property

Specifies whether to use temporary LOBs for writing input and input/output LOB parameters into database when executing SQL statements.

Class
TOraSQL

Syntax

```
property TemporaryLobUpdate: boolean default False;
```

Remarks

If the TemporaryLobUpdate property is True, temporary LOBs are used to write input and input/output LOB parameters into database when executing SQL statements.
5.15.1.24.3 Methods

Methods of the TOraSQL class.

For a complete list of the TOraSQL class members, see the TOraSQL Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BreakExec</td>
<td>(inherited from TCustomDASQL) Breaks execution of an SQL statement on the server.</td>
</tr>
<tr>
<td>CreateProcCall</td>
<td>Assigns a PL/SQL block that calls stored procedure</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td>Obtains zero-based starting byte position of a parse error detected by an Oracle server while processing SQL statement.</td>
</tr>
<tr>
<td>Execute</td>
<td>(inherited from TCustomDASQL) Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td>Executing</td>
<td>(inherited from TCustomDASQL) Checks whether TCustomDASQL still executes a SQL statement.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>(inherited from TCustomDASQL) Finds a macro with the specified name.</td>
</tr>
<tr>
<td>FindParam</td>
<td>Searches for and returns a parameter with the specified name.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>(inherited from TCustomDASQL) Finds a macro with the specified name.</td>
</tr>
<tr>
<td>ParamByName</td>
<td>Searches for and returns a parameter with the specified name.</td>
</tr>
<tr>
<td>Prepare</td>
<td>(inherited from TCustomDASQL) Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>(inherited from TCustomDASQL) Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>WaitExecuting</td>
<td>(inherited from TCustomDASQL) Waits until TCustomDASQL</td>
</tr>
</tbody>
</table>
5.15.1.24.3.1  CreateProcCall Method

Assigns a PL/SQL block that calls stored procedure

Class

TOraSQL

Syntax

```plaintext
procedure CreateProcCall(Name: string; Overload: integer = 0);
```

Parameters

- **Name**
  - Holds the stored procedure name.
- **Overload**
  - Holds the number of overloaded procedure.

Remarks

Call the CreateProcCall method to assign a PL/SQL block that calls stored procedure specified by Name to the SQL property. The Overload parameter must contain the number of overloaded procedure. Retrieves the information about parameters of the procedure from Oracle. After calling CreateProcCall you can execute a stored procedure by Execute method.

See Also

- TCustomDASQL.Execute
- TCustomDACConnection.ExecProc
- TOraStoredProc
5.15.1.24.3.2 ErrorOffset Method

Obtains zero-based starting byte position of a parse error detected by an Oracle server while processing SQL statement.

Class

TOraSQL

Syntax

```
function ErrorOffset: integer;
```

Return Value

Holds the position of a parse error detected by an Oracle server while processing SQL statement.

Remarks

Call the ErrorOffset method to obtain zero-based starting byte position of a parse error detected by an Oracle server while processing SQL statement.

Note that statements which are longer than 64KB will have unpredictable effect on the ErrorOffset value. Refer to Programmer's Guide to the Oracle Call Interface for further information.

5.15.1.24.3.3 FindParam Method

Searches for and returns a parameter with the specified name.

Class

TOraSQL

Syntax

```
function FindParam(const Value: string): TOraParam;
```

Parameters

Value

Holds the stored procedure name.

Return Value

the parameter, if a match was found. Nil otherwise.

© 1997-2021 Devart. All Rights Reserved.
Remarks

Call the FindParam method to find a parameter with the name passed in the Name argument. If a match was found, FindParam returns the parameter. Otherwise, it returns nil.

See Also

- TOraParam
- ParamByName

5.15.1.24.3.4  ParamByName Method

Searches for and returns a parameter with the specified name.

Class

TOraSQL

Syntax

function ParamByName(const Value: string): TOraParam;

Parameters

Value

Holds the parameter name.

Return Value

The parameter, if a match was found. Otherwise an exception is raised.

Remarks

Call the ParamByName method to find a parameter with the name passed in the Name argument.

If a match is found, ParamByName returns the parameter. Otherwise, an exception is raised.

Example

OraSQL1.Execute;
Edit1.Text := OraSQL1.ParamsByName('Contact').AsString;

See Also

- TOraParam
5.15.1.25 TOraStoredProc Class

A component for accessing and executing stored procedures and functions.

For a list of all members of this type, see TOraStoredProc members.

Unit
ora

Syntax

TOraStoredProc = class(TCustomOraQuery);

Remarks

Use TOraStoredProc to access stored procedures on the database server.

You need only to define the StoredProcName property, and the SQL statement to call the stored procedure will be generated automatically.

Use the Execute method at runtime to generate request that instructs server to execute procedure and PrepareSQL to describe parameters at run time.

Inheritance Hierarchy

TMemDataSet
  TCustomDADataset
    TOraDataSet
      TCustomOraQuery
        TOraStoredProc

See Also

- TOraQuery
- TOraSQL
- Updating Data with ODAC Dataset Components
- TCustomDACConnection.ExecProc
## TOraStoredProc class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BaseSQL</strong></td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td><strong>CachedUpdates</strong></td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>ChangeNotification</strong></td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td><strong>CheckMode</strong></td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td><strong>CommandTimeout</strong></td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>Used to add WHERE conditions to a query.</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>Used to specify the fields that correspond to the foreign key fields from</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>FetchAll</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>IsPLSQL</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td><strong>IsQuery</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>KeyFields</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>KeySequence</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>LocalConstraints</td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td>LocalUpdate</td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td>LockMode</td>
<td></td>
</tr>
<tr>
<td>MacroCount</td>
<td><strong>TCustomDADataset</strong></td>
</tr>
<tr>
<td>Macros</td>
<td><strong>TCustomDADataset</strong></td>
</tr>
<tr>
<td>MasterFields</td>
<td><strong>TCustomDADataset</strong></td>
</tr>
<tr>
<td>MasterSource</td>
<td><strong>TCustomDADataset</strong></td>
</tr>
<tr>
<td>NonBlocking</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>Options</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>OptionsDS</td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td>Overload</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td><strong>ParamCheck</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>ParamCount</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>Params</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>Ranged</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>ReadOnly</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>RefreshMode</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>RefreshOptions</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>ReturnParams</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>RowsAffected</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>RowsProcessed</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>SequenceMode</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>SmartFetch</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td>SQLLock</td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td>StoredProcName</td>
<td>Used to specify the name of the stored procedure to call on the server.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
</tbody>
</table>
**UpdateObject** (inherited from **TOraDataSet**)  
Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

**UpdateRecordTypes** (inherited from **TMemDataSet**)  
Used to indicate the update status for the current record when cached updates are enabled.

**UpdatesPending** (inherited from **TMemDataSet**)  
Used to check the status of the cached updates buffer.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddWhere</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td><strong>ApplyRange</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Generates the stored procedure call.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>DeleteWhere</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>ErrorOffset</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the parse error offset.</td>
</tr>
<tr>
<td><strong>ExecProc</strong></td>
<td>Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Execute</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to find out whether TCustomDADataSet has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to learn whether TCustomDADataSet is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to learn whether TCustomDADataSet is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraArray</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetBlob</code></td>
<td>(inherited from <code>TMemDataSet</code>) Overloaded. Retrieves <code>TBlob</code> object for a field or current record when only its name is known.</td>
</tr>
<tr>
<td><code>GetDataType</code></td>
<td>(inherited from <code>TCustomDADataset</code>) Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td><code>GetErrorPos</code></td>
<td>(inherited from <code>TOraDataSet</code>) Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td><code>GetFieldObject</code></td>
<td>(inherited from <code>TCustomDADataset</code>) Returns a multireference shared object from field.</td>
</tr>
<tr>
<td><code>GetFieldPrecision</code></td>
<td>(inherited from <code>TCustomDADataset</code>) Retrieves the precision of a number field.</td>
</tr>
<tr>
<td><code>GetFieldScale</code></td>
<td>(inherited from <code>TCustomDADataset</code>) Retrieves the scale of a number field.</td>
</tr>
<tr>
<td><code>GetFile</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a <code>TOraFile</code> object for a field with known name.</td>
</tr>
<tr>
<td><code>GetInterval</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a <code>TOraInterval</code> object for a field with known name.</td>
</tr>
<tr>
<td><code>GetKeyFieldNames</code></td>
<td>(inherited from <code>TCustomDADataset</code>) Provides a list of available key field names.</td>
</tr>
<tr>
<td><code>GetKeyList</code></td>
<td>(inherited from <code>TOraDataSet</code>) Returns the list of table primary key fields.</td>
</tr>
<tr>
<td><code>GetLob</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a <code>TOraLob</code> object for a field with known name.</td>
</tr>
<tr>
<td><code>GetLobLocator</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a <code>TOraLob</code> object for a field with known name.</td>
</tr>
<tr>
<td><code>GetObject</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a <code>TOraObject</code> object for a field with known name.</td>
</tr>
<tr>
<td><code>GetOrderBy</code></td>
<td>(inherited from <code>TCustomDADataset</code>) Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td><code>GetRef</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a <code>TOraRef</code> object for a field with known name.</td>
</tr>
<tr>
<td><code>GetTable</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieve a <code>TOraNestTable</code> object for a field with known name.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetTimeStam (inherited from TOracleDataSet)</td>
<td>Retrieves a TOracleTimeStam object for a field with known name.</td>
</tr>
<tr>
<td>GotoCurrent (inherited from TCustomDADataSet)</td>
<td>Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td>Locate (inherited from TMemDataSet)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx (inherited from TMemDataSet)</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Lock (inherited from TCustomDADataSet)</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td>MacroByName (inherited from TCustomDADataSet)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>OpenNext (inherited from TOracleDataSet)</td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td>ParamByName (inherited from TOracleDataSet)</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td>Prepare (inherited from TCustomDADataSet)</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>PrepareSQL</td>
<td>Describes the parameters of a stored procedure.</td>
</tr>
<tr>
<td>RefreshRecord (inherited from TCustomDADataSet)</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td>RestoreSQL (inherited from TCustomDADataSet)</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td>RestoreUpdates (inherited from TMemDataSet)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord (inherited from TMemDataSet)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveSQL (inherited from TCustomDADataSet)</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
</tbody>
</table>
### SaveToXML
(inherited from `TMemDataSet`)

Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.

### SetOrderBy
(inherited from `TCustomDADataset`)

Builds an ORDER BY clause of a SELECT statement.

### SetRange
(inherited from `TMemDataSet`)

Sets the starting and ending values of a range, and applies it.

### SetRangeEnd
(inherited from `TMemDataSet`)

Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.

### SetRangeStart
(inherited from `TMemDataSet`)

Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.

### SQLSaved
(inherited from `TCustomDADataset`)

Determines if the SQL property value was saved to the BaseSQL property.

### UnLock
(inherited from `TCustomDADataset`)

 Releases a record lock.

### UnPrepare
(inherited from `TMemDataSet`)

Frees the resources allocated for a previously prepared query on the server and client sides.

### UpdateResult
(inherited from `TMemDataSet`)

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

### UpdateStatus
(inherited from `TMemDataSet`)

Indicates the current update status for the dataset when cached updates are enabled.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AfterExecute</strong></td>
<td>(inherited from <code>TCustomDADataset</code>) Occurs after a component has executed a query to database.</td>
</tr>
</tbody>
</table>
### Properties of the `TOraStoredProc` Class

For a complete list of the `TOraStoredProc` class members, see the [TOraStoredProc Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BaseSQL</code> (inherited from <code>TCustomDADataset</code>)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td><code>CachedUpdates</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><code>ChangeNotification</code> (inherited from <code>TOraDataSet</code>)</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CheckMode</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td><strong>CommandTimeout</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to set data type mapping rules.</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>FetchAll</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>FilterSQL</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>FinalSQL</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>IsPLSQL</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>IsQuery</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>KeyFields</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>KeySequence</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>MacroCount</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Macros</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>MasterFields</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>MasterSource</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Options</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>OptionsDS</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>ParamCount</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Params</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Prepared</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Ranged</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>RefreshMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>RowsAffected</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>RowsProcessed</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td><strong>SequenceMode</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td><strong>Session</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td><strong>SmartFetch</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td><strong>SQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td><strong>SQLDelete</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td><strong>SQLInsert</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td><strong>SQLLock</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td><strong>SQLRecCount</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td><strong>SQLRefresh</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the <strong>TCustomDADataset.Refresh</strong>Record procedure.</td>
</tr>
<tr>
<td><strong>SQLType</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to get the typecode of</td>
</tr>
</tbody>
</table>
the SQL statement being processed by Oracle database server.

**SQLUpdate** (inherited from **TCustomDADataset**)  
Used to specify a SQL statement that will be used when applying an update to a dataset.

**StrictUpdate** (inherited from **TOraDataSet**)  
Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.

**UniDirectional** (inherited from **TCustomDADataset**)  
Used if an application does not need bidirectional access to records in the result set.

**UpdateObject** (inherited from **TOraDataSet**)  
Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.

**UpdateRecordTypes** (inherited from **TMemDataSet**)  
Used to indicate the update status for the current record when cached updates are enabled.

**UpdatesPending** (inherited from **TMemDataSet**)  
Used to check the status of the cached updates buffer.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LockMode</strong></td>
<td>Used to specify what kind of lock will be performed when editing a record.</td>
</tr>
<tr>
<td><strong>Overload</strong></td>
<td>Used to specify the overloading number.</td>
</tr>
<tr>
<td><strong>StoredProcName</strong></td>
<td>Used to specify the name of the stored procedure to call on the server.</td>
</tr>
</tbody>
</table>

### See Also
- **TOraStoredProc Class**
- **TOraStoredProc Class Members**

© 2021 Devart
5.15.1.25.2.1 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

Class

T0raStoredProc

Syntax

| property  | LockMode: TLockMode default lmNone; |

Remarks

Use the LockMode property to define what kind of lock will be performed when editing a record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time.

Locking is performed by the RefreshRecord method.

The default value is lmNone.

See Also

- T0raQuery.LockMode
- T0raTable.LockMode

5.15.1.25.2.2 Overload Property

Used to specify the overloading number.

Class

T0raStoredProc

Syntax

| property  | Overload: integer default 0; |

Remarks
Use the Overload property to specify the overloading number in case the procedure or function is a part of a package and is overloaded.

5.15.1.25.3 StoredProcName Property

Used to specify the name of the stored procedure to call on the server.

Class

TOraStoredProc

Syntax

```property StoredProcName: string;```

Remarks

Use the StoredProcName property to specify the name of the stored procedure to call on the server. If StoredProcName does not match the name of an existing stored procedure on the server, then when the application attempts to prepare the procedure prior to execution, an exception is raised.

5.15.1.25.3 Methods

Methods of the TOraStoredProc class.

For a complete list of the TOraStoredProc class members, see the TOraStoredProc Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere</td>
<td>(inherited from TCustomDADataset) Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange</td>
<td>(inherited from TMemDataSet) Applies a range to the dataset.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>DeleteWhere</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>ErrorOffset</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the parse error offset.</td>
</tr>
<tr>
<td><strong>ExecProc</strong></td>
<td>Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Execute</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to find out whether</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TCustomDADataset has fetched all rows.</td>
<td></td>
</tr>
<tr>
<td><strong>Fetching</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to learn whether TCustomDADataset is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to learn whether TCustomDADataset is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraArray object for a field when only its name is known.</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetDataType</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td><strong>GetErrorPos</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Returns a multireference shared object from field.</td>
</tr>
<tr>
<td><strong>GetFieldPrecision</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td><strong>GetFieldScale</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td><strong>GetFile</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>GetInterval</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>GetKeyList</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetLob</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetLobLocator</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetObject</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>GetRef</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetTable</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Locate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LocateEx</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Lock</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>MacroByName</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>OpenNext</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited from</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td><strong>ParamByName</strong></td>
<td><strong>TOraDataSet</strong></td>
</tr>
<tr>
<td><strong>Prepare</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>PrepareSQL</strong></td>
<td></td>
</tr>
<tr>
<td><strong>RefreshRecord</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>RestoreSQL</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>RevertRecord</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>SaveSQL</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>SaveToXML</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>SetOrderBy</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td><strong>SetRange</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>SetRangeEnd</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>SetRangeStart</strong></td>
<td><strong>TMemDataSet</strong></td>
</tr>
<tr>
<td><strong>SQLSaved</strong></td>
<td><strong>TCustomDADataSet</strong></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>UnLock</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Releases a record lock.</td>
</tr>
<tr>
<td><strong>UnPrepare</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><strong>UpdateResult</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdateStatus</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

**See Also**
- **TOraStoredProc Class**
- **TOraStoredProc Class Members**

**5.15.1.25.3.1 ExecProc Method**

Executes a SQL statement on the server.

**Class**

**TOraStoredProc**

**Syntax**

```plaintext
procedure ExecProc;
```

**Remarks**

The ExecProc method is the same as **TCustomDADataSet.Execute** method. It is included for compatibility with TStoredProc.

**See Also**

- **TCustomDADataSet.Execute**
5.15.1.25.3.2 PrepareSQL Method

Describes the parameters of a stored procedure.

Class
TOraStoredProc

Syntax

```
procedure PrepareSQL;
```

Remarks

Use the PrepareSQL method to describe parameters of a stored procedure. If it is necessary, Execute method calls it automatically. You can define parameters at design time if ParametersEditor is opened.

5.15.1.26 TOraTimeStampField Class

A class providing access to the Oracle timestamp fields.

For a list of all members of this type, see TOraTimeStampField members.

Unit
Ora

Syntax

```
TOraTimeStampField = class(TField);
```

Remarks

TOraTimeStampField provides access to Oracle timestamp fields. The TOraTimeStampField class supports three data types: TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE. According to this, the TOraTimeStampField.DataType property has three valid values ftTimeStamp, ftTimeStampTZ and ftTimeStampLTZ.

You can access actual timestamp value using the AsDateTime,AsString and...
AsOraTimeStamp properties.

See Also

- TOraTimeStamp

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTimeStamp</td>
<td>Used to provide access to a TOraTimeStamp object.</td>
</tr>
<tr>
<td>Format</td>
<td>Used to get or set the date-time format model for operations with the AsString property.</td>
</tr>
</tbody>
</table>

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTimeStamp</td>
<td>Used to provide access to a TOraTimeStamp object.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>Used to get or set the date-time format model for</td>
</tr>
</tbody>
</table>
5.15.1.26.2.1  AsTimeStamp Property

Used to provide access to a TOraTimeStamp object.

Class
TOraTimeStampField

Syntax

| property AsTimeStamp: TOraTimeStamp; |

Remarks

Use the AsTimeStamp property to get access to a TOraTimeStamp object which you can use for manipulations with timestamp value.

See Also
- TOraTimeStamp

5.15.1.26.2.2  Format Property

Used to get or set the date-time format model for operations with the AsString property.

Class
TOraTimeStampField

Syntax

| property Format: string; |
Remarks
Use the Format property to get or set the date-time format model for operations with the AsString property. Format string should be an Oracle date-time string.

See Also
- TOraTimeStamFormat

5.15.1.27 TOraTrace Class

A component allowing starting and stopping a SQL trace for a specified session. This component provides access to the DBMS_TRACE package.

For a list of all members of this type, see TOraTrace members.

Unit
ora

Syntax

TOraTrace = class(TComponent);

Remarks
Use the TOraTrace component to start and stop a SQL trace for a session. The component also provides access to functionality of DBMS_TRACE package to control the PL/SQL trace.

SQL trace can be useful in performance diagnostics. It can help to determine in detail how applications/users access the database.

The TOraTrace component automatically starts the trace when its TOraSession component connects to a database or when you assign already connected session to the TOraTrace.Session property. The SQL trace is started if TOraTrace.SqlTraceMode <> []. The PL/SQL trace is started if TOraTrace.PISqlTraceMode <> [].
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>Used to enable the trace.</td>
</tr>
<tr>
<td><strong>MaxTraceFileSize</strong></td>
<td>Used to limit the size of the SQL trace dump file.</td>
</tr>
<tr>
<td><strong>PLSqlTraceMode</strong></td>
<td>Defines the level of PL/SQL trace.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>used to specify the session on which a trace will be enabled.</td>
</tr>
<tr>
<td><strong>SqlTraceMode</strong></td>
<td>Used to determine the level of SQL trace.</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>Used to define whether SQL trace and PL/SQL trace are active.</td>
</tr>
<tr>
<td><strong>TraceFileIdentifier</strong></td>
<td>Used to add a string to the SQL trace dump file name.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetSessionPID</strong></td>
<td>Returns the operating system process identifier for a process that owns the current session.</td>
</tr>
<tr>
<td><strong>GetTraceFileName</strong></td>
<td>Returns the name of a dump file on the database server to which SQL trace data is written.</td>
</tr>
<tr>
<td><strong>PLSqlTraceComment</strong></td>
<td>Provides a comment on the PL/SQL trace.</td>
</tr>
<tr>
<td><strong>PLSqlTraceLimit</strong></td>
<td>Limits the amount of storage used in the database for the PL/SQL trace data.</td>
</tr>
<tr>
<td><strong>PLSqlTracePause</strong></td>
<td>makes a pause in PL/SQL tracing.</td>
</tr>
<tr>
<td><strong>PLSqlTraceResume</strong></td>
<td>Resumes PL/SQL trace.</td>
</tr>
<tr>
<td><strong>PLSqlTraceRunNumber</strong></td>
<td>Returns a run number for the current PL/SQL trace.</td>
</tr>
<tr>
<td><strong>PLSqlTraceStart</strong></td>
<td>Starts PL/SQL trace data collection.</td>
</tr>
</tbody>
</table>
5.15.1.27.2 Properties

Properties of the TOraTrace class.

For a complete list of the TOraTrace class members, see the TOraTrace Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>Used to define whether SQL trace and PL/SQL trace are active.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>Used to enable the trace.</td>
</tr>
<tr>
<td>MaxTraceFileSize</td>
<td>Used to limit the size of the SQL trace dump file.</td>
</tr>
<tr>
<td>PLSqlTraceMode</td>
<td>Defines the level of PL/SQL trace.</td>
</tr>
<tr>
<td>Session</td>
<td>used to specify the session on which a trace will be enabled.</td>
</tr>
<tr>
<td>SqlTraceMode</td>
<td>Used to determine the level of SQL trace.</td>
</tr>
<tr>
<td>TraceFileIdentifier</td>
<td>Used to add a string to the SQL trace dump file name.</td>
</tr>
</tbody>
</table>

See Also

- TOraTrace Class
- TOraTrace Class Members

© 1997-2021

Request Support  DAC Forum  Provide Feedback
5.15.1.27.2.1 Enabled Property

Used to enable the trace.

Class
T0raTrace

Syntax

```
property Enabled: boolean default True;
```

Remarks

Use the Enabled property to enable or disable the trace. Set Enabled to False to disable the trace.

See Also

- SqlTraceStart
- SqlTraceStop

© 1997-2021
Devart. All Rights Reserved.

5.15.1.27.2.2 MaxTraceFileSize Property

Used to limit the size of the SQL trace dump file.

Class
T0raTrace

Syntax

```
property MaxTraceFileSize: integer default DEFAULT_TRACE_FILE_SIZE;
```

Remarks

Use the MaxTraceFileSize property to limit the size of the SQL trace dump file.
5.15.1.27.2.3 PlSqlTraceMode Property

Defines the level of PL/SQL trace.

Class

TOraTrace

Syntax

```delphi
property PlSqlTraceMode: TPlSqlTraceMode default [];```

Remarks

Use the PlSqlTraceMode property to define the level of PL/SQL trace.

Note: PL/SQL program unit is enabled when it is compiled with debug information.

5.15.1.27.2.4 Session Property

used to specify the session on which a trace will be enabled.

Class

TOraTrace

Syntax

```delphi
property Session: TOraSession;```

Remarks

Use the Session property to specify the session on which a trace will be enabled.

5.15.1.27.2.5 SqlTraceMode Property

Used to determine the level of SQL trace.

Class
**T0raTrace**

**Syntax**

```plaintext
property SqlTraceMode: TSqlTraceMode default [smTypicalStatistics, smTimedStatistics];
```

**Remarks**

Use the SqlTraceMode property to define the level of SQL trace.

---

**Class**

**T0raTrace**

**Syntax**

```plaintext
property State: TTraceState;
```

**Remarks**

Use the State property to detect whether SQL trace and PL/SQL trace are active.

---

**TraceFileIdentifier Property**

Used to add a string to the SQL trace dump file name.

**Class**

**T0raTrace**

**Syntax**

```plaintext
property TraceFileIdentifier: string;
```

**Remarks**
Use the TraceFileIdentifier property to add a string to the name of the SQL trace dump file. If you set TraceFileIdentifier property to some non-empty string then this string will be added to the name of SQL trace dump file. It makes finding of the SQL trace dump file easier.

See Also
- GetTraceFileName
- GetSessionPID

Methods of the TOraTrace class.

For a complete list of the TOraTrace class members, see the TOraTrace Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSessionPID</td>
<td>Returns the operating system process identifier for a process that owns the current session.</td>
</tr>
<tr>
<td>GetTraceFileName</td>
<td>Returns the name of a dump file on the database server to which SQL trace data is written.</td>
</tr>
<tr>
<td>PlSqlTraceComment</td>
<td>Provides a comment on the PL/SQL trace.</td>
</tr>
<tr>
<td>PlSqlTraceLimit</td>
<td>Limits the amount of storage used in the database for the PL/SQL trace data.</td>
</tr>
<tr>
<td>PlSqlTracePause</td>
<td>makes a pause in PL/SQL tracing.</td>
</tr>
<tr>
<td>PlSqlTraceResume</td>
<td>Resumes PL/SQL trace.</td>
</tr>
<tr>
<td>PlSqlTrace_RunNumber</td>
<td>Returns a run number for the current PL/SQL trace.</td>
</tr>
<tr>
<td>PlSqlTraceStart</td>
<td>Starts PL/SQL trace data collection.</td>
</tr>
<tr>
<td>PlSqlTraceStop</td>
<td>Stops the PL/SQL trace.</td>
</tr>
<tr>
<td>SqlTraceStart</td>
<td>Starts the SQL trace.</td>
</tr>
</tbody>
</table>
SqlTraceStop

See Also

- TOraTrace Class
- TOraTrace Class Members

5.15.1.27.3.1  GetSessionPID Method

Returns the operating system process identifier for a process that owns the current session.

Class

TOraTrace

Syntax

function GetSessionPID: integer;

Return Value

the operating system process identifier for a process that owns the current session.

Remarks

Call the GetSessionPID method to return the operating system process identifier for a process that owns the current session. This identifier can be useful to find the dump file in which SQL trace data is written. The name of the dump file contains this identifier.

See Also

- GetTraceFileName
- TraceFileIdentifier

5.15.1.27.3.2  GetTraceFileName Method

Returns the name of a dump file on the database server to which SQL trace data is written.

Class
**T0raTrace**

**Syntax**

```plaintext
function GetTraceFileName: string;
```

**Return Value**

the name of a dump file on the database server to which SQL trace data is written.

**Remarks**

Call the GetTraceFileName method to return the name of a dump file on the database server to which SQL trace data is written. The file name is returned with the path to the file.

**Note:** In some versions of Oracle database the dump file name format can be different from the one returned by the GetTraceFileName method.

**See Also**

- `GetSessionPID`
- `TraceFileIdentifier`

---

**5.15.1.27.3.3 PlSqlTraceComment Method**

Provides a comment on the PL/SQL trace.

**Class**

`T0raTrace`

**Syntax**

```plaintext
procedure PlSqlTraceComment(const Comment: string);
```

**Parameters**

`Comment`

Holds the comment for the PL/SQL trace.

**Remarks**

Call the PlSqlTraceComment method to set a comment on the PL/SQL trace.
5.15.1.27.3.4 PlSqlTraceLimit Method

Limits the amount of storage used in the database for the PL/SQL trace data.

Class
TOraTrace

Syntax

```
procedure PlSqlTraceLimit(Limit: integer = 8192);
```

Parameters
Limit
Holds the limit size for the storage used for the PL/SQL trace data.

Remarks
Call the PlSqlTraceLimit method to limit the amount of storage used in the database for the PL/SQL trace data.

5.15.1.27.3.5 PlSqlTracePause Method

makes a pause in PL/SQL tracing.

Class
TOraTrace

Syntax

```
procedure PlSqlTracePause;
```

Remarks
Call the PlSqlTracePause method to pause PL/SQL trace.

See Also
- PlSqlTraceResume
5.15.1.27.3.6  PlSqlTraceResume Method

Resumes PL/SQL trace.

Class

TOraTrace

Syntax

```
procedure PlSqlTraceResume;
```

Remarks

Call the PlSqlTraceResume method to resume PL/SQL trace.

See Also

- PlSqlTracePause

5.15.1.27.3.7  PlSqlTraceRunNumber Method

Returns a run number for the current PL/SQL trace.

Class

TOraTrace

Syntax

```
function PlSqlTraceRunNumber: integer;
```

Return Value

- a run number for the current PL/SQL trace.

Remarks

Call the PlSqlTraceRunNumber method to return a run number for the current PL/SQL trace. This number can be used to retrieve information from the trace tables.
5.15.1.27.3.8  PLSqlTraceStart Method

Starts PL/SQL trace data collection.

Class

TOraTrace

Syntax

```
procedure PLSqlTraceStart;
```

Remarks

When Enabled property is False, call the PLSqlTraceStart method to start PL/SQL trace data collection. Setting Enabled property to True is another way to start the PL/SQL trace.

See Also

- Enabled
- PLSqlTraceStop
- PLSqlTracePause
- PLSqlTraceResume

5.15.1.27.3.9  PLSqlTraceStop Method

Stops the PL/SQL trace.

Class

TOraTrace

Syntax

```
procedure PLSqlTraceStop;
```

Remarks

Call the PLSqlTraceStop method to stop the PL/SQL trace. Setting the Enabled property to False is another way to stop the PL/SQL trace.

See Also
5.15.1.27.3.10  SqlTraceStart Method

Starts the SQL trace.

Class
TOraTrace

Syntax

procedure SqlTraceStart;

Remarks

When the Enabled property is False, call the SqlTraceStart method to start the SQL trace. Setting the Enabled property to True is another way to start the SQL trace.

See Also

- Enabled
- SqlTraceStop

5.15.1.27.3.11  SqlTraceStop Method

Stops the SQL trace.

Class
TOraTrace

Syntax

procedure SqlTraceStop;
Remarks

Call the SqlTraceStop method to stop the SQL trace. Setting the Enabled property to False is another way to stop the SQL trace.

See Also

- Enabled
- SqlTraceStart

Syntax

```
TOraUpdateSQL = class(TCustomDAUpdateSQL);
```

Remarks

Use the TOraUpdateSQL component to provide DML statements for the dataset components that return read-only result set. This component also allows setting objects that can be used for executing update operations. You may prefer to use directly SQLInsert, SQLUpdate, and SQLDelete properties of the TCustomDADataset descendants.

Inheritance Hierarchy

TCustomDAUpdateSQL
    TOraUpdateSQL

See Also

- TOraDataSet.UpdateObject
### 5.15.1.28.1 Members

**TOraUpdateSQL** class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DataSet</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used to hold a reference to the TCustomDADataSet object that is being updated.</td>
</tr>
<tr>
<td><strong>DeleteObject</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Provides ability to perform advanced adjustment of the delete operations.</td>
</tr>
<tr>
<td><strong>DeleteSQL</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used when deleting a record.</td>
</tr>
<tr>
<td><strong>InsertObject</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Provides ability to perform advanced adjustment of insert operations.</td>
</tr>
<tr>
<td><strong>InsertSQL</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used when inserting a record.</td>
</tr>
<tr>
<td><strong>LockObject</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Provides ability to perform advanced adjustment of lock operations.</td>
</tr>
<tr>
<td><strong>LockSQL</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used to lock the current record.</td>
</tr>
<tr>
<td><strong>ModifyObject</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Provides ability to perform advanced adjustment of modify operations.</td>
</tr>
<tr>
<td><strong>ModifySQL</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used when updating a record.</td>
</tr>
<tr>
<td><strong>RefreshObject</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Provides ability to perform advanced adjustment of refresh operations.</td>
</tr>
<tr>
<td><strong>RefreshSQL</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used to specify an SQL statement that will be used for refreshing the current record by <strong>TCustomDADataSet.RefreshRecord</strong> procedure.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>(inherited from <strong>TCustomDAUpdateSQL</strong>) Used to return a SQL statement for one of the <strong>ModifySQL</strong>, <strong>InsertSQL</strong>, or <strong>DeleteSQL</strong> properties.</td>
</tr>
</tbody>
</table>
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Apply</strong></td>
<td>Sets parameters for a SQL statement and executes it to update a record.</td>
</tr>
<tr>
<td><strong>ExecSQL</strong></td>
<td>Executes a SQL statement.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

5.15.1.29 TOraXMLField Class

A class providing access to the Oracle SYS.XMLTYPE objects.

For a list of all members of this type, see TOraXMLField members.

Unit
ora

Syntax

```
TOraXMLField = class(TField);
```

Remarks

TOraXMLField provides access to the Oracle SYS.XMLTYPE objects.

The TMSXMLField.DataType property values equal to ftXML. You can access actual XML document using the AsString and TOraXMLField.AsXML properties.

See Also

- TOraXML

© 1997-2021
Devart. All Rights Reserved.

5.15.1.29.1 Members

TOraXMLField class overview.

Properties
### 5.15.1.29.2 Properties

Properties of the `TOraXMLField` class.

For a complete list of the `TOraXMLField` class members, see the [TOraXMLField Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsXML</td>
<td>Used to provide access to a <code>TOraXML</code> object.</td>
</tr>
</tbody>
</table>

**See Also**

- [TOraXMLField Class](#)
- [TOraXMLField Class Members](#)

Used to provide access to a `TOraXML` object.

### Class

`TOraXMLField`

### Syntax

```property
AsXML: TOraXML;
```

### Remarks

Use the AsXML property to get access to `TOraXML` object that can be used for manipulations with an XML document.
5.15.2 Types

Types in the Ora unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TConnectChangeEvent</td>
<td>This type is used for the TOraSession.OnConnectChangeEvent event.</td>
</tr>
<tr>
<td>TFailoverEvent</td>
<td>This Type is used for the TOraSession.OnFailover event.</td>
</tr>
<tr>
<td>TOraChangeNotificationEvent</td>
<td>This type is used for the TOraChangeNotification.OnChange event.</td>
</tr>
<tr>
<td>TPISqlTraceMode</td>
<td>Specifies the level of PL/SQL trace.</td>
</tr>
<tr>
<td>TSqlTraceMode</td>
<td>Specifies the level of SQL trace statistics level.</td>
</tr>
</tbody>
</table>

5.15.2.1 TConnectChangeEvent Procedure Reference

This type is used for the TOraSession.OnConnectChangeEvent event.

Unit

Ora

Syntax

```
TConnectChangeEvent = procedure (Sender: TObject; Connected: boolean) of object;
```

Parameters
5.15.2.2 TFailoverEvent Procedure Reference

This Type is used for the TOraSession.OnFailover event.

Unit
ora

Syntax

```
TFailoverEvent = procedure (Sender: TObject; FailoverState: TFailoverState; FailoverType: TFailoverType; var Retry: boolean) of object;
```

Parameters

- **Sender**: An object that raised the event.
- **FailoverState**: The failover state.
- **FailoverType**: The type of failover.
- **Retry**: True, if performing of the failover should be retried.

5.15.2.3 TOraChangeNotificationEvent Procedure Reference

This type is used for the TOraChangeNotificationOnChange event.

Unit
ora

Syntax
TOraChangeNotificationEvent = procedure (Sender: TObject; NotifyType: TChangeNotifyEventType; TableChanges: TNotifyTableChanges) of object;

Parameters

Sender
An object that raised the event.

NotifyType
The type of the event that occurred.

TableChanges
Holds the information on all table changes.

5.15.2.4 TPlSqlTraceMode Set

Specifies the level of PL/SQL trace.

Unit
Ora

Syntax

TPlSqlTraceMode = set of ( pmAllCalls, pmEnabledCalls, pmAllExceptions, pmEnabledExceptions, pmAllSql, pmEnabledSql, pmAllLines, pmEnabledLines);

5.15.2.5 TSqlTraceMode Set

Specifies the level of SQL trace statistics level.

Unit
Ora

Syntax

TSqlTraceMode = set of ( smBasicStatistics, smTypicalStatistics, smAllStatistics, smBindVariables, smWaitEvents, smTimedStatistics);
5.15.3 Enumerations

Enumerations in the **Ora** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFailoverState</td>
<td>Indicates the failover state.</td>
</tr>
<tr>
<td>TFailoverType</td>
<td>Specifies the failover type.</td>
</tr>
<tr>
<td>TOraIsolationLevel</td>
<td>Specifies the way the transactions containing database modifications are handled.</td>
</tr>
<tr>
<td>TRefreshMode</td>
<td>Defines when to refresh an editing record.</td>
</tr>
<tr>
<td>TSequenceMode</td>
<td>Specifies the method used internally to generate sequenced field.</td>
</tr>
</tbody>
</table>

### 5.15.3.1 TFailoverState Enumeration

Indicates the failover state.

**Unit**

**ora**

**Syntax**

```plaintext
TFailoverState = (fsEnd, fsAbort, fsReauth, fsBegin, fsError);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>fsAbort</td>
<td>Indicates that failover was unsuccessful and there is no option of retrying.</td>
</tr>
<tr>
<td>fsBegin</td>
<td>Indicates that failover has detected a lost connection and failover is starting.</td>
</tr>
<tr>
<td>fsEnd</td>
<td>Indicates successful completion of failover.</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>fsError</td>
<td>Indicates that an error occurred while trying to re-establish the connection.</td>
</tr>
<tr>
<td>fsReauth</td>
<td>Indicates that a user handle has been re-authenticated.</td>
</tr>
</tbody>
</table>

### 5.15.3.2 TFailoverType Enumeration

Specifies the failover type.

**Unit**

**Ora**

**Syntax**

```
TFailoverType = (ftSession, ftSelect);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftSelect</td>
<td>Indicates that the user has requested select failover as well.</td>
</tr>
<tr>
<td>ftSession</td>
<td>Indicates that the user has requested only session failover.</td>
</tr>
</tbody>
</table>

### 5.15.3.3 TOraIsolationLevel Enumeration

Specifies the way the transactions containing database modifications are handled.

**Unit**

**Ora**

**Syntax**

```
TOraIsolationLevel = (ilReadCommitted, ilSerializable, ilReadOnly);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ilReadCommitted</td>
<td>Read committed isolation level</td>
</tr>
<tr>
<td>ilSerializable</td>
<td>Serializable isolation level</td>
</tr>
<tr>
<td>ilReadOnly</td>
<td>Read only isolation level</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ilReadCommitted</strong></td>
<td>If the transaction contains DML that requires row locks held by another transaction, then the DML statement waits until the row locks are released. The default Oracle behavior.</td>
</tr>
<tr>
<td><strong>ilReadOnly</strong></td>
<td>All subsequent queries in that transaction only see changes committed before the transaction began. Read-only transactions are useful for reports that run multiple queries against one or more tables while other users update these same tables.</td>
</tr>
<tr>
<td><strong>ilSerializable</strong></td>
<td>Specifies serializable transaction isolation mode as defined in the SQL92 standard. If a serializable transaction contains data manipulation language (DML) that attempts to update any resource that may have been updated in a transaction uncommitted at the start of the serializable transaction, then the DML statement fails.</td>
</tr>
</tbody>
</table>

### 5.15.3.4 TRefreshMode Enumeration

Defines when to refresh an editing record.

**Unit**

**ora**

**Syntax**

\[ TRefreshMode = (rmNone, rmAfterInsert, rmAfterUpdate, rmAlways); \]

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmAfterInsert</td>
<td>Refresh is performed after inserting a record.</td>
</tr>
<tr>
<td>rmAfterUpdate</td>
<td>Refresh is performed after updating a record.</td>
</tr>
<tr>
<td>rmAlways</td>
<td>Refresh is performed after inserting and updating a record.</td>
</tr>
<tr>
<td>rmNone</td>
<td>No refresh is performed. The default value.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
5.15.3.5 TSequenceMode Enumeration

Specifies the method used internally to generate sequenced field.

Unit

Ora

Syntax

TSequenceMode = (smInsert, smPost);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>smInsert</td>
<td>New record is inserted into the dataset where the first key field is populated with a sequenced value. An application may modify this field before posting the record to the database.</td>
</tr>
<tr>
<td>smPost</td>
<td>Database server populates the key field with a sequenced value when application posts the record to the database. Any value put into the key field before post will be overwritten.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.15.4 Variables

Variables in the Ora unit.

Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefSession</td>
<td>Read this variable to get pointer to default session object. Same as DefaultSession function.</td>
</tr>
<tr>
<td>OraQueryCompatibilityMode</td>
<td>All TOracleQuery components in project become editable, and can be modified by the end users.</td>
</tr>
<tr>
<td>Sessions</td>
<td>Holds pointers to all TOracleSession objects of an application.</td>
</tr>
<tr>
<td>UseDefSession</td>
<td>When set to True enables TOracleDataSet and TOracleSQL</td>
</tr>
</tbody>
</table>
5.15.4.1 DefSession Variable

Read this variable to get pointer to default session object. Same as DefaultSession function.

Unit
Ora

Syntax

```delphi
DefSession: TOraSession;
```

5.15.4.2 OraQueryCompatibilityMode Variable

All TOraQuery components in project become editable, and can be modified by the end users.

Unit
Ora

Syntax

```delphi
OraQueryCompatibilityMode: boolean = False;
```

Remarks

Before ODAC 6, TOraQuery could be editable only when InsertSQL, UpdateSQL, and DeleteSQL properties are assigned. The ability to generate update SQL statements with TOraQuery automatically was added in ODAC 6.00.0.4. Therefore, after upgrading your ODAC to the sixth version, all TOraQuery components in your project become editable, and can be modified by the end users. To restore the old behavior, set the OraQueryCompatibilityMode variable to True.
5.15.4.3 Sessions Variable

Holds pointers to all TOraSession objects of an application.

Unit
ora

Syntax

```pascal
Sessions: TSessionList;
```

5.15.4.4 UseDefSession Variable

When set to True enables TOraDataSet and TOraSQL components to use default session if they are not attached to any session.

Unit
ora

Syntax

```pascal
useDefSession: boolean;
```

5.16 OraAlerter

This unit contains implementation of the TOraAlerter component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraAlerter</td>
<td>A component allowing transferring messages between sessions or client applications.</td>
</tr>
</tbody>
</table>
Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOnEventEvent</td>
<td>This type is used for the TOraAlerter.OnEvent event.</td>
</tr>
<tr>
<td>TOnTimeOutEvent</td>
<td>This type is used for the TOraAlerter.OnTimeOut event.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEventType</td>
<td>Specifies the kind of messages used by the TOraAlerter component.</td>
</tr>
</tbody>
</table>

5.16.1 Classes

Classes in the OraAlerter unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraAlerter</td>
<td>A component allowing transferring messages between sessions or client applications.</td>
</tr>
</tbody>
</table>

5.16.1.1 TOraAlerter Class

A component allowing transferring messages between sessions or client applications.

For a list of all members of this type, see TOraAlerter members.

Unit

OraAlerter
Syntax

```
TOrAlerter = class(TDaAlerter);
```

Remarks

The TOrAlerter component allows to transfer messages between sessions or client applications. Use the EventType property to specify what kind of messages will be used. The TOrAlerter component supports Oracle alerts by DBMS_ALERT and pipes by DBMS_PIPE. You can send and get messages with the help of this component. When TOrAlerter is started by Start method it waits messages in a background thread. If a message is received, the OnEvent event occurs. If no messages were received during the TimeOut time, the OnTimeOut event occurs.

**Note:** Alerts are transaction-based. This means that the waiting session does not get alert until the transaction signaling the alert commits.

Inheritance Hierarchy

```
TDAAlerter
   TOrAlerter
```

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong> (inherited from TDAAlerter)</td>
<td>Used to determine if TDAAlerter waits for messages.</td>
</tr>
<tr>
<td><strong>AutoCommit</strong></td>
<td>Used to specify whether the TOrAlerter object should commit transaction in its Session property after calling the SendEvent method to send a named message to an Oracle server.</td>
</tr>
</tbody>
</table>
### AutoRegister (inherited from TDAAlerter)

Used to automatically register events whenever connection opens.

### Connection (inherited from TDAAlerter)

Used to specify the connection for TDAAlerter.

### Events

Used to set the names of waiting alerts or pipes.

### EventType

Used to define the kind of messages used by the TOraAlerter component.

### Interval

Specifies the time after which TOraAlerter starts waiting process.

### Session

Specifies the session for TOraAlerter to create an internal TOraSession object based on this session settings.

### TimeOut

Used to set the time for the TOraAlerter component to wait for a message.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetMessage</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>NextItemType</td>
<td>Returns the datatype of the next message found in the received named pipe local buffer.</td>
</tr>
<tr>
<td>NextMessageType</td>
<td>Returns the datatype of the next message found in the received named pipe local buffer.</td>
</tr>
<tr>
<td>PackMessage</td>
<td>Places messages into the named pipe local buffer.</td>
</tr>
<tr>
<td>PurgePipe</td>
<td>Removes all incoming messages from the pipe's input buffer.</td>
</tr>
<tr>
<td>PutMessage</td>
<td>Places messages into the named pipe local buffer.</td>
</tr>
<tr>
<td>SendEvent</td>
<td>Sends an event with Name and content Message.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SendMessage</td>
<td>Sends a named pipe event to other listening sessions.</td>
</tr>
<tr>
<td>SendPipeMessage</td>
<td>Sends a named pipe event to other listening sessions.</td>
</tr>
<tr>
<td>Start (inherited from TDAAlerter)</td>
<td>Starts waiting process.</td>
</tr>
<tr>
<td>Stop (inherited from TDAAlerter)</td>
<td>Stops waiting process.</td>
</tr>
<tr>
<td>UnpackMessage</td>
<td>Overloaded. Retrieves messages from a named pipe local buffer.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError (inherited from TDAAlerter)</td>
<td>Occurs if an exception occurs in waiting process</td>
</tr>
<tr>
<td>OnEvent</td>
<td>Occurs if a message is received by waiting process.</td>
</tr>
<tr>
<td>OnTimeOut</td>
<td>Occurs if there were no messages during the TimeOut time.</td>
</tr>
</tbody>
</table>

Properties of the TOraAlerter class.

For a complete list of the TOraAlerter class members, see the TOraAlerter Members topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active (inherited from TDAAlerter)</td>
<td>Used to determine if TDAAlerter waits for messages.</td>
</tr>
<tr>
<td>AutoRegister (inherited from TDAAlerter)</td>
<td>Used to automatically register events whenever connection opens.</td>
</tr>
<tr>
<td>Connection (inherited from TDAAlerter)</td>
<td>Used to specify the connection for TDAAlerter.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AutoCommit</td>
<td>Used to specify whether the TOraAlerter object should commit transaction in its Session property after calling the SendEvent method to send a named message to an Oracle server.</td>
</tr>
<tr>
<td>Events</td>
<td>Used to set the names of waiting alerts or pipes.</td>
</tr>
<tr>
<td>EventType</td>
<td>Used to define the kind of messages used by the TOraAlerter component.</td>
</tr>
<tr>
<td>Interval</td>
<td>Specifies the time after which TOraAlerter starts waiting process.</td>
</tr>
<tr>
<td>Session</td>
<td>Specifies the session for TOraAlerter to create an internal TOraSession object based on this session settings.</td>
</tr>
<tr>
<td>TimeOut</td>
<td>Used to set the time for the TOraAlerter component to wait for a message.</td>
</tr>
</tbody>
</table>

See Also

- TOraAlerter Class
- TOraAlerter Class Members

Used to specify whether the TOraAlerter object should commit transaction in its Session property after calling the SendEvent method to send a named message to an Oracle server.

Class

TOraAlerter
Syntax

```objectivec
property AutoCommit: boolean;
```

Remarks

Set the AutoCommit property to specify whether the TOraAlerter object should commit transaction in its Session property after calling the SendEvent method to send a named message to an Oracle server.

Setting AutoCommit to True instructs TOraAlerter to commit its session each time an event is sent. Otherwise listening session will have to wait till current transaction terminates and only then it will be notified by the server.

See Also

- `TDAAlerter.SendEvent`

© 1997-2021 Devart. All Rights Reserved.

5.16.1.1.2.2 Events Property

Used to set the names of waiting alerts or pipes.

Class

`TOraAlerter`

Syntax

```objectivec
property Events: string;
```

Remarks

Use the Events property to set the names of alerts or pipes which are waiting.

© 1997-2021 Devart. All Rights Reserved.

5.16.1.1.2.3 EventType Property

Used to define the kind of messages used by the TOraAlerter component.

Class
**TOraAlerter**

Syntax

```object
property EventType: TEventType;
```

Remarks

Use `EventType` to specify what kind of messages is used by the TOraAlerter component.

---

5.16.1.2.4 Interval Property

Specifies the time after which TOraAlerter starts waiting process.

Class

**TOraAlerter**

Syntax

```object
property Interval: integer default 0;
```

Remarks

If `Interval` property is greater than 0, TOraAlerter starts waiting process in `Interval` seconds after time out occurred.

---

5.16.1.2.5 Session Property

Specifies the session for TOraAlerter to create an internal TOraSession object based on this session settings.

Class

**TOraAlerter**

Syntax

```object
property Session: TOraSession;
```
Remarks
Specifies the session for **TOraAlerter** to create an internal **TOraSession** object based on this session settings.

See Also
- **TOraSession**

5.16.1.1.2.6 TimeOut Property

Used to set the time for the TOraAlerter component to wait for a message.

Class
**TOraAlerter**

Syntax

```
property TimeOut: integer default 0;
```

Remarks
Use TimeOut property to set the time of waiting message by the TOraAlerter component.

5.16.1.1.3 Methods

Methods of the **TOraAlerter** class.

For a complete list of the **TOraAlerter** class members, see the **TOraAlerter Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetMessage</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>NextItemType</td>
<td>Returns the datatype of the next message found in the received named pipe local buffer.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>NextMessageType</td>
<td>Returns the datatype of the next message found in the received named pipe local buffer.</td>
</tr>
<tr>
<td>PackMessage</td>
<td>Places messages into the named pipe local buffer.</td>
</tr>
<tr>
<td>PurgePipe</td>
<td>Removes all incoming messages from the pipe's input buffer.</td>
</tr>
<tr>
<td>PutMessage</td>
<td>Places messages into the named pipe local buffer.</td>
</tr>
<tr>
<td>SendEvent (inherited from TDAAlerter)</td>
<td>Sends an event with Name and content Message.</td>
</tr>
<tr>
<td>SendMessage</td>
<td>Sends a named pipe event to other listening sessions.</td>
</tr>
<tr>
<td>SendPipeMessage</td>
<td>Sends a named pipe event to other listening sessions.</td>
</tr>
<tr>
<td>Start (inherited from TDAAlerter)</td>
<td>Starts waiting process.</td>
</tr>
<tr>
<td>Stop (inherited from TDAAlerter)</td>
<td>Stops waiting process.</td>
</tr>
<tr>
<td>UnpackMessage</td>
<td>Overloaded. Retrieves messages from a named pipe local buffer.</td>
</tr>
</tbody>
</table>

See Also

- TораAlerter Class
- TораAlerter Class Members

© 1997-2021 Devart. All Rights Reserved.

5.16.1.1.3.1 GetMessage Method

Class

TораAlerter

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetMessage</td>
<td>Retrieves messages from a named pipe local buffer.</td>
</tr>
<tr>
<td>GetMessage(var Item: variant)</td>
<td>Retrieves messages from a named pipe</td>
</tr>
</tbody>
</table>
Retrieves messages from a named pipe local buffer.

Class

T0raAlerter

Syntax

```pascal
function GetMessage: variant; overload;
```

Return Value

True, if the Item parameter holds not Null variant value or False otherwise.

Remarks

Call the GetMessage method to retrieve messages from a named pipe local buffer.

GetMessage is designed to work only with Oracle DBMS_PIPE communication package which is the case only if T0raAlerter.EventType property has been set to etPipe.

Implementation with the parameter will return True if the Item parameter holds not Null variant value or False otherwise.

**Note:** This method is considered obsolete now. In newer projects use functionally equivalent T0raAlerter.UnpackMessage method instead.
**Item**

Holds a value received from the pipe.

**Return Value**

True, if the Item parameter holds not Null variant value or False otherwise.

---

5.16.1.1.3.2 NextItemType Method

Returns the datatype of the next message found in the received named pipe local buffer.

**Class**

`TOraAlerter`

**Syntax**

```delphi
function NextItemType: TMessageType;
```

**Return Value**

the datatype of the next message found in the received named pipe local buffer.

**Remarks**

Use NextItemType to retrieve the datatype of the next message found in the received named pipe local buffer.

NextItemType is designed to work only with Oracle DBMS_Pipe communication package which is the case only if `EventType` property has been set to `etPipe`.

The `mtNone` return value indicates that no more messages are found in the local buffer.

---

5.16.1.1.3.3 NextMessageType Method

Returns the datatype of the next message found in the received named pipe local buffer.

**Class**

`TOraAlerter`

**Syntax**


```function
NextMessageType: TMessageType;
```

**Return Value**

- the datatype of the next message found in the received named pipe local buffer.

**Remarks**

Use `NextMessageType` to retrieve the datatype of the next message found in the received named pipe local buffer.

`NextMessageType` is designed to work only with Oracle DBMS_PIPE communication package which is the case only if `EventType` property has been set to `etPipe`.

The return value of `mtNone` indicates that no more messages are found in the local buffer.

**Note:** This method is considered obsolete now. In newer projects use functionally equivalent `NextItemType` method instead.

5.16.1.1.3.4 PackMessage Method

Places messages into the named pipe local buffer.

**Class**

`T0raAlert`er

**Syntax**

```procedure
PackMessage(Item: variant);
```

**Parameters**

- `Item`
  - Holds the value to be sent in a message.

**Remarks**

Call `PackMessage` to place messages into the named pipe local buffer. Local buffer is limited in size to 8192 bytes and besides the actual message values accommodates other internal data. Item will be dropped out if it doesn't fit into available free buffer space.

`PackMessage` is designed to work only with the Oracle DBMS_PIPE communication package which is the case only if the `EventType` property has been set to `etPipe`.

© 1997-2021 Devart. All Rights Reserved.

© 2021 Devart
5.16.1.1.3.5 PurgePipe Method

Removes all incoming messages from the pipe's input buffer.

Class

T0raAlerter

Syntax

```
procedure PurgePipe;
```

Remarks

Call PurgePipe to clear the pipe's input buffer.

PurgePipe is designed to work only with the Oracle DBMS_PIPE communication package. So it can be called only if Event_Type property has been set to etPipe.

© 1997-2021

Devart. All Rights Reserved.

5.16.1.1.3.6 PutMessage Method

Places messages into the named pipe local buffer.

Class

T0raAlerter

Syntax

```
procedure PutMessage(Item: variant);
```

Parameters

- **Item**
  
  Holds the value to be sent in a message.

Remarks

Call PutMessage to place messages into the named pipe local buffer. Local buffer is limited in size to 8192 bytes and besides the actual message values accommodates other internal data. Item will be dropped out if it doesn't fit into free buffer space.
PutMessage is designed to work only with the Oracle DBMS_PIPE communication package which is the case only if EventType property has been set to etPipe.

**Note:** This method is now considered obsolete. In newer projects use functionally equivalent PackMessage method instead.

### 5.16.1.1.3.7 SendMessage Method

Sends a named pipe event to other listening sessions.

**Class**

TораAlerter

**Syntax**

```pascal
procedure SendMessage(Name: string = '');
```

**Parameters**

- **Name**
  - Holds the name of the pipe.

**Remarks**

Use SendMessage procedure to send a named pipe event to other listening sessions. The event internally is a local buffer which has been previously filled in by the PutMessage method.

SendMessage is designed to work only with the Oracle DBMS_PIPE communication package which is the case only if EventType property has been set to etPipe.

**Note:** This method is considered obsolete now. In newer projects use functionally equivalent SendPipeMessage method instead.

### 5.16.1.1.3.8 SendPipeMessage Method

Sends a named pipe event to other listening sessions.

**Class**
**Syntax**

```pascal
procedure SendPipeMessage(Name: string = '');
```

**Parameters**

- **Name**
  - Holds the name of the pipe.

**Remarks**

Use `SendPipeMessage` procedure to send a named pipe event to other listening sessions. The event internally is a local buffer which has been previously filled in by the `PackMessage` method.

`SendPipeMessage` is designed to work only with the Oracle DBMS_PIPE communication package which is the case only if `EventType` property has been set to `etPipe`.

---

5.16.1.3.9 UnpackMessage Method

Retrieves messages from a named pipe local buffer.

**Class**

**TOraAlerter**

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnpackMessage</td>
<td>Retrieves messages from a named pipe local buffer.</td>
</tr>
<tr>
<td>UnpackMessage(var Item: variant)</td>
<td>Retrieves messages from a named pipe local buffer.</td>
</tr>
</tbody>
</table>

---

© 1997-2021 Devart. All Rights Reserved.
**TOraAlerter**

**Syntax**

```delphi
function UnpackMessage: variant; overload;
```

**Return Value**

True, if the Item parameter holds not Null variant value or False otherwise.

**Remarks**

UnpackMessage function is used to retrieve messages from a named pipe local buffer.

UnpackMessage is designed to work only with the Oracle DBMS_PIPE communication package which is the case only if `TOraAlerter.Event_Type` property has been set to etPipe.

An implementation with the parameter will return True if the Item parameter holds not Null variant value or False otherwise.

© 1997-2021 Devart. All Rights Reserved.

Retrieves messages from a named pipe local buffer.

**Class**

**TOraAlerter**

**Syntax**

```delphi
function UnpackMessage(var Item: variant): variant; overload;
```

**Parameters**

*Item*

Holds a value received from the pipe.

**Return Value**

True, if the Item parameter holds not Null variant value or False otherwise.

© 1997-2021 Devart. All Rights Reserved.

5.16.1.1.4 Events

Events of the **TOraAlerter** class.
For a complete list of the `TOraAlerter` class members, see the `TOraAlerter Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnError</code> (inherited from <code>TDAAlerter</code>)</td>
<td>Occurs if an exception occurs in waiting process</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnEvent</code></td>
<td>Occurs if a message is received by waiting process.</td>
</tr>
<tr>
<td><code>OnTimeOut</code></td>
<td>Occurs if there were no messages during the TimeOut time.</td>
</tr>
</tbody>
</table>

See Also
- `TOraAlerter Class`
- `TOraAlerter Class Members`

5.16.1.1.4.1 OnEvent Event

Occurs if a message is received by waiting process.

Class

`TOraAlerter`

Syntax

```pascal
property OnEvent: TOnEventEvent;
```

Remarks

Occurs when waiting process receives some message. The event parameter is the name of an event (alert or pipe) and Message is its content.

See Also
- `OnTimeOut`
5.16.1.4.2 OnTimeOut Event

Occurs if there were no messages during the TimeOut time.

Class

TOnTimeOutEvent

Syntax

property OnTimeOut: TOnTimeOutEvent;

Remarks

Occurs when there were no messages during the TimeOut time. Assign True to the Continue parameter to continue waiting messages. If Continue is False (by default) waiting process is stopped.

See Also

- TimeOut
- OnEvent

5.16.2 Types

Types in the OraAlerter unit.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOnEventEvent</td>
<td>This type is used for the TOnTimeOutEvent event.</td>
</tr>
<tr>
<td>TOnTimeOutEvent</td>
<td>This type is used for the TOnTimeOutEvent event.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
5.16.2.1 TOnEventEvent Procedure Reference

This type is used for the TOnEventOnEvent event.

Unit

OraAlerter

Syntax

TOnEventEvent = procedure (Sender: TObject; Event: string; Message: string) of object;

Parameters

Sender
An object that raised the event.

Event
A name of event (alert or pipe).

Message
The content of message waiting process receives.

5.16.2.2 TOnTimeOutEvent Procedure Reference

This type is used for the TOnEventOnTimeOut event.

Unit

OraAlerter

Syntax

TOnTimeOutEvent = procedure (Sender: TObject; var Continue: boolean) of object;

Parameters

Sender
An object that raised the event.

Continue
True, if waiting messages process should be continued. If False (by default), the waiting process is stopped.
Reserved.

5.16.3 Enumerations

Enumerations in the OraAlerter unit.

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEventType</td>
<td>Specifies the kind of messages used by the TOraAlerter component.</td>
</tr>
</tbody>
</table>

5.16.3.1 TEventType Enumeration

Specifies the kind of messages used by the TOraAlerter component.

Unit

OraAlerter

Syntax

TEventType = (etAlert, etPipe);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>etAlert</td>
<td>Lets all listening sessions be notified when another session broadcasts its message with the corresponding name parameter. The default value.</td>
</tr>
<tr>
<td>etPipe</td>
<td>Causes only one of the listening sessions to receive a message.</td>
</tr>
</tbody>
</table>

5.17 OraAQ

This unit contains ODAC components for working with Oracle Advanced Queueing.

Classes
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDequeueOptions</td>
<td>A base class for setting default dequeue options for the queue or for using special dequeue options when calling the TOraQueue.Dequeue or TOraQueue.DequeueArray methods.</td>
</tr>
<tr>
<td>TEnqueueOptions</td>
<td>A base class for setting default or special enqueue options for a queue.</td>
</tr>
<tr>
<td>TOraQueue</td>
<td>A component providing access to Oracle Streams Advanced Queuing.</td>
</tr>
<tr>
<td>TOraQueueAdmin</td>
<td>A component for managing queues in a database.</td>
</tr>
<tr>
<td>TOraQueueTable</td>
<td>A component managing queue tables in a database.</td>
</tr>
<tr>
<td>TQueueAgent</td>
<td>A class representing a producer or a consumer of a queue message.</td>
</tr>
<tr>
<td>TQueueAgents</td>
<td>A class holding a collection of the TQueueAgent objects.</td>
</tr>
<tr>
<td>TQueueMessage</td>
<td>A class representing a queue message.</td>
</tr>
<tr>
<td>TQueueMessageProperties</td>
<td>A class for setting or providing queue message properties.</td>
</tr>
</tbody>
</table>

### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQueueMessageEvent</td>
<td>This type is used for the TOraQueue.OnMessage event.</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDequeueMode</td>
<td>Specifies the locking behavior associated with the dequeue.</td>
</tr>
<tr>
<td>Class Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>TQueueDeliveryMode</strong></td>
<td>Specifies the type of the message that will be dequeued.</td>
</tr>
<tr>
<td><strong>TQueueMessageGrouping</strong></td>
<td>Specifies the message grouping behavior in the queues based on this table.</td>
</tr>
<tr>
<td><strong>TQueueMessageState</strong></td>
<td>Specifies the message state.</td>
</tr>
<tr>
<td><strong>TQueueNavigation</strong></td>
<td>Specifies the position of the message that will be retrieved.</td>
</tr>
<tr>
<td><strong>TQueueSequenceDeviation</strong></td>
<td>Specifies if a message should be enqueued before other messages.</td>
</tr>
<tr>
<td><strong>TQueueSortList</strong></td>
<td>Specifies the column that will be used as a sort key.</td>
</tr>
<tr>
<td><strong>TQueueType</strong></td>
<td>Specifies whether the queue being created is an exception queue or a normal queue.</td>
</tr>
<tr>
<td><strong>TQueueVisibility</strong></td>
<td>Specifies the transaction behaviour of the dequeue or enqueue request.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.17.1 Classes

Classes in the **OraAQ** unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDequeueOptions</td>
<td>A base class for setting default dequeue options for the queue or for using special dequeue options when calling the TOraQueue.Dequeue or TOraQueue.DequeueArray methods.</td>
</tr>
<tr>
<td>TEnqueueOptions</td>
<td>A base class for setting default or special enqueue options.</td>
</tr>
</tbody>
</table>
### 5.17.1.1 TDequeueOptions Class

A base class for setting default dequeue options for the queue or for using special dequeue options when calling the `TOraQueue.Dequeue` or `TOraQueue.DequeueArray` methods.

For a list of all members of this type, see `TDequeueOptions` members.

**Unit**

`OraAQ`

**Syntax**

```delphi
TDequeueOptions = class(TPersistent);
```

**Remarks**

Use the `TDequeueOptions` class for setting default dequeue options for the queue or for using special dequeue options when calling `TOraQueue.Dequeue` or `TOraQueue.DequeueArray` methods.

**See Also**
5.17.1.1.1 Members

**TDequeueOptions** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsumerName</td>
<td>Used to specify the name of the consumer.</td>
</tr>
<tr>
<td>Correlation</td>
<td>Used to specify the correlation identifier of the message to be dequeued.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Used to specify the type of the message that will be dequeued.</td>
</tr>
<tr>
<td>DequeueCondition</td>
<td>Used to specify the conditional expression based on the message properties, message data properties, and PL/SQL functions.</td>
</tr>
<tr>
<td>DequeueMode</td>
<td>Used to specify the locking behaviour associated with the dequeue.</td>
</tr>
<tr>
<td>MessageId</td>
<td>Used to specify the message ID for the message to be dequeued.</td>
</tr>
<tr>
<td>Navigation</td>
<td>Used to specify the position of the message that will be retrieved.</td>
</tr>
<tr>
<td>Transformation</td>
<td>Used to specify a transformation that will be applied after dequeuing the message.</td>
</tr>
<tr>
<td>Visibility</td>
<td>Used to specify the transactional behavior of the dequeue request.</td>
</tr>
</tbody>
</table>
WaitTimeout

Specifies the time to wait if there is no message matching the search criteria.

5.17.1.1.2 Properties

Properties of the TDequeueOptions class.

For a complete list of the TDequeueOptions class members, see the TDequeueOptions Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageId</td>
<td>Used to specify the message ID for the message to be dequeued.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConsumerName</td>
<td>Used to specify the name of the consumer.</td>
</tr>
<tr>
<td>Correlation</td>
<td>Used to specify the correlation identifier of the message to be dequeued.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Used to specify the type of the message that will be dequeued.</td>
</tr>
<tr>
<td>DequeueCondition</td>
<td>Used to specify the conditional expression based on the message properties, the message data properties, and PL/SQL functions.</td>
</tr>
<tr>
<td>DequeueMode</td>
<td>Used to specify the locking behaviour associated with the dequeue.</td>
</tr>
<tr>
<td>Navigation</td>
<td>Used to specify the position of the message that will be</td>
</tr>
<tr>
<td><strong>Transformation</strong></td>
<td>Used to specify a transformation that will be applied after dequeuing the message.</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Visibility</strong></td>
<td>Used to specify the transactional behavior of the dequeue request.</td>
</tr>
<tr>
<td><strong>WaitTimeout</strong></td>
<td>Specifies the time to wait if there is no message matching the search criteria.</td>
</tr>
</tbody>
</table>

**See Also**
- **TDequeueOptions Class**
- **TDequeueOptions Class Members**

**5.17.1.1.2.1 ConsumerName Property**

Used to specify the name of the consumer.

**Class**

**TDequeueOptions**

**Syntax**

```plaintext
property ConsumerName: string;
```

**Remarks**

Use ConsumerName property to specify the name of the consumer.

**5.17.1.1.2.2 Correlation Property**

Used to specify the correlation identifier of the message to be dequeued.

**Class**
**TDequeueOptions**

**Syntax**

```property```

**Correlation:** string;
```

**Remarks**

Use `Correlation` property to specify the correlation identifier of the message to be dequeued.

5.17.1.1.2.3 **DeliveryMode Property**

Used to specify the type of the message that will be dequeued.

**Class**

**TDequeueOptions**

**Syntax**

```property```

**DeliveryMode:** TQueueDeliveryMode default qdmPersistent;
```

**Remarks**

Use the `DeliveryMode` property to specify the type of the message that will be dequeued. Use it with Oracle 10 and higher.

**See Also**

- `TEnqueueOptions.DeliveryMode`
- `TQueueMessageProperties.DeliveryMode`

5.17.1.1.2.4 **DequeueCondition Property**

Used to specify the conditional expression based on the message properties, the message data properties, and PL/SQL functions.

**Class**

**TDequeueOptions**
Syntax

```[property] DequeueCondition: [string];```

Remarks

Use DequeueCondition property to specify the conditional expression based on the message properties, the message data properties, and PL/SQL functions. It should be a boolean expression like a SQL WHERE clause. DequeueCondition should not exceed 4000 characters.

Class

`TDequeueOptions`

Syntax

```[property] DequeueMode: TDequeueMode default dqmRemove;```

Remarks

Use DequeueMode property to specify the locking behavior associated with the dequeue.

Class

`TDequeueOptions`

Syntax

```[property] MessageId: [string];```

Remarks

Used to specify the message ID for the message to be dequeued.

Class

`TDequeueOptions`
Remarks

Use the MessageId property to specify the message ID for the message to be dequeued.

5.17.1.1.2.7 Navigation Property

Used to specify the position of the message that will be retrieved.

Class

**TDequeueOptions**

Syntax

```pascal
property Navigation: TQueueNavigation default qnNextMessage;
```

Remarks

Use the Navigation property to specify the position of the message that will be retrieved.

5.17.1.1.2.8 Transformation Property

Used to specify a transformation that will be applied after dequeuing the message.

Class

**TDequeueOptions**

Syntax

```pascal
property Transformation: string;
```

Remarks

Use the Transformation property to specify a transformation that will be applied after dequeuing the message. Use it with Oracle 10 and higher.

See Also

- **TEnqueueOptions.Transformation**
5.17.1.2.9 Visibility Property

Used to specify the transactional behavior of the dequeue request.

Class

`TDequeueOptions`

Syntax

```
property Visibility: TQueueVisibility default qvOnCommit;
```

Remarks

Use Visibility property to specify the transactional behavior of the dequeue request.

See Also

- `TEnqueueOptions.Visibility`

5.17.1.2.10 WaitTimeout Property

Specifies the time to wait if there is no message matching the search criteria.

Class

`TDequeueOptions`

Syntax

```
property WaitTimeout: integer default AQ_FOREVER;
```

Remarks

Use WaitTimeout property to specify the wait time in seconds if there is currently no message matching the search criteria available. You can use constants AQ_FOREVER (the default value - wait forever) and AQ_NO_WAIT (do not wait).
5.17.1.2 TEnqueueOptions Class

A base class for setting default or special enqueue options for a queue.

For a list of all members of this type, see TEnqueueOptions members.

Unit

OraAQ

Syntax

TEnqueueOptions = class(TPersistent);

Remarks

Use the TEnqueueOptions class for setting default enqueue options for a queue or for using special enqueue options when calling TOraQueue.Enqueue or the TOraQueue.EnqueueArray method.

See Also

- TOraQueue.Enqueue
- TOraQueue.EnqueueArray
- TOraQueue.EnqueueOptions

© 1997-2021 Devart. All Rights Reserved.

5.17.1.2.1 Members

TEnqueueOptions class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliveryMode</td>
<td>Used to specify the type of the message that will be enqueued.</td>
</tr>
<tr>
<td>RelativeMessageId</td>
<td>Used to specify the message identifier of the message which is used in the sequence deviation operation.</td>
</tr>
<tr>
<td>SequenceDeviation</td>
<td>Used to specify whether the</td>
</tr>
</tbody>
</table>
5.17.1.2.2 Properties

Properties of the **TEnqueueOptions** class.

For a complete list of the **TEnqueueOptions** class members, see the **TEnqueueOptions Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelativeMessageId</td>
<td>Used to specify the message identifier of the message which is used in the sequence deviation operation.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeliveryMode</td>
<td>Used to specify the type of the message that will be enqueued.</td>
</tr>
<tr>
<td>SequenceDeviation</td>
<td>Used to specify whether the enqueued message should be dequeued before other messages that are in the queue already.</td>
</tr>
<tr>
<td>Transformation</td>
<td>Used to specify the transformation that will be applied before enqueuing a message.</td>
</tr>
</tbody>
</table>
transformation that will be applied before enqueuing a message.

| Visibility | Used to specify the transactional behavior of the enqueue request. |

See Also
- TEnqueueOptions Class
- TEnqueueOptions Class Members

5.17.1.2.2.1 DeliveryMode Property

Used to specify the type of the message that will be enqueued.

Class
TEnqueueOptions

Syntax
```
property DeliveryMode: TQueueDeliveryMode default qdmPersistent;
```

Remarks
Use the DeliveryMode property to specify the type of the message that will be enqueued. Use it with Oracle 10 and higher.

See Also
- TDequeueOptions.DeliveryMode
- TQueueMessageProperties.DeliveryMode

5.17.1.2.2.2 RelativeMessageId Property

Used to specify the message identifier of the message which is used in the sequence deviation operation.
Class

**TEnqueueOptions**

Syntax

```pascal
property RelativeMessageId: string;
```

Remarks

Use the RelativeMsgid property to specify the message identifier of the message which is used in the sequence deviation operation. Ignored if SequenceDeviation is not set to sdBefore.

See Also

- **SequenceDeviation**

© 1997-2021 Devart. All Rights Reserved.

5.17.1.2.2.3 SequenceDeviation Property

Used to specify whether the enqueued message should be dequeued before other messages that are in the queue already.

Class

**TEnqueueOptions**

Syntax

```pascal
property SequenceDeviation: TQueueSequenceDeviation default qsdNone;
```

See Also

- **RelativeMessageId**

© 1997-2021 Devart. All Rights Reserved.

5.17.1.2.2.3 SequenceDeviation Property
5.17.1.2.2.4 Transformation Property

Used to specify the transformation that will be applied before enqueuing a message.

Class
TEnqueueOptions

Syntax

```plaintext
property Transformation: string;
```

Remarks
Use the Transformation property to specify a transformation that will be applied before enqueuing a message. Use it with Oracle 10 and higher.

See Also
- TDequeueOptions.Transformation

5.17.1.2.2.5 Visibility Property

Used to specify the transactional behavior of the enqueue request.

Class
TEnqueueOptions

Syntax

```plaintext
property Visibility: TQueueVisibility default qvOnCommit;
```

Remarks
Use the Visibility property to specify the transactional behavior of the enqueue request.

See Also
- TDequeueOptions.Visibility
5.17.1.3 **TOraQueue Class**

A component providing access to Oracle Streams Advanced Queuing.

For a list of all members of this type, see **TOraQueue** members.

**Unit**

**OraAQ**

**Syntax**

```pascal
TOraQueue = class(TComponent);
```

**Remarks**

The TOraQueue component provides access to Oracle Streams Advanced Queuing. It allows to enqueue and dequeue messages and to listen to the queue.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsyncNotification</td>
<td>Used to generate the OnMessage event each time when a new message is enqueued.</td>
</tr>
<tr>
<td>DequeueOptions</td>
<td>Used to set the default message dequeuing options.</td>
</tr>
<tr>
<td>EnqueueMessageProperties</td>
<td>Used to set the default enqueuing message properties.</td>
</tr>
<tr>
<td>EnqueueOptions</td>
<td>Used to set the default message enqueuing options.</td>
</tr>
<tr>
<td>PayloadArrayTypeName</td>
<td>Contains the type name of associative array, VARRAY, or nested table of queue</td>
</tr>
</tbody>
</table>
Properties of the TOraQueue class.

For a complete list of the TOraQueue class members, see the TOraQueue Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PayloadArrayTypeName</td>
<td>Contains the type name of associative array, VARRAY, or nested table of queue payload type.</td>
</tr>
<tr>
<td>PayloadTypeName</td>
<td>Contains the payload type for the queue.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsyncNotification</td>
<td>Used to generate the OnMessage event each time when a new message is enqueued.</td>
</tr>
<tr>
<td>DequeueOptions</td>
<td>Used to set the default message dequeuing options.</td>
</tr>
<tr>
<td>EnqueueMessageProperties</td>
<td>Used to set the default enqueuing message properties.</td>
</tr>
<tr>
<td>EnqueueOptions</td>
<td>Used to set the default message enqueuing options.</td>
</tr>
<tr>
<td>QueueName</td>
<td>Used to set the name of the Oracle queue to operation.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session through which a queue will be controlled.</td>
</tr>
</tbody>
</table>

See Also

- TOraQueue Class
- TOraQueue Class Members

© 1997-2021 Devart. All Rights Reserved.
### AsyncNotification Property

**Syntax**

```plaintext
property AsyncNotification: boolean default False;
```

**Remarks**

If the AsyncNotification property is True, the TOraQueue component will generate the OnMessage event each time when a new message is enqueued.

When setting the AsyncNotification value to True, a message subscription is created in the queue. When setting the property value to False, the subscription is unregistered.

Active session is required to change the value of AsyncNotification, but TOraQueue doesn't need a session to get notifications. You can disconnect after setting the AsyncNotification value.

AsyncNotification property works only if OCI is initialized with the OCI_EVENTS flag. By default ODAC sets this flag if the OCI version is 9.0 or higher. If you need to use AsyncNotification with Oracle client 8.1, you should set the OCIEventsVersion variable from OraCall unit to 8100 in the initialization section of one of your program units.

AsyncNotification is not supported in Direct mode.

**Note**, the Oracle client version 8.1 has a bug when the NAMES.DEFAULT_DOMAIN parameter in the sqlnet.ora file is not working when the OCI_EVENTS flag is set.

### DequeueOptions Property

**Class**

`TOraQueue`

**Syntax**

```plaintext
property DequeueOptions: TDequeueOptions;
```

**Remarks**

Used to set the default message dequeuing options.
Use the DequeueOptions property to set the default message dequeuing options.

See Also
- TDequeueOptions
- EnqueueOptions

5.17.1.3.2.3 EnqueueMessageProperties Property

Used to set the default enqueuing message properties.

Class
T0raQueue

Syntax

```
property EnqueueMessageProperties: TQueueMessageProperties;
```

Remarks

Use the EnqueueMessageProperties property to set the default enqueuing message properties.

See Also
- TQueueMessageProperties

5.17.1.3.2.4 EnqueueOptions Property

Used to set the default message enqueuing options.

Class
T0raQueue

Syntax

```
property EnqueueOptions: TEnqueueOptions;
```
Remarks

Use the EnqueueOptions property to set the default message enqueuing options.

See Also

- **TEnqueueOptions**
- **DequeueOptions**

## 5.17.1.3.2.5 PayloadArrayTypeName Property

Contains the type name of associative array, VARRAY, or nested table of queue payload type.

Class

**T0raQueue**

Syntax

```plaintext
property PayloadArrayTypeName: string;
```

Remarks

The PayloadArrayTypeName property contains the type name of associative array, VARRAY, or nested table of queue payload type. It is used by EnqueueArray and DequeueArray functions.

See Also

- **EnqueueArray**
- **DequeueArray**
- **PayloadTypeName**

## 5.17.1.3.2.6 PayloadTypeName Property

Contains the payload type for the queue.

Class
**T0raQueue**

**Syntax**

```
property PayloadTypeName: string;
```

**Remarks**

The PayloadTypeName property contains the payload type for the queue. PayloadType can be 'RAW' or the name of an object type. PayloadType value is used in Dequeue method. If value of the property is not set then Dequeue method will get payload type from the server before dequeuing message.

© 1997-2021
Devart. All Rights Reserved.

5.17.1.3.2.7 QueueName Property

Used to set the name of the Oracle queue to operation.

**Class**

**T0raQueue**

**Syntax**

```
property QueueName: string;
```

**Remarks**

Use the QueueName property to set the name of the Oracle queue to operation.

© 1997-2021
Devart. All Rights Reserved.

5.17.1.3.2.8 Session Property

Used to specify the session through which a queue will be controlled.

**Class**

**T0raQueue**

**Syntax**

```
property Session: T0raSession;
```
Remarks

Use the Session property to specify the session through which a queue will be controlled.

See Also
- TOraSession

5.17.1.3.3 Methods

Methods of the TOraQueue class.

For a complete list of the TOraQueue class members, see the TOraQueue Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dequeue</td>
<td>Overloaded. Dequeues messages.</td>
</tr>
<tr>
<td>DequeueArray</td>
<td>Dequeues an array of messages.</td>
</tr>
<tr>
<td>Enqueue</td>
<td>Overloaded. Enqueues messages.</td>
</tr>
<tr>
<td>EnqueueArray</td>
<td>Enqueues an array of messages.</td>
</tr>
<tr>
<td>Listen</td>
<td>Overloaded. Listens to one or more queues on behalf of the list of agents.</td>
</tr>
</tbody>
</table>

See Also
- TOraQueue Class
- TOraQueue Class Members

5.17.1.3.3.1 Dequeue Method

Dequeues messages.

Class
**T0raQueue**

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dequeue(Payload: T0raObject; MessageProperties: TQueueMessageProperties; DequeueOptions: TDequeueOptions)</td>
<td>Dequeues a message as a T0raObject.</td>
</tr>
<tr>
<td>Dequeue(Message: TQueueMessage; DequeueOptions: TDequeueOptions)</td>
<td>Dequeues messages.</td>
</tr>
<tr>
<td>Dequeue(out Payload: TBytes; MessageProperties: TQueueMessageProperties; DequeueOptions: TDequeueOptions)</td>
<td>Dequeues a message as array of Byte.</td>
</tr>
<tr>
<td>Dequeue(out Payload: string; MessageProperties: TQueueMessageProperties; DequeueOptions: TDequeueOptions)</td>
<td>Dequeues string messages.</td>
</tr>
</tbody>
</table>

Dequeues a message as a T0raObject.

**Class**

**T0raQueue**

**Syntax**

```plaintext
function Dequeue(Payload: T0raObject; MessageProperties: TQueueMessageProperties = nil; DequeueOptions: TDequeueOptions = nil): TMessageId; overload;
```

**Parameters**

*Payload*
- Holds a message content as a T0raObject.

*MessageProperties*
- Holds the properties of a message that will be dequeued.

*DequeueOptions*
- Holds the options for dequeuing messages.

**Return Value**
message ID as string.

Remarks

Use one of Dequeue method overloads to dequeue messages. Use overloads with string or TBytes payload for queues with the RAW payload type. For queues with the object payload type pass TOraObject instance with appropriate object type to Dequeue method. If DequeueOptions parameter was not specified, the TOraQueue.DequeueOptions property of TOraQueue component will be used. MessageProperties parameter will be filled with the properties of the dequeued message.

In Direct mode only queues with RAW payload are supported.

Dequeue us a message as array of Byte.

Class

TOraQueue

Syntax

function Dequeue(Message: TQueueMessage; DequeueOptions: TDequeueOptions = nil): TMessageId; overload;

Parameters

Message

Holds the message content.

DequeueOptions

Holds the options for dequeueing messages.

Return Value

message ID as string.

Class

TOraQueue
Syntax

```pascal
function Dequeue(out Payload: TBytes; MessageProperties: TQueueMessageProperties = nil; DequeueOptions: TDequeueOptions = nil): TMessageId; overload;
```

Parameters

- **Payload**: Holds the message content as array of Byte.
- **MessageProperties**: Holds the properties of a message that will be dequeued.
- **DequeueOptions**: Holds the options for dequeueing messages.

Return Value

- message ID as string.

Dequeues string messages.

Class

**T0raQueue**

Syntax

```pascal
function Dequeue(out Payload: string; MessageProperties: TQueueMessageProperties = nil; DequeueOptions: TDequeueOptions = nil): TMessageId; overload;
```

Parameters

- **Payload**: Holds a message as string.
- **MessageProperties**: Holds the properties of a message that will be dequeued.
- **DequeueOptions**: Holds the options for dequeueing messages.

Return Value

- message ID as string.

See Also
5.17.1.3.3.2 DequeueArray Method

Dequeues an array of messages.

Class
TOraQueue

Syntax

```pascal
function DequeueArray(const MessageArray: array of TQueueMessage; out DequeuedSize: integer; DequeueOptions: TDequeueOptions = nil): TMessageIds;
```

Parameters

- **MessageArray**
  An array of messages to dequeue.
- **DequeuedSize**
  Returns the number of dequeued messages.
- **DequeueOptions**
  Holds the options for dequeuing messages.

Return Value

an array of message IDs.

Remarks

Use DequeueArray method to dequeue an array of messages. If DequeueOptions parameter was not specified, DequeueOptions property of TOraQueue component will be used. DequeuedSize returns the number of the dequeued messages.
See Also

- OnMessage
- TDequeueOptions
- Dequeue
- EnqueueArray
- TQueueMessage
- TQueueMessageProperties
- TOraObject

© 1997-2021
Devart. All Rights Reserved.

5.17.1.3.3.3 Enqueue Method

Enqueues messages.

Class

TOraQueue

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Enqueue(Payload: TOraObject;</code></td>
<td>Enqueues a message as TOraObject.</td>
</tr>
<tr>
<td><code>MessageProperties: TQueueMessageProperties;</code></td>
<td></td>
</tr>
<tr>
<td><code>EnqueueOptions: TEnqueueOptions)</code></td>
<td></td>
</tr>
<tr>
<td><code>Enqueue(Message: TQueueMessage;</code></td>
<td>Enqueues messages.</td>
</tr>
<tr>
<td><code>EnqueueOptions: TEnqueueOptions)</code></td>
<td></td>
</tr>
<tr>
<td><code>Enqueue(const Payload: TBytes;</code></td>
<td>Enqueues a message as array of Byte.</td>
</tr>
<tr>
<td><code>MessageProperties: TQueueMessageProperties;</code></td>
<td></td>
</tr>
<tr>
<td><code>EnqueueOptions: TEnqueueOptions)</code></td>
<td></td>
</tr>
<tr>
<td><code>Enqueue(const Payload: string;</code></td>
<td>Enqueues string messages.</td>
</tr>
<tr>
<td><code>MessageProperties: TQueueMessageProperties;</code></td>
<td></td>
</tr>
<tr>
<td><code>EnqueueOptions: TEnqueueOptions)</code></td>
<td></td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.
Enqueue a message as TOraObject.

Class

TOraQueue

Syntax

function Enqueue(Payload: TOraObject; MessageProperties: TQueueMessageProperties = nil; EnqueueOptions: TEnqueueOptions = nil): TMessageId;

overload;

Parameters

Payload

a message content as a TOraObject.

MessageProperties

Holds the properties of the message which will be enqueued.

EnqueueOptions

Holds the options for enqueuing messages.

Return Value

a message ID.

Remarks

Use one of the Enqueue method overloads to enqueue messages. Use overloads with string or TBytes payload for queues with the RAW payload type. For queues with object payload type pass the TOraObject instance with appropriate object type to the Enqueue method. If EnqueueOptions or MessageProperties parameters were not specified, the TOraQueue.EnqueueOptions and TOraQueue.EnqueueMessageProperties properties of TOraQueue component will be used.

In Direct mode only queues with RAW payload are supported.

© 1997-2021

Devart. All Rights Reserved.

Request Support DAC Forum Provide Feedback

Enqueues messages.

Class

TOraQueue

Syntax
**function** Enqueue(Message: TQueueMessage; EnqueueOptions: TEnqueueOptions = nil): TMessageId; overload;

**Parameters**

*Message*
- Holds the message content.

*EnqueueOptions*
- Holds the options for enqueuing messages.

**Return Value**
- a message ID.

Enqueues a message as array of Byte.

**Class**

T0raQueue

**Syntax**

**function** Enqueue(const Payload: TBytes; MessageProperties: TQueueMessageProperties = nil; EnqueueOptions: TEnqueueOptions = nil): TMessageId; overload;

**Parameters**

*Payload*
- Holds the message content as array of Byte.

*MessageProperties*
- Holds the properties of the message which will be enqueued.

*EnqueueOptions*
- Holds the options for enqueuing messages.

**Return Value**
- a message ID.

Enqueues string messages.

**Class**
### T0raQueue

#### Syntax

```delphi
function Enqueue(const Payload: string; MessageProperties: TQueueMessageProperties = nil; EnqueueOptions: TEnqueueOptions = nil): TMessageId; overload;
```

#### Parameters

- **Payload**
  - Holds a message as a string.
- **MessageProperties**
  - Holds the properties of the message which will be enqueued.
- **EnqueueOptions**
  - Holds the options for enqueuing messages.

#### Return Value

- A message ID.

#### See Also

- T0raQueue.OnMessage
- T0raQueue.EnqueueOptions
- T0raQueue.EnqueueMessageProperties
- T0raQueue.Dequeue
- TQueueMessage
- TEnqueueOptions
- TQueueMessageProperties
- T0raObject

---

5.17.1.3.4 EnqueueArray Method

Enqueues an array of messages.

### Class

**T0raQueue**

#### Syntax

```delphi
function EnqueueArray(const MessageArray: array of TQueueMessage);
```
EnqueueOptions: \text{TEnqueueOptions} = \text{nil}): \text{TMessageIds};

\textbf{Parameters}

\textit{MessageArray}
- Holds an array of messages to enqueue.

\textit{EnqueueOptions}
- Holds the options for enqueueing messages.

\textbf{Return Value}
- an array of message IDs.

\textbf{Remarks}

Use the EnqueueArray method to enqueue an array of messages. If the EnqueueOptions parameter was not specified, the \textit{EnqueueOptions} property of \textit{TOraQueue} will be used.

DequeuedSize returns the number of dequeued messages.

© 1997-2021 Devart. All Rights Reserved.

5.17.1.3.3.5 Listen Method

LISTENS to one or more queues on behalf of the list of agents.

\textbf{Class}
\textit{TOraQueue}

\textbf{Overload List}

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Listen(Agents: TQueueAgents; Agent: TQueueAgent; WaitTimeout: integer)}</td>
<td>Listens to one or more queues on behalf of the list of agents.</td>
</tr>
<tr>
<td>\text{Listen(Agents: TQueueAgents; ListDeliveryMode: TQueueDeliveryMode; Agent: TQueueAgent; \textbf{var} MessageDeliveryMode: TQueueDeliveryMode; WaitTimeout: integer)}</td>
<td>Listens to one or more queues on behalf of the list of agents.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
Listens to one or more queues on behalf of the list of agents.

Class

T0raQueue

Syntax

procedure Listen(Agents: TQueueAgents; Agent: TQueueAgent;
WaitTimeout: integer = AQ_FOREVER); overload;

Parameters

Agents
Holds the list of the agents.

Agent
Holds the agent name when monitoring multiconsumer queues.

WaitTimeout
Holds the amount of time to wait if there are no messages found.

Remarks

Call the Listen method to listen to one or more queues on behalf of the list of agents. You specify the queue to be monitored in the address field of each agent listed. You must also specify the name of the agent when monitoring multiconsumer queues. For single-consumer queues, an agent name must not be specified. Only local queues are supported as addresses.

This is a blocking call that returns when there is a message ready for consumption for an agent in the list. If there are messages for more than one agent, only the first agent listed is returned. If there are no messages found when the wait time expires, an error is raised.

A successful return from the LISTEN call is only an indication that there is a message for one of the listed agents in one of the specified queues. The interested agent must still dequeue the relevant message.

Second overload with DeliveryMode is supported starting with Oracle 10. DeliveryMode possible values are described in TDequeueOptions.DeliveryMode topic.

See Also

- TDequeueOptions.DeliveryMode
- TQueueAgent
- TQueueAgents

© 1997-2021 Request Support DAC Forum Provide Feedback
Devart. All Rights Reserved.

Listens to one or more queues on behalf of the list of agents.

Class

**TOraQueue**

Syntax

```pascal
procedure Listen(Agents: TQueueAgents; ListDeliveryMode: TQueueDeliveryMode; Agent: TQueueAgent; var MessageDeliveryMode: TQueueDeliveryMode; WaitTimeout: integer = AQ_FOREVER); overload;
```

**Parameters**

*Agents*
- Holds the list of the agents.

*ListDeliveryMode*
- Holds the message type (persistent, buffered messages or both).

*Agent*
- Holds the agent name when monitoring multiconsumer queues.

*MessageDeliveryMode*
- Holds the message type along with the queue and consumer for whom there is a message.

*WaitTimeout*
- Holds the amount of time to wait if there are no messages found.

© 1997-2021 Devart. All Rights Reserved.

5.17.1.3.4 Events

Events of the **TOraQueue** class.

For a complete list of the **TOraQueue** class members, see the [TOraQueue Members](#) topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnMessage</strong></td>
<td>Occurs when new messages are enqueued.</td>
</tr>
</tbody>
</table>

See Also
5.17.1.3.4.1  OnMessage Event

Occurs when new messages are enqueued.

Class

**TOraQueue**

Syntax

```.delphi
property OnMessage: TQueueMessageEvent;
```

Remarks

Use OnMessage event handler to get notifications when new messages are enqueued. Set the `AsyncNotification` property to True to get OnMessage events.

To get the payload for the message assign `MessageId` value passed to the event handler to the `TDequeueOptions.MessageId` property in `DequeueOptions` and then call `TOraQueue.Dequeue`.

See Also

- `AsyncNotification`
- `TQueueMessageProperties`
- `DequeueOptions`
- `TDequeueOptions.MessageId`

5.17.1.4  TOraQueueAdmin Class

A component for managing queues in a database.

For a list of all members of this type, see `TOraQueueAdmin` members.
Syntax

TOraQueueAdmin = class(TComponent);

Remarks

Use the TOraQueueAdmin component to manage queues in a database.

Database user must have AQ_ADMINISTRATOR_ROLE to work with TOraQueueAdmin component.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>Used to get or set the user-specified description of the queue.</td>
</tr>
<tr>
<td>MaxRetries</td>
<td>Used to specify the number of attempts to dequeue message.</td>
</tr>
<tr>
<td>MultipleConsumers</td>
<td>Enables the possibility of using definite table for queues that can have multiple consumers for each message.</td>
</tr>
<tr>
<td>QueueName</td>
<td>Used to set the name of the Oracle queue to operate.</td>
</tr>
<tr>
<td>QueueTableName</td>
<td>Used to set or get the queue table for the queue.</td>
</tr>
<tr>
<td>QueueType</td>
<td>Indicates whether the queue being created is an exception queue or a normal queue.</td>
</tr>
<tr>
<td>RetentionTime</td>
<td>Used to set the time in seconds for which a message remains in the</td>
</tr>
</tbody>
</table>
queue after being dequeued.

**RetryDelay**
Used to get or set delay time in seconds before the message that failed to be dequeued, will be scheduled to processing again.

**Session**
Used to specify the session through which queues will be managed.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSubscriber</td>
<td>Adds a subscriber to the queue.</td>
</tr>
<tr>
<td>AlterComment</td>
<td>Alters the user-defined queue description.</td>
</tr>
<tr>
<td>AlterMaxRetries</td>
<td>Alters the number of attempts to dequeue a message.</td>
</tr>
<tr>
<td>AlterPropagationSchedule</td>
<td>Alters parameters for a propagation schedule.</td>
</tr>
<tr>
<td>AlterQueue</td>
<td>Alters queue properties.</td>
</tr>
<tr>
<td>AlterRetentionTime</td>
<td>Alters the time in seconds during which a message remains in the queue after being dequeued.</td>
</tr>
<tr>
<td>AlterRetryDelay</td>
<td>Alters the delay time in seconds before the message, which failed to be dequeued, will be scheduled to processing again.</td>
</tr>
<tr>
<td>AlterSubscriber</td>
<td>Alters the rule and transformation properties of a queue subscriber.</td>
</tr>
<tr>
<td>CreateQueue</td>
<td>Creates a queue with the name, specified by the TOraQueueAdmin.QueueName property.</td>
</tr>
<tr>
<td>DisablePropagationSchedule</td>
<td>Disables a propagation schedule.</td>
</tr>
<tr>
<td>DropQueue</td>
<td>Drops the queue specified</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>EnablePropagationSchedule</strong></td>
<td>Enables a previously disabled propagation schedule.</td>
</tr>
<tr>
<td><strong>GetSubscribers</strong></td>
<td>Provides the list of queue subscribers.</td>
</tr>
<tr>
<td><strong>GrantQueuePrivilege</strong></td>
<td>Grants queue privilege to the grantee.</td>
</tr>
<tr>
<td><strong>ReadQueueProperties</strong></td>
<td>Reads the information about a queue specified by the <code>TOraQueueAdmin.QueueName</code> property from the database to a <code>TOraQueueAdmin</code> component.</td>
</tr>
<tr>
<td><strong>RemoveSubscriber</strong></td>
<td>Removes a subscriber from the queue.</td>
</tr>
<tr>
<td><strong>RevokeQueuePrivilege</strong></td>
<td>Revokes queue privilege from a grantee.</td>
</tr>
<tr>
<td><strong>SchedulePropagation</strong></td>
<td>Schedules the propagation of messages from the queue to a destination identified by a specific database link.</td>
</tr>
<tr>
<td><strong>StartDequeue</strong></td>
<td>Enables dequeueing on a queue.</td>
</tr>
<tr>
<td><strong>StartEnqueue</strong></td>
<td>Enables enqueueing on a queue.</td>
</tr>
<tr>
<td><strong>StartQueue</strong></td>
<td>Enables enqueueing, dequeueing, or both on a queue.</td>
</tr>
<tr>
<td><strong>StopDequeue</strong></td>
<td>Disables dequeueing on a queue.</td>
</tr>
<tr>
<td><strong>StopEnqueue</strong></td>
<td>Disables enqueueing on a queue.</td>
</tr>
<tr>
<td><strong>StopQueue</strong></td>
<td>Stops enqueueing, dequeueing, or both on a queue.</td>
</tr>
<tr>
<td><strong>UnschedulePropagation</strong></td>
<td>Unschedules previously scheduled propagation of messages from the queue to the specified destination.</td>
</tr>
<tr>
<td><strong>VerifyQueueTypes</strong></td>
<td>Verifies that the current...</td>
</tr>
</tbody>
</table>
Properties of the **TOraQueueAdmin** class.

For a complete list of the **TOraQueueAdmin** class members, see the **TOraQueueAdmin Members** topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comment</strong></td>
<td>Used to get or set the user-specified description of the queue.</td>
</tr>
<tr>
<td><strong>MaxRetries</strong></td>
<td>Used to specify the number of attempts to dequeue message.</td>
</tr>
<tr>
<td><strong>MultipleConsumers</strong></td>
<td>Enables the possibility of using definite table for queues that can have multiple consumers for each message.</td>
</tr>
<tr>
<td><strong>QueueName</strong></td>
<td>Used to set the name of the Oracle queue to operate.</td>
</tr>
<tr>
<td><strong>QueueTableName</strong></td>
<td>Used to set or get the queue table for the queue.</td>
</tr>
<tr>
<td><strong>QueueType</strong></td>
<td>Indicates whether the queue being created is an exception queue or a normal queue.</td>
</tr>
<tr>
<td><strong>RetentionTime</strong></td>
<td>Used to set the time in seconds for which a message remains in the queue after being dequeued.</td>
</tr>
<tr>
<td><strong>RetryDelay</strong></td>
<td>Used to get or set delay time in seconds before the message that failed to be dequeued, will be scheduled</td>
</tr>
</tbody>
</table>
### Comment Property

**Used to get or set the user-specified description of the queue.**

**Class**

**T0raQueueAdmin**

**Syntax**

```plaintext
property Comment: string;
```

**Remarks**

Call the Comment property to get or set the user-specified description of the queue.

### MaxRetries Property

**Used to specify the number of attempts to dequeue message.**

**Class**

**T0raQueueAdmin**

**Syntax**

```plaintext
property MaxRetries: integer default AQ_NOT_DEFINED;
```

**Remarks**
Use the MaxRetries property to specify the number of attempts to dequeue message.

See Also
- RetryDelay

### 5.17.1.4.2.3 MultipleConsumers Property

Enables the possibility of using definite table for queues that can have multiple consumers for each message.

**Class**

`TOraQueueAdmin`

**Syntax**

```
property MultipleConsumers: boolean default False;
```

**Remarks**

Should be set to True to use definite table for queues that can have multiple consumers for each message. False is the default value and means that queues, based on this table can have only one consumer for each message.

### 5.17.1.4.2.4 QueueName Property

Used to set the name of the Oracle queue to operate.

**Class**

`TOraQueueAdmin`

**Syntax**

```
property QueueName: string;
```

**Remarks**

Use the QueueName property to set the name of the Oracle queue to operate.
5.17.1.4.2.5 QueueTableName Property

Used to set or get the queue table for the queue.

Class

T0raQueueAdmin

Syntax

```text
property QueueTableName: string;
```

Remarks

Use the QueueTableName property to set or get the queue table for the queue.

5.17.1.4.2.6 QueueType Property

Indicates whether the queue being created is an exception queue or a normal queue.

Class

T0raQueueAdmin

Syntax

```text
property QueueType: TQueueType default qtNormalQueue;
```

Remarks

Use the QueueType property to specify whether the queue being created is an exception queue or a normal queue. Only the dequeue operation is allowed in the exception queue. The default value is qtNormalQueue.
5.17.1.4.2.7 RetentionTime Property

Used to set the time in seconds for which a message remains in the queue after being dequeued.

Class
T0raQueueAdmin

Syntax

```property RetentionTime: integer default 0;```

Remarks
Use the RetentionTime property to set the time in seconds for which a message remains in the queue after being dequeued. The default value is 0.

5.17.1.4.2.8 RetryDelay Property

Used to get or set delay time in seconds before the message that failed to be dequeued, will be scheduled to processing again.

Class
T0raQueueAdmin

Syntax

```property RetryDelay: integer default 0;```

Remarks
Use the RetryDelay property to get or set delay time in seconds before the message that failed to be dequeued, will be scheduled to processing again. The default value is 0. Useless if MaxRetries property is set to 0.
5.17.1.4.2.9 Session Property

Used to specify the session through which queues will be managed.

Class

TOraQueueAdmin

Syntax

```property`` Session: TOraSession;
```

Remarks

Use the Session property to specify the session through which queues will be managed.

See Also

• TOraSession

© 1997-2021

Devart. All Rights Reserved.

5.17.1.4.3 Methods

Methods of the TOraQueueAdmin class.

For a complete list of the TOraQueueAdmin class members, see the TOraQueueAdmin Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSubscriber</td>
<td>Adds a subscriber to the queue.</td>
</tr>
<tr>
<td>AlterComment</td>
<td>Alters the user-defined queue description.</td>
</tr>
<tr>
<td>AlterMaxRetries</td>
<td>Alters the number of attempts to dequeue a message.</td>
</tr>
<tr>
<td>AlterPropagationSchedule</td>
<td>Alters parameters for a propagation schedule.</td>
</tr>
<tr>
<td>AlterQueue</td>
<td>Alters queue properties.</td>
</tr>
<tr>
<td>AlterRetentionTime</td>
<td>Alters the time in seconds during which a message</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AlterRetryDelay</td>
<td>Alters the delay time in seconds before the message, which failed to be dequeued, will be scheduled to processing again.</td>
</tr>
<tr>
<td>AlterSubscriber</td>
<td>Alters the rule and transformation properties of a queue subscriber.</td>
</tr>
<tr>
<td>CreateQueue</td>
<td>Creates a queue with the name, specified by the TOraQueueAdmin.QueueName property.</td>
</tr>
<tr>
<td>DisablePropagationSchedule</td>
<td>Disables a propagation schedule.</td>
</tr>
<tr>
<td>DropQueue</td>
<td>Drops the queue specified by the TOraQueueAdmin.QueueName property.</td>
</tr>
<tr>
<td>EnablePropagationSchedule</td>
<td>Enables a previously disabled propagation schedule.</td>
</tr>
<tr>
<td>GetSubscribers</td>
<td>Provides the list of queue subscribers.</td>
</tr>
<tr>
<td>GrantQueuePrivilege</td>
<td>Grants queue privilege to the grantee.</td>
</tr>
<tr>
<td>ReadQueueProperties</td>
<td>Reads the information about a queue specified by the TOraQueueAdmin.QueueName property from the database to a TOraQueueAdmin component.</td>
</tr>
<tr>
<td>RemoveSubscriber</td>
<td>Removes a subscriber from the queue.</td>
</tr>
<tr>
<td>RevokeQueuePrivilege</td>
<td>Revokes queue privilege from a grantee.</td>
</tr>
<tr>
<td>SchedulePropagation</td>
<td>Schedules the propagation of messages from the queue to a destination identified by a specific database link.</td>
</tr>
<tr>
<td>StartDequeue</td>
<td>Enables dequeuing on a queue.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StartEnqueue</td>
<td>Enables enqueueing on a queue.</td>
</tr>
<tr>
<td>StartQueue</td>
<td>Enables enqueueing, dequeueing, or both on a queue.</td>
</tr>
<tr>
<td>StopDequeue</td>
<td>Disables dequeueing on a queue.</td>
</tr>
<tr>
<td>StopEnqueue</td>
<td>Disables enqueueing on a queue.</td>
</tr>
<tr>
<td>StopQueue</td>
<td>Stops enqueueing, dequeueing, or both on a queue.</td>
</tr>
<tr>
<td>UnschedulePropagation</td>
<td>Unschedules previously scheduled propagation of messages from the queue to</td>
</tr>
<tr>
<td></td>
<td>the specified destination.</td>
</tr>
<tr>
<td>VerifyQueueTypes</td>
<td>Verifies that the current queue and destination queue have the same type.</td>
</tr>
</tbody>
</table>

See Also
- TOraQueueAdmin Class
- TOraQueueAdmin Class Members

5.17.1.4.3.1 AddSubscriber Method

Adds a subscriber to the queue.

Class
TOraQueueAdmin

Syntax

```pascal
procedure AddSubscriber(Subscriber: TQueueAgent; const Rule: string = ''; const Transformation: string = ''; QueueToQueue: boolean = False; DeliveryMode: TQueueDeliveryMode = qdmPersistent);
```

Parameters

- **Subscriber**
  - Holds an agent on whose behalf the subscription is being defined.
Rule
Holds a conditional expression based on the message properties, the message data properties, and PL/SQL functions.

Transformation
Holds the transformation that will be applied when this subscriber dequeues the message.

QueueToQueue
Is True, if propagation is from queue-to-queue.

DeliveryMode
Holds the delivery mode of the messages the subscriber is interested in. See TDequeueOptions.DeliveryMode for more information.

Remarks
Call the AddSubscriber method to add a subscriber to the queue.

See Also
- AlterSubscriber
- RemoveSubscriber
- TDequeueOptions.DeliveryMode
- TQueueAgent

© 1997-2021
Devart. All Rights Reserved.

5.17.1.4.3.2 AlterComment Method

Alters the user-defined queue description.

Class
T0raQueueAdmin

Syntax

```
procedure AlterComment(const Comment: string);
```

Parameters

Comment
Holds the user-defined queue description.

Remarks
Call the AlterComment method to alter the user-defined queue description.
5.17.1.4.3.3 AlterMaxRetries Method

Alters the number of attempts to dequeue a message.

Class

TOraQueueAdmin

Syntax

```procedure AlterMaxRetries(MaxRetries: integer);```

Parameters

- **MaxRetries**
  
  Holds the number of attempts that can be taken to dequeue a message.

Remarks

Call the AlterMaxRetries method to alter the number of attempts to dequeue a message.

See Also

- AlterQueue
- MaxRetries

5.17.1.4.3.4 AlterPropagationSchedule Method

Alters parameters for a propagation schedule.

Class

TOraQueueAdmin

Syntax
procedure AlterPropagationSchedule(const Destination: string; Duration: integer; const NextTime: string; Latency: integer; const DestinationQueue: string = '');

Parameters

Destination
Holds the destination database link.

Duration
Holds the duration of the propagation window in seconds (NULL value means the propagation window is unscheduled forever or until the propagation).

NextTime
Holds the date function to compute the start of the next propagation window from the end of the current window.

Latency
Holds the maximum wait time in seconds in the propagation window for a message to be propagated after it is enqueued.

DestinationQueue
Holds the name of the destination queue. This parameter is supported starting with Oracle 10.

Remarks

Call the AlterPropagationSchedule method to alter parameters for a propagation schedule.

See Also

- SchedulePropagation
- DisablePropagationSchedule
- UnschedulePropagation
AQ_NOT_DEFINED; RetentionTime: integer = AQ_NOT_DEFINED; const 
Comment: string = '');

Parameters

MaxRetries
Holds the number of attempts to dequeue message.

RetryDelay
Holds the delay time before the message that failed to be dequeued, will be scheduled to processing again.

RetentionTime
Holds the time for which a message remains in the queue after being dequeued.

Comment
Holds the user-specified description of the queue.

Remarks

Call the AlterQueue method to alter queue properties.

See Also
• MaxRetries
• RetryDelay
• RetentionTime
• Comment
• AlterMaxRetries
• AlterRetryDelay
• AlterRetentionTime
• AlterComment

© 1997-2021
Devart. All Rights Reserved.

5.17.1.4.3.6 AlterRetentionTime Method

Alters the time in seconds during which a message remains in the queue after being dequeued.

Class
TOraQueueAdmin

Syntax
**procedure** AlterRetentionTime(RetentionTime: integer);

**Parameters**

*RetentionTime*

Holds the time for which a message remains in the queue after being dequeued.

**Remarks**

Call the AlterRetentionTime method to alter the time in seconds during which a message remains in the queue after being dequeued.

**See Also**

- [AlterQueue](#)
- [RetentionTime](#)

---

Alters the delay time in seconds before the message, which failed to be dequeued, will be scheduled to processing again.

**Class**

*T0raQueueAdmin*

**Syntax**

```
procedure AlterRetryDelay(RetryDelay: integer);
```

**Parameters**

*RetryDelay*

Holds the delay time before the message that failed to be dequeued, will be scheduled to processing again.

**Remarks**

Call the AlterRetryDelay method to alter delay time in seconds before the message, which failed to be dequeued, will be scheduled to processing again.

**See Also**

- [AlterQueue](#)
- [RetryDelay](#)
5.17.1.4.3.8 AlterSubscriber Method

Alters the rule and transformation properties of a queue subscriber.

Class

T0raQueueAdmin

Syntax

```
procedure AlterSubscriber(Subscriber: TQueueAgent; const
Rule: string; const Transformation: string = '');
```

Parameters

- **Subscriber**
  
  Holds an agent on whose behalf the subscription is being defined.

- **Rule**

  Holds a conditional expression based on the message properties, the message data properties, and PL/SQL functions.

- **Transformation**

  Holds the transformation that will be applied when this subscriber dequeues the message.

Remarks

Call the AlterSubscriber method to alter the rule and transformation properties of a queue subscriber.

See Also

- [AddSubscriber](#)
- [RemoveSubscriber](#)
- [TQueueAgent](#)

5.17.1.4.3.9 CreateQueue Method

Creates a queue with the name, specified by the QueueName property.

Class
T0raQueueAdmin

Syntax

```procedure CreateQueue(NonPersistent: boolean = False);```

Parameters

**NonPersistent**

Holds True, if the created queue should be persistent. False otherwise.

Remarks

Call the CreateQueue method to create a queue with the name, specified by `QueueName` property. When creating a persistent queue, properties

- `QueueTableName`
- `QueueType`,
- `MaxRetries`,
- `RetryDelay`,
- `RetentionTime`,
- `Comment`

will be used as parameters for the new queue.

When creating a non-persistent queue, T0raQueueAdmin component uses properties `MultipleConsumers` and `Comment`.

Disables a propagation schedule.

Class

T0raQueueAdmin

Syntax

```procedure DisablePropagationSchedule(const Destination: string; const DestinationQueue: string = '');```

Parameters

**Destination**

Holds the destination database link.
**DestinationQueue**

Holds the name of the destination queue.

**Remarks**

Call the DisablePropagationSchedule method to disable a propagation schedule. The DestinationQueue parameter is supported starting with Oracle 10.

**See Also**

- [EnablePropagationSchedule](#)
- [SchedulePropagation](#)
- [UnschedulePropagation](#)
- [AlterPropagationSchedule](#)

---

### DropQueue Method

Drops the queue specified by the QueueName property.

**Class**

**T0raQueueAdmin**

**Syntax**

```pascal
procedure DropQueue;
```

**Remarks**

Call the DropQueue method to drop the queue specified by the QueueName property.

**See Also**

- [QueueName](#)

---

### EnablePropagationSchedule Method

Enables a previously disabled propagation schedule.

**Class**
### TOraQueueAdmin

#### Syntax

```delphi
procedure EnablePropagationSchedule(const Destination: string;
const DestinationQueue: string = '');
```

#### Parameters

- **Destination**
  - Holds the destination database link.

- **DestinationQueue**
  - Holds the name of the destination queue.

#### Remarks

Use the `EnablePropagationSchedule` method to enable a previously disabled propagation schedule. The `DestinationQueue` parameter is supported starting with Oracle 10.

#### See Also

- [DisablePropagationSchedule](#)
- [SchedulePropagation](#)
- [AlterPropagationSchedule](#)

---

### GetSubscribers Method

Provides the list of queue subscribers.

#### Class

`TOraQueueAdmin`

#### Syntax

```delphi
procedure GetSubscribers(Subscribers: TQueueAgents);
```

#### Parameters

- **Subscribers**
  - Holds a list of TOraAgent objects.

#### Remarks
Call the GetSubscribers method to get queue subscribers.

See Also
- AddSubscriber
- RemoveSubscriber
- TQueueAgents

Class
T0raQueueAdmin

Syntax

```pascal
procedure GrantQueuePrivilege(Privilege: TQueuePrivilege; const
Grantee: string; GrantOption: boolean = False);
```

Parameters

- **Privilege**
  - Holds the kind of privilege.
- **Grantee**
  - Holds the grantee name.
- **GrantOption**
  - True, if the privilege will be granted with GRANT OPTION. False otherwise.

Remarks

Use the GrantQueuePrivilege method to grant queue privilege to the grantee. Privilege can be qpEnqueue (ENQUEUE privilege), qpDequeue (DEQUEUE privilege) or qpAll (both ENQUEUE and DEQUEUE privileges). If GrantOption parameter is set to True, the privilege will be granted with GRANT OPTION.

See Also
- RevokeQueuePrivilege
5.17.1.4.3.15 ReadQueueProperties Method

Reads the information about a queue specified by the QueueName property from the database to a TOraQueueAdmin component.

Class

TOraQueueAdmin

Syntax

procedure ReadQueueProperties;

Remarks

Call the ReadQueueProperties method to read the information about a queue specified by the QueueName property from the database to a TOraQueueAdmin component. After calling this method use properties

- QueueTableName
- QueueType,
- MaxRetries,
- RetryDelay,
- RetentionTime,
- Comment

to get queue parameters.

© 1997-2021
Devart. All Rights Reserved.

5.17.1.4.3.16 RemoveSubscriber Method

Removes a subscriber from the queue.

Class

TOraQueueAdmin

Syntax

procedure RemoveSubscriber(Subscriber: TQueueAgent);

Parameters

Subscriber

Holds an agent on whose behalf the subscription is being defined.
Remarks

Call the RemoveSubscriber method to remove a subscriber from the queue. All references to the subscriber in existing messages will be also removed.

See Also

- GetSubscribers
- AddSubscriber
- TQueueAgent

RevokeQueuePrivilege Method

Revoke queue privilege from a grantee.

Class

T0raQueueAdmin

Syntax

```plaintext
procedure RevokeQueuePrivilege(Privilege: TQueuePrivilege; const Grantee: string);
```

Parameters

- **Privilege**
  - Holds the kind of privilege.
- **Grantee**
  - Holds the grantee name.

Remarks

Call the RevokeQueuePrivilege method to revoke queue privilege from a grantee. The privilege can be qpEnqueue (ENQUEUE privilege), qpDequeue (DEQUEUE privilege) or qpAll (both ENQUEUE and DEQUEUE privileges).

See Also

- GrantQueuePrivilege
5.17.1.4.3.18 SchedulePropagation Method

Schedules the propagation of messages from the queue to a destination identified by a specific database link.

Class

T0raQueueAdmin

Syntax

```delphi
procedure SchedulePropagation(const Destination: string; StartTime: TDateTime; Duration: integer; const NextTime: string; Latency: integer; const DestinationQueue: string = '');
```

Parameters

- **Destination**: Holds the destination database link.
- **StartTime**: Holds the initial start time for the propagation window for messages from the queue to the destination.
- **Duration**: Holds the duration of the propagation window in seconds.
- **NextTime**: Holds the date function to compute the start of the next propagation window from the end of the current window.
- **Latency**: Holds the maximum wait time in seconds in the propagation window for a message to be propagated after it is enqueued.
- **DestinationQueue**: Holds the name of the destination queue. DestinationQueue parameter is supported starting with Oracle 10.

Remarks

Call the SchedulePropagation metod to schedule the propagation of messages from the queue to a destination identified by a specific database link.

See Also

- UnschedulePropagation
- AlterPropagationSchedule
- EnablePropagationSchedule

© 1997-2021 Request Support DAC Forum Provide Feedback
5.17.1.4.3.19 StartDequeue Method

Enables dequeueing on a queue.

Class

TOraQueueAdmin

Syntax

procedure StartDequeue;

Remarks

Call the StartDequeue method to enable dequeueing on a queue.

See Also

• StartQueue
• StartEnqueue
• StopDequeue

5.17.1.4.3.20 StartEnqueue Method

Enables enqueueing on a queue.

Class

TOraQueueAdmin

Syntax

procedure StartEnqueue;

Remarks

Call the StartEnqueue method to enable enqueueing on a queue.

See Also

• StartQueue
5.17.1.4.3.21 StartQueue Method

Enables enqueueing, dequeueing, or both on a queue.

Class

TOraQueueAdmin

Syntax

procedure StartQueue(Enqueue: boolean = True; Dequeue: boolean = True);

Parameters

Enqueue
True, if enqueueing is enabled. False otherwise.

Dequeue
True, if dequeueing is enabled. False otherwise.

Remarks

Call the StopQueue method to enable enqueueing, dequeueing, or both on a queue.

See Also

• StartEnqueue
• StartDequeue
• StopQueue

5.17.1.4.3.22 StopDequeue Method

Disables dequeueing on a queue.

Class

TOraQueueAdmin
**Syntax**

```plaintext
procedure StopDequeue(wait: boolean = True);
```

**Parameters**

*Wait*

True, if waiting for the completion of the currently active queue transactions is enabled.

**Remarks**

Call the `StopDequeue` method to disable dequeueing on a queue. The `Wait` parameter specifies whether to wait for the completion of the currently active queue transactions.

**See Also**

- `StopQueue`
- `StopEnqueue`
- `StartDequeue`

---

**5.17.1.4.3.23 StopEnqueue Method**

Disables enqueueing on a queue.

**Class**

`TOraQueueAdmin`

**Syntax**

```plaintext
procedure StopEnqueue(wait: boolean = True);
```

**Parameters**

*Wait*

True, if wait for the completion of the currently active queue transactions is enabled.

**Remarks**

Call the `StopEnqueue` method to disable enqueueing on a queue. The `Wait` parameter specifies whether to wait for the completion of the currently active queue transactions.

**See Also**

- `StopQueue`

© 2021 Devart
5.17.1.4.3.24  StopQueue Method

Stops enqueueing, dequeueing, or both on a queue.

Class

TOraQueueAdmin

Syntax

procedure StopQueue(Enqueue: boolean = True; Dequeue: boolean = True; Wait: boolean = True);

Parameters

* Enqueue
  True, if enqueueing is stopped. False otherwise.

* Dequeue
  True, if dequeueing is stopped. False otherwise.

* Wait
  True, if waiting for the completion of the currently active queue transactions is enabled.

Remarks

Call the StopQueue method to stop enqueueing, dequeueing, or both on a queue. The Wait parameter specifies whether to wait for the completion of the currently active queue transactions.

See Also

* StopEnqueue
* StopDequeue
* StartQueue

© 1997-2021 Devart. All Rights Reserved.
5.17.1.4.3.25 UnscheudlePropagation Method

Unschedules previously scheduled propagation of messages from the queue to the specified destination.

Class

TOraQueueAdmin

Syntax

```plaintext
procedure UnscheudlePropagation(const Destination: string; const DestinationQueue: string = '');
```

Parameters

- **Destination**
  - Holds the destination database link.

- **DestinationQueue**
  - Holds the name of the destination queue. This parameter is supported starting with Oracle 10.

Remarks

Call the UnscheudlePropagation method to unschedule previously scheduled propagation of messages from the queue to the specified destination (database link). DestinationQueue parameter is supported starting with Oracle 10.

See Also

- SchedulePropagation
- AlterPropagationSchedule
- DisablePropagationSchedule

5.17.1.4.3.26 VerifyQueueTypes Method

Verifies that the current queue and destination queue have the same type.

Class

TOraQueueAdmin

Syntax
function VerifyQueueTypes(const DestQueueName: string; const Destination: string): boolean;

Parameters

DestQueueName
Holds the name of the destination queue.

Destination
Holds the destination database link.

Return Value
True, if the queues have the same type.

Remarks

Call the VerifyQueueTypes method to verify that the current queue and destination queue have the same type.

© 1997-2021 Devart. All Rights Reserved.

5.17.1.5 TOraQueueTable Class

A component managing queue tables in a database.

For a list of all members of this type, see TOraQueueTable members.

Unit
oraAQ

Syntax

TOraQueueTable = class(TComponent);

Remarks

Use the TOraQueueTable component to manage queue tables in a database.

Database user must have AQ_ADMINISTRATOR_ROLE to work with the TOraQueueTable component.

See Also
- TOraQueue, TOraQueueAdmin and TOraQueueTable Components
- TOraQueueAdmin
- TOraQueue
**TOraQueueTable** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comment</strong></td>
<td>Used to det or set a user-specified description of the queue table.</td>
</tr>
<tr>
<td><strong>Compatible</strong></td>
<td>Used to define the lowest database version the queue is compatible with.</td>
</tr>
<tr>
<td><strong>MessageGrouping</strong></td>
<td>Used to specify the message grouping behavior in the queues based on this table.</td>
</tr>
<tr>
<td><strong>MultipleConsumers</strong></td>
<td>Used if a definite table for queues that can have multiple consumers for each message is needed.</td>
</tr>
<tr>
<td><strong>PayloadTypeName</strong></td>
<td>Used to get or set the object type name for the queue payload.</td>
</tr>
<tr>
<td><strong>PrimaryInstance</strong></td>
<td>Used to set the primary owner of the queue table.</td>
</tr>
<tr>
<td><strong>QueueTableName</strong></td>
<td>Used to get or set the queue table name.</td>
</tr>
<tr>
<td><strong>SecondaryInstance</strong></td>
<td>Used to set the secondary owner of the queue table.</td>
</tr>
<tr>
<td><strong>Secure</strong></td>
<td>Used for queue tables that will be used for secure queues.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>Used to specify the session through which a queue table will be controlled.</td>
</tr>
<tr>
<td><strong>SortList</strong></td>
<td>Used to specify the columns to be used as the sort key.</td>
</tr>
<tr>
<td><strong>StorageClause</strong></td>
<td>Used to get or set the table storage clause.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlterComment</td>
<td>Alters queue table user-defined description.</td>
</tr>
<tr>
<td>AlterPrimaryInstance</td>
<td>Changes queue table primary_instance parameter.</td>
</tr>
<tr>
<td>AlterQueueTable</td>
<td>Changes existing queue table properties.</td>
</tr>
<tr>
<td>AlterSecondaryInstance</td>
<td>Changes the queue table secondary_instance parameter.</td>
</tr>
<tr>
<td>CreateQueueTable</td>
<td>Creates a queue table in a database.</td>
</tr>
<tr>
<td>DropQueueTable</td>
<td>Drops the queue table.</td>
</tr>
<tr>
<td>GrantSystemPrivilege</td>
<td>Grants system queue privilege.</td>
</tr>
<tr>
<td>MigrateQueueTable</td>
<td>Used to upgrade or downgrade a queue table to the desired Compatible parameter value.</td>
</tr>
<tr>
<td>PurgeQueueTable</td>
<td>Purges records from the queue table.</td>
</tr>
<tr>
<td>ReadQueueTableProperties</td>
<td>Reads information about a queue table specified by the QueueTableName property from the database to the TOraQueueTable component.</td>
</tr>
<tr>
<td>RevokeSystemPrivilege</td>
<td>Revokes system queue privilege.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

Request Support  DAC Forum  Provide Feedback

5.17.1.5.2 Properties

Properties of the **TOraQueueTable** class.

For a complete list of the **TOraQueueTable** class members, see the **TOraQueueTable Members** topic.
### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comment</strong></td>
<td>Used to set or set a user-specified description of the queue table.</td>
</tr>
<tr>
<td><strong>Compatible</strong></td>
<td>Used to define the lowest database version the queue is compatible with.</td>
</tr>
<tr>
<td><strong>MessageGrouping</strong></td>
<td>Used to specify the message grouping behavior in the queues based on this table.</td>
</tr>
<tr>
<td><strong>MultipleConsumers</strong></td>
<td>Used if a definite table for queues that can have multiple consumers for each message is needed.</td>
</tr>
<tr>
<td><strong>PayloadTypeName</strong></td>
<td>Used to get or set the object type name for the queue payload.</td>
</tr>
<tr>
<td><strong>PrimaryInstance</strong></td>
<td>Used to set the primary owner of the queue table.</td>
</tr>
<tr>
<td><strong>QueueTableName</strong></td>
<td>Used to get or set the queue table name.</td>
</tr>
<tr>
<td><strong>SecondaryInstance</strong></td>
<td>Used to set the secondary owner of the queue table.</td>
</tr>
<tr>
<td><strong>Secure</strong></td>
<td>Used for queue tables that will be used for secure queues.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>Used to specify the session through which a queue table will be controlled.</td>
</tr>
<tr>
<td><strong>SortList</strong></td>
<td>Used to specify the columns to be used as the sort key.</td>
</tr>
<tr>
<td><strong>StorageClause</strong></td>
<td>Used to get or set the table storage clause.</td>
</tr>
</tbody>
</table>

### See Also
- [TOraQueueTable Class](#)
- [TOraQueueTable Class Members](#)
5.17.1.5.2.1 Comment Property

Used to det or set a user-specified description of teh queue table.

Class

T0raQueueTable

Syntax

```property Comment: string;```

Remarks

Use The Comment property to get user-specified description of the queue table or to set it. It is used by the CreateQueueTable method.

See Also

- CreateQueueTable

5.17.1.5.2.2 Compatible Property

Used to define the lowest database version the queue is compatible with.

Class

T0raQueueTable

Syntax

```property Compatible: TQueueCompatible default qcDefault;```

Remarks

Use the Compatible property to determine the lowest database version with which the queue is compatible. The default value is qcDefault. The meaning of this value depends on the database compatible mode. For more information read the Oracle documentation. This property is used by the CreateQueueTable method.

See Also
5.17.1.5.2.3 MessageGrouping Property

Used to specify the message grouping behavior in the queues based on this table.

Class

TOraQueueTable

Syntax

```property```
MessageGrouping: TQueueMessageGrouping default qmgNone;
```

Remarks

Use the MessageGrouping property to determine message grouping behavior in the queues based on this table. mgNone means that each queue message is treated individually, mgTransactional means that messages, enqueued in one transaction, will belong to the same group. The default value is mgNone. This property is used by the CreateQueueTable method.

See Also

- CreateQueueTable

5.17.1.5.2.4 MultipleConsumers Property

Used if a definite table for queues that can have multiple consumers for each message is needed.

Class

TOraQueueTable

Syntax

```property```
MultipleConsumers: boolean default False;
```

Remarks
The MultipleConsumers property should be set to True to use a definite table for queues that can have multiple consumers for each message. False is the default value and means that queues, based on this table can have only one consumer for each message. This property is used by the `CreateQueueTable` method.

See Also
- `CreateQueueTable`

5.17.1.5.2.5 PayloadTypeName Property

Used to get or set the object type name for the queue payload.

Class
`TOraQueueTable`

Syntax

```delphi
property PayloadTypeName: string stored IsPayloadTypeNameStored;
```

Remarks
Use the PayloadTypeName property to get or set the object type name for the queue payload. This property is used by the `CreateQueueTable` method. This property is set to 'RAW' for RAW payload.

See Also
- `CreateQueueTable`

5.17.1.5.2.6 PrimaryInstance Property

Used to set the primary owner of the queue table.

Class
`TOraQueueTable`

Syntax
**property** PrimaryInstance: integer default 0;

Remarks
Use the PrimaryInstance property to get or set primary owner of the queue table. Queue monitor scheduling and propagation for the queues in the queue table are done in this instance. The default value for primary instance is 0, what means that queue monitor scheduling and propagation will be done in any available instance. This property is used by CreateQueueTable method.

See Also
- CreateQueueTable

© 1997-2021 Devart. All Rights Reserved.

5.17.1.5.2.7 QueueTableName Property

Used to get or set the queue table name.

Class
TOraQueueTable

Syntax

```
**property** QueueTableName: string;
```

Remarks
Use the QueueTableName property to get or set the queue table name. This property is used by the CreateQueueTable method.

See Also
- CreateQueueTable

© 1997-2021 Devart. All Rights Reserved.

5.17.1.5.2.8 SecondaryInstance Property

Used to set the secondary owner of the queue table.

Class
**T0raQueueTable**

**Syntax**

```plaintext
property SecondaryInstance: integer default 0;
```

**Remarks**

The queue table fails over the secondary instance if the primary instance is not available. The default value is 0. It means that the queue table will fail over any available instance. This property is used by `CreateQueueTable` method.

**See Also**

- `CreateQueueTable`

---

**Secure Property**

Used for queue tables that will be used for secure queues.

**Class**

**T0raQueueTable**

**Syntax**

```plaintext
property Secure: boolean default False;
```

**Remarks**

The Secure property must be True for queue tables that will be used for secure queues. It is used by the `CreateQueueTable` method.

**See Also**

- `CreateQueueTable`
5.17.1.5.2.10  Session Property

Used to specify the session through which a queue table will be controlled.

Class

T0raQueueTable

Syntax

```pascal
property Session: T0raSession;
```

Remarks

Use the Session property to specify the session through which a queue table will be controlled.

See Also

• T0raSession

5.17.1.5.2.11  SortList Property

Used to specify the columns to be used as the sort key.

Class

T0raQueueTable

Syntax

```pascal
property SortList: TQueueSortList default qslDefault;
```

Remarks

Use the SortList property to specify the columns to be used as the sort key in the ascending order.

See Also

• CreateQueueTable
5.17.1.5.2.12 StorageClause Property

Used to get or set the table storage clause.

Class

TOraQueueTable

Syntax

property StorageClause: string;

Remarks

Use the StorageClause property to get or set the table storage clause. It is used by the CreateQueueTable method. May contain any SQL code that can be in the CREATE TABLE storage clause.

See Also

• CreateQueueTable

5.17.1.5.3 Methods

Methods of the TOraQueueTable class.

For a complete list of the TOraQueueTable class members, see the TOraQueueTable Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlterComment</td>
<td>Alters queue table user-defined description.</td>
</tr>
<tr>
<td>AlterPrimaryInstance</td>
<td>Changes queue table primary_instance parameter.</td>
</tr>
<tr>
<td>AlterQueueTable</td>
<td>Changes existing queue table properties.</td>
</tr>
<tr>
<td>AlterSecondaryInstance</td>
<td>Changes the queue table secondary_instance parameter.</td>
</tr>
</tbody>
</table>
### CreateQueueTable
Creates a queue table in a database.

### DropQueueTable
Drops the queue table.

### GrantSystemPrivilege
Grants system queue privilege.

### MigrateQueueTable
Used to upgrade or downgrade a queue table to the desired Compatible parameter value.

### PurgeQueueTable
Purges records from the queue table.

### ReadQueueTableProperties
Reads information about a queue table specified by the QueueTableName property from the database to the TOraQueueTable component.

### RevokeSystemPrivilege
Revokes system queue privilege.

#### See Also
- **TOraQueueTable Class**
- **TOraQueueTable Class Members**

© 1997-2021 Devart. All Rights Reserved.

5.17.1.5.3.1 AlterComment Method

Alters queue table user-defined description.

**Class**

**TOraQueueTable**

**Syntax**

```pascal
procedure AlterComment(const Comment: string);
```

**Parameters**

- **Comment**
  Holds the user-defined description of the queue-table.
Remarks
Call the AlterComment method to alter queue table user-defined description.

See Also
- Comment
- AlterQueueTable

5.17.1.5.3.2 AlterPrimaryInstance Method

Changes queue table primary_instance parameter.

Class
TOraQueueTable

Syntax

```plaintext
procedure AlterPrimaryInstance(PrimaryInstance: integer);
```

Parameters

- `PrimaryInstance`
  Holds the primary owner of the queue table.

Remarks
Call the AlterPrimaryInstance method to alter queue table primary_instance parameter.

See Also
- PrimaryInstance
- AlterQueueTable

5.17.1.5.3.3 AlterQueueTable Method

Changes existing queue table properties.

Class
TOraQueueTable

Syntax

```delphi
procedure AlterQueueTable(const Comment: string; PrimaryInstance: integer = AQ_NOT_DEFINED; SecondaryInstance: integer = AQ_NOT_DEFINED);
```

Parameters

- **Comment**: Holds the user-defined description of the queue-table.
- **PrimaryInstance**: Holds the primary owner of the queue table.
- **SecondaryInstance**: Holds the secondary owner of the queue table.

Remarks

Call the `AlterQueueTable` method to alter existing queue table properties. You can alter the following properties: `Comment`, `PrimaryInstance`, `SecondaryInstance`.

See Also

- `Comment`
- `PrimaryInstance`
- `SecondaryInstance`
- `AlterComment`
- `AlterSecondaryInstance`
- `AlterPrimaryInstance`

© 1997-2021 Devart. All Rights Reserved.

5.17.1.5.3.4 AlterSecondaryInstance Method

Changes the queue table secondary_instance parameter.

Class

TOraQueueTable

Syntax

```delphi
procedure AlterSecondaryInstance(SecondaryInstance: integer);
```
Parameters

SecondaryInstance
Holds the secondary owner of the queue table.

Remarks

Call the AlterSecondaryInstance method to alter queue table secondary_instance parameter.

See Also

- SecondaryInstance
- AlterQueueTable

CreateQueueTable Method

Creates a queue table in a database.

Class

TOraQueueTable

Syntax

procedure CreateQueueTable;

Remarks

Call the CreateQueueTable method to create a queue table in a database. The QueueTableName property of the TOraQueueTable component will be used as a name for the queue table. Properties

- StorageClause,
- PayloadTypeName,
- SortList,
- MultipleConsumers,
- MessageGrouping,
- Comment,
- PrimaryInstance,
- SecondaryInstance,
- Compatible,
5.17.1.5.3.6 DropQueueTable Method

Drops the queue table.

Class

T0raQueueTable

Syntax

```plaintext
procedure DropQueueTable(Force: boolean = False);
```

Parameters

Force

If True, stops and drops all queues in the table automatically. Otherwise the table will not be dropped if there are any queues in it.

Remarks

Call the DropQueueTable method to drop the queue table. If the Force parameter is set to False, the table will not be dropped if there are any queues in the table. Otherwise all queues will be stopped and dropped automatically.

5.17.1.5.3.7 GrantSystemPrivilege Method

Grants system queue privilege.

Class

T0raQueueTable

Syntax

```plaintext
procedure GrantSystemPrivilege(Privilege: TQueueSystemPrivilege;
const Grantee: string; AdminOption: boolean = False);
```

Parameters
**Privilege**
Holds a system privilege to a grant.

**Grantee**
Holds the user name, a role, or a PUBLIC role.

**AdminOption**
Specifies whether the privilege will be granted with ADMIN OPTION.

**Remarks**
Call the GrantSystemPrivilege method to grant system queue privilege. Privilege parameter is a system privilege to grant. Can be qspEnqueueAny (ENQUEUE_ANY privilege), qspDequeueAny (DEQUEUE_ANY privilege), qspManageAny (MANAGE_ANY privilege). Grantee parameter is a grantee - can be a user, a role, or the PUBLIC role. AdminOption parameter specifies whether the privilege will be granted with ADMIN OPTION.

**See Also**
- RevokeSystemPrivilege

---

**5.17.1.5.3.8 MigrateQueueTable Method**

Used to upgrade or downgrade a queue table to the desired Compatible parameter value.

**Class**
**TOraQueueTable**

**Syntax**

```plaintext
procedure MigrateQueueTable(Compatible: TQueueCompatible);
```

**Parameters**

- **Compatible**
  Holds the lowest database version the queue is compatible with.

**Remarks**

Call the MigrateQueueTable method to upgrade or downgrade a queue table to the desired Compatible parameter value.

**See Also**
- Compatible
5.17.1.5.3.9 PurgeQueueTable Method

Purges records from the queue table.

Class

`TOraQueueTable`

Syntax

```plaintext
procedure PurgeQueueTable(const PurgeCondition: string; Block: boolean = False; DeliveryMode: TQueueDeliveryMode = qdmPersistent);
```

Parameters

- **PurgeCondition**
  Specifies the purge condition, which must be in the format of a SQL WHERE clause and is based on the columns of `aq$queue_table_name` view.

- **Block**
  Specifies whether to hold an exclusive lock on all the queues in the queue table while purging the queue table.

- **DeliveryMode**
  Specifies which type of messages should be purged. Possible values are described in `TDequeueOptions.DeliveryMode` topic

Remarks

Call the PurgeQueueTable method to purge records from the queue table.

See Also

- `TDequeueOptions.DeliveryMode`

5.17.1.5.3.10 ReadQueueTableProperties Method

Reads information about a queue table specified by the QueueTableName property from the database to the TOraQueueTable component.

Class
TOraQueueTable

Syntax

procedure ReadQueueTableProperties;

Remarks

Call the ReadQueueTableProperties method to read information about a queue table specified by the QueueTableName property from the database to the TOraQueueTable component. After calling this method use properties

- PayloadTypeName,
- SortList,
- MultipleConsumers,
- MessageGrouping,
- Comment,
- PrimaryInstance,
- SecondaryInstance,
- Compatible,
- Secure

to get queue table parameters.

© 1997-2021 Devart. All Rights Reserved.

5.17.1.5.3.11 RevokeSystemPrivilege Method

RevokeSystemPrivilege Method

Revoke system queue privilege.

Class

TOraQueueTable

Syntax

procedure RevokeSystemPrivilege(Privilege: TQueueSystemPrivilege;
const Grantee: string);

Parameters

Privilege

- Holds the system privilege to revoke.

Grantee
Holds the user name, a role, or a PUBLIC role.

Remarks

Call the RevokeSystemPrivilege method to revoke system queue privilege. Privilege parameter is a system privilege to revoke. Can be qspEnqueueAny (ENQUEUE_ANY privilege), qspDequeueAny (DEQUEUE_ANY privilege), qspManageAny (MANAGE_ANY privilege). Grantee parameter is a grantee - can be a user, a role, or the PUBLIC role.

5.17.1.6 TQueueAgent Class

A class representing a producer or a consumer of a queue message.

For a list of all members of this type, see TQueueAgent members.

Unit

OraAQ

Syntax

TQueueAgent = class(TCollectionItem);

Remarks

The TQueueAgent class represents a producer or a consumer of a queue message. Use the TQueueAgent class to set producers or consumers of queue messages, queue subscribers etc.

See Also

- TQueueAgents
- TQueueMessageProperties.SenderId
- TOraQueue.Listen
- TOraQueueAdmin.AddSubscriber
- TOraQueueAdmin.RemoveSubscriber
- TOraQueueAdmin.AlterSubscriber
5.17.1.6.1 Members

**TQueueAgent** class overview.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Used to get or set the recipient protocol-specific address.</td>
</tr>
<tr>
<td>Name</td>
<td>Used to get or set a name of the queue agent.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Used to set a protocol to interpret the address and propagate the message.</td>
</tr>
</tbody>
</table>

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>Used to get or set the recipient protocol-specific address.</td>
</tr>
<tr>
<td>Name</td>
<td>Used to get or set a name of the queue agent.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Used to set a protocol to interpret the address and propagate the message.</td>
</tr>
</tbody>
</table>

See Also
- **TQueueAgent Class**
- **TQueueAgent Class Members**
5.17.1.6.2.1 Address Property

Used to get or set the recipient protocol-specific address.

Class

TQueueAgent

Syntax

```property Address: string;```

Remarks

Use the Address property to get or set the recipient protocol-specific address. If the Protocol property is 0, then the address should have the [schema.]queue[@dblink] form.

5.17.1.6.2.2 Name Property

Used to get or set a name of the queue agent.

Class

TQueueAgent

Syntax

```property Name: string;```

Remarks

Use the Name property to get or set a name of the queue agent.

5.17.1.6.2.3 Protocol Property

Used to set a protocol to interpret the address and propagate the message.

Class
**TQueueAgent**

Syntax

```plaintext
property Protocol: integer default 0;
```

Remarks

Use the Protocol property to set a protocol to interpret the address and propagate the message.

© 1997-2021
Devart. All Rights Reserved.

5.17.1.7 **TQueueAgents Class**

A class holding a collection of the **TQueueAgent** objects.

For a list of all members of this type, see **TQueueAgents** members.

Unit

```plaintext
OraAQ
```

Syntax

```plaintext
TQueueAgents = class (TOwnedCollection);
```

Remarks

Each TQueueAgents holds a collection of the **TQueueAgent** objects. TQueueAgents maintains an index of the columns in its Items array. The Count property contains the number of columns in the collection.

See Also

- **TQueueAgent**
- **TQueueMessageProperties.RecipientList**
- **TOraQueue.Listen**
- **TOraQueueAdmin.GetSubscribers**

© 1997-2021
Devart. All Rights Reserved.
5.17.1.7.1 Members

**TQueueAgents** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access individual columns.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a TQueueAgent object to the collection.</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts a TQueueAgent object in the TQueueAgents collection to the required place.</td>
</tr>
</tbody>
</table>

Properties of the **TQueueAgents** class.

For a complete list of the **TQueueAgents** class members, see the **TQueueAgents Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access individual columns.</td>
</tr>
</tbody>
</table>

See Also
- **TQueueAgents Class**
- **TQueueAgents Class Members**

© 1997-2021 Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback
5.17.1.7.2.1 Items Property(Indexer)

Used to access individual columns.

Class

TQueueAgents

Syntax

```plaintext
property Items[Index: integer]: TQueueAgent; default;
```

Parameters

*Index*

Holds the index of a TQueueAgent object.

Remarks

Use the Items property to access individual columns. The value of the Index parameter corresponds to the Index property of TQueueAgent.

See Also

* TQueueAgent

© 1997-2021

Devart. All Rights Reserved.

5.17.1.7.3 Methods

Methods of the TQueueAgents class.

For a complete list of the TQueueAgents class members, see the TQueueAgents Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a TQueueAgent object to the collection.</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts a TQueueAgent object in the TQueueAgents collection to the required place.</td>
</tr>
</tbody>
</table>
### Add Method

Adds a TQueueAgent object to the collection.

**Class**

TQueueAgents

**Syntax**

```pascal
function Add: TQueueAgent;
```

**Return Value**

A TQueueAgent object with empty properties.

**Remarks**

Call the Add method to add a TQueueAgent object with empty properties to the collection.

### Insert Method

Inserts a TQueueAgent object in the TQueueAgents collection to the required place.

**Class**

TQueueAgents

**Syntax**

```pascal
function Insert(Index: Integer): TQueueAgent;
```

**Parameters**

*Index*

Holds the index of the place to insert a TQueueAgent object to.

**Return Value**
a TQueueAgent object.

Remarks

Call the Insert method to insert a TQueueAgent object in the TQueueAgents collection to the required place, specified with Index parameter.

5.17.1.8 TQueueMessage Class

A class representing a queue message.

For a list of all members of this type, see TQueueMessage members.

Unit

OraAQ

Syntax

TQueueMessage = class(System TObject);

Remarks

The TQueueMessage class represents a queue message. Use the TQueueMessage class for setting or reading message parameters and payload when sending or receiving a queue message.

See Also

• TOraQueue.Dequeue
• TOraQueue.DequeueArray
• TOraQueue.Enqueue
• TOraQueue.EnqueueArray
### 5.17.1.8.2 Properties

Properties of the `TQueueMessage` class.

For a complete list of the `TQueueMessage` class members, see the [TQueueMessage Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageId</td>
<td>Used to provide a message ID.</td>
</tr>
<tr>
<td>MessageProperties</td>
<td>Used to get or set queue the queue message parameters.</td>
</tr>
<tr>
<td>RawPayload</td>
<td>Used to set or get payload of the queue message as RAW payload.</td>
</tr>
<tr>
<td>StringPayload</td>
<td>Used to set or get the payload of a queue message as string payload.</td>
</tr>
</tbody>
</table>

**See Also**
- [TQueueMessage Class](#)
- [TQueueMessage Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

[Request Support] [DAC Forum] [Provide Feedback]
5.17.1.8.2.1 MessageId Property

Used to provide a message ID.

Class
TQueueMessage

Syntax

property MessageId: TMessageId;

Remarks
Use the MessageId property to get the message ID for a message.

See Also
- TQueueMessageProperties

5.17.1.8.2.2 MessageProperties Property

Used to get or set queue the queue message parameters.

Class
TQueueMessage

Syntax

property MessageProperties: TQueueMessageProperties;

Remarks
Use the MessageProperties property to get or set queue the queue message parameters that AQ uses to manage individual messages.

See Also
- TQueueMessageProperties
5.17.1.8.2.3 Raw Payload Property

Used to set or get payload of the queue message as RAW payload.

Class

TQueueMessage

Syntax

```property`` RawPayload: TBytes;
```

Remarks

Use the RawPayload method to set or get payload of the queue message as RAW payload.

© 1997-2021 Devart. All Rights Reserved.

5.17.1.8.2.4 StringPayload Property

Used to set or get the payload of a queue message as string payload.

Class

TQueueMessage

Syntax

```property`` StringPayload: string;
```

Remarks

Use the StringPayload property to set or get the payload of a queue message as string payload.

© 1997-2021 Devart. All Rights Reserved.

5.17.1.9 TQueueMessageProperties Class

A class for setting or providing queue message properties.

For a list of all members of this type, see [TQueueMessageProperties](#) members.
OraAQ

Syntax

```
TQueueMessageProperties = class(TPersistent);
```

Remarks

Use the TQueueMessageProperties class to set or get queue message properties.

See Also

- `TQueueMessage.MessageProperties`
- `TOraQueue.EnqueueMessageProperties`
- `TOraQueue.OnMessage`
- `TOraQueue.Dequeue`
- `TOraQueue.DequeueArray`
- `TOraQueue.Enqueue`
- `TOraQueue.EnqueueArray`

© 1997-2021 Devart. All Rights Reserved.

5.17.1.9.1 Members

**TQueueMessageProperties** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attempts</td>
<td>Contains the number of attempts that have been made to dequeue a message.</td>
</tr>
<tr>
<td>Correlation</td>
<td>Contains the identifier supplied by the message producer at enqueue time.</td>
</tr>
<tr>
<td>Delay</td>
<td>Contains the delay of the enqueued message.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Used to indicate whether the message will be enqueued as buffered or persistent and to specify what kinds of messages should be</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Attempts</strong></td>
<td>Contains the number of attempts that have been made to dequeue a message.</td>
</tr>
<tr>
<td><strong>Correlation</strong></td>
<td>Contains the identifier supplied by the message producer at enqueue time.</td>
</tr>
</tbody>
</table>

Properties of the `TQueueMessageProperties` class.

For a complete list of the `TQueueMessageProperties` class members, see the `TQueueMessageProperties Members` topic.
<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>EnqueueTime</td>
<td>Contains the time when a message was enqueued.</td>
</tr>
<tr>
<td>OriginalMessageId</td>
<td>Contains the parameter for propagating messages.</td>
</tr>
<tr>
<td>State</td>
<td>Contains the state of a message.</td>
</tr>
<tr>
<td>TransactionGroup</td>
<td>Contains the transaction_group for the dequeued message.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th><strong>Name</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td>Contains the delay of the enqueued message.</td>
</tr>
<tr>
<td>DeliveryMode</td>
<td>Used to indicate whether the message will be enqueued as buffered or persistent and to specify what kinds of messages should be dequeued.</td>
</tr>
<tr>
<td>ExceptionQueue</td>
<td>Contains the name of the queue into which the message is moved if it cannot be processed successfully.</td>
</tr>
<tr>
<td>Expiration</td>
<td>Contains the expiration of a message.</td>
</tr>
<tr>
<td>Priority</td>
<td>Contains the message priority.</td>
</tr>
<tr>
<td>RecipientList</td>
<td>Contains the list of agents that receive a message.</td>
</tr>
<tr>
<td>SenderId</td>
<td>Contains application-sender identification.</td>
</tr>
</tbody>
</table>

See Also

- TQueueMessageProperties Class
- TQueueMessageProperties Class Members
5.17.1.9.2.1 Attempts Property

Contains the number of attempts that have been made to dequeue a message.

Class

TQueueMessageProperties

Syntax

property Attempts: integer;

Remarks

The Attempts property contains the number of attempts that have been made to dequeue a message.

5.17.1.9.2.2 Correlation Property

Contains the identifier supplied by the message producer at enqueue time.

Class

TQueueMessageProperties

Syntax

property Correlation: string;

Remarks

The Correlation property contains the identifier supplied by the message producer at enqueue time.

5.17.1.9.2.3 Delay Property

Contains the delay of the enqueued message.

Class
**TQueueMessageProperties**

**Syntax**

```property Delay: integer default AQ_NO_DELAY;```

**Remarks**

The Delay property contains the delay of the enqueued message. The delay represents the number of seconds after which a message is available for dequeuing. The default value is AQ_NO_DELAY. That means the message will be available for immediate dequeuing.

**Class**

**TQueueMessageProperties**

**Syntax**

```property DeliveryMode: TQueueDeliveryMode default qdmPersistent;```

**Remarks**

Use the DeliveryMode property to indicate whether the message will be enqueued as buffered or persistent when enqueueing it, and to specify what kinds of messages should be dequeued (buffered, persistent or both), when dequeueing messages. qdmPersistentOrBuffered value can be set only when dequeueing a message.

**See Also**

- `TEnqueueOptions.DeliveryMode`
- `TDequeueOptions.DeliveryMode`
5.17.1.9.2.5 EnqueueTime Property

Contains the time when a message was enqueued.

Class

TQueueMessageProperties

Syntax

property EnqueueTime: TDateTime;

Remarks

The EnqueueTime property contains the time when a message was enqueued.

5.17.1.9.2.6 ExceptionQueue Property

Contains the name of the queue into which the message is moved if it cannot be processed successfully.

Class

TQueueMessageProperties

Syntax

property ExceptionQueue: string;

Remarks

The ExceptionQueue property contains the name of the queue into which the message is moved if it cannot be processed successfully.

5.17.1.9.2.7 Expiration Property

Contains the expiration of a message.

Class
**TQueueMessageProperties**

**Syntax**

```plaintext
property Expiration: integer default AQ_NEVER;
```

**Remarks**

The Expiration property contains the expiration of a message in seconds. It is the duration the message is available for dequeuing. The default value is AQ_NEVER. It means that the message will never expire.

5.17.1.9.2.8 OriginalMessageId Property

Contains the parameter for propagating messages.

**Class**

*TQueueMessageProperties*

**Syntax**

```plaintext
property OriginalMessageId: string;
```

**Remarks**

The OriginalMessageId property contains the parameter used by Oracle Streams AQ for propagating messages.

5.17.1.9.2.9 Priority Property

Contains the message priority.

**Class**

*TQueueMessageProperties*

**Syntax**

```plaintext
property Priority: integer default 1;
```
Remarks

The Priority property contains the message priority. A smaller number indicates higher priority. The priority can be any number, including negative numbers.

Class

TQueueMessageProperties

Syntax

property RecipientList: TQueueAgents stored
IsRecipientListStored;

Remarks

Contains the list of agents that receive a message.

See Also

- TQueueAgents

Class

TQueueMessageProperties

Syntax

property SenderId: TQueueAgent;

Remarks

Contains application-sender identification.
The SenderId property contains application-sender identification specified at the enqueue time by the message producer.

See Also

- TQueueAgent

5.17.1.9.2.12 State Property

Contains the state of a message.

Class

TQueueMessageProperties

Syntax

```property State: TQueueMessageState;```

Remarks

The State property contains the state of a message.

5.17.1.9.2.13 TransactionGroup Property

Contains the transaction_group for the dequeued message.

Class

TQueueMessageProperties

Syntax

```property TransactionGroup: string;```

Remarks

The TransactionGroup property contains the transaction_group for the dequeued message. Messages belonging to the same transaction group will have the same value for this attribute.
5.17.2 Types

Types in the OraAQ unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TQueueMessageEvent</td>
<td>This type is used for the TOraQueue.OnMessage event.</td>
</tr>
</tbody>
</table>

5.17.2.1 TQueueMessageEvent Procedure Reference

This type is used for the TOraQueue.OnMessage event.

Unit

oraAQ

Syntax

TQueueMessageEvent = procedure (Sender: TOrQueue; const MessageId: string; const MessageProperties: TQueueMessageProperties) of object;

Parameters

Sender

An object that raised the event.

MessageId

Holds the payload for the message.

MessageProperties

Holds the message properties.
5.17.3 Enumerations

Enumerations in the OraAQ unit.

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDequeueMode</td>
<td>Specifies the locking behavior associated with the dequeue.</td>
</tr>
<tr>
<td>TQueueDeliveryMode</td>
<td>Specifies the type of the message that will be dequeued.</td>
</tr>
<tr>
<td>TQueueMessageGrouping</td>
<td>Specifies the message grouping behavior in the queues based on this table.</td>
</tr>
<tr>
<td>TQueueMessageState</td>
<td>Specifies the message state.</td>
</tr>
<tr>
<td>TQueueNavigation</td>
<td>Specifies the position of the message that will be retrieved.</td>
</tr>
<tr>
<td>TQueueSequenceDeviation</td>
<td>Specifies if a message should be enqueued before other messages.</td>
</tr>
<tr>
<td>TQueueSortList</td>
<td>Specifies the column that will be used as a sort key.</td>
</tr>
<tr>
<td>TQueueType</td>
<td>Specifies whether the queue being created is an exception queue or a normal queue.</td>
</tr>
<tr>
<td>TQueueVisibility</td>
<td>Specifies the transaction behavior of the dequeue or enqueue request.</td>
</tr>
</tbody>
</table>

5.17.3.1 TDequeueMode Enumeration

Specifies the locking behavior associated with the dequeue.

Unit
oraAQ
Syntax

\[ \text{TDequeueMode} = (\text{dqmBrowse}, \text{dqmLocked}, \text{dqmRemove}, \text{dqmRemoveNoData}); \]

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>dqmBrowse</td>
<td>Messages will be dequeued without any locking.</td>
</tr>
<tr>
<td>dqmLocked</td>
<td>Sets write locks on the dequeued messages. The locks last for the durations of the transaction.</td>
</tr>
<tr>
<td>dqmRemove</td>
<td>Messages are removed after dequeuing. The default value.</td>
</tr>
<tr>
<td>dqmRemoveNoData</td>
<td>Messages are marked as updated or deleted after dequeuing. The message can be retained in the queue table based on the retention properties.</td>
</tr>
</tbody>
</table>

Remarks

Use DequeueMode property to specify the locking behavior associated with the dequeue.

© 1997-2021 Devart. All Rights Reserved.

5.17.3.2 TQueueDeliveryMode Enumeration

Specifies the type of the message that will be dequeued.

Unit

oraAQ

Syntax

\[ \text{TQueueDeliveryMode} = (\text{qdmPersistent}, \text{qdmBuffered}, \text{qdmPersistentOrBuffered}); \]

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qdmBuffered</td>
<td>Only buffered message will be dequeued.</td>
</tr>
<tr>
<td>qdmPersistent</td>
<td>Only persistent message will be dequeued.</td>
</tr>
<tr>
<td>qdmPersistentOrBuffered</td>
<td>Suited message of any type can be dequeued. Is not used when enqueueing a message.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart
5.17.3.3 **TQueueMessageGrouping Enumeration**

Specifies the message grouping behavior in the queues based on this table.

**Unit**

`oraAQ`

**Syntax**

```plaintext
TQueueMessageGrouping = (qmgNone, qmgTransactional);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qmgNone</td>
<td>Each queue message is treated individually. The default value.</td>
</tr>
<tr>
<td>qmgTransactional</td>
<td>Messages, enqueued in one transaction, will belong to the same group.</td>
</tr>
</tbody>
</table>

5.17.3.4 **TQueueMessageState Enumeration**

Specifies the message state.

**Unit**

`oraAQ`

**Syntax**

```plaintext
TQueueMessageState = (qmsReady, qmsWaiting, qmsProcessed, qmsExpired);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qmsExpired</td>
<td>The message has been moved to the exception queue.</td>
</tr>
<tr>
<td>qmsProcessed</td>
<td>The message has been processed and is retained.</td>
</tr>
<tr>
<td>qmsReady</td>
<td>The message is ready to be processed.</td>
</tr>
<tr>
<td>qmsWaiting</td>
<td>The message delay has not yet been reached.</td>
</tr>
</tbody>
</table>
5.17.3.5 TQueueNavigation Enumeration

Specifies the position of the message that will be retrieved.

Unit
oraAQ

Syntax

TQueueNavigation = (qnNextMessage, qnNextTransaction,
qnFirstMessage, qnFirstMessageMultiGroup, qnNextMessageMultiGroup);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qnFirstMessage</td>
<td>The first message which is available and that matches the search criteria will be dequeued. This setting resets the position to the beginning of the queue.</td>
</tr>
<tr>
<td>qnFirstMessageMultiGroup</td>
<td>Available matching messages from the beginning of the queue will be dequeued (possibly across different transaction groups) until reaching the limit of result message array size. This setting resets the position to the beginning of the queue.</td>
</tr>
<tr>
<td>qnNextMessage</td>
<td>The next message that is available and that matches the search criteria will be dequeued. The default value.</td>
</tr>
<tr>
<td>qnNextMessageMultiGroup</td>
<td>The next set of available matching messages from the beginning of the queue will be dequeued (possibly across different transaction groups) until reaching the limit of the result message array size. This setting resets the position to the beginning of the queue.</td>
</tr>
<tr>
<td>qnNextTransaction</td>
<td>The first message of the next transaction group that is available and that matches the search criteria will be dequeued.</td>
</tr>
</tbody>
</table>

5.17.3.6 TQueueSequenceDeviation Enumeration

Specifies if a message should be enqueued before other messages.

Unit
OraAQ

Syntax

TQueueSequenceDeviation = (qsdNone, qsdBefore, qsdTop);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qsdBefore</td>
<td>Message is enqueued before the message specified by RelativeMsgid property.</td>
</tr>
<tr>
<td>qsdNone</td>
<td>Message is enqueued in usual order. The default value.</td>
</tr>
<tr>
<td>qsdTop</td>
<td>The message will be enqueued before all queued messages.</td>
</tr>
</tbody>
</table>

5.17.3.7 TQueueSortList Enumeration

Specifies the column that will be used as a sort key.

Unit

OraAQ

Syntax

TQueueSortList = (qslDefault, qslPriority, qslEnqueueTime, qslPriorityEnqueueTime, qslEnqueueTimePriority);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qslDefault</td>
<td>No sorting is specified, the table will be sorted in the enqueue time in the ascending order by default. The default value.</td>
</tr>
<tr>
<td>qslEnqueueTime</td>
<td>The table will be sorted by enq_time column.</td>
</tr>
<tr>
<td>qslEnqueueTimePriority</td>
<td>The table will be sorted by both enq_time and priority columns. Enq_time column defines the most significant order.</td>
</tr>
<tr>
<td>qslPriority</td>
<td>The table will be sorted by priority column.</td>
</tr>
<tr>
<td>qslPriorityEnqueueTime</td>
<td>The table will be sorted by both the enq_time and priority columns. Priority column defines the most significant order.</td>
</tr>
</tbody>
</table>
5.17.3.8 TQueueType Enumeration

Specifies whether the queue being created is an exception queue or a normal queue.

Unit
OraAQ

Syntax
TQueueType = (qtNormalQueue, qtExceptionQueue);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qtExceptionQueue</td>
<td>The queue being created is an exception queue.</td>
</tr>
<tr>
<td>qtNormalQueue</td>
<td>The queue being created is a normal queue. The Default value.</td>
</tr>
</tbody>
</table>

5.17.3.9 TQueueVisibility Enumeration

Specifies the transaction behaviour of the dequeue or enqueue request.

Unit
OraAQ

Syntax
TQueueVisibility = (qvOnCommit, qvImmediate);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>qvImmediate</td>
<td>The dequeue or enqueue is being made in an autonomous transaction.</td>
</tr>
<tr>
<td>qvOnCommit</td>
<td>The dequeue or enqueue is a part of the current transaction. The operation is complete when the transaction commits. The default value.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

Request Support   DAC Forum   Provide Feedback
5.18 OraCall

Defines Oracle Call Interface routines.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOracleHome</td>
<td>A class representing the TOraSession.Home property for using Oracle Client.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.18.1 Classes

Classes in the OraCall unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOracleHome</td>
<td>A class representing the TOraSession.Home property for using Oracle Client.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.18.1.1 TOracleHome Class

A class representing the TOraSession.Home property for using Oracle Client.

For a list of all members of this type, see TOracleHome members.

Unit
oraCall

Syntax

TOracleHome = class(System.TObject);
Remarks

Use `TOraSession.Home` property to select which Oracle client will be used in your application. Use this property in cases when there is a number of Oracle clients on the machine. ODAC searches all available homes in the `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\ALL_HOMES` registry folder.

See Also
- `TOraSession.Home`

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Defines the name of Oracle Client - <code>ORACLE_HOME_NAME</code></td>
</tr>
<tr>
<td>OCICallStyle</td>
<td>Defines belonging to the OCI function set {None, OCI73, OCI80}</td>
</tr>
<tr>
<td>OCIclientDLL</td>
<td>Defines the name and full path to the ClientOCI library</td>
</tr>
<tr>
<td>OCIDLL</td>
<td>Defines the name and full path to the OCI library</td>
</tr>
<tr>
<td>OCIVersion</td>
<td>Defines the version of the OCI library as integer</td>
</tr>
<tr>
<td>OCIVersionSt</td>
<td>Defines the version of the OCI library</td>
</tr>
<tr>
<td>Path</td>
<td>Defines the path to Oracle Client</td>
</tr>
<tr>
<td>PossibleOCICallStyles</td>
<td>Holds the list of supported OCI function sets</td>
</tr>
<tr>
<td>TNSPath</td>
<td>Defines the value for the TNS_ADMIN variable</td>
</tr>
</tbody>
</table>
5.18.1.1.2 Properties

Properties of the TOracleHome class.

For a complete list of the TOracleHome class members, see the TOracleHome Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Defines the name of Oracle Client - ORACLE_HOME_NAME</td>
</tr>
<tr>
<td>OCICallStyle</td>
<td>Defines belonging to the OCI function set {None, OCI73, OCI80}</td>
</tr>
<tr>
<td>OCIClientDLL</td>
<td>Defines the name and full path to the ClientOCI library</td>
</tr>
<tr>
<td>OCIDLL</td>
<td>Defines the name and full path to the OCI library</td>
</tr>
<tr>
<td>OCIVersion</td>
<td>Defines the version of the OCI library as integer</td>
</tr>
<tr>
<td>OCIVersionSt</td>
<td>Defines the version of the OCI library</td>
</tr>
<tr>
<td>Path</td>
<td>Defines the path to Oracle Client</td>
</tr>
<tr>
<td>PossibleOCICallStyles</td>
<td>Holds the list of supported OCI function sets</td>
</tr>
<tr>
<td>TNSPath</td>
<td>Defines the value for the TNS_ADMIN variable</td>
</tr>
</tbody>
</table>

See Also
- TOracleHome Class
- TOracleHome Class Members

© 1997-2021 Devart. All Rights Reserved.
5.18.1.1.2.1 Name Property

Defines the name of Oracle Client - ORACLE_HOME_NAME

Class

TOracleHome

Syntax

```property Name: string;```

© 1997-2021
Devart. All Rights Reserved.

5.18.1.1.2.2 OCICallStyle Property

Defines belonging to the OCI function set {None, OCI73, OCI80}

Class

TOracleHome

Syntax

```property OCICallStyle: TOCICallStyle;```

© 1997-2021
Devart. All Rights Reserved.

5.18.1.1.2.3 OCIClientDLL Property

Defines the name and full path to the ClientOCI library

Class

TOracleHome

Syntax

```property OCIClientDLL: string;```

© 1997-2021
Devart. All Rights Reserved.
5.18.1.1.2.4 OCIDLL Property

Defines the name and full path to the OCI library

**Class**

*TOracleHome*

**Syntax**

```?
property OCIDLL: string;
```

© 1997-2021

Devart. All Rights Reserved.

5.18.1.1.2.5 OCIVersion Property

Defines the version of the OCI library as integer

**Class**

*TOracleHome*

**Syntax**

```?
property OCIVersion: word;
```

© 1997-2021

Devart. All Rights Reserved.

Provide Feedback

5.18.1.1.2.6 OCIVersionSt Property

Defines the version of the OCI library

**Class**

*TOracleHome*

**Syntax**

```?
property OCIVersionSt: string;
```

© 1997-2021

Devart. All Rights Reserved.

Provide Feedback
5.18.1.1.2.7 Path Property

Defines the path to Oracle Client

Class

TOracleHome

Syntax

property Path: string;

5.18.1.1.2.8 PossibleOCICallStyles Property

Holds the list of supported OCI function sets

Class

TOracleHome

Syntax

property PossibleOCICallStyles: TOCICallStyleSet;

5.18.1.1.2.9 TNSPath Property

Defines the value for the TNS_ADMIN variable

Class

TOracleHome

Syntax

property TNSPath: string;
5.19 OraClasses

OraClasses unit defines following data type constants: dtRowId dtCursor dtOraBlob
dtOraClob dBFILE dtCFILE dtLabel dtFixedChar dtUndefined dtTimeStamp dtTimeStampTZ
dtTimeStampLTZ dtIntervalYM dtIntervalDS // obsolete dtBLOBLocator = dtOraBlob
dtCLOBLocator = dtOraClob

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNotifyTableChanges</td>
<td>Allows getting the list of tables and changes made to them using the</td>
</tr>
<tr>
<td></td>
<td>TNotifyTableChanges.Changes property.</td>
</tr>
<tr>
<td>TOraCursor</td>
<td>A class holding the internal Oracle Call Interface data CDA for Oracle 7 and</td>
</tr>
<tr>
<td></td>
<td>the OCISIdxt descriptor for Oracle 8.</td>
</tr>
<tr>
<td>TOraFile</td>
<td>A class holding the value of the BFile field and parameter.</td>
</tr>
<tr>
<td>TOraInterval</td>
<td>A class providing support for Oracle 9 interval datatypes.</td>
</tr>
<tr>
<td>TOraLob</td>
<td>A class holding the value of the BLOB and CLOB fields and parameters.</td>
</tr>
<tr>
<td>TOraNumber</td>
<td>A class supporting</td>
</tr>
<tr>
<td>TOraTimeStamp</td>
<td>A class supporting Oracle 9 timestamp datatypes.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TChangeNotifyEventType</td>
<td>Contains the event type.</td>
</tr>
<tr>
<td>TConnectMode</td>
<td>Specifies the system privileges used when a user connects to the server.</td>
</tr>
<tr>
<td>TMessageType</td>
<td>Defines the type of the next message found in the received named pipe local</td>
</tr>
</tbody>
</table>
Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloatPrecision</td>
<td>Set this constant to define the type of NUMBER fields with Scale &gt; 0.</td>
</tr>
<tr>
<td>IntegerPrecision</td>
<td>Set this constant to define the type of NUMBER fields with precision less than or equal to IntegerPrecision as dtInteger.</td>
</tr>
<tr>
<td>LargeIntPrecision</td>
<td>Set this constant to define the type of NUMBER fields with precision greater than the IntegerPrecision and less than or equal to LargeIntPrecision as dtLargeInt.</td>
</tr>
</tbody>
</table>

5.19.1 Classes

Classes in the OraClasses unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNotifyTableChanges</td>
<td>Allows getting the list of tables and changes made to them using the TNotifyTableChanges.Changes property.</td>
</tr>
<tr>
<td>TOraCursor</td>
<td>A class holding the internal Oracle Call Interface data CDA for Oracle 7 and the OCISQmt descriptor for Oracle 8.</td>
</tr>
</tbody>
</table>
### TNotifyTableChanges Class

Allows getting the list of tables and changes made to them using the `TNotifyTableChanges.Changes` property.

For a list of all members of this type, see [TNotifyTableChanges members](https://www.devart.com/tdb/tdbaccess/tda/oraclasses/tnotifytablechanges.html).

#### Unit

**OraClasses**

#### Syntax

```pascal
TNotifyTableChanges = class(TCustomNotifyChanges);
```

#### Inheritance Hierarchy

TCustomNotifyChanges

**TNotifyTableChanges**

© 1997-2021 Devart. All Rights Reserved.  

#### Members

[TNotifyTableChanges](https://www.devart.com/tdb/tdbaccess/tda/oraclasses/tnotifytablechanges.html) class overview.

#### Properties
5.19.1.2 Properties

Properties of the **TNotifyTableChanges** class.

For a complete list of the **TNotifyTableChanges** class members, see the **TNotifyTableChanges Members** topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changes</td>
<td>Contains the list of tables and changes made to them.</td>
</tr>
</tbody>
</table>

**See Also**
- **TNotifyTableChanges Class**
- **TNotifyTableChanges Class Members**

### Changes Property(Indexer)

Contains the list of tables and changes made to them.

### Class

**TNotifyTableChanges**

### Syntax

```pascal
property Changes[Index: integer]: TNotifyTableChange; default;
```

**Parameters**

*Index*

© 1997-2021 Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback
5.19.1.2  TOraCursor Class

A class holding the internal Oracle Call Interface data CDA for Oracle 7 and the OCISstmt descriptor for Oracle 8.

For a list of all members of this type, see TOraCursor members.

Unit
oraClasses

Syntax

TOraCursor = class(TCRCursor);

Remarks

TOraCursor holds the internal Oracle Call Interface data CDA for Oracle 7 and the OCISstmt descriptor for Oracle 8.

Inheritance Hierarchy

TSharedObject
   TCRCursor
   TOraCursor

See Also

- TCursorField
- TOraDataSet.Cursor
- TOraParam.AsCursor

© 1997-2021
Devart. All Rights Reserved.

5.19.1.2.1  Members

TOraCursor class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDA</td>
<td>Used to return CDA.</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocCursor</strong></td>
<td>Allocates CDA for Oracle 7 and OCIStmt handler for Oracle 8.</td>
</tr>
<tr>
<td><strong>CanFetch</strong></td>
<td>Verifies if the cursor state permits data fetching.</td>
</tr>
<tr>
<td><strong>FreeCursor</strong></td>
<td>Releases the cursor data.</td>
</tr>
<tr>
<td><strong>Release</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

### Properties

Properties of the **TOraCursor** class.

For a complete list of the **TOraCursor** class members, see the **TOraCursor Members** topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CDA</strong></td>
<td>Used to return CDA.</td>
</tr>
<tr>
<td><strong>OCICallStyle</strong></td>
<td>Used to get or set the Oracle Client version for a subsequent cursor operations.</td>
</tr>
</tbody>
</table>
**Oracle Data Access Components**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCIStmt</td>
<td>Used to return the OCIStmt handler.</td>
</tr>
<tr>
<td>RefCount</td>
<td>(inherited from TSharedObject) Used to return the count of reference to a</td>
</tr>
<tr>
<td></td>
<td>TSharedObject object.</td>
</tr>
<tr>
<td>State</td>
<td>Used to set the cursor state.</td>
</tr>
</tbody>
</table>

See Also
- [TOraCursor Class](#)
- [TOraCursor Class Members](#)

**5.19.1.2.2.1 CDA Property**

Used to return CDA.

**Class**

[TOraCursor](#)

**Syntax**

```liquid
property CDA: PCDA;
```

**Remarks**

Use the CDA property to return CDA. You can use this property for Oracle 7.

**5.19.1.2.2.2 OCICallStyle Property**

Used to get or set the Oracle Client version for a subsequent cursor operations.

**Class**

[TOraCursor](#)

**Syntax**
**property** OCICallStyle: TOCICallStyle;

**Remarks**

Use the OCICallStyle property to get or set the Oracle Client version for a subsequent cursor operations.

5.19.1.2.2.3 OCIStmt Property

Used to return the OCIStmt handler.

**Class**

**TOraCursor**

**Syntax**

```property** OCIStmt: pOCIStmt;
```

**Remarks**

Use the OCIStmt property to return the OCIStmt handler. You can use this property for Oracle 8.

5.19.1.2.2.4 State Property

Used to set the cursor state.

**Class**

**TOraCursor**

**Syntax**

```property** State: TCursorState;
```

**See Also**

- **TCursorState**
5.19.1.2.3  Methods

Methods of the `TOraCursor` class.

For a complete list of the `TOraCursor` class members, see the `TOraCursor Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from <code>TSharedObject</code>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocCursor</td>
<td>Allocates CDA for Oracle 7 and OCStmt handler for Oracle 8.</td>
</tr>
<tr>
<td>CanFetch</td>
<td>Verifies if the cursor state permits data fetching.</td>
</tr>
<tr>
<td>FreeCursor</td>
<td>Releases the cursor data.</td>
</tr>
<tr>
<td>Release (inherited from <code>TSharedObject</code>)</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

See Also
- `TOraCursor Class`
- `TOraCursor Class Members`

5.19.1.2.3.1 AllocCursor Method

Allocates CDA for Oracle 7 and OCStmt handler for Oracle 8.

Class
- `TOraCursor`

Syntax

```pascal
procedure AllocCursor(StatementMode: TStatementMode = smAllocated);
```
Parameters
AStatementCache

Remarks
Call the AllocCursor procedure to allocate CDA for Oracle 7 and OCISmt handler for Oracle 8.

See Also
• FreeCursor

5.19.1.2.3.2 CanFetch Method

Verifies if the cursor state permits data fetching.

Class
TOraCursor

Syntax
function CanFetch: boolean; override;

Return Value
True, if data fetching is permitted, False otherwise.

Remarks
Call the CanFetch function to verify whether the cursor is in the state that permits data fetching.

5.19.1.2.3.3 FreeCursor Method

Releases the cursor data.

Class
TOraCursor
Syntax

```plaintext
procedure FreeCursor;
```

Remarks

Call the FreeCursor method to release the cursor data.

See Also

- AllocCursor

5.19.1.3 TOraFile Class

A class holding the value of the BFile field and parameter.

For a list of all members of this type, see TOraFile members.

Unit

oraClasses

Syntax

```plaintext
TOraFile = class(TOraLob);
```

Remarks

TOraFile is a descendant of the TOraLob class. It holds the value of the BFile field and parameter.

The BFile datatype provides access to the file LOBs that are stored in file systems outside an Oracle database. Oracle 8 currently supports access to binary files, or BFILES. The BFILE datatype allows the read-only support of large binary files; you cannot modify a file through Oracle.

Inheritance Hierarchy

- TSharedObject
  - TBlob
    - TCompressedBlob
      - TOraLob
**TOraFile**

**See Also**
- `TOraLob`
- `TBFileField`
- `TOraParam.AsBFile`

© 1997-2021 Devart. All Rights Reserved.

5.19.1.3.1 Members

**TOraFile class overview.**

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsString</code> (inherited from <code>TBlob</code>)</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td><code>AsWideString</code> (inherited from <code>TBlob</code>)</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td><code>Cached</code> (inherited from <code>TOraLob</code>)</td>
<td>Used to indicate where the LOB data is.</td>
</tr>
<tr>
<td><code>Compressed</code> (inherited from <code>TCompressedBlob</code>)</td>
<td>Used to indicate if the Blob is compressed.</td>
</tr>
<tr>
<td><code>CompressedSize</code> (inherited from <code>TCompressedBlob</code>)</td>
<td>Used to indicate compressed size of the Blob data.</td>
</tr>
<tr>
<td><code>FileDir</code></td>
<td>Determines the directory alias of where the file associated with the BFile field is stored.</td>
</tr>
<tr>
<td><code>FileName</code></td>
<td>Used to determine the name of a file associated with the BFile field.</td>
</tr>
<tr>
<td><code>IsUnicode</code> (inherited from <code>TBlob</code>)</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
<tr>
<td><code>OCILobLocator</code> (inherited from <code>TOraLob</code>)</td>
<td>Used to get or set the OCILobLocator handle.</td>
</tr>
<tr>
<td><code>OCILobLocatorPtr</code> (inherited from <code>TOraLob</code>)</td>
<td>Used to retrieve the LOB value type locator.</td>
</tr>
</tbody>
</table>
### OCISvcCtx (inherited from TOpal)
- **Description**: Used to assign a service context handle.

### RefCount (inherited from TSharedObject)
- **Description**: Used to return the count of reference to a TSharedObject object.

### Size (inherited from TBlob)
- **Description**: Used to learn the size of the TBlob value in bytes.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from TSharedObject)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocLob (inherited from TOpal)</td>
<td>Allocates and initializes the LOB locator.</td>
</tr>
<tr>
<td>Assign (inherited from TBlob)</td>
<td>Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td>Clear (inherited from TBlob)</td>
<td>Deletes the current value in TBlob object.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes a file opened by Open.</td>
</tr>
<tr>
<td>CreateTemporary (inherited from TOpal)</td>
<td>Allocates and initializes a temporary LOB locator of the specified type.</td>
</tr>
<tr>
<td>DisableBuffering (inherited from TOpal)</td>
<td>Disables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td>EnableBuffering (inherited from TOpal)</td>
<td>Enables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td>Exists</td>
<td>Determines whether a file associated with the BFile field exists.</td>
</tr>
<tr>
<td>FreeLob (inherited from TOpal)</td>
<td>Releases the LOB locator descriptor.</td>
</tr>
<tr>
<td>FreeTemporary (inherited from TOpal)</td>
<td>Frees a temporary LOB.</td>
</tr>
<tr>
<td>Init (inherited from TOpal)</td>
<td>Initializes the OCILOBLocator handle.</td>
</tr>
<tr>
<td>IsInit (inherited from TOpal)</td>
<td>Verifies if the LOB locator is initialized.</td>
</tr>
<tr>
<td>IsOpen</td>
<td>Determines whether the file associated with the BFile field is opened.</td>
</tr>
</tbody>
</table>
### 5.19.1.3.2 Properties

Properties of the **TOraFile** class.

For a complete list of the **TOraFile** class members, see the [TOraFile Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IsTemporary</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Indicates whether the LOB is temporary.</td>
</tr>
<tr>
<td><strong>LengthLob</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Returns the number of bytes contained in the LOB object.</td>
</tr>
<tr>
<td><strong>LoadFromFile</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td><strong>LoadFromStream</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td><strong>Open</strong></td>
<td>Opens the file associated with the BFile field.</td>
</tr>
<tr>
<td><strong>Read</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
<tr>
<td><strong>ReadLob</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Reads the LOB content from the server.</td>
</tr>
<tr>
<td><strong>Refresh</strong></td>
<td>Reloads a file associated the BFile field opened by Open.</td>
</tr>
<tr>
<td><strong>Release</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td><strong>SaveToFile</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Saves the contents of the TBlob object to a file.</td>
</tr>
<tr>
<td><strong>SaveToStream</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Copies the contents of a TBlob object to a stream.</td>
</tr>
<tr>
<td><strong>Truncate</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Sets new TBlob size and discards all data over it.</td>
</tr>
<tr>
<td><strong>Write</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Stores a raw sequence of bytes into a TBlob object.</td>
</tr>
<tr>
<td><strong>WriteLob</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Writes a LOB value to the server.</td>
</tr>
</tbody>
</table>
| **AsString** (inherited from **TBlob**)
| **Used to manipulate BLOB value as string.** |
| **AsWideString** (inherited from **TBlob**)
| **Used to manipulate BLOB value as Unicode string.** |
| **Cached** (inherited from **TOraLob**)
| **Used to indicate where the LOB data is.** |
| **Compressed** (inherited from **TCompressedBlob**)
| **Used to indicate if the Blob is compressed.** |
| **CompressedSize** (inherited from **TCompressedBlob**)
| **Used to indicate compressed size of the Blob data.** |
| **FileDir**
| **Determines the directory alias of where the file associated with the BFile field is stored.** |
| **FileName**
| **Used to determine the name of a file associated with the BFile field.** |
| **IsUnicode** (inherited from **TBlob**)
| **Gives choice of making TBlob store and process data in Unicode format or not.** |
| **OCILobLocator** (inherited from **TOraLob**)
| **Used to get or set the OCILobLocator handle.** |
| **OCILobLocatorPtr** (inherited from **TOraLob**)
| **Used to retrieve the LOB value type locator.** |
| **OCISvcCtx** (inherited from **TOraLob**)
| **Used to assign a service context handle.** |
| **RefCount** (inherited from **TSharedObject**)
| **Used to return the count of reference to a TSharedObject object.** |
| **Size** (inherited from **TBlob**)
| **Used to learn the size of the TBlob value in bytes.** |

See Also
- **TOraFile Class**
- **TOraFile Class Members**
5.19.1.3.2.1 FileDir Property

Determines the directory alias of where the file associated with the BFile field is stored.

Class

TOraFile

Syntax

```properties
FileDir: string;
```

Remarks

Use FileDir to determine the directory alias where the file associated with the BFile field is stored.

To create a directory alias use CREATE DIRECTORY.

See Also

- FileName

5.19.1.3.2.2 FileName Property

Used to determine the name of a file associated with the BFile field.

Class

TOraFile

Syntax

```properties
FileName: string;
```

Remarks

Use the FileName property to determine the name of a file associated with the BFile field.

See Also

- FileDir
Reserved.

5.19.1.3.3 Methods

Methods of the **TOraFile** class.

For a complete list of the **TOraFile** class members, see the **TOraFile Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocLob</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Allocates and initializes the LOB locator.</td>
</tr>
<tr>
<td><strong>Assign</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td><strong>Clear</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Deletes the current value in TBlob object.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes a file opened by Open.</td>
</tr>
<tr>
<td><strong>CreateTemporary</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Allocates and initializes a temporary LOB locator of the specified type.</td>
</tr>
<tr>
<td><strong>DisableBuffering</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Disables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td><strong>EnableBuffering</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Enables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Determines whether a file associated with the BFile field exists.</td>
</tr>
<tr>
<td><strong>FreeLob</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Releases the LOB locator descriptor.</td>
</tr>
<tr>
<td><strong>FreeTemporary</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Frees a temporary LOB.</td>
</tr>
<tr>
<td><strong>Init</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Initializes the OCILobLocator handle.</td>
</tr>
<tr>
<td><strong>IsInit</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Verifies if the LOB locator is initialized.</td>
</tr>
<tr>
<td><strong>IsOpen</strong></td>
<td>Determines whether the file associated with the BFile field is opened.</td>
</tr>
<tr>
<td><strong>IsTemporary</strong> (inherited from <strong>TOraLob</strong>)</td>
<td>Indicates whether the LOB is temporary.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LengthLob</td>
<td>Returns the number of bytes contained in the LOB object.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td>Open</td>
<td>Opens the file associated with the BFile field.</td>
</tr>
<tr>
<td>Read</td>
<td>Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
<tr>
<td>ReadLob</td>
<td>Reads the LOB content from the server.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Reloads a file associated the BFile field opened by Open.</td>
</tr>
<tr>
<td>Release</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>SaveToFile</td>
<td>Saves the contents of the TBlob object to a file.</td>
</tr>
<tr>
<td>SaveToStream</td>
<td>Copies the contents of a TBlob object to a stream.</td>
</tr>
<tr>
<td>Truncate</td>
<td>Sets new TBlob size and discards all data over it.</td>
</tr>
<tr>
<td>Write</td>
<td>Stores a raw sequence of bytes into a TBlob object.</td>
</tr>
<tr>
<td>WriteLob</td>
<td>Writes a LOB value to the server.</td>
</tr>
</tbody>
</table>

See Also
- TOraFile Class
- TOraFile Class Members

© 1997-2021 Devart. All Rights Reserved.

5.19.1.3.3.1 Close Method

Closes a file opened by Open.

Class
**T0raFile**

**Syntax**

```plaintext
procedure close;
```

**Remarks**

Call the Close procedure to close a file specified by `FileDir` and `FileName` properties of the BFile field, opened earlier by Open.

**See Also**

- [Open](#)

---

5.19.1.3.3.2 Exists Method

Determines whether a file associated with the BFile field exists.

**Class**

**T0raFile**

**Syntax**

```plaintext
function Exists: boolean;
```

**Return Value**

True, if a file associated with the BFile field exists, False otherwise.

**Remarks**

Call the Exists method to determine if a file associated with the BFile field exists.

**See Also**

- [Open](#)
5.19.1.3.3 IsOpen Method

Determines whether the file associated with the BFile field is opened.

Class

T0raFile

Syntax

function IsOpen: boolean;

Return Value

True, if the file is opened, False otherwise.

Remarks

Use the IsOpen method to determine whether the file associated with the BFile field is opened.

See Also

• Open

© 1997-2021 Devart. All Rights Reserved.

5.19.1.3.4 Open Method

Opens the file associated with the BFile field.

Class

T0raFile

Syntax

procedure Open;

Remarks

Call the Open procedure to open the file associated with the BFile field.

See Also

• Close
  • IsOpen
5.19.1.3.5 Refresh Method

Reloads a file associated the BFile field opened by Open.

Class

TOraFile

Syntax

```plaintext
procedure Refresh;
```

Remarks

Call the Refresh procedure to reload the file associated the BFile field, which was opened earlier by Open.

See Also

• Open

5.19.1.4 TOraInterval Class

A class providing support for Oracle 9 interval datatypes.

For a list of all members of this type, see TOraInterval members.

Unit

oraclasses

Syntax

```plaintext
TOraInterval = class(TSharedObject);
```

Remarks

TOraInterval is used to support Oracle 9 interval datatypes. There are two interval datatypes:
INTERVAL YEAR TO MONTH,
INTERVAL DAY TO SECOND.
They can be distinguished by the DescriptorType property.

Inheritance Hierarchy

TSharedObject
  TOraInterval

© 1997-2021
Devart. All Rights Reserved.

5.19.1.4.1 Members

**TOraInterval** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsString</td>
<td>Used to get and set the interval value as string.</td>
</tr>
<tr>
<td>DescriptorType</td>
<td>Used to define the descriptor type allocated to operate with the interval value.</td>
</tr>
<tr>
<td>FracPrecision</td>
<td>Used to get or set the interval fractional second precision.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to get or set the null interval value flag.</td>
</tr>
<tr>
<td>LeadPrecision</td>
<td>Used to get or set the leading field precision.</td>
</tr>
<tr>
<td>OCIInterval</td>
<td>Used to get or set the OCIInterval descriptor handle.</td>
</tr>
<tr>
<td>OCIIntervalPtr</td>
<td>Used to get the OCIInterval type locator.</td>
</tr>
<tr>
<td>RefCount</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

Methods

© 2021 Devart
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong></td>
<td>(inherited from <strong>TSharedObject</strong>) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocInterval</strong></td>
<td>Allocates the OCIInterval descriptor handle.</td>
</tr>
<tr>
<td><strong>Compare</strong></td>
<td>Compares the current TOraInterval value with the Dest value.</td>
</tr>
<tr>
<td><strong>FreeInterval</strong></td>
<td>Frees the OCIInterval descriptor handle.</td>
</tr>
<tr>
<td><strong>GetDaySecond</strong></td>
<td>Gets the values of the day and second from an interval.</td>
</tr>
<tr>
<td><strong>GetYearMonth</strong></td>
<td>Gets the values of the year and month from an interval.</td>
</tr>
<tr>
<td><strong>Release</strong></td>
<td>(inherited from <strong>TSharedObject</strong>) Decrements the reference count.</td>
</tr>
<tr>
<td><strong>SetDaySecond</strong></td>
<td>Used to set the values of the day and second from in an interval.</td>
</tr>
<tr>
<td><strong>SetYearMonth</strong></td>
<td>Sets the values of the year and month in an interval.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.19.1.4.2 Properties

Properties of the **TOraInterval** class.

For a complete list of the **TOraInterval** class members, see the [TOraInterval Members](#) topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsString</strong></td>
<td>Used to get and set the interval value as string.</td>
</tr>
<tr>
<td><strong>DescriptorType</strong></td>
<td>Used to define the descriptor type allocated to operate with the interval value.</td>
</tr>
<tr>
<td><strong>FracPrecision</strong></td>
<td>Used to get or set the interval fractional second</td>
</tr>
</tbody>
</table>
### AsString Property

**Remarks**

Use the AsString property to get and set the interval value as string. Oracle establishes certain formats for each interval type.

**INTERVAL YEAR TO MONTH format:**

'\[year-\]month'.

**INTERVAL DAY TO SECOND formats:**

'seconds',

---

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsNull</td>
<td>Used to get or set the null interval value flag.</td>
</tr>
<tr>
<td>LeadPrecision</td>
<td>Used to get or set the leading field precision.</td>
</tr>
<tr>
<td>OCIInterval</td>
<td>Used to get or set the OCIInterval descriptor handle.</td>
</tr>
<tr>
<td>OCIIntervalPtr</td>
<td>Used to get the OCIInterval type locator.</td>
</tr>
<tr>
<td>RefCount</td>
<td>(inherited from <strong>TSharedObject</strong>) Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>
'minutes:seconds',

'hours:minutes:seconds]','

days[ hours:minutes:seconds]]'.

Optional fields are surrounded by brackets.

Reading AsString property when IsNull is True returns an empty string.

See Also
• IsNull

5.19.1.4.2.2 DescriptorType Property

Used to define the descriptor type allocated to operate with the interval value.

Class
TOraInterval

Syntax

| property | DescriptorType: cardinal; |

Remarks

Use the DescriptorType property to define the type of the descriptor which is allocated to operate with the interval value. In most cases you don't need to set this property directly: it is adjusted automatically when working with fields and parameters.

Valid values for this property are:

OCI_DTYPE_INTERVAL_DS,

OCI_DTYPE_INTERVAL_YM.

They are defined in OraCall unit.
5.19.1.4.2.3 FracPrecision Property

Used to get or set the interval fractional second precision.

Class
T0raInterval

Syntax

| property FracPrecision: byte; |

Remarks

Use the FracPrecision property to get or set fractional second precision of the interval (the number of digits that are used to represent fractional seconds). This property affects only when reading the AsString property. The default value of the property is 6.

See Also
- AsString

5.19.1.4.2.4 IsNull Property

Used to get or set the null interval value flag.

Class
T0raInterval

Syntax

| property IsNull: boolean; |

Remarks

Use the IsNull property to get or set the null interval value flag. When IsNull is True getting values via the GetDaySecond and GetYearMonth methods can raise an exception. The AsString property in this case returns an empty string.

See Also
- GetDaySecond
5.19.1.4.2.5  LeadPrecision Property

Used to get or set the leading field precision.

Class

T0raInterval

Syntax

```pascal
property LeadPrecision: byte;
```

Remarks

Use the LeadPrecision property to get or set the leading field precision (the number of digits that are used to represent leading interval part). This property has effect only when reading the AsString property. The default value of the property is 2.

See Also

• AsString

5.19.1.4.2.6  OCIInterval Property

Used to get or set the OCIIInterval descriptor handle.

Class

T0raInterval

Syntax

```pascal
property OCIIInterval: pOCIIInterval;
```

Remarks

Use the OCIIInterval property to get or set the OCIIInterval descriptor handle.
5.19.1.4.2.7 OCIIntervalPtr Property

Used to get the OCIInterval type locator.

Class

**TOraInterval**

Syntax

```pascal
property OCIIntervalPtr: ppOCIInterval;
```

Remarks

Use the OCIIntervalPtr property to get the OCIInterval type locator.

5.19.1.4.3 Methods

Methods of the **TOraInterval** class.

For a complete list of the **TOraInterval** class members, see the [TOraInterval Members](#) topic.

Protected

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllocInterval</td>
<td>Allocates the OCIInterval descriptor handle.</td>
</tr>
<tr>
<td>FreeInterval</td>
<td>Frees the OCIInterval descriptor handle.</td>
</tr>
</tbody>
</table>

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares the current</td>
</tr>
</tbody>
</table>
### AllocInterval Method

Allocates the OCIInterval descriptor handle.

**Class**

**TOraInterval**

**Syntax**

```plaintext
procedure AllocInterval;
```

**Remarks**

Call the AllocInterval method to allocate the OCIInterval descriptor handle according to the **DescriptorType** property.

**See Also**

- **DescriptorType**

5.19.1.4.3.2 Compare Method

Compares the current TOraInterval value with the Dest value.

Class

TOraInterval

Syntax

function Compare(Dest: TOraInterval): integer;

Parameters

Dest

Holds the Dest value.

Return Value

the negative value if current interval is shorter than Dest, zero if intervals are equal, and positive value if the current interval is longer than Dest.

Remarks

Call the Compare method to compare the current TOraInterval value with Dest value.

Returns negative value if current interval is shorter than Dest, zero if intervals are equal and positive value if current interval is longer than Dest.

© 1997-2021 Devart. All Rights Reserved.

5.19.1.4.3.3 FreeInterval Method

Frees the OCIInterval descriptor handle.

Class

TOraInterval

Syntax

procedure FreeInterval;

Remarks

Call the FreeInterval procedure frees the OCIInterval descriptor handle. After the FreeInterval call the IsNull property is set to True.
5.19.1.4.3.4  GetDaySecond Method

Gets the values of the day and second from an interval.

Class

**TораInterval**

Syntax

```plaintext
procedure GetDaySecond(var Day: integer; var Hour: integer; var Min: integer; var Sec: integer; var FSec: integer);
```

Parameters

- **Day**
  - Holds the day value.
- **Hour**
  - Holds the hour value.
- **Min**
  - Holds the minute value.
- **Sec**
  - Holds the second value.
- **FSec**
  - Holds the fraction second value.

Remarks

Call the GetDaySecond procedure to get the values of the day and second from an interval.

See Also

- **IsNull**

See Also

- **SetDaySecond**
5.19.1.4.3.5 GetYearMonth Method

Gets the values of the year and month from an interval.

Class

TOraInterval

Syntax

```pascal
procedure GetYearMonth(var Year: integer; var Month: integer);
```

Parameters

- **Year**
  - Holds the year value.

- **Month**
  - Holds the month value.

Remarks

Call the GetYearMonth procedure to get the values of the year and month from an interval.

See Also

- [SetYearMonth](#)

5.19.1.4.3.6 SetDaySecond Method

Used to set the values of the day and second from in an interval.

Class

TOraInterval

Syntax

```pascal
procedure SetDaySecond(Day: integer; Hour: integer; Min: integer;
                       Sec: integer; FSec: integer);
```

Parameters

- **Day**
  - Holds the day value.

- **Hour**
Holds the hour value.

Min
Holds the minute value.

Sec
Holds the second value.

FSec
Holds the fraction second value.

Remarks
Call the SetDaySecond method to set the values of the day and second from in an interval.

See Also
- GetDaySecond

Sets the values of the year and month in an interval.

Class
T0raInterval

Syntax

```
procedure SetYearMonth(Year: integer; Month: integer);
```

Parameters

Year
Holds the year value.

Month
Holds the month value.

Remarks
Call the SetYearMonth method to set the values of the year and month in an interval.

See Also
- GetYearMonth
5.19.1.5 T0raLob Class

A class holding the value of the BLOB and CLOB fields and parameters.

For a list of all members of this type, see T0raLob members.

Unit
oraClasses

Syntax

\[
\text{T0raLob} = \text{class}(\text{TCompressedBlob});
\]

Remarks

T0raLob is a descendant of the TBlob class. It holds value of BLOB and CLOB fields and parameters.

Note: You can affect performance of reading/writing LOBs by changing MemData.DefaultPieceSize variable to a different value. DefaultPieceSize defines the size of data portion transferred through network at a single OCI call.

Inheritance Hierarchy

TSharedObject
  \[
  \text{TBlob}
  \]
  \[
  \text{TCompressedBlob}
  \]
  \[
  \text{T0raLob}
  \]

See Also

- TBlob
- T0raDataSet.GetLob
- T0raParam.AsBLOBLocator
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsString</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td><strong>AsWideString</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td><strong>Cached</strong></td>
<td>Used to indicate where the LOB data is.</td>
</tr>
<tr>
<td><strong>Compressed</strong> (inherited from <strong>TCompressedBlob</strong>)</td>
<td>Used to indicate if the Blob is compressed.</td>
</tr>
<tr>
<td><strong>CompressedSize</strong> (inherited from <strong>TCompressedBlob</strong>)</td>
<td>Used to indicate compressed size of the Blob data.</td>
</tr>
<tr>
<td><strong>IsUnicode</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
<tr>
<td><strong>OCILobLocator</strong></td>
<td>Used to get or set the OCILobLocator handle.</td>
</tr>
<tr>
<td><strong>OCILobLocatorPtr</strong></td>
<td>Used to retrieve the LOB value type locator.</td>
</tr>
<tr>
<td><strong>OCISvcCtx</strong></td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td><strong>RefCount</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td><strong>Size</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to learn the size of the TBlob value in bytes.</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocLob</strong></td>
<td>Allocates and initializes the LOB locator.</td>
</tr>
<tr>
<td><strong>Assign</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td><strong>Clear</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Deletes the current value in</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateTemporary</td>
<td>Allocates and initializes a temporary LOB locator of the specified type.</td>
</tr>
<tr>
<td>DisableBuffering</td>
<td>Disables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td>EnableBuffering</td>
<td>Enables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td>FreeLob</td>
<td>Releases the LOB locator descriptor.</td>
</tr>
<tr>
<td>FreeTemporary</td>
<td>Frees a temporary LOB.</td>
</tr>
<tr>
<td>Init</td>
<td>Initializes the OCILobLocator handle.</td>
</tr>
<tr>
<td>IsInit</td>
<td>Verifies if the LOB locator is initialized.</td>
</tr>
<tr>
<td>IsTemporary</td>
<td>Indicates whether the LOB is temporary.</td>
</tr>
<tr>
<td>LengthLob</td>
<td>Returns the number of bytes contained in the LOB object.</td>
</tr>
<tr>
<td>LoadFromFile</td>
<td>Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td>Read</td>
<td>Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
<tr>
<td>ReadLob</td>
<td>Reads the LOB content from the server.</td>
</tr>
<tr>
<td>Release</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>SaveToFile</td>
<td>Saves the contents of the TBlob object to a file.</td>
</tr>
<tr>
<td>SaveToStream</td>
<td>Copies the contents of a TBlob object to a stream.</td>
</tr>
<tr>
<td>Truncate</td>
<td>Sets new TBlob size and discards all data over it.</td>
</tr>
<tr>
<td>Write</td>
<td>Stores a raw sequence of bytes into a TBlob object.</td>
</tr>
<tr>
<td>WriteLob</td>
<td>Writes a LOB value to the server.</td>
</tr>
</tbody>
</table>
Properties of the **TOraLob** class.

For a complete list of the **TOraLob** class members, see the **TOraLob Members** topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsString</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as string.</td>
</tr>
<tr>
<td><strong>AsWideString</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to manipulate BLOB value as Unicode string.</td>
</tr>
<tr>
<td><strong>Cached</strong></td>
<td>Used to indicate where the LOB data is.</td>
</tr>
<tr>
<td><strong>Compressed</strong> (inherited from <strong>TCompressedBlob</strong>)</td>
<td>Used to indicate if the Blob is compressed.</td>
</tr>
<tr>
<td><strong>CompressedSize</strong> (inherited from <strong>TCompressedBlob</strong>)</td>
<td>Used to indicate compressed size of the Blob data.</td>
</tr>
<tr>
<td><strong>IsUniverse</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Gives choice of making TBlob store and process data in Unicode format or not.</td>
</tr>
<tr>
<td><strong>OCILobLocator</strong></td>
<td>Used to get or set the OCILobLocator handle.</td>
</tr>
<tr>
<td><strong>OCILobLocatorPtr</strong></td>
<td>Used to retrieve the LOB value type locator.</td>
</tr>
<tr>
<td><strong>OCISvcCtx</strong></td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td><strong>RefCount</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td><strong>Size</strong> (inherited from <strong>TBlob</strong>)</td>
<td>Used to learn the size of the TBlob value in bytes.</td>
</tr>
</tbody>
</table>

### See Also

- **TOraLob Class**
- **TOraLob Class Members**

© 1997-2021 Devart. All Rights Reserved.
5.19.1.5.2.1 Cached Property

Used to indicate where the LOB data is.

Class

TOraLob

Syntax

```
property Cached: boolean;
```

Remarks

Use the Cached property to indicate whether the LOB data is cached on a client or it is accessed remotely on the server. In most cases you don't need to set the value of this property directly. To enable or disable LOB caching use the `TOraDataSet.Options` property.

See Also

- `TOraDataSet.Options`

© 1997-2021 Devart. All Rights Reserved.

5.19.1.5.2.2 OCILobLocator Property

Used to get or set the OCILobLocator handle.

Class

TOraLob

Syntax

```
property OCILobLocator: pOCILobLocator;
```

Remarks

Use the OCILobLocator property to get or set the OCILobLocator handle.
5.19.1.5.2.3 OCILobLocatorPtr Property

Used to retrieve the LOB value type locator.

Class

T0raLob

Syntax

```property OCILobLocatorPtr: ppOCILobLocator;```

Remarks

Use the OCILobLocatorPtr property to retrieve the LOB value type locator.

5.19.1.5.2.4 OCISvcCtx Property

Used to assign a service context handle.

Class

T0raLob

Syntax

```property OCISvcCtx: TOCISvcCtx;```

Remarks

Use the OCISvcCtx property to assign a service context handle. Some operations with LOBs require service context handle. To get a service context handle use T0raSession.OCISvcCtx.

See Also

- T0raSession.OCISvcCtx
Methods of the **TOraLob** class.

For a complete list of the **TOraLob** class members, see the [TOraLob Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocLob</td>
<td>Allocates and initializes the LOB locator.</td>
</tr>
<tr>
<td>Assign (inherited from <strong>TBlob</strong>)</td>
<td>Sets BLOB value from another TBlob object.</td>
</tr>
<tr>
<td>Clear (inherited from <strong>TBlob</strong>)</td>
<td>Deletes the current value in TBlob object.</td>
</tr>
<tr>
<td>CreateTemporary</td>
<td>Allocates and initializes a temporary LOB locator of the specified type.</td>
</tr>
<tr>
<td>DisableBuffering</td>
<td>Disables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td>EnableBuffering</td>
<td>Enables the LOB buffering for the LOB locator.</td>
</tr>
<tr>
<td>FreeLob</td>
<td>Releases the LOB locator descriptor.</td>
</tr>
<tr>
<td>FreeTemporary</td>
<td>Frees a temporary LOB.</td>
</tr>
<tr>
<td>Init</td>
<td>Initializes the OCILOBLocator handle.</td>
</tr>
<tr>
<td>IsInit</td>
<td>Verifies if the LOB locator is initialized.</td>
</tr>
<tr>
<td>IsTemporary</td>
<td>Indicates whether the LOB is temporary.</td>
</tr>
<tr>
<td>LengthLob</td>
<td>Returns the number of bytes contained in the LOB object.</td>
</tr>
<tr>
<td>LoadFromFile (inherited from <strong>TBlob</strong>)</td>
<td>Loads the contents of a file into a TBlob object.</td>
</tr>
<tr>
<td>LoadFromStream (inherited from <strong>TBlob</strong>)</td>
<td>Copies the contents of a stream into the TBlob object.</td>
</tr>
<tr>
<td>Read (inherited from <strong>TBlob</strong>)</td>
<td>Acquires a raw sequence of bytes from the data stored in TBlob.</td>
</tr>
</tbody>
</table>
### ReadLob
Reads the LOB content from the server.

### Release (inherited from TSharedObject)
Decrements the reference count.

### SaveToFile (inherited from TBlob)
Saves the contents of the TBlob object to a file.

### SaveToStream (inherited from TBlob)
Copies the contents of a TBlob object to a stream.

### Truncate (inherited from TBlob)
Sets new TBlob size and discards all data over it.

### Write (inherited from TBlob)
Stores a raw sequence of bytes into a TBlob object.

### WriteLob
Writes a LOB value to the server.

### See Also
- **TOraLob Class**
- **TOraLob Class Members**

### AllocLob Method
Allocates and initializes the LOB locator.

#### Class
**TOraLob**

#### Syntax
```pascal
procedure AllocLob; virtual;
```

#### Remarks
Call the AllocLob method to allocate and initialize the LOB locator.
5.19.1.5.3.2 CreateTemporary Method

Allocates and initializes a temporary LOB locator of the specified type.

Class

T0raLob

Syntax

procedure CreateTemporary(LobType: TLobType);

Parameters

LobType
  Holds the type of the temporary LOB locator.

Remarks

Call the CreateTemporary method to allocate and initialize a temporary LOB locator of the specified type.

5.19.1.5.3.3 DisableBuffering Method

Disables the LOB buffering for the LOB locator.

Class

T0raLob

Syntax

procedure DisableBuffering;

Remarks

Call the DisableBuffering method to disable the LOB buffering for the LOB locator. The next time data is read from or written to the LOB through the input locator, the LOB buffering subsystem is not used.

See Also

- EnableBuffering
5.19.1.5.3.4 EnableBuffering Method

Enables the LOB buffering for the LOB locator.

Class
T0raLob

Syntax

```
procedure EnableBuffering;
```

Remarks

Call the EnableBuffering method to enable the LOB buffering for the LOB locator. The next time data is read from or written to the LOB through the input locator, the LOB buffering subsystem is used.

See Also

- DisableBuffering

5.19.1.5.3.5 FreeLob Method

Releases the LOB locator descriptor.

Class
T0raLob

Syntax

```
procedure FreeLob;
```

Remarks

Call the FreeLob method to release the LOB locator descriptor.
5.19.1.5.3.6 FreeTemporary Method

Frees a temporary LOB.

Class

TOraLob

Syntax

procedure FreeTemporary;

Remarks

Use the FreeTemporary method to free a temporary LOB.

See Also

• FreeLob

© 1997-2021 Devart. All Rights Reserved.

5.19.1.5.3.7 Init Method

Initializes the OCILobLocator handle.

Class

TOraLob

Syntax

procedure Init;

Remarks

Call the Init method to initialize the OCILobLocator handle.

© 1997-2021 Devart. All Rights Reserved.

5.19.1.5.3.8 IsInit Method

Verifies if the LOB locator is initialized.

Class
**T0raLob**

**Syntax**

```pascal
function IsInit: boolean;
```

**Return Value**

True, if the LOB locator is initialized. False otherwise.

**Remarks**

Call the IsInit method to verify that the LOB locator is initialized.

© 1997-2021 Devart. All Rights Reserved.

**5.19.1.5.3.9 IsTemporary Method**

Indicates whether the LOB is temporary.

**Class**

**T0raLob**

**Syntax**

```pascal
function IsTemporary: LongBool;
```

**Return Value**

True, if the LOB is temporary. False otherwise.

**Remarks**

Call the IsTemporary method to indicate whether the LOB is temporary.

© 1997-2021 Devart. All Rights Reserved.

**5.19.1.5.3.10 LengthLob Method**

Returns the number of bytes contained in the LOB object.

**Class**

**T0raLob**
Syntax

**function** LengthLob: Cardinal;

**Return Value**
the LOB object size in bytes.

**Remarks**
Call the LengthLob method to return the number of bytes contained in the LOB object.

© 1997-2021 Devart. All Rights Reserved.

5.19.1.5.3.11 ReadLob Method

Reads the LOB content from the server.

**Class**
**TOraLob**

**Syntax**

**procedure** ReadLob(*var* SharedPiece: PPieceHeader; *PrefetchLobSize*: Integer); **overload:** **procedure** ReadLob; **overload**;

**Remarks**
Call the ReadLob method to get the LOB content from the server. When reading such properties as AsString or AsWideString this method is called automatically.

© 1997-2021 Devart. All Rights Reserved.

5.19.1.5.3.12 WriteLob Method

Writes a LOB value to the server.

**Class**
**TOraLob**

**Syntax**

**procedure** writeLob;
Remarks
Call the WriteLob method to write a LOB value to the server.

5.19.1.6 TOraNumber Class

A class supporting

For a list of all members of this type, see TOraNumber members.

Unit
oraClasses

Syntax

TOraNumber = class(TSharedObject);

Remarks
TOraNumber is used to support the Oracle numbers in native format. Support of the internal Oracle number format on clients allows to use full number precision without accuracy losses.

Inheritance Hierarchy
TSharedObject
TOraNumber

© 1997-2021 Devart. All Rights Reserved.

5.19.1.6.1 Members

TOraNumber class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsFloat</td>
<td>Used to get and set the number value as float.</td>
</tr>
<tr>
<td>AsInteger</td>
<td>Used to get and set the number value as integer.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AsLargeInt</td>
<td>Used to get and set the number value as Int64.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to get and set the number value as string.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to set the null number value flag.</td>
</tr>
<tr>
<td>OCINumber</td>
<td>Used to get or set the OCINumber opaque structure value.</td>
</tr>
<tr>
<td>OCINumberPtr</td>
<td>Used to get a pointer to the OCINumber structure.</td>
</tr>
<tr>
<td>RefCount</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AssignTo</td>
<td>Copies all properties of the current TOraNumber instance to the Dest TOraNumber instance.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares the current TOraNumber value with the Dest value.</td>
</tr>
<tr>
<td>Release</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AsFloat</td>
<td>Used to get and set the number value as float.</td>
</tr>
<tr>
<td>AsInteger</td>
<td>Used to get and set the number value as integer.</td>
</tr>
<tr>
<td>AsLargeInt</td>
<td>Used to get and set the number value as Int64.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to get and set the number value as string.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to set the null number value flag.</td>
</tr>
<tr>
<td>OCINumber</td>
<td>Used to get or set the OCINumber opaque structure value.</td>
</tr>
<tr>
<td>OCINumberPtr</td>
<td>Used to get a pointer to the OCINumber structure.</td>
</tr>
<tr>
<td>RefCount</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

See Also

- TOraNumber Class
- TOraNumber Class Members

5.19.1.6.2.1 AsFloat Property

Used to get and set the number value as float.

**Class**

TOraNumber

**Syntax**

```delphi
property AsFloat: double;
```

**Remarks**

Use the AsFloat property to get and set the number value as float.

Reading AsFloat property when IsNull is True returns 0.
5.19.1.6.2.2  AsInteger Property

Used to get and set the number value as integer.

**Class**

*TOraNumber*

**Syntax**

```plaintext
property AsInteger: Integer;
```

**Remarks**

Use the AsInteger property to get and set the number value as integer.

Reading AsInteger property when **IsNull** is True returns 0.

**See Also**

- **IsNull**

© 1997-2021 Devart. All Rights Reserved.

5.19.1.6.2.3  AsLargeInt Property

Used to get and set the number value as Int64.

**Class**

*TOraNumber*

**Syntax**

```plaintext
property AsLargeInt: Int64;
```

**Remarks**

Use the AsLargeInt property to get and set the number value as Int64.

© 2021 Devart
Reading `AsLargeInt` property when `IsNull` is True returns 0.

See Also
- `IsNull`

5.19.1.6.2.4 AsString Property

Used to get and set the number value as string.

Class

`T0raNumber`

Syntax

```plaintext
property AsString: string;
```

Remarks

Use the `AsString` property to get and set the number value as string.

Reading `AsString` property when `IsNull` is True returns empty string.

See Also
- `IsNull`

5.19.1.6.2.5 IsNull Property

Used to set the null number value flag.

Class

`T0raNumber`

Syntax

```plaintext
property IsNull: boolean;
```

Remarks
Use the IsNull property to get or set the null number value flag.

5.19.1.6.2.6 OCINumber Property

Used to get or set the OCINumber opaque structure value.

Class

TOraNumber

Syntax

```property``
``OCINumber: OCINumber;``

Remarks

Use the OCINumber property to get or set the OCINumber opaque structure value.

5.19.1.6.2.7 OCINumberPtr Property

Used to get a pointer to the OCINumber structure.

Class

TOraNumber

Syntax

```property``
``OCINumberPtr: pOCINumber;``

Remarks

Use the OCINumberPtr property to get a pointer to the OCINumber structure.

See Also

- OCINumber
5.19.1.6.3 Methods

Methods of the **TOraNumber** class.

For a complete list of the **TOraNumber** class members, see the **TOraNumber Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AssignTo</td>
<td>Copies all properties of the current TOraNumber instance to the Dest TOraNumber instance.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares the current TOraNumber value with the Dest value.</td>
</tr>
<tr>
<td>Release (inherited from <strong>TSharedObject</strong>)</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

See Also

- **TOraNumber Class**
- **TOraNumber Class Members**

© 1997-2021 Devart. All Rights Reserved.

5.19.1.6.3.1 AssignTo Method

Copies all properties of the current TOraNumber instance to the Dest TOraNumber instance.

Class

**TOraNumber**

Syntax

```
procedure AssignTo(Dest: TOraNumber);
```

Parameters
Dest
Holds a Dest TOraNumber instance.

Remarks
Call AssignTo method to copy all properties of the current TOraNumber instance to the Dest TOraNumber instance.

5.19.1.6.3.2 Compare Method

Compared the current TOraNumber value with the Dest value.

Class
TOraNumber

Syntax

```pascal
function Compare(Dest: TOraNumber): integer;
```

Parameters

Dest
Holds the Dest value to compare the current TOraNumber value with.

Return Value
a negative value if current number is less than Dest, zero if numbers are equal, and a positive value if the current number is greater than Dest.

Remarks
Call the Compare method to compare the current TOraNumber value with the Dest value.

Returns a negative value if current number is less than Dest, zero if numbers are equal, and a positive value if the current number is greater than Dest.

5.19.1.7 TOraTimeStamp Class

A class supporting Oracle 9 timestamp datatypes.

For a list of all members of this type, see TOraTimeStamp members.
Unit
oraClasses

Syntax

TOraTimeStamp = class(TSharedObject);

Remarks

TOraTimeStamp is used to support Oracle 9 timestamp datatypes. There are three timestamp datatypes:

TIMESTAMP,

TIMESTAMP WITH TIME ZONE,

TIMESTAMP WITH LOCAL TIME ZONE.

They can be distinguished by DescriptorType property.

Inheritance Hierarchy

TSharedObject
   TOraTimeStamp

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsDateTime</td>
<td>Used to get and set the timestamp value as TDateTime.</td>
</tr>
<tr>
<td>AsString</td>
<td>Used to get and set the timestamp value as string.</td>
</tr>
<tr>
<td>DescriptorType</td>
<td>Defines the type of the descriptor allocated to operate with the interval value.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Format</td>
<td>Used to get or set the date-time format model for operations with the TOraTimeStamp.AsString property.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to get or set the null timestamp value flag.</td>
</tr>
<tr>
<td>OCIDateTime</td>
<td>Used to set the OCIDateTime descriptor handle.</td>
</tr>
<tr>
<td>OCIDateTimePtr</td>
<td>Used to provide the OCIDateTime type locator.</td>
</tr>
<tr>
<td>Precision</td>
<td>Used to learn or set the precision with which the fractional seconds are returned.</td>
</tr>
<tr>
<td>RefCount (inherited from TSharedObject)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>TimeZone</td>
<td>Used to provide a time zone name portion of the datetime value.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from TSharedObject)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocDateTime</td>
<td>Allocates the OCIDateTime descriptor handle according to the TOraTimeStamp.Descriptor Type property.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares the current TOraTimeStamp value with the Dest value.</td>
</tr>
<tr>
<td>Construct</td>
<td>Constructs datetime descriptor.</td>
</tr>
<tr>
<td>GetDate</td>
<td>Provides a date portion of the datetime value.</td>
</tr>
<tr>
<td>GetTime</td>
<td>Provides a time portion of the datetime value.</td>
</tr>
</tbody>
</table>
### GetTimeZoneOffset
- Provides a time zone portion of the datetime value.

### Release
- (inherited from `TSharedObject`)
- Decrements the reference count.

### SetDate
- Provides a date portion of the datetime value.

### SetTime
- Sets the time into a datetime value.

### SetTimeZoneOffset
- Provides a time zone portion of the datetime value.

---

5.19.1.7.2 Properties

Properties of the `TOraTimeStamp` class.

For a complete list of the `TOraTimeStamp` class members, see the [TOraTimeStamp Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsDateTime</code></td>
<td>Used to get and set the timestamp value as <code>TDateTime</code>.</td>
</tr>
<tr>
<td><code>AsString</code></td>
<td>Used to get and set the timestamp value as string.</td>
</tr>
<tr>
<td><code>DescriptorType</code></td>
<td>Defines the type of the descriptor allocated to operate with the interval value.</td>
</tr>
<tr>
<td><code>Format</code></td>
<td>Used to get or set the date-time format model for operations with the <code>TOraTimeStamp.AsString</code> property.</td>
</tr>
<tr>
<td><code>IsNull</code></td>
<td>Used to get or set the null timestamp value flag.</td>
</tr>
<tr>
<td><code>OCIDateTime</code></td>
<td>Used to set the OCIDateTime descriptor handle.</td>
</tr>
</tbody>
</table>
**OCIDateTimePtr**

Used to provide the OCIDateTime type locator.

**Precision**

Used to learn or set the precision with which the fractional seconds are returned.

**RefCount** (inherited from TSharedObject)

Used to return the count of reference to a TSharedObject object.

**TimeZone**

Used to provide a time zone name portion of the datetime value.

See Also
- TOraTimeStamp Class
- TOraTimeStamp Class Members

© 1997-2021
Devart. All Rights Reserved.

5.19.1.7.2.1 AsDateTime Property

Used to get and set the timestamp value as TDateTime.

### Class

**TOraTimeStamp**

### Syntax

```property
AsDateTime: TDateTime;
```

### Remarks

Use the AsDateTime property to get and set the timestamp value as TDateTime.

Reading the AsDateTime property when `IsNull` is True returns 0.

See Also
- `IsNull`

© 1997-2021
Devart. All Rights Reserved.
5.19.1.7.2.2 AsString Property

Used to get and set the timestamp value as string.

Class

T0raTimeStamp

Syntax

property AsString: string;

Remarks

Use the AsString property to get and set the timestamp value as string. Format of the string is specified by the Format property.

Reading the AsString property when IsNull is True returns empty string.

See Also

- Format
- IsNull

5.19.1.7.2.3 DescriptorType Property

Defines the type of the descriptor allocated to operate with the interval value.

Class

T0raTimeStamp

Syntax

property DescriptorType: cardinal;

Remarks

Use the DescriptorType property to define the type of the descriptor that is allocated to operate with the interval value. In most cases you don't need to set this property directly: it is adjusted automatically when working with fields and parameters.

Valid values for this property are:
OCI_DTYPE_TIMESTAMP,
OCI_DTYPE_TIMESTAMP_TZ,
OCI_DTYPE_TIMESTAMP_LTZ

They are defined in the OraCall unit.

5.19.1.7.2.4 Format Property

Used to get or set the date-time format model for operations with the `AsString` property.

Class

`TOraTimeStamp`

Syntax

```delphi
property Format: string;
```

Remarks

Use the Format property to get or set the date-time format model for operations with the `AsString` property. For example, the date format model for string ‘17:54:23’ is ‘HH24:MM:SS’.

For complete description of the date-time models refer to the Oracle documentation.

See Also

- `AsString`

5.19.1.7.2.5 IsNull Property

Used to get or set the null timestamp value flag.

Class

`TOraTimeStamp`

Syntax

```delphi
property IsNull: boolean;
```
Remarks

Use the IsNull property to get or set the null timestamp value flag. When IsNull is true getting values via the GetDate, GetTime and SetTimeZoneOffset methods can raise an exception. TheAsString property in this case returns empty string. The AsDateTime property returns 0.

See Also

- GetDate
- GetTime
- SetTimeZoneOffset
-AsString
-AsDateTime

© 1997-2021
Devart. All Rights Reserved.
5.19.1.7.2.8 Precision Property

Used to learn or set the precision with which the fractional seconds are returned.

Class

TOraTimeStamp

Syntax

[property] Precision: byte;

Remarks

Use the Precision property to get or set the precision in which the fractional seconds are returned when reading the AsString property. The default value of the property is 6.

See Also

• AsString

5.19.1.7.2.9 TimeZone Property

Used to provide a time zone name portion of the datetime value.

Class

TOraTimeStamp

Syntax

[property] TimeZone: string;
Remarks
Use the TimeZone property to get a time zone name portion of the datetime value.

© 1997-2021 Devart. All Rights Reserved.

5.19.1.7.3 Methods

Methods of the TOraTimeStamp class.

For a complete list of the TOraTimeStamp class members, see the TOraTimeStamp Members topic.

Protected

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllocDateTime</td>
<td>Allocates the OCIDateTime descriptor handle according to the TOraTimeStamp.Descriptor Type property.</td>
</tr>
</tbody>
</table>

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from TSharedObject)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>Compare</td>
<td>Compares the current TOraTimeStamp value with the Dest value.</td>
</tr>
<tr>
<td>Construct</td>
<td>Constructs datetime descriptor.</td>
</tr>
<tr>
<td>GetDate</td>
<td>Provides a date portion of the datetime value.</td>
</tr>
<tr>
<td>GetTime</td>
<td>Provides a time portion of the datetime value.</td>
</tr>
<tr>
<td>GetTimeZoneOffset</td>
<td>Provides a time zone portion of the datetime value.</td>
</tr>
<tr>
<td>Release (inherited from TSharedObject)</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>SetDate</td>
<td>Provides a date portion of the datetime value.</td>
</tr>
<tr>
<td>SetTime</td>
<td>Sets the time into a datetime value.</td>
</tr>
<tr>
<td>SetTimeZoneOffset</td>
<td>Provides a time zone portion of the datetime value.</td>
</tr>
</tbody>
</table>

See Also
- TOraTimeStamp Class
- TOraTimeStamp Class Members

5.19.1.7.3.1 AllocDateTime Method

Allocates the OCIDateTime descriptor handle according to the DescriptorType property.

Class
TOraTimeStamp

Syntax

```pascal
procedure AllocDateTime;
```

Remarks

Use the AllocDateTime method to allocate the OCIDateTime descriptor handle according to the DescriptorType property.

See Also
- DescriptorType

5.19.1.7.3.2 Compare Method

Compares the current TOraTime Stamp value with the Dest value.

Class
TOraTimeStamp
Syntax

```pascal
function Compare(Dest: TOraTimeStamp): integer;
```

**Parameters**

`Dest`
Holds the Dest value.

**Return Value**
negative value if the current timestamp is less than Dest, zero if timestamp is equal, and positive value if the current timestamp is greater than Dest.

**Remarks**
Call the Compare method to compare the current TOraTimeStamp value with the Dest value.
Returns negative value if current timestamp is less than Dest, zero if timestamp are equal and positive value if current timestamp is greater than Dest.

5.19.1.7.3.3  Construct Method

Constructs datetime descriptor.

**Class**

`TOraTimeStamp`

Syntax

```pascal
procedure Construct(Year: smallint; Month: byte; Day: byte; Hour: byte; Min: byte; Sec: byte; FSec: cardinal; TimeZone: string);
```

**Parameters**

`Year`
Holds the year.

`Month`
Holds the month.

`Day`
Holds the day.

`Hour`
Holds the hour.

`Min`
Holds the minute.
Holds the minute.

Sec
Holds the second.

FSec
Holds the fraction second.

TimeZone
Holds the time zone name.

Remarks
Call the Construct method to construct datetime descriptor. When calling the Construct procedure only relevant fields based on the datetime type are used. For the type with time zone, the date and time fields are assumed to be in the local time of the specified time zone. If the time zone is not specified, then session default time zone is assumed.

See Also
• SetDate
• SetTime
• SetTimeZoneOffset

Provides a date portion of the datetime value.

Class
TOraTimeStamp

Syntax

```pascal
procedure GetDate(var Year: smallint; var Month: byte; var Day: byte);
```

Parameters

Year
Holds the year.

Month
Holds the month.

Day
Holds the day.
Remarks

Call the GetDate method to get a date (year, month, day) portion of the datetime value.

See Also

- GetTime
- SetTimeZoneOffset

Provides a time portion of the datetime value.

Class

TOraTimeStamp

Syntax

```pascal
procedure GetTime(var Hour: byte; var Min: byte; var Sec: byte; var FSec: cardinal);
```

Parameters

- **Hour**
  - Holds the hour.
- **Min**
  - Holds the minute.
- **Sec**
  - Holds the second.
- **FSec**
  - Holds the fraction second.

Remarks

Call the GetTime method to get a time (hour, minute, second, fractional second) of the datetime value.

See Also

- GetDate
- SetTimeZoneOffset
5.19.1.7.6  GetTimeZoneOffset Method

Provides a time zone portion of the datetime value.

Class

TOraTimeStamp

Syntax

procedure GetTimeZoneOffset(var TZHour: shortint; var TZMin: shortint);

Parameters

TZHour
TZMin

Remarks

Call the GetTimeZoneOffset method to get a time zone (hour, minute) portion of the datetime value.

See Also

- GetDate
- GetTime

5.19.1.7.7  SetDate Method

Provides a date portion of the datetime value.

Class

TOraTimeStamp

Syntax

procedure SetDate(Year: smallint; Month: byte; Day: byte);
**Parameters**

*Year*
- Holds the year.

*Month*
- Holds the month.

*Day*
- Holds the day.

**Remarks**

Call the SetDate method to set a date (year, month, day) portion in a datetime value.

**See Also**
- [Construct](#)

© 1997-2021 Devart. All Rights Reserved.

Sets the time into a datetime value.

**Class**

*TOraTimeStamp*

**Syntax**

```plaintext
procedure SetTime(Hour: byte; Min: byte; Sec: byte; FSec: cardinal);
```

**Parameters**

*Hour*
- Holds the hour.

*Min*
- Holds the minute.

*Sec*
- Holds the second.

*FSec*
- Holds the fraction second.

**Remarks**
Call the SetTime method to set the time (hour, minute, second, fractional second) into a datetime value.

See Also
• Construct

5.19.1.7.3.9 SetTimeZoneOffset Method

Provides a time zone portion of the datetime value.

Class
TOraTimeStamp

Syntax

procedure SetTimeZoneOffset(TZHour: shortint; TZMin: shortint);

Parameters
TZHour
Holds the hour.
TZMin
Holds the minute.

Remarks
Call the SetTimeZoneOffset method to set a time zone (hour, minute) portion of the datetime value.

See Also
• Construct

5.19.2 Enumerations

Enumerations in the OraClasses unit.

Enumerations
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TChangeNotifyEventType</td>
<td>Contains the event type.</td>
</tr>
<tr>
<td>TConnectMode</td>
<td>Specifies the system privileges used when a user connects to the server.</td>
</tr>
<tr>
<td>TMessageType</td>
<td>Defines the type of the next message found in the received named pipe local buffer.</td>
</tr>
<tr>
<td>TOptimizerMode</td>
<td>Specifies the optimizer mode for connection.</td>
</tr>
</tbody>
</table>

### 5.19.2.1 TChangeNotifyEventType Enumeration

Contains the event type.

**Unit**
oraClasses

**Syntax**

```
TChangeNotifyEventType = (cneNone, cneStartup, cneShutdown, cneShutdownAny, cneDropDB, cneDereg, cneObjChange, cneQueryChange);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>cneDereg</td>
<td>Registration has been removed.</td>
</tr>
<tr>
<td>cneDropDB</td>
<td>Database has been dropped.</td>
</tr>
<tr>
<td>cneNone</td>
<td>No further information about the continuous query notification.</td>
</tr>
<tr>
<td>cneObjChange</td>
<td>Object change notification.</td>
</tr>
<tr>
<td>cneQueryChange</td>
<td>Query change notification.</td>
</tr>
<tr>
<td>cneShutdown</td>
<td>Instance shutdown notification.</td>
</tr>
<tr>
<td>cneShutdownAny</td>
<td>Any instance shutdown when running RAC.</td>
</tr>
<tr>
<td>cneStartup</td>
<td>Instance startup notification.</td>
</tr>
</tbody>
</table>
5.19.2.2 TConnectMode Enumeration

Specifies the system privileges used when a user connects to the server.

Unit
oraClasses

Syntax

TConnectMode = (cmNormal, cmSysOper, cmSysDBA, cmSysASM);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmNormal</td>
<td>Connect as an ordinary user. The default value.</td>
</tr>
<tr>
<td>cmSysASM</td>
<td>Connect with the SYSASM role.</td>
</tr>
<tr>
<td>cmSysDBA</td>
<td>Connect with the SYSDBA role.</td>
</tr>
<tr>
<td>cmSysOper</td>
<td>Connect with the SYSOPER role.</td>
</tr>
</tbody>
</table>

5.19.2.3 TMessageType Enumeration

Defines the type of the next message found in the received named pipe local buffer.

Unit
oraClasses

Syntax

TMessageType = (mtNone, mtNumber, mtString, mtDate);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>mtDate</td>
<td>Indicates that the messages found in the local buffer are of the Date type.</td>
</tr>
<tr>
<td>mtNone</td>
<td>Indicates that no more messages are found in the local buffer.</td>
</tr>
<tr>
<td>mtNumber</td>
<td>Indicates that the messages found in the local buffer are of the Number type.</td>
</tr>
<tr>
<td>mtString</td>
<td>Indicates that the messages found in the local buffer are of the</td>
</tr>
</tbody>
</table>
### 5.19.2.4 `TOptimizerMode` Enumeration

Specifies the optimizer mode for connection.

**Unit**

`oraClasses`

**Syntax**

```
TOptimizerMode = (omDefault, omFirstRows1000, omFirstRows100, omFirstRows10, omFirstRows1, omFirstRows, omAllRows, omChoose, omRule);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>omAllRows</code></td>
<td>Explicitly chooses the cost-based approach to optimize a statement block with a goal of best throughput (that is minimum total resource consumption).</td>
</tr>
<tr>
<td><code>omChoose</code></td>
<td>Causes the optimizer to choose between the rule-based and cost-based approaches for a SQL statement. The optimizer selection is based on the presence of statistics for the tables accessed by the statement. If the data dictionary has statistics for at least one of these tables, then the optimizer uses the cost-based approach and optimizes with the goal of the best throughput. If the data dictionary does not have statistics for these tables, then it uses the rule-based approach.</td>
</tr>
<tr>
<td><code>omDefault</code></td>
<td>Session optimizer mode will not be changed.</td>
</tr>
<tr>
<td><code>omFirstRows</code></td>
<td>This mode is retained for backward compatibility and plan stability. It optimizes for the best plan to return the first single row.</td>
</tr>
<tr>
<td><code>omFirstRows1</code></td>
<td>Instruct Oracle to optimize a SQL statement for fast response. It instructs Oracle to choose the plan that returns the first row most efficiently. If you use the version server lower than Oracle 9.0, these values have the same effect as <code>omFirstRows</code>.</td>
</tr>
<tr>
<td><code>omFirstRows10</code></td>
<td>Instruct Oracle to optimize an SQL statement for fast response. It instructs Oracle to choose the plan that returns the first 10 rows most efficiently. If you use the version server lower than Oracle 9.0, these values have the same effect as <code>omFirstRows</code>.</td>
</tr>
</tbody>
</table>
**omFirstRows100**  
Instruct Oracle to optimize an SQL statement for fast response. It instructs Oracle to choose the plan that returns the first 100 rows most efficiently. If you use the version server lower than Oracle 9.0, these values have the same effect as `omFirstRows`.

**omFirstRows1000**  
Instruct Oracle to optimize an SQL statement for fast response. It instructs Oracle to choose the plan that returns the first 1000 rows most efficiently. If you use the version server lower than Oracle 9.0, these values have the same effect as `omFirstRows`.

**omRule**  
Chooses rule-based optimization (RBO). Any other value causes the optimizer to choose cost-based optimization (CBO). The rule-based optimizer is the archaic optimizer mode from the earliest releases of Oracle Database.

### 5.19.3 Variables

Variables in the **OraClasses** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FloatPrecision</td>
<td>Set this constant to define the type of NUMBER fields with Scale &gt; 0.</td>
</tr>
<tr>
<td>IntegerPrecision</td>
<td>Set this constant to define the type of NUMBER fields with precision less than or equal to IntegerPrecision as dtInteger.</td>
</tr>
<tr>
<td>LargeIntPrecision</td>
<td>Set this constant to define the type of NUMBER fields with precision greater than the IntegerPresision and less than or equal to LargeIntPrecision as dtLargeInt.</td>
</tr>
</tbody>
</table>
5.19.3.1 FloatPrecision Variable

Set this constant to define the type of NUMBER fields with Scale > 0.

Unit
oraClasses

Syntax

```plaintext
FloatPrecision: integer = 15;
```

Remarks

Set this constant to define the type of NUMBER fields with Scale > 0. If its precision is less than or equal to FloatPrecision the field will be defined as TFloatField. Otherwise it will be defined as TOraNumberField (if TOraSessionOptions.EnableNumbers option is set to True).

According to these constants and TOraSessionOptions.EnableIntegers and TOraSessionOptions.EnableNumbers options, Oracle NUMBER type is mapped to ODAC field classes in the following way:

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Field class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision &lt;= IntegerPrecision, Scale = 0,</td>
<td>TIntegerField</td>
</tr>
<tr>
<td>TOraSessionOptions.EnableIntegers = True</td>
<td></td>
</tr>
<tr>
<td>IntegerPrecision &lt; Precision &lt;=</td>
<td>TLargeIntField</td>
</tr>
<tr>
<td>LargeIntPrecision, Scale = 0,</td>
<td></td>
</tr>
<tr>
<td>TOraSessionOptions.EnableIntegers = True</td>
<td></td>
</tr>
<tr>
<td>Precision &gt; FloatPrecision, Scale &gt; 0,</td>
<td>TOraNumberField</td>
</tr>
<tr>
<td>TOraSessionOptions.EnableNumbers = True</td>
<td></td>
</tr>
<tr>
<td>In other cases</td>
<td>TFloatField</td>
</tr>
</tbody>
</table>

Example

```plaintext
FloatPrecision: integer = 15;
```
5.19.3.2 IntegerPrecision Variable

Set this constant to define the type of NUMBER fields with precision less than or equal to IntegerPrecision as dtInteger.

Unit
oraClasses

Syntax

IntegerPrecision: integer = 9;

5.19.3.3 LargeIntPrecision Variable

Set this constant to define the type of NUMBER fields with precision greater than the IntegerPrecision and less than or equal to LargeIntPrecision as dtLargeInt.

Unit
oraClasses

Syntax

LargeIntPrecision: integer = 0;

5.20 OraConnectionPool

This unit contains the TOraConnectionPoolManager class for managing connection pool.

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraPoolingType</td>
<td>Specifies the pool type.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

Request Support  DAC Forum  Provide Feedback
5.20.1 Enumerations

Enumerations in the OraConnectionPool unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraPoolingType</td>
<td>Specifies the pool type.</td>
</tr>
</tbody>
</table>

5.20.1.1 TOraPoolingType Enumeration

Specifies the pool type.

Unit

OraConnectionPool

Syntax

TOraPoolingType = (optLocal, optOCI, optMTS);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>optLocal</td>
<td>Pool is created and controlled by ODAC.</td>
</tr>
<tr>
<td>optMTS</td>
<td>Pool is created and controlled by MTS.</td>
</tr>
<tr>
<td>optOCI</td>
<td>Pool is created and controlled by OCI.</td>
</tr>
</tbody>
</table>

5.21 OraErrHand

This unit contains the TOraErrorHandler component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraErrorHandler</td>
<td>A component allowing translating of error</td>
</tr>
</tbody>
</table>
Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOnOraErrorEvent</td>
<td>This type is used for the TOraErrorHandler.OnError event.</td>
</tr>
</tbody>
</table>

5.21.1 Classes

Classes in the OraErrHand unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraErrorHandler</td>
<td>A component allowing translating of error messages.</td>
</tr>
</tbody>
</table>

5.21.1.1 TOraErrorHandler Class

A component allowing translating of error messages.

For a list of all members of this type, see TOraErrorHandler members.

Unit

OraErrHand

Syntax

TOraErrorHandler = class(TComponent);

Remarks

TOraErrorHandler allows to translate error messages. Messages is stored in error table or can be got out of OnError event handler.
Structure of the error table must be as following:

| ErrorCode | INTEGER |
| Constraint | VARCHAR2 |
| Message | VARCHAR2 |
| SenderName | VARCHAR2 |
| ErrorClass | VARCHAR2 |

You can create and edit error table by ErrorHandler Editor in design time.

See Also
- EOraError

© 1997-2021 Devart. All Rights Reserved.
### Properties

Properties of the `TOraErrorHandler` class.

For a complete list of the `TOraErrorHandler` class members, see the `TOraErrorHandler Members` topic.

#### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>Activates searching messages in error table.</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used for an error message to contain additional information.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>Used to specify the session in which the dataset will be executed.</td>
</tr>
<tr>
<td><strong>TableName</strong></td>
<td>Used to define the error table name.</td>
</tr>
</tbody>
</table>

#### See Also

- `TOraErrorHandler Class`
- `TOraErrorHandler Class Members`

© 1997-2021
Devar. All Rights Reserved.

© 2021 Devar
**TOraErrorHandler**

**Syntax**

```plaintext
property Active: boolean default False;
```

**Remarks**

When the Active property is True, ErrorHandler searches messages in error table.

**See Also**

- Open

© 1997-2021
Devart. All Rights Reserved.

5.21.1.1.2.2  Debug Property

Used for an error message to contain additional information.

**Class**

**TOraErrorHandler**

**Syntax**

```plaintext
property Debug: boolean default False;
```

**Remarks**

When the Debug property is True, error message contains additional information: sender name, constraint name and error code.

**Note:** To use this property you should explicitly include OdacVcl (OdacClx under Linux) unit to your project.

If TOraSQLMonitor is used in the project and the TOraSQLMonitor.Active property is set to False, the debug window is not displayed.
5.21.1.1.2.3 Session Property

Used to specify the session in which the dataset will be executed.

Class
   TOraErrorHandler

Syntax

```property
Session: TOraSession;
```

Remarks

Use the Session property to specify the session in which the dataset will be executed. If Session is not connected, the Open method calls Session.Connect.

See Also
- TOraSession

© 1997-2021 Devart. All Rights Reserved.

5.21.1.1.2.4 TableName Property

Used to define the error table name.

Class
   TOraErrorHandler

Syntax

```property
TableName: string;
```

Remarks

Use the TableName property to determine the error table name. The error table must exist.

© 1997-2021 Devart. All Rights Reserved.
Methods of the TOraErrorHandler class.

For a complete list of the TOraErrorHandler class members, see the TOraErrorHandler Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Close</td>
<td>Sets the Active property of a table to False.</td>
</tr>
<tr>
<td>Open</td>
<td>Sets the Active property to True.</td>
</tr>
</tbody>
</table>

See Also
- TOraErrorHandler Class
- TOraErrorHandler Class Members

Call the Close method to set the Active property of a table to False. When Active is False, the table is closed; the table cannot read data from or write data to the database.

See Also
- Open
5.21.1.3.2 Open Method

Sets the Active property to True.

Class

TOraErrorHandler

Syntax

```pascal
procedure Open;
```

Remarks

Call the Open method to set the Active property for the table to True. When Active is True, data can be read from and written to the database.

See Also

- Close
- Active

5.21.1.4 Events

Events of the TOraErrorHandler class.

For a complete list of the TOraErrorHandler class members, see the TOraErrorHandler Members topic.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Occurs if an appropriate error can not be found in the error table or Active is False.</td>
</tr>
</tbody>
</table>

See Also

- TOraErrorHandler Class
- TOraErrorHandler Class Members

© 1997-2021 Devart
5.21.1.4.1 OnError Event

Occurs if an appropriate error can not be found in the error table or Active is False.

Class

TOraErrorHandler

Syntax

```pascal
property OnError: TOnOraErrorEvent;
```

Remarks

Occurs when ErrorHandler can not find an appropriate error in the error table or Active is False.

See Also

- TCustomDACConnection.OnError

5.21.2 Types

Types in the OraErrHand unit.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOnOraErrorEvent</td>
<td>This type is used for the TOraErrorHandler.OnError event.</td>
</tr>
</tbody>
</table>

5.21.2.1 TOnOraErrorEvent Procedure Reference

This type is used for the TOraErrorHandler.OnError event.
**OraErrHand**

**Syntax**

```
TOnOraErrorEvent = procedure (Sender: TObject; E: Exception; 
ErrorCode: integer; const ConstraintName: string; 
var Msg: string) of object;
```

**Parameters**

- **Sender**
  - An object that raised the event.
- **E**
  - Holds the reference to an exception object.
- **ErrorCode**
  - The code of an error.
- **ConstraintName**
  - Holds the name of the constraint that raised the error.
- **Msg**
  - Holds the error message (can be set individually).

---

**5.22 OraError**

This unit contains the EOraError exception class.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOraError</td>
<td>Raised when a component detects Oracle error.</td>
</tr>
</tbody>
</table>

---

**5.22.1 Classes**

Classes in the **OraError** unit.

**Classes**
5.22.1.1 **EOraError Class**

Raised when a component detects Oracle error.

For a list of all members of this type, see **EOraError** members.

**Unit**

*OraError*

**Syntax**

```plaintext
EOraError = class (EDAError);  
```

**Remarks**

EOraError is raised when a component detects Oracle error. Use EOraError in an exception handling block.

**Inheritance Hierarchy**

*EDAError*  
  *EOraError*

**See Also**

* TOracleErrorHandler

---

**5.22.1.1.1 Members**

**EOraError** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOraError</td>
<td>Raised when a component detects Oracle error.</td>
</tr>
</tbody>
</table>
### Properties of the **EOraError** class.

For a complete list of the **EOraError** class members, see the [EOraError Members](#) topic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component</strong> (inherited from <strong>EDAError</strong>)</td>
<td>Contains the component that caused the error.</td>
</tr>
<tr>
<td><strong>ErrorCode</strong> (inherited from <strong>EDAError</strong>)</td>
<td>Determines the error code returned by the server.</td>
</tr>
<tr>
<td><strong>Sender</strong></td>
<td>Holds reference to the sender if exception is raised by the TComponent instance.</td>
</tr>
</tbody>
</table>

**See Also**
- [EOraError Class](#)
- [EOraError Class Members](#)

© 1997-2021
Devart. All Rights Reserved.

5.22.1.2.1 **Sender Property**

Holds reference to the sender if exception is raised by the TComponent instance.

© 2021 Devart
### E0raError

**Syntax**

```plaintext
property Sender: TComponent;
```

**Remarks**

The `Sender` property holds reference to the sender if exception is raised by the `TComponent` instance.

© 1997-2021 Devart. All Rights Reserved.

#### 5.23 OraLoader

This unit contains implementation of the `TOraLoader` component.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TDPColumn</code></td>
<td>A base class holding a collection of <code>TDPColumn</code> objects.</td>
</tr>
<tr>
<td><code>TOraLoader</code></td>
<td><code>TOraLoader</code> allows to load external data into the server database.</td>
</tr>
</tbody>
</table>

**Types**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TDPErrorEvent</code></td>
<td>This type is used for the <code>TOraLoader.OnError</code> event.</td>
</tr>
<tr>
<td><code>TDPGetColumnDataEvent</code></td>
<td>This type is used for the <code>TOraLoader.OnGetColumnData</code> event.</td>
</tr>
<tr>
<td><code>TDPPutDataEvent</code></td>
<td>This type is used for the <code>TOraLoader.OnPutData</code> event.</td>
</tr>
</tbody>
</table>

**Enumerations**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 5.23.1 Classes

Classes in the **OraLoader** unit.

#### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TDPColumn</strong></td>
<td>A base class holding a collection of TDPColumn objects.</td>
</tr>
<tr>
<td><strong>TOraLoader</strong></td>
<td>TOraLoader allows to load external data into the server database.</td>
</tr>
</tbody>
</table>

#### 5.23.1.1 TDPColumn Class

A base class holding a collection of TDPColumn objects.

For a list of all members of this type, see **TDPColumn** members.

**Unit**

**OraLoader**

**Syntax**

```pascal
TDPColumn = class(TDAColumn);
```

**Remarks**
Each TOraLoader uses TDPColumns to maintain a collection of TDPColumn objects. TDPColumn object represents the attributes for column loading. Every TDPColumn object corresponds to one of the table fields with the same name as its Name property.

To create columns at design time use column editor of TOraLoader component.

Inheritance Hierarchy

TDAColumn
    TDPColumn

See Also
- TOraLoader
- TQueueAgents

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateFormat</td>
<td>Used to specify a conversion mask for a column.</td>
</tr>
<tr>
<td>FieldType</td>
<td>(inherited from TDAColumn) Used to specify the types of values that will be loaded.</td>
</tr>
<tr>
<td>Name</td>
<td>(inherited from TDAColumn) Used to specify the field name of loading table.</td>
</tr>
<tr>
<td>Precision</td>
<td>Used to set the precision of number columns.</td>
</tr>
<tr>
<td>Scale</td>
<td>Used to set the precision of number columns.</td>
</tr>
<tr>
<td>Size</td>
<td>Used to set the maximum size in bytes of data for a column.</td>
</tr>
</tbody>
</table>
Properties of the **TDPColumn** class.

For a complete list of the **TDPColumn** class members, see the [TDPColumn Members](#) topic.

### Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DateFormat</strong></td>
<td>Used to specify a conversion mask for a column.</td>
</tr>
<tr>
<td><strong>FieldType</strong></td>
<td>(inherited from <strong>TDAColumn</strong>) Used to specify the types of values that will be loaded.</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>(inherited from <strong>TDAColumn</strong>) Used to specify the field name of loading table.</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>Used to set the precision of number columns.</td>
</tr>
<tr>
<td><strong>Scale</strong></td>
<td>Used to set the precision of number columns.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Used to set the maximum size in bytes of data for a column.</td>
</tr>
</tbody>
</table>

### See Also

- [TDPColumn Class](#)
- [TDPColumn Class Members](#)

© 1997-2021 Devart. All Rights Reserved. Request Support DAC Forum Provide Feedback

5.23.1.2.1.1 **DateFormat Property**

Used to specify a conversion mask for a column.

**Class**

**TDPColumn**

**Syntax**

```java
property DateFormat: string;
```

**Remarks**
Set DateFormat to specify a conversion mask for a column. TOraLoader uses DateFormat to convert string representation of date to its internal representation. If not set, the date format defaults to the date conversion mask set in the direct path context.

See Also
- TDAColumn.Name

Class
TDPColumn

Syntax

```delphi
property Precision: integer default 0;
```

Remarks
Use Precision property to set the precision of number columns.

See Also
- TDAColumn.FieldType

Class
TDPColumn

Syntax

```delphi
property Scale: integer default 0;
```
Remarks

Use Scale property to set the scale of number columns.

See Also

- `TDAColumn.FieldType`

Class

`TDPColumn`

Syntax

```
property Size: integer default 0;
```

Remarks

Use Size property to set the maximum size in bytes of data for a column. Size is used only for string columns.

See Also

- `TDAColumn.FieldType`

Unit

`OraLoader`

Syntax

```
TOraLoader Class

TOraLoader allows to load external data into the server database.

For a list of all members of this type, see `TOraLoader` members.
```

© 2021 Devart
TOraLoader = class(TDALoader);

Remarks

TOraLoader serves for fast loading of data to the server. It uses direct path load interface to speed up loading. To specify the name of the loading table set TDALoader.TableName property. Use TDALoader.Columns property to access individual columns. Write TDALoader.OnGetColumnData or TDALoader.OnPutData event handlers to read external data and pass it to the database. Call TDALoader.Load method to start loading data.

Limitations and Restrictions

TOraLoader has the following limitations similar to those of SQL*Loader:

- triggers are not supported
- check constraints are not supported
- referential integrity constraints are not supported
- clustered tables are not supported
- loading of remote objects is not supported
- user-defined types are not supported
- LOBs must be specified after all scalar columns
- LONGs must be specified last
- You cannot use TOraLoader in a threaded OCI environment in direct mode with Oracle client 8.17 or lower.
- ODAC sets T:Devart.Odac.Units.OraCall := False when you use OraLoader unit in your application. You can set it to True for Oracle client 9.2 or higher.

Inheritance Hierarchy

TDALoader
  TOraLoader

See Also

- TOraLoader Component
- TDALoader
- TQueueAgents
- TOraSession

© 1997-2021
Devart. All Rights Reserved.
### TOraLoader class overview.

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns (inherited from TDALoader)</td>
<td>Used to add a TDAColumn object for each field that will be loaded.</td>
</tr>
<tr>
<td>Connection (inherited from TDALoader)</td>
<td>See the TOraLoader.Session property.</td>
</tr>
<tr>
<td>LoadMode</td>
<td>Used to specify which access mode to use for a TOraLoader object when a database table is being modified.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session in which TOraLoader will be executed.</td>
</tr>
<tr>
<td>TableName (inherited from TDALoader)</td>
<td>Used to specify the name of the table to which data will be loaded.</td>
</tr>
</tbody>
</table>

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateColumns (inherited from TDALoader)</td>
<td>Creates TDAColumn objects for all fields of the table with the same name as TDALoader.TableName.</td>
</tr>
<tr>
<td>Load (inherited from TDALoader)</td>
<td>Starts loading data.</td>
</tr>
<tr>
<td>LoadFromDataSet (inherited from TDALoader)</td>
<td>Loads data from the specified dataset.</td>
</tr>
<tr>
<td>PutColumnData (inherited from TDALoader)</td>
<td>Overloaded. Puts the value of individual columns.</td>
</tr>
</tbody>
</table>

#### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Occurs when processing errors that are raised during</td>
</tr>
</tbody>
</table>
OnGetColumnData

Occurs when it is needed to put column values.

OnProgress (inherited from TDALoader)

Occurs if handling data loading progress of the TDALoader.LoadFromData Set method is needed.

OnPutData

Occurs when putting loading data by rows is needed.

Properties

Properties of the TOraLoader class.

For a complete list of the TOraLoader class members, see the TOraLoader Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns (inherited from TDALoader)</td>
<td>Used to add a TDAColumn object for each field that will be loaded.</td>
</tr>
<tr>
<td>Connection (inherited from TDALoader)</td>
<td>See the TOraLoader.Session property.</td>
</tr>
<tr>
<td>LoadMode</td>
<td>Used to specify which access mode to use for a TOraLoader object when a database table is being modified.</td>
</tr>
<tr>
<td>TableName (inherited from TDALoader)</td>
<td>Used to specify the name of the table to which data will be loaded.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session</td>
<td>Used to specify the session in which TOraLoader will be executed.</td>
</tr>
</tbody>
</table>
5.23.1.2.2.1 LoadMode Property

Used to specify which access mode to use for a TOraLoader object when a database table is being modified.

Class

TOraLoader

Syntax

```objective-c
property LoadMode: TLoadMode;
```

Remarks

Use the LoadMode property to specify which access mode to use for a TOraLoader object when a database table is being modified.

Set this property to lmDirect to make all modifications pass through internal data buffers or set it to lmDML to construct relevant DML statement which applies updates to the database table.

See Also

- TOraLoader Component

© 1997-2021
Devart. All Rights Reserved.  Request Support  DAC Forum  Provide Feedback
Syntax

```property
Session: TOraSession;
```

Remarks

Use the Session property to specify the session in which TOraLoader will be executed. If Session is not connected, Load method calls Session.Connect.

See Also

- TOraSession

5.23.1.2.3 Events

Events of the TOraLoader class.

For a complete list of the TOraLoader class members, see the TOraLoader Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnProgress</td>
<td>Occurs if handling data loading progress of the TDALoader.LoadFromDataSet method is needed.</td>
</tr>
</tbody>
</table>

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnError</td>
<td>Occurs when processing errors that are raised during loading is needed.</td>
</tr>
<tr>
<td>OnGetColumnData</td>
<td>Occurs when it is needed to put column values.</td>
</tr>
<tr>
<td>OnPutData</td>
<td>Occurs when putting loading data by rows is needed.</td>
</tr>
</tbody>
</table>

See Also

- TOraLoader Class
5.23.1.2.3.1 OnError Event

Occurs when processing errors that are raised during loading is needed.

Class

TOraLoader

Syntax

```property OnError: TDPErrorEvent;```

Remarks

Write the OnError event handler to process errors that occur during loading. Handler is used only if LoadMode = lmDirect, i.e. when using Oracle Direct Path interface. E parameter is an exception that was raised. Col and Row parameters are appropriate column and row values for data that loader failed to write to a table.

See Also

• LoadMode

5.23.1.2.3.2 OnGetColumnData Event

Occurs when it is needed to put column values.

Class

TOraLoader

Syntax

```property OnGetColumnData: TDPGetColumnDataEvent;```

Remarks

Write OnGetColumnData event handler to put column values. TOraLoader calls
OnGetColumnData event handler for each column in the loop. Column points to TDAColumn object that corresponds to the current loading column. Use its Name or Index property to identify what column is loading. The Row parameter indicates the current loading record. TDALoader increments Row parameter when all the columns of the current record are loaded. The first row is 1. Set EOF to True to stop data loading. Fill Value parameter by column values. To start loading call the TDALoader.Load method.

Another way to load data is using OnPutData event.

See Also
- TDALoader.OnPutData
- TDALoader.Load
- OnPutData

5.23.1.2.3.3 OnPutData Event

Occurs when putting loading data by rows is needed.

Class
TD0raLoader

Syntax

| property OnPutData: TDPPutDataEvent; |

Remarks

Note that rows should be loaded from the first in ascending order. To start loading, call TDALoader.Load method.

To start loading, call the TDALoader.Load method.

See Also
- TDALoader.PutColumnData
- TDALoader.Load
- TDALoader.OnGetColumnData
- OnGetColumnData
5.23.2 Types

Types in the *OraLoader* unit.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDPErrorEvent</td>
<td>This type is used for the <code>TOraLoader.OnError</code> event.</td>
</tr>
<tr>
<td>TDPGetColumnDataEvent</td>
<td>This type is used for the <code>TOraLoader.OnGetColumnData</code> event.</td>
</tr>
<tr>
<td>TDPPutDataEvent</td>
<td>This type is used for the <code>TOraLoader.OnPutData</code> event.</td>
</tr>
</tbody>
</table>

5.23.2.1 TDPErrorEvent Procedure Reference

This type is used for the `TOraLoader.OnError` event.

Unit

*OraLoader*

Syntax

```pascal
TDPErrorEvent = procedure (Sender: TOraLoader; E: Exception; Col: integer; Row: integer; var Action: TDPErrorAction) of object;
```

Parameters

*Sender*

An object that raised the event.

*E*

The exception that was raised.

*Col*

The column value for data that loader failed to write to a table.

*Row*

The row value for data that loader failed to write to a table.
5.23.2.2  TDPGetColumnDataEvent Procedure Reference

This type is used for the TOraLoader.OnGetColumnData event.

Unit

OraLoader

Syntax

TDPGetColumnDataEvent = procedure (Sender: TObject; Column: TDPColumn; Row: integer; var Value: variant; var IsEOF: boolean) of object;

Parameters

Sender
An object that raised the event.

Column
Points to TDAColumn object that corresponds to the current loading column.

Row
Indicates the current loading record.

Value
The column values.

IsEOF
True, if data loading needs to be stopped.

5.23.2.3  TDPPutDataEvent Procedure Reference

This type is used for the TOraLoader.OnPutData event.

Unit

OraLoader

Syntax
TDPPutDataEvent = \texttt{procedure} (\texttt{Sender: TOraLoader}) \texttt{of object};

\textbf{Parameters}

\textit{Sender} \hspace{1cm} An object that raised the event.

\section*{5.23.3 Enumerations}

Enumerations in the \texttt{OraLoader} unit.

\begin{center}
\begin{tabular}{|l|p{20cm}|}
\hline
\textbf{Name} & \textbf{Description} \\
\hline
\texttt{TDPErrorAction} & Specifies the action for \texttt{TOraLoader} to take to process errors that occur during loading. \\
\hline
\texttt{TLoadMode} & Specifies the access mode to use for a \texttt{TOraLoader} object when a database table is being modified. \\
\hline
\end{tabular}
\end{center}

\subsection*{5.23.3.1 \texttt{TDPErrorAction} Enumeration}

Specifies the action for \texttt{TOraLoader} to take to process errors that occur during loading.

\underline{Unit} \hspace{1cm} \texttt{OraLoader}

\underline{Syntax} \hspace{1cm} \texttt{TDPErrorAction} = (dpAbort, dpFail, dpIgnore);

\underline{Values}

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>dpAbort</td>
<td>\texttt{TOraLoader} silently interrupts the current operation.</td>
</tr>
</tbody>
</table>
5.23.3.2 TLoadMode Enumeration

Specifies the access mode to use for a TOraLoader object when a database table is being modified.

Unit
OraLoader

Syntax

TLoadMode = (lmDirect, lmDML);

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>lmDirect</td>
<td>All modifications are passed through internal data buffers.</td>
</tr>
<tr>
<td>lmDML</td>
<td>Constructs relevant DML statement which applies updates to the database table.</td>
</tr>
</tbody>
</table>

Remarks

Use the LoadMode property to specify which access mode to use for a TOraLoader object when a database table is being modified.

Set this property to lmDirect to make all modifications pass through internal data buffers or set it to lmDML to construct relevant DML statement which applies updates to the database table.

5.24 OraNet

This unit implements the Direct Mode in ODAC.

Enumerations
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSecurityLevel</td>
<td>Used to turn on the Oracle Advanced Security encryption and integrity.</td>
</tr>
</tbody>
</table>

### Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataIntegrityLevel</td>
<td>Use this variable to prevent unauthorized data modification when it is passed over the network.</td>
</tr>
<tr>
<td>EncryptionLevel</td>
<td>Use this variable to prevent unauthorized data viewing when it is passed over the network.</td>
</tr>
<tr>
<td>PacketSize</td>
<td>Use this constant to specify the size of transferred packets.</td>
</tr>
</tbody>
</table>

5.24.1 Enumerations

Enumerations in the OraNet unit.

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSecurityLevel</td>
<td>Used to turn on the Oracle Advanced Security encryption and integrity.</td>
</tr>
</tbody>
</table>

5.24.1.1 TSecurityLevel Enumeration

Used to turn on the Oracle Advanced Security encryption and integrity.
### OraNet

#### Syntax

```
TSecurityLevel = (slAccepted = 0, slRejected = 1, slRequested = 2, slRequired = 3);
```

#### Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>slAccepted</td>
<td>Select this value to enable the security service if required or requested by the other side.</td>
</tr>
<tr>
<td>slRejected</td>
<td>Select this value if you do not elect to enable the security service, even if required by the other side.</td>
</tr>
<tr>
<td>slRequested</td>
<td>Select this value to enable the security service if the other side permits it.</td>
</tr>
<tr>
<td>slRequired</td>
<td>Select this value to enable the security service or preclude the connection.</td>
</tr>
</tbody>
</table>

### 5.24.2 Variables

Variables in the OraNet unit.

#### Variables

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataIntegrityLevel</td>
<td>Use this variable to prevent unauthorized data modification when it is passed over the network.</td>
</tr>
<tr>
<td>EncryptionLevel</td>
<td>Use this variable to prevent unauthorized data viewing when it is passed over the network.</td>
</tr>
<tr>
<td>PacketSize</td>
<td>Use this constant to specify the size of transferred packets.</td>
</tr>
</tbody>
</table>
5.24.2.1 DataIntegrityLevel Variable

Use this variable to prevent unauthorized data modification when it is passed over the network.

Unit
OraNet

Syntax

DataIntegrityLevel: TSecurityLevel;

Remarks

Use the DataIntegrityLevel variable to prevent unauthorized data modification when it is passed over the network. To select the data integrity level, specify one of the possible values of the TSecurityLevel variable. The default value is Accepted.

© 1997-2021
Devart. All Rights Reserved.

5.24.2.2 EncryptionLevel Variable

Use this variable to prevent unauthorized data viewing when it is passed over the network.

Unit
OraNet

Syntax

EncryptionLevel: TSecurityLevel;

Remarks

Use the EncryptionLevel variable to prevent unauthorized data viewing when it is passed over the network. To select the data encryption level, specify one of the existing values of the TSecurityLevel variable. The default value is Accepted.

© 1997-2021
Devart. All Rights Reserved.
5.24.2.3 PacketSize Variable

Use this constant to specify the size of transferred packets.

Unit

OraNet

Syntax

PacketSize: Integer;

Remarks

Use this constant to specify the size of transferred packets. If you set this variable to an optimal value (that depends on the network you are working in), this can significantly increase performance for VPN, Wireless and other networks.

Allowed values are within the range from 512 to 65535 bytes. The default value is 8192 bytes.

5.25 OraObjects

This unit contains classes for Oracle OBJECT, ARRAY, TABLE and XMLTYPE data types.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraArray</td>
<td>A class representing the value of the Oracle array data type.</td>
</tr>
<tr>
<td>TOraNestTable</td>
<td>A class representing a value of the Oracle nested table data type.</td>
</tr>
<tr>
<td>TOraObject</td>
<td>A class representing the Oracle object data type value.</td>
</tr>
<tr>
<td>TOraRef</td>
<td>A class representing the Oracle reference data type value.</td>
</tr>
<tr>
<td>TOraType</td>
<td>A class holding information about Oracle type required for TOraObject objects.</td>
</tr>
</tbody>
</table>
### 5.25.1 Classes

Classes in the **OraObjects** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraArray</td>
<td>A class representing the value of the Oracle array data type.</td>
</tr>
<tr>
<td>TOraNestTable</td>
<td>A class representing a value of the Oracle nested table data type.</td>
</tr>
<tr>
<td>TOraObject</td>
<td>A class representing the Oracle object data type value.</td>
</tr>
<tr>
<td>TOraRef</td>
<td>A class representing the Oracle reference data type value.</td>
</tr>
<tr>
<td>TOraType</td>
<td>A class holding information about Oracle type required for TOraObject objects.</td>
</tr>
<tr>
<td>TOraXML</td>
<td>A class representing a value of the Oracle SYS.XMLTYPE type.</td>
</tr>
</tbody>
</table>

### 5.25.1.1 TOraArray Class

A class representing the value of the Oracle array data type.

For a list of all members of this type, see [TOraArray](#) members.
OraObjects

Syntax

```pascal
TOraArray = class(TOraObject);
```

Remarks

TOraArray represents the value of the Oracle array data type. TOraArray is inherited from TOraObject. You can get a TOraArray object by the TOraDataSet.GetArray method after fetching rows containing array field.

Inheritance Hierarchy

- TSharedObject
  - TDBObject
    - TOraObject
      - TOraArray

See Also

- TOraObject
- TOraDataSet.GetArray
- TOraParam.AsArray

© 1997-2021 Devart. All Rights Reserved.

5.25.1.1.1 Members

**TOraArray class overview.**

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttrAsArray</td>
<td>(inherited from TOraObject) Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td>AttrAsDateTime</td>
<td>(inherited from TOraObject) Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsFloat</td>
<td>(inherited from TOraObject) Used to read the attribute's data value into a double, or</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AttrAsInteger (inherited from TOracle)</td>
<td>to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsLob (inherited from TOracle)</td>
<td>Used to read the attribute’s data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsObject (inherited from TOracle)</td>
<td>Used to read the attribute’s data value as TOracle, or to assign a TOracle value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIDate (inherited from TOracle)</td>
<td>Used to read the attribute’s data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCINumber (inherited from TOracle)</td>
<td>Used to read the attribute’s data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIString (inherited from TOracle)</td>
<td>Used to read the attribute’s data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsString (inherited from TOracle)</td>
<td>Used to read the attribute’s data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrIsNull (inherited from TOracle)</td>
<td>Used to indicate if the attribute value is NULL.</td>
</tr>
<tr>
<td>Indicator (inherited from TOracle)</td>
<td>Used to get a pointer to the indicator structure of an object.</td>
</tr>
<tr>
<td>Instance (inherited from TOracle)</td>
<td>Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td>IsNull (inherited from TOracle)</td>
<td>Used to verify if an object is empty.</td>
</tr>
<tr>
<td>ItemAsAnsiString</td>
<td>Used to read the value of the</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ItemAsString</td>
<td>Used to read the value of the array element into an AnsiString or to assign an AnsiString to the array element.</td>
</tr>
<tr>
<td>ItemAsDateTime</td>
<td>Used to read the value of the array element into a TDateTime variable or to assign a TDateTime variable to the array element.</td>
</tr>
<tr>
<td>ItemAsFloat</td>
<td>Used to read the value of the array element into a Double or to assign a Double to the array element.</td>
</tr>
<tr>
<td>ItemAsInteger</td>
<td>Used to read the value of the array element into an Integer or to assign an Integer to the array element.</td>
</tr>
<tr>
<td>ItemAsInterval</td>
<td>Used to read the value of the array element into a TOraInterval variable or to assign a TOraInterval variable to the array element.</td>
</tr>
<tr>
<td>ItemAsLargeInt</td>
<td>Used to read the value of the array element into an Int64 or to assign an Int64 to the array element.</td>
</tr>
<tr>
<td>ItemAsLob</td>
<td>Used to read the value of the array element into a TOraLob variable or to assign a TOraLob variable to the array element.</td>
</tr>
<tr>
<td>ItemAsObject</td>
<td>Used to read the value of the array element into a TOraObject variable or to assign a TOraObject variable to the array element.</td>
</tr>
<tr>
<td>ItemAsOCIStrine</td>
<td>Used to read the value of the array element into a generic OCIStrine pointer or to assign the OCIStrine pointer to the array element.</td>
</tr>
<tr>
<td>ItemAsRef</td>
<td>Used to read the value of the array element into a TOraRef variable or to assign a TOraRef variable to the array element.</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ItemAsString</td>
<td>Used to read the value of the array element into a String or to assign a String to the array element.</td>
</tr>
<tr>
<td>ItemAsTimeStamp</td>
<td>Used to read the value of the array element into a TOraTimeStamp variable or to assign a TOraTimeStamp variable to the array element.</td>
</tr>
<tr>
<td>ItemAsWideString</td>
<td>Used to read the value of the array element into a WideString or to assign a WideString to the array element.</td>
</tr>
<tr>
<td>ItemExists</td>
<td>Indicates whether an element with the specified index exists in the array.</td>
</tr>
<tr>
<td>ItemIsNull</td>
<td>Indicates whether the array element contains a value.</td>
</tr>
<tr>
<td>ItemSubType</td>
<td>Indicates the subtype of an array element.</td>
</tr>
<tr>
<td>ItemType</td>
<td>Indicates the type of an array element.</td>
</tr>
<tr>
<td>MaxSize</td>
<td>Defines the maximum number of elements that an array object may hold.</td>
</tr>
<tr>
<td>ObjectType (inherited from TOraObject)</td>
<td>Used to indicate the object type.</td>
</tr>
<tr>
<td>OCISvcCtx (inherited from TOraObject)</td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td>RefCount (inherited from TSharedObject)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>Size</td>
<td>Holds the size of an array.</td>
</tr>
</tbody>
</table>

Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from <code>TSharedObject</code>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocObject (inherited from <code>TOraObject</code>)</td>
<td>Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td>AppendItem</td>
<td>Appends an element to the end of an array.</td>
</tr>
<tr>
<td>Assign</td>
<td>Copies properties and attributes of one object to another one.</td>
</tr>
<tr>
<td>Clear</td>
<td>Removes all elements from an array.</td>
</tr>
<tr>
<td>CreateObject (inherited from <code>TOraObject</code>)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td>Exists (inherited from <code>TOraObject</code>)</td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td>Flush (inherited from <code>TOraObject</code>)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td>FreeObject (inherited from <code>TOraObject</code>)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td>InsertItem</td>
<td>Inserts an element at a specific index.</td>
</tr>
<tr>
<td>IsDirty (inherited from <code>TOraObject</code>)</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td>IsLocked (inherited from <code>TOraObject</code>)</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td>Lock (inherited from <code>TOraObject</code>)</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td>MarkDelete (inherited from <code>TOraObject</code>)</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td>MarkUpdate (inherited from <code>TOraObject</code>)</td>
<td>Marks an object as being updated.</td>
</tr>
<tr>
<td>Refresh (inherited from <code>TOraObject</code>)</td>
<td>Retrieves the latest database image for the object.</td>
</tr>
<tr>
<td>Release (inherited from <code>TSharedObject</code>)</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>Unmark (inherited from <code>TOraObject</code>)</td>
<td>Marks an object as not being dirty.</td>
</tr>
</tbody>
</table>
5.25.1.1.2 Properties

Properties of the **ToraArray** class.

For a complete list of the **ToraArray** class members, see the [ToraArray Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AttrAsArray</strong></td>
<td>Used to get the ToraArray type attribute value.</td>
</tr>
<tr>
<td><strong>AttrAsDateTime</strong></td>
<td>Used to read the attribute’s data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsFloat</strong></td>
<td>Used to read the attribute’s data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsInteger</strong></td>
<td>Used to read the attribute’s data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsLob</strong></td>
<td>Used to get reference to the ToraLob object that represents the LOBAttribute value.</td>
</tr>
<tr>
<td><strong>AttrAsObject</strong></td>
<td>Used to read the attribute’s data value as ToraObject, or to assign a ToraObject value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCIDate</strong></td>
<td>Used to read the attribute’s data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCINumber</strong></td>
<td>Used to read the attribute’s data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCIString</strong></td>
<td>Used to read the attribute’s data value into OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AttrAsString</td>
<td>Used to read the attribute's data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrIsNull</td>
<td>Used to indicate if the attribute value is NULL.</td>
</tr>
<tr>
<td>Indicator</td>
<td>Used to get a pointer to the indicator structure of an object.</td>
</tr>
<tr>
<td>Instance</td>
<td>Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to verify if an object is empty.</td>
</tr>
<tr>
<td>ItemAsAnsiString</td>
<td>Used to read the value of the array element into an AnsiString or to assign an AnsiString to the array element.</td>
</tr>
<tr>
<td>ItemAsDateTime</td>
<td>Used to read the value of the array element into a TDateTime variable or to assign a TDateTime variable to the array element.</td>
</tr>
<tr>
<td>ItemAsFloat</td>
<td>Used to read the value of the array element into a Double or to assign a Double to the array element.</td>
</tr>
<tr>
<td>ItemAsInteger</td>
<td>Used to read the value of the array element into an Integer or to assign an Integer to the array element.</td>
</tr>
<tr>
<td>ItemAsInterval</td>
<td>Used to read the value of the array element into a TOraInterval variable or to assign a TOraInterval variable to the array element.</td>
</tr>
<tr>
<td>ItemAsLargeInt</td>
<td>Used to read the value of the array element into an Int64</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>ItemAsLob</code></td>
<td>Used to read the value of the array element into a <code>TOraLob</code> variable or to assign a <code>TOraLob</code> variable to the array element.</td>
</tr>
<tr>
<td><code>ItemAsObject</code></td>
<td>Used to read the value of the array element into a <code>TOraObject</code> variable or to assign a <code>TOraObject</code> variable to the array element.</td>
</tr>
<tr>
<td><code>ItemAsOCIString</code></td>
<td>Used to read the value of the array element into a generic OCIString pointer or to assign the OCIString pointer to the array element.</td>
</tr>
<tr>
<td><code>ItemAsRef</code></td>
<td>Used to read the value of the array element into a <code>TOraRef</code> variable or to assign a <code>TOraRef</code> variable to the array element.</td>
</tr>
<tr>
<td><code>ItemAsString</code></td>
<td>Used to read the value of the array element into a String or to assign a String to the array element.</td>
</tr>
<tr>
<td><code>ItemAsTimeStamp</code></td>
<td>Used to read the value of the array element into a <code>TOraTimeStamp</code> variable or to assign a <code>TOraTimeStamp</code> variable to the array element.</td>
</tr>
<tr>
<td><code>ItemAsWideString</code></td>
<td>Used to read the value of the array element into a WideString or to assign a WideString to the array element.</td>
</tr>
<tr>
<td><code>ItemExists</code></td>
<td>Indicates whether an element with the specified index exists in the array.</td>
</tr>
<tr>
<td><code>ItemIsNull</code></td>
<td>Indicates whether the array element contains a value.</td>
</tr>
<tr>
<td><code>ItemSubType</code></td>
<td>Indicates the subtype of an array element.</td>
</tr>
<tr>
<td><strong>ItemType</strong></td>
<td>Indicates the type of an array element.</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>MaxSize</strong></td>
<td>Defines the maximum number of elements that an array object may hold.</td>
</tr>
<tr>
<td><strong>ObjectType</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to indicate the object type.</td>
</tr>
<tr>
<td><strong>OCISvcCtx</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td><strong>RefCount</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>Holds the size of an array.</td>
</tr>
</tbody>
</table>

See Also

- **TOraArray Class**
- **TOraArray Class Members**

© 1997-2021 Devart. All Rights Reserved.

5.25.1.1.2.1 ItemAsAnsiString Property(Indexer)

Used to read the value of the array element into an AnsiString or to assign an AnsiString to the array element.

**Class**

**TOraArray**

**Syntax**

```property ItemAsAnsiString[Index: integer]: AnsiString;```

**Parameters**

- **Index**
  - Holds the index of the array element.

**Remarks**

Use the ItemAsAnsiString property to read the value of the array element into an AnsiString or to assign an AnsiString to the array element.

© 1997-2021 Devart
5.25.1.1.2.2 ItemAsDateTime Property (Indexer)

Used to read the value of the array element into a TDateTime variable or to assign a TDateTime variable to the array element.

Class

T0raArray

Syntax

property ItemAsDateTime[Index: integer]: TDateTime;

Parameters

Index

Holds the index of the array element.

Remarks

Use the ItemAsDateTime property to read the value of the array element into a TDateTime variable or to assign a TDateTime variable to the array element.

5.25.1.1.2.3 ItemAsFloat Property (Indexer)

Used to read the value of the array element into a Double or to assign a Double to the array element.

Class

T0raArray

Syntax

property ItemAsFloat[Index: integer]: double;

Parameters

Index

Holds the index of the array element.

Remarks
Use the ItemAsFloat property to read the value of the array element into a Double or to assign a Double to the array element.

### 5.25.1.1.2.4 ItemAsInteger Property (Indexer)

Used to read the value of the array element into an Integer or to assign an Integer to the array element.

**Class**

TOraArray

**Syntax**

```property
ItemAsInteger[Index: integer]: integer;
```

**Parameters**

- **Index**
  
  Holds the index of the array element.

**Remarks**

Use the ItemAsInteger property to read the value of the array element into an Integer or to assign an Integer to the array element.

### 5.25.1.1.2.5 ItemAsInterval Property (Indexer)

Used to read the value of the array element into a TOraInterval variable or to assign a TOraInterval variable to the array element.

**Class**

TOraArray

**Syntax**

```property
ItemAsInterval[Index: integer]: TOraInterval;
```

**Parameters**


Index
Holds the index of the array element.

Remarks
Use the ItemAsInterval property to read the value of the array element into a **TOraInterval** variable or to assign a **TOraInterval** variable to the array element.

See Also
- **TOraInterval**

### 5.25.1.1.2.6 ItemAsLargeInt Property(Indexer)

Used to read the value of the array element into an Int64 or to assign an Int64 to the array element.

**Class**
**TOraArray**

**Syntax**
```property ItemAsLargeInt[Index: integer]: int64;```

**Parameters**
- **Index**
  Holds the index of the array element.

**Remarks**
Use the ItemAsLargeInt property to read the value of the array element into an Int64 or to assign an Int64 to the array element.

### 5.25.1.1.2.7 ItemAsLob Property(Indexer)

Used to read the value of the array element into a **TOraLob** variable or to assign a **TOraLob** variable to the array element.

**Class**
**TOraArray**

**Syntax**

```
property ItemAsLob[Index: integer]: TOraLob;
```

**Parameters**

- **Index**
  - Holds the index of the array element.

**Remarks**

Use the ItemAsLob property to read the value of the array element into a TOraLob variable or to assign a TOraLob variable to the array element.

**See Also**

- TOraLob

---

5.25.1.1.2.8 ItemAsObject Property(Indexer)

Used to read the value of the array element into a TOraObject variable or to assign a TOraObject variable to the array element.

**Class**

**TOraArray**

**Syntax**

```
property ItemAsObject[Index: integer]: TOraObject;
```

**Parameters**

- **Index**
  - Holds the index of the array element.

**Remarks**

Use the ItemAsObject property to read the value of the array element into a TOraObject variable or to assign a TOraObject variable to the array element.

**See Also**
5.25.1.1.2.9  ItemAsOCIString Property(Indexer)

Used to read the value of the array element into a generic OCIString pointer or to assign the OCIString pointer to the array element.

Class

TOraArray

Syntax

```plaintext
property ItemAsOCIString[Index: integer]: pOCIString;
```

Parameters

- **Index**
  - Holds the index of the array element.

Remarks

Use the ItemAsOCIString property to read the value of the array element into a generic OCIString pointer or to assign the OCIString pointer to the array element.

5.25.1.1.2.10  ItemAsRef Property(Indexer)

Used to read the value of the array element into a TOraRef variable or to assign a TOraRef variable to the array element.

Class

TOraArray

Syntax

```plaintext
property ItemAsRef[Index: integer]: TOraRef;
```

Parameters

- **Index**
  - Holds the index of the array element.
Remarks
Use the ItemAsRef property to read the value of the array element into a TOraRef variable or to assign a TOraRef variable to the array element.

See Also
• TOraRef

5.25.1.1.2.11 ItemAsString Property(Indexer)

Used to read the value of the array element into a String or to assign a String to the array element.

Class
TOraArray

Syntax

property ItemAsString[Index: integer]: string;

Parameters

Index
Holds the index of the array element.

Remarks
Use the ItemAsString property to read the value of the array element into a String or to assign a String to the array element.

5.25.1.1.2.12 ItemAsTimeStamp Property(Indexer)

Used to read the value of the array element into a TOraTimeStamp variable or to assign a TOraTimeStamp variable to the array element.

Class
TOraArray
Syntax

```property`` ItemAsTimeStamp[Index: integer]: TOraTimeStamp;
```

**Parameters**

*Index*

Holds the index of the array element.

**Remarks**

Use the ItemAsTimeStamp property to read the value of the array element into a `TOraTimeStamp` variable or to assign a `TOraTimeStamp` variable to the array element.

**See Also**

- `TOraTimeStamp`

© 1997-2021 Devart. All Rights Reserved.

5.25.1.1.2.13  ItemAsWideString Property(Indexer)

Used to read the value of the array element into a WideString or to assign a WideString to the array element.

**Class**

`TOraArray`

**Syntax**

```property`` ItemAsWideString[Index: integer]: string;
```

**Parameters**

*Index*

Holds the index of the array element.

**Remarks**

Use the ItemAsWideString property to read the value of the array element into a WideString or to assign a WideString to the array element.

© 1997-2021 Devart. All Rights Reserved.
5.25.1.1.2.14  ItemExists Property(Indexer)

Indicates whether an element with the specified index exists in the array.

Class

T0raArray

Syntax

property ItemExists[Index: integer]: boolean;

Parameters

Index

Holds the index of the array element.

Remarks

Use the ItemExists property to check whether an element with the specified index exists in the array.

See Also

• Size

5.25.1.1.2.15  ItemIsNull Property(Indexer)

Indicates whether the array element contains a value.

Class

T0raArray

Syntax

property ItemIsNull[Index: integer]: boolean;

Parameters

Index

Holds the index of the array element.

Remarks

Use IsNull to determine whether the array element with at specific index contains a value. If
IsNull is True, the element is empty. IfIsNull is False, the element has a value.

5.25.1.1.2.16  ItemSubType Property

Indicates the subtype of an array element.

Class

T0raArray

Syntax

```pascal
property ItemSubType: word;
```

Remarks

Use the ItemSubType property to return the subtype of an array element.

5.25.1.1.2.17  ItemType Property

Indicates the type of an array element.

Class

T0raArray

Syntax

```pascal
property ItemType: word;
```

Remarks

Use the ItemType property to return the type of an array element.
5.25.1.1.2.18 MaxSize Property

Defines the maximum number of elements that an array object may hold.

Class
TOraArray

Syntax

```property```
```MaxSize: integer;```

Remarks
Use the MaxSize property to get the maximum number of elements that an array object may hold.

See Also
- Size

© 1997-2021
Devart. All Rights Reserved.

5.25.1.1.2.19 Size Property

Holds the size of an array.

Class
TOraArray

Syntax

```property```
```Size: integer;```

Remarks
Use the Size property to learn the length of an array.

See Also
- ItemExists
### Methods

Methods of the **TOraArray** class.

For a complete list of the **TOraArray** class members, see the [TOraArray Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocObject</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td><strong>AppendItem</strong></td>
<td>Appends an element to the end of an array.</td>
</tr>
<tr>
<td><strong>Assign</strong></td>
<td>Copies properties and attributes of one object to another one.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Removes all elements from an array.</td>
</tr>
<tr>
<td><strong>CreateObject</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td><strong>Exists</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td><strong>FreeObject</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td><strong>InsertItem</strong></td>
<td>Inserts an element at a specific index.</td>
</tr>
<tr>
<td><strong>IsDirty</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td><strong>IsLocked</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td><strong>Lock</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td><strong>MarkDelete</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td><strong>MarkUpdate</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Marks an object as being updated.</td>
</tr>
<tr>
<td><strong>Refresh</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Retrieves the latest database image for the</td>
</tr>
</tbody>
</table>
### AppendItem Method

Appends an element to the end of an array.

#### Class

**T0raArray**

#### Syntax

```pascal
function AppendItem: integer;
```

#### Return Value

The index of the appended element.

#### Remarks

Call the AppendItem function to append an element to the end of an array. Returns the index of the appended element.

#### See Also

- **InsertItem**
5.25.1.1.3.2 Assign Method

Copies properties and attributes of one object to another one.

Class

T0raArray

Syntax

```plaintext
procedure Assign(Source: T0raObject); override;
```

Parameters

Source

Holds the source object which properties and attributes will be copied.

Remarks

Call the Assign method to copy the properties and attributes of one object to another one.

See Also

- T0raObject

5.25.1.1.3.3 Clear Method

Removes all elements from an array.

Class

T0raArray

Syntax

```plaintext
procedure Clear;
```

Remarks

Call this method to remove all elements from an array.
5.25.1.3.4 InsertItem Method

Inserts an element at a specific index.

Class

TOraArray

Syntax

procedure InsertItem(Index: integer);

Parameters

Index

Holds the index position.

Remarks

Call the InsertItem procedure to insert an item at a specific index.

See Also

- AppendItem

5.25.1.2 TOraNestTable Class

A class representing a value of the Oracle nested table data type.

For a list of all members of this type, see TOraNestTable members.

Unit

oraObjects

Syntax

TOraNestTable = class(TOraArray);

Remarks

TOraNestTable represents a value of the Oracle nested table data type. TOraNestTable is inherited from TOraArray. You can get a TOraNestTable object using the TOraDataSet.GetTable method after fetching rows containing array field. Also you can get it using the TOraParam.AsTable method.
Inheritance Hierarchy

**TSharedObject**  
**TDBObject**  
**TOraObject**  
**TOraArray**  
**TOraNestTable**

See Also
- **TOraArray**  
- **TOraNestedTable**  
- **TOraDataSet.GetTable**  
- **TOraParam.AsTable**

© 1997-2021
Devar. All Rights Reserved.

5.25.1.2.1 Members

**TOraNestTable** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AttrAsArray         | (inherited from **TOraObject**)
|                     | Used to get the TOraArray type attribute value.                             |
| AttrAsDateTime      | (inherited from **TOraObject**)
|                     | Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute. |
| AttrAsFloat         | (inherited from **TOraObject**)
|                     | Used to read the attribute's data value into a double, or to assign a double value to the contents of an attribute. |
| AttrAsInteger       | (inherited from **TOraObject**)
|                     | Used to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute. |
| AttrAsLob           | (inherited from **TOraObject**)
<p>|                     | Used to get reference to the TOraLob object that represents the LOBAttribute |</p>
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AttrAsObject</strong></td>
<td>Used to read the attribute's data value as <strong>TOraObject</strong>, or to assign a <strong>TOraObject</strong> value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCIDate</strong></td>
<td>Used to read the attribute's data value as <strong>OCIDate</strong>, or to assign an <strong>OCIDate</strong> value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCINumber</strong></td>
<td>Used to read the attribute's data value into <strong>OCINumber</strong>, or to assign an <strong>OCINumber</strong> value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCIString</strong></td>
<td>Used to read the attribute's data value as <strong>OCIString</strong>, or to assign an <strong>OCIString</strong> value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsString</strong></td>
<td>Used to read the attribute's data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrIsNull</strong></td>
<td>Used to indicate if the attribute value is <strong>NULL</strong>.</td>
</tr>
<tr>
<td><strong>Indicator</strong></td>
<td>Used to get a pointer to the indicator structure of an object.</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td><strong>IsNull</strong></td>
<td>Used to verify if an object is empty.</td>
</tr>
<tr>
<td><strong>ItemAsAnsiString</strong></td>
<td>Used to read the value of the array element into an <strong>AnsiString</strong> or to assign an <strong>AnsiString</strong> to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsDateTime</strong></td>
<td>Used to read the value of the array element into a <strong>TDateTime</strong> variable or to assign a <strong>TDateTime</strong> variable to the array element.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ItemAsFloat</strong></td>
<td>Used to read the value of the array element into a Double or to assign a Double to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsInteger</strong></td>
<td>Used to read the value of the array element into an Integer or to assign an Integer to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsInterval</strong></td>
<td>Used to read the value of the array element into a TOraInterval variable or to assign a TOraInterval variable to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsLargeInt</strong></td>
<td>Used to read the value of the array element into an Int64 or to assign an Int64 to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsLob</strong></td>
<td>Used to read the value of the array element into a TOraLob variable or to assign a TOraLob variable to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsObject</strong></td>
<td>Used to read the value of the array element into a TOraObject variable or to assign a TOraObject variable to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsOCIString</strong></td>
<td>Used to read the value of the array element into a generic OCIString pointer or to assign the OCIString pointer to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsRef</strong></td>
<td>Used to read the value of the array element into a TOraRef variable or to assign a TOraRef variable to the array element.</td>
</tr>
<tr>
<td><strong>ItemAsString</strong></td>
<td>Used to read the value of the array element into a String or to assign a String to the array element.</td>
</tr>
</tbody>
</table>
| **ItemAsTimeStamp**   | Used to read the value of the array element into a }
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ItemAsWideString</strong></td>
<td>(inherited from TOraArray) Used to read the value of the array element into a WideString or to assign a WideString to the array element.</td>
</tr>
<tr>
<td><strong>ItemExists</strong></td>
<td>(inherited from TOraArray) Indicates whether an element with the specified index exists in the array.</td>
</tr>
<tr>
<td><strong>ItemIsNull</strong></td>
<td>(inherited from TOraArray) Indicates whether the array element contains a value.</td>
</tr>
<tr>
<td><strong>ItemSubType</strong></td>
<td>(inherited from TOraArray) Indicates the subtype of an array element.</td>
</tr>
<tr>
<td><strong>ItemType</strong></td>
<td>(inherited from TOraArray) Indicates the type of an array element.</td>
</tr>
<tr>
<td><strong>MaxSize</strong></td>
<td>(inherited from TOraArray) Defines the maximum number of elements that an array object may hold.</td>
</tr>
<tr>
<td><strong>ObjectType</strong></td>
<td>(inherited from TOraObject) Used to indicate the object type.</td>
</tr>
<tr>
<td><strong>OCISvcCtx</strong></td>
<td>(inherited from TOraObject) Used to assign a service context handle.</td>
</tr>
<tr>
<td><strong>RefCount</strong></td>
<td>(inherited from TSharedObject) Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td><strong>Size</strong></td>
<td>(inherited from TOraArray) Holds the size of an array.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong></td>
<td>(inherited from TSharedObject) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocObject</strong></td>
<td>(inherited from TOraObject) Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td><strong>AppendItem</strong></td>
<td>(inherited from TOraArray) Appends an element to the end of an array.</td>
</tr>
<tr>
<td><strong>Assign</strong></td>
<td>(inherited from TOraArray) Copies properties and attributes of one object to</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Clear (inherited from TOraArray)</td>
<td>Removes all elements from an array.</td>
</tr>
<tr>
<td>CreateObject (inherited from TOraObject)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td>DeleteItem</td>
<td>Deletes an item pointed by its index.</td>
</tr>
<tr>
<td>Exists (inherited from TOraObject)</td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td>Flush (inherited from TOraObject)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td>FreeObject (inherited from TOraObject)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td>InsertItem (inherited from TOraArray)</td>
<td>Inserts an element at a specific index.</td>
</tr>
<tr>
<td>IsDirty (inherited from TOraObject)</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td>IsLocked (inherited from TOraObject)</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td>Lock (inherited from TOraObject)</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td>MarkDelete (inherited from TOraObject)</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td>MarkUpdate (inherited from TOraObject)</td>
<td>Marks an object as being updated.</td>
</tr>
<tr>
<td>Refresh (inherited from TOraObject)</td>
<td>Retrieves the latest database image for the object.</td>
</tr>
<tr>
<td>Release (inherited from TSharedObject)</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>Unmark (inherited from TOraObject)</td>
<td>Marks an object as not being dirty.</td>
</tr>
</tbody>
</table>

Methods of the TOraNestTable class.

For a complete list of the TOraNestTable class members, see the TOraNestTable Members topic.
Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from TSharedObject)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocObject (inherited from TOraObject)</td>
<td>Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td>AppendItem (inherited from TOraArray)</td>
<td>Appends an element to the end of an array.</td>
</tr>
<tr>
<td>Assign (inherited from TOraArray)</td>
<td>Copies properties and attributes of one object to another one.</td>
</tr>
<tr>
<td>Clear (inherited from TOraArray)</td>
<td>Removes all elements from an array.</td>
</tr>
<tr>
<td>CreateObject (inherited from TOraObject)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td>DeleteItem</td>
<td>Deletes an item pointed by its index.</td>
</tr>
<tr>
<td>Exists (inherited from TOraObject)</td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td>Flush (inherited from TOraObject)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td>FreeObject (inherited from TOraObject)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td>InsertItem (inherited from TOraArray)</td>
<td>Inserts an element at a specific index.</td>
</tr>
<tr>
<td>IsDirty (inherited from TOraObject)</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td>IsLocked (inherited from TOraObject)</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td>Lock (inherited from TOraObject)</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td>MarkDelete (inherited from TOraObject)</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td>MarkUpdate (inherited from TOraObject)</td>
<td>Marks an object as being updated.</td>
</tr>
<tr>
<td>Refresh (inherited from TOraObject)</td>
<td>Retrieves the latest database image for the object.</td>
</tr>
<tr>
<td>Release (inherited from TSharedObject)</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>
Unmark (inherited from TOraObject)

Marks an object as not being dirty.

See Also
- TOraNestTable Class
- TOraNestTable Class Members

5.25.1.2.2.1 DeleteItem Method

Deletes an item pointed by its index.

Class
TOraNestTable

Syntax

procedure DeleteItem(Index: integer);

Parameters
Index

Holds the index of the item to delete.

Remarks
Call the DeleteItem method to delete an item pointed by Index.

5.25.1.3 TOraObject Class

A class representing the Oracle object data type value.

For a list of all members of this type, see TOraObject members.

Unit
oraobjects

Syntax
**Remarks**

TOraObject represents a value of the Oracle object data type. You can get the description of the object type using the ObjectType method. Use the AllocObject method to create an object of a certain type. To access the attribute value use AttrAsInteger, AttrAsString, etc. You can get a TOraObject object using TOraDataSet.GetObject method after fetching rows containing an array field. Also you can get it using the TOraParam.AsObject method.

**Example**

```pascal
var
    MyType : TOraType;
    MyObject : TOraObject;

    MyType := TOraType.Create;
    MyType.Describe('SCOTT.PERSON');
    MyObject := TOraObject.Create(Obj.ObjectType);
    MyObject.AttrAsString('Name') := 'Ja';
```

**Inheritance Hierarchy**

- TSharedObject
  - TDBObject
    - TOraObject

**See Also**

- TOraType
- TOraDataSet.GetObject
- TOraParam.AsObject
- TOraRef
- TOraArray
- TOraNestTable

© 1997-2021 Devart. All Rights Reserved.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttrAsArray</td>
<td>Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td>AttrAsDateTime</td>
<td>Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsFloat</td>
<td>Used to read the attribute's data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsInteger</td>
<td>Used to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsLob</td>
<td>Used to get reference to the TOraLob object that represents the LOBAAttribute value.</td>
</tr>
<tr>
<td>AttrAsObject</td>
<td>Used to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIDate</td>
<td>Used to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCINumber</td>
<td>Used to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIString</td>
<td>Used to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsString</td>
<td>Used to read the attribute’s data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrIsNull</td>
<td>Used to indicate if the</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Indicator</strong></td>
<td>Used to get a pointer to the indicator structure of an object.</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td><strong>IsNull</strong></td>
<td>Used to verify if an object is empty.</td>
</tr>
<tr>
<td><strong>ObjectType</strong></td>
<td>Used to indicate the object type.</td>
</tr>
<tr>
<td><strong>OCISvcCtx</strong></td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td><strong>RefCount</strong></td>
<td>(inherited from <strong>TSharedObject</strong>) Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocObject</strong></td>
<td>Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td><strong>Assign</strong></td>
<td>Copies properties or other attributes from another object.</td>
</tr>
<tr>
<td><strong>CreateObject</strong></td>
<td>Creates an object.</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td><strong>Flush</strong></td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td><strong>FreeObject</strong></td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td><strong>IsLocked</strong></td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Marks an object as locked for update.</td>
</tr>
</tbody>
</table>
### Properties of the **TOraObject** class.

For a complete list of the **TOraObject** class members, see the [TOraObject Members](#) topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttrAsArray</td>
<td>Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td>AttrAsDateTime</td>
<td>Used to read the attribute’s data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsFloat</td>
<td>Used to read the attribute’s data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsInteger</td>
<td>Used to read the attribute’s data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsLob</td>
<td>Used to get reference to the TOraLob object that represents the LOBAttribute value.</td>
</tr>
<tr>
<td>AttrAsObject</td>
<td>Used to read the attribute’s</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AttrAsOCIDate</td>
<td>Used to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCINumber</td>
<td>Used to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIString</td>
<td>Used to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsString</td>
<td>Used to read the attribute's data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrIsNull</td>
<td>Used to indicate if the attribute value is NULL.</td>
</tr>
<tr>
<td>Indicator</td>
<td>Used to get a pointer to the indicator structure of an object.</td>
</tr>
<tr>
<td>Instance</td>
<td>Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td>IsNull</td>
<td>Used to verify if an object is empty.</td>
</tr>
<tr>
<td>ObjectType</td>
<td>Used to indicate the object type.</td>
</tr>
<tr>
<td>OCISvcCtx</td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td>RefCount</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

See Also
- TOraObject Class
- TOraObject Class Members
5.25.1.3.2.1  AttrAsArray Property(Indexer)

Used to get the TOraArray type attribute value.

Class
TOraObject

Syntax

```pascal
property AttrAsArray[const Name: string]: TOraArray;
```

Parameters
Name
Holds an attribute name.

Remarks
Use the AttrAsArray property to get the value of an attribute of the TOraArray type.

Provide an attribute name in the Name index parameter.

5.25.1.3.2.2  AttrAsDateTime Property(Indexer)

Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.

Class
TOraObject

Syntax

```pascal
property AttrAsDateTime[const Name: string]: TDateTime;
```

Parameters
Name
Holds an attribute name.

Remarks
Use the AttrAsDateTime property to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.

5.25.1.3.2.3 AttrAsFloat Property(Indexer)

Used to read the attribute's data value into a double, or to assign a double value to the contents of an attribute.

Class
TораObject

Syntax

```pascal
property AttrAsFloat[const Name: string]: double;
```

Parameters

Name
Holds an attribute name.

Remarks
Use the AttrAsFloat property to read the attribute's data value into a double, or to assign a double value to the contents of an attribute.

5.25.1.3.2.4 AttrAsInteger Property(Indexer)

Used to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute.

Class
TораObject

Syntax

```pascal
property AttrAsInteger[const Name: string]: integer;
```

Parameters
**Name**
Holds an attribute name.

**Remarks**
Use the AttrAsInteger property to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute.

© 1997-2021
Devart. All Rights Reserved.

5.25.1.3.2.5 AttrAsLob Property(Indexer)

Used to get reference to the TOraLob object that represents the LOBAttribute value.

**Class**
**TOraObject**

**Syntax**
```
property AttrAsLob[const Name: string]: TOraLob;
```

**Parameters**
**Name**
Holds an attribute name.

**Remarks**
Use the AttrAsLob property to get reference to the TOraLob object that represents the LOBAttribute value.

**See Also**
- **TOraLob**

© 1997-2021
Devart. All Rights Reserved.

5.25.1.3.2.6 AttrAsObject Property(Indexer)

Used to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute.

**Class**
TOraObject

Syntax

```delphi
property AttrAsObject[const Name: string] : TOraObject;
```

Parameters

- **Name**
  
  Holds an attribute name.

Remarks

Use the AttrAsObject property to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute. Type of an attribute specified by the Name parameter must be Object. You can use AttrAs... properties to access its attributes after.

Another way to get an attribute value of a nested object is using of the name path.

Example

```delphi
Example 1.
Street1 := MyObject.AttrAsObject('Address').AsString('Street');
Street2 := MyObject.AttrAsString('Address.Street');
```

See Also

- TOraObject

© 1997-2021

© 2021 Devart

Devart. All Rights Reserved.
Holds an attribute name.

Remarks

Use the AttrAsOCIDate property to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.

Class

TOraObject

Syntax

```delphi
property AttrAsOCIDate[const Name: string]: OCIString;
```

Parameters

Name
Holds an attribute name.

Remarks

Use the AttrAsOCIDate property to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.

Class

TOraObject

Syntax

```delphi
property AttrAsOCINumber[const Name: string]: OCINumber;
```

Parameters

Name
Holds an attribute name.

Remarks

Use the AttrAsOCINumber property to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.

Class

TOraObject

Syntax

```delphi
property AttrAsOCIString[const Name: string]: OCIString;
```

Parameters

Name
Holds an attribute name.

Remarks

Use the AttrAsOCIString property to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.
property AttrAsOCIString[const Name: string]: pOCIString;

Parameters
Name
Holds an attribute name.

Remarks
Use AttrAsOCIString property to read the attribute's data value as OCIStr, or to assign an OCIStr value to the contents of an attribute.

Class
T0raObject

Syntax
property AttrAsString[const Name: string]: string;

Parameters
Name
Holds an attribute name.

Remarks
Use the AttrAsString property to read the attribute's data value as string, or to assign a string value to the contents of an attribute.

Class
**TOraObject**

**Syntax**

```
property AttrIsNull[const Name: string]: boolean;
```

**Parameters**

Name

Holds an attribute name.

**Remarks**

Check the AttrIsNULL property to learn whether the attribute value is NULL.

© 1997-2021
Devart. All Rights Reserved.

5.25.1.3.2.12 Indicator Property

Used to get a pointer to the indicator structure of an object.

**Class**

TOraObject

**Syntax**

```
property Indicator: IntPtr;
```

**Remarks**

Use the Indicator property to get a pointer to the indicator structure of an object.

© 1997-2021
Devart. All Rights Reserved.

5.25.1.3.2.13 Instance Property

Used to get a pointer to the internal representation of an object.

**Class**

TOraObject

**Syntax**
**property** Instance: IntPtr;

Remarks
Use the Instance property to get a pointer to the internal representation of an object.

5.25.1.3.2.14 IsNull Property

Used to verify if an object is empty.

Class
**TOraObject**

Syntax
**property** IsNull: boolean;

Remarks
Use the IsNull property to verify whether this object is empty. Assign a value to this property to set the object to Null or not Null.

5.25.1.3.2.15 ObjectType Property

Used to indicate the object type.

Class
**TOraObject**

Syntax
**property** ObjectType: **TOraType**;

Remarks
Read the ObjectType property to learn the type of an object.

See Also
5.25.1.3.2.16  OCISvcCtx Property

Used to assign a service context handle.

Class

TOraObject

Syntax

```pascal
property OCISvcCtx: TOCISvcCtx;
```

Remarks

Use the OCISvcCtx property to assign a service context handle. Some operations with objects require a service context handle. To get a service context handle use TOraSession.OCISvcCtx.

See Also

- TOraSession.OCISvcCtx

5.25.1.3.3  Methods

Methods of the TOraObject class.

For a complete list of the TOraObject class members, see the TOraObject Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from TSharedObject)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocObject</td>
<td>Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td>Procedure</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Assign</td>
<td>Copies properties or other attributes from another object.</td>
</tr>
<tr>
<td>CreateObject</td>
<td>Creates an object.</td>
</tr>
<tr>
<td>Exists</td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td>Flush</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td>FreeObject</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td>IsLocked</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td>Lock</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td>MarkDelete</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td>MarkUpdate</td>
<td>Marks an object as being updated.</td>
</tr>
<tr>
<td>Refresh</td>
<td>Retrieves the latest database image for the object.</td>
</tr>
<tr>
<td>Release (inherited from TSharedObject)</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td>Unmark</td>
<td>Marks an object as not being dirty.</td>
</tr>
</tbody>
</table>

See Also
- TOraObject Class
- TOraObject Class Members

© 1997-2021
Devart. All Rights Reserved.

5.25.1.3.3.1 AllocObject Method

Allocates an object instance.

Class
- TOraObject
Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllocObject</td>
<td>Allocates an object instance.</td>
</tr>
<tr>
<td>AllocObject(OCISvcCtx: TOCISvcCtx)</td>
<td>Allocates an object instance.</td>
</tr>
<tr>
<td>AllocObject(OCISvcCtx: TOCISvcCtx; const TypeName: string)</td>
<td>Allocates an object instance.</td>
</tr>
<tr>
<td>AllocObject(const TypeName: string)</td>
<td>Allocates an object instance.</td>
</tr>
</tbody>
</table>

Class

TOraObject

Syntax

```plaintext
procedure AllocObject; overload; virtual;
```

Remarks

Call the AllocObject method to allocate an object instance. Overloaded procedures with parameters modify either service context handle or object type properties before allocating the object.

© 1997-2021 Devart. All Rights Reserved.
Holds the OCI service context.

Remarks

Call the AllocObject method to allocate an object instance. Overloaded procedures with parameters modify either service context handle or object type properties before allocating the object.

© 1997-2021 Devart. All Rights Reserved.

Allocates an object instance.

Class

TOraObject

Syntax

```pascal
procedure AllocObject(OCISvcCtx: TOCISvcCtx; const TypeName: string); overload; virtual;
```

Parameters

**OCISvcCtx**
Holds the OCI service context.

**TypeName**
Holds the name of an Oracle object type that must be allocated.

Remarks

Call the AllocObject method to allocate an object instance. Overloaded procedures with parameters modify either service context handle or object type properties before allocating the object.

© 1997-2021 Devart. All Rights Reserved.

Allocates an object instance.

Class

TOraObject

Syntax
procedure AllocObject(const TypeName: string); overload;
virtual;

Parameters

TypeName
Holds the name of an Oracle object type that must be allocated.

Remarks

Call the Assign method to copy the properties or other attributes from another object.

5.25.1.3.3.2 Assign Method

Copies properties or other attributes from another object.

Class

T0raObject

Syntax

procedure Assign(Source: T0raObject); virtual;

Parameters

Source
Holds the source object to copy properties or other attributes from.

Remarks

Call the Assign method to copy the properties or other attributes from another object.

5.25.1.3.3.3 CreateObject Method

Creates an object.

Class

T0raObject

Syntax

procedure CreateObject(OCISvcCtx: TOCISvcCtx; const TypeName: string); deprecated;
Parameters

SvcCtx
  Holds an OCI service context handle.
TypeName
  Holds the type of the object to create.

Remarks

Call the CreateObject method to create an object.

**Note:** This method is obsolete, so it's better to use the AllocObject method instead.

5.25.1.3.3.4 Exists Method

Verifies if an object instance exists in a database.

Class

T0raObject

Syntax

function Exists: boolean;

Remarks

Call the Exists method to verify whether an instance of an object exists in the database.

5.25.1.3.3.5 Flush Method

Places modifications made to the object to the database.

Class

T0raObject

Syntax

procedure Flush;
Remarks

Call the Flush method to place modifications made to the object into the database.

See the OCIObjectFlush function description in Oracle references for more detailed description of this method.

5.25.1.3.3.6 FreeObject Method

Deallocates and frees an object instance.

Class

TOraObject

Syntax

procedure FreeObject(FreeChild: boolean = True); virtual;

Parameters

FreeChild
Holds True, if there is an object to be deallocated and freed.

Remarks

Call the FreeObject method to deallocate and free an object instance.

5.25.1.3.3.7 IsDirty Method

Verifies if an object instance is marked as dirty.

Class

TOraObject

Syntax

function IsDirty: boolean;

Return Value
True, if the instance is marked as dirty.

Remarks

Call the IsDirty method to verify whether the object instance is marked as dirty.

The return value is True if the instance is dirty.

Class

**TOraObject**

Syntax

```pascal
function IsLocked: boolean;
```

Return Value

True, if the instance is locked.

Remarks

Call the IsLocked method to verify whether the object instance is marked as locked.

The return value is True if the instance is locked.

Class

**TOraObject**

Syntax

```pascal
procedure Lock;
```
Remarks

Call the Lock method to mark an object as locked for update.

See the OCIObjectLock function description in the Oracle references for more detailed description of this method.

MarkDelete Method

Marks an object as being deleted.

Class

TOraObject

Syntax

procedure MarkDelete;

Remarks

Call the MarkDelete method to mark an object as being deleted.

See the OCIObjectMarkDelete function description in the Oracle references for more detailed description of this method.

MarkUpdate Method

Marks an object as being updated.

Class

TOraObject

Syntax

procedure MarkUpdate;

Remarks
Call MarkUpdate method to mark this object as being updated.

See the OCIOBJECTMARKUPDATE function description in the Oracle references for more detailed description of this method.

© 1997-2021 Devart. All Rights Reserved.

5.25.1.3.3.12 Refresh Method

Retrieves the latest database image for the object.

Class
T0raObject

Syntax

procedure Refresh;

Remarks
Call the Refresh method to retrieve the latest database image for the object.

© 1997-2021 Devart. All Rights Reserved.

5.25.1.3.3.13 Unmark Method

Marks an object as not being dirty.

Class
T0raObject

Syntax

procedure Unmark;

Remarks
Call the Unmark method to mark an object as not being dirty.

See the OCIOBJECTUNMARK function description in the Oracle references for more detailed description of this method.

© 1997-2021 Devart. All Rights Reserved.
5.25.1.4 TOraRef Class

A class representing the Oracle reference data type value.

For a list of all members of this type, see TOraRef members.

Unit
oraObjects

Syntax

TOraRef = class(TOraObject);

Remarks

TOraRef represents a value of Oracle reference data type. TOraRef is inherited from TOraObject. You can get TOraRef object by TOraDataSet.GetRef method after fetching rows containing array field. Also you can get it using TOraParam.AsRef method.

Inheritance Hierarchy

TSharedObject
    TDBObject
        TOraObject
            TOraRef

See Also

• TOraObject
• TOraType
• TOraDataSet.GetRef
• TOraParam.AsRef

© 1997-2021
Devart. All Rights Reserved.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsHex</td>
<td>Used to convert the REF value converted to the hexadecimal string.</td>
</tr>
<tr>
<td>AttrAsArray (inherited from TOraObject)</td>
<td>Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td>AttrAsDateTime (inherited from TOraObject)</td>
<td>Used to read the attribute’s data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsFloat (inherited from TOraObject)</td>
<td>Used to read the attribute’s data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsInteger (inherited from TOraObject)</td>
<td>Used to read the attribute’s data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsLob (inherited from TOraObject)</td>
<td>Used to get reference to the TOraLob object that represents the LOBAttribute value.</td>
</tr>
<tr>
<td>AttrAsObject (inherited from TOraObject)</td>
<td>Used to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIDate (inherited from TOraObject)</td>
<td>Used to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCINumber (inherited from TOraObject)</td>
<td>Used to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIString (inherited from TOraObject)</td>
<td>Used to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsString (inherited from TOraObject)</td>
<td>Used to read the attribute's data value as string, or to</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td><strong>AttrIsNull</strong></td>
<td>(inherited from <code>TOraObject</code>) assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>Indicator</strong></td>
<td>(inherited from <code>TOraObject</code>) Used to indicate if the attribute value is NULL.</td>
</tr>
<tr>
<td><strong>Instance</strong></td>
<td>(inherited from <code>TOraObject</code>) Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td><strong>IsNull</strong></td>
<td>(inherited from <code>TOraObject</code>) Used to verify if an object is empty.</td>
</tr>
<tr>
<td><strong>ObjectType</strong></td>
<td>(inherited from <code>TOraObject</code>) Used to indicate the object type.</td>
</tr>
<tr>
<td><strong>OCIRef</strong></td>
<td>Used to get or set the OCIRef handle of reference.</td>
</tr>
<tr>
<td><strong>OCIRefPtr</strong></td>
<td>Provides reference to OCIRef handle.</td>
</tr>
<tr>
<td><strong>OCISvcCtx</strong></td>
<td>(inherited from <code>TOraObject</code>) Used to assign a service context handle.</td>
</tr>
<tr>
<td><strong>RefCount</strong></td>
<td>(inherited from <code>TSharedObject</code>) Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong></td>
<td>(inherited from <code>TSharedObject</code>) Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocObject</strong></td>
<td>(inherited from <code>TOraObject</code>) Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td><strong>Assign</strong></td>
<td>(inherited from <code>TOraObject</code>) Copies properties or other attributes from another object.</td>
</tr>
<tr>
<td><strong>Clear</strong></td>
<td>Invalidates reference so that it would no longer point to an object.</td>
</tr>
<tr>
<td><strong>CreateObject</strong></td>
<td>(inherited from <code>TOraObject</code>) Creates an object.</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>(inherited from <code>TOraObject</code>) Verifies if an object instance exists in a database.</td>
</tr>
</tbody>
</table>
Properties of the TOraRef class.

For a complete list of the TOraRef class members, see the TOraRef Members topic.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsHex</td>
<td>Used to convert the REF</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>AttrAsArray</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td><strong>AttrAsDateTime</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsFloat</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsInteger</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsLob</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to get reference to the TOraLob object that represents the LOBAttribute value.</td>
</tr>
<tr>
<td><strong>AttrAsObject</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCIDate</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCINumber</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsOCIString</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td><strong>AttrAsString</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Used to read the attribute's data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
</tbody>
</table>
### AttrIsNull (inherited from TOraObject)
Used to indicate if the attribute value is NULL.

### Indicator (inherited from TOraObject)
Used to get a pointer to the indicator structure of an object.

### Instance (inherited from TOraObject)
Used to get a pointer to the internal representation of an object.

### IsNull (inherited from TOraObject)
Used to verify if an object is empty.

### ObjectType (inherited from TOraObject)
Used to indicate the object type.

### OCIRef
Used to get or set the OCIRef handle of reference.

### OCIRefPtr
Provides reference to OCIRef handle.

### OCISvcCtx (inherited from TOraObject)
Used to assign a service context handle.

### RefCount (inherited from TSharedObject)
Used to return the count of reference to a TSharedObject object.

---

**See Also**
- TOraRef Class
- TOraRef Class Members

---

**5.25.1.4.2.1 AsHex Property**

Used to convert the REF value converted to the hexadecimal string.

**Class**
TOraRef

**Syntax**

```
property AsHex: string;
```

**Remarks**

Use the AsHex property to get the REF value converted to hexadecimal string.
5.25.1.4.2.2 OCIRef Property

Used to get or set the OCIRef handle of reference.

Class

T0raRef

Syntax

```pascal
property OCIRef: pOCIRef;
```

Remarks

Use the OCIRef property to get or set the OCIRef handle of reference.

See Also

- T0raRef

5.25.1.4.2.3 OCIRefPtr Property

Provides reference to OCIRef handle.

Class

T0raRef

Syntax

```pascal
property OCIRefPtr: ppOCIRef;
```

Remarks

Use the OCIRefPtr property to get reference to OCIRef handle.

See Also

- T0raRef
Methods of the **TOraRef** class.

For a complete list of the **TOraRef** class members, see the [TOraRef Members](#) topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AddRef</code> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><code>AllocObject</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Overloaded. Allocates an object instance.</td>
</tr>
<tr>
<td><code>Assign</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Copies properties or other attributes from another object.</td>
</tr>
<tr>
<td><code>Clear</code></td>
<td>Invalidates reference so that it would no longer point to an object.</td>
</tr>
<tr>
<td><code>CreateObject</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td><code>Exists</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance exists in a database.</td>
</tr>
<tr>
<td><code>Flush</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td><code>FreeObject</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td><code>IsDirty</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td><code>IsLocked</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td><code>Lock</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td><code>MarkDelete</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td><code>MarkUpdate</code> (inherited from <strong>TOraObject</strong>)</td>
<td>Marks an object as being updated.</td>
</tr>
</tbody>
</table>
### Clear Method

Invalidates reference so that it would no longer point to an object.

#### Class

**T0raRef**

#### Syntax

```pascal
procedure Clear;
```

#### Remarks

Call the Clear method to invalidate reference so that it would no longer point to an object.

#### See Also

- **T0raRef**
5.25.1.4.3.2 Pin Method

Pins a referenceable object.

Class

**TOraRef**

Syntax

```plaintext
procedure Pin;
```

Remarks

Call the Pin method to pin a referenceable object.

See the OCIOBJECTPIN function description in the Oracle documentation for more information.

See Also

- **TOraRef**

© 1997-2021 Devart. All Rights Reserved.

5.25.1.4.3.3 RefIsNull Method

Verifies whether reference is associated with an object and the identifier of that object is currently Null.

Class

**TOraRef**

Syntax

```plaintext
function RefIsNull: boolean;
```

Return Value

True, if object identifier is Null. False otherwise.

Remarks

Call the RefIsNull method to verify whether reference is associated with an object and the identifier of that object is currently Null.

The return value is True if object's identifier is Null.
5.25.1.4.3.4  Unpin Method

Unpins a referenceable object.

Class

TOraRef

Syntax

```
procedure Unpin;
```

Remarks

Call the Unpin method to unpin a referenceable object.

See the OCIObjectUnpin function description in the Oracle documentation for more information.

See Also

- TOraRef

5.25.1.5  TOraType Class

A class holding information about Oracle type required for TOraObject objects.

For a list of all members of this type, see TOraType members.

Unit

OraObjects

Syntax

```
TOraType = class(TObjectType);
```
Remarks

The TOraType class represents Oracle type and holds information about type required for TOraObject objects.

Inheritance Hierarchy

TSharedObject
   TObjectType
   TOraType

See Also

- TObjectType
- TOraObject

Members

**TOraType** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeCount (inherited from TObjectType)</td>
<td>Used to indicate the number of attributes of type.</td>
</tr>
<tr>
<td>Attributes (inherited from TObjectType)</td>
<td>Used to access separate attributes.</td>
</tr>
<tr>
<td>DataSize</td>
<td>Used to specify the size for allocating an Oracle object in the memory.</td>
</tr>
<tr>
<td>DataType (inherited from TObjectType)</td>
<td>Used to indicate the type of object dtObject, dtArray or dtTable.</td>
</tr>
<tr>
<td>IndicatorSize</td>
<td>Contains the size of the indicator structure for Oracle object.</td>
</tr>
<tr>
<td>RefCount (inherited from TSharedObject)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
<tr>
<td>Size (inherited from TObjectType)</td>
<td>Used to learn the size of an object instance.</td>
</tr>
</tbody>
</table>
TDO

Used to retrieve the type descriptor object for type.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>FindAttribute (inherited from <strong>TObjectType</strong>)</td>
<td>Indicates whether a specified Attribute component is referenced in the TAttributes object.</td>
</tr>
<tr>
<td>Release (inherited from <strong>TSharedObject</strong>)</td>
<td>Decrements the reference count.</td>
</tr>
</tbody>
</table>

5.25.1.5.2 Properties

Properties of the **TOraType** class.

For a complete list of the **TOraType** class members, see the **TOraType Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AttributeCount (inherited from <strong>TObjectType</strong>)</td>
<td>Used to indicate the number of attributes of type.</td>
</tr>
<tr>
<td>Attributes (inherited from <strong>TObjectType</strong>)</td>
<td>Used to access separate attributes.</td>
</tr>
<tr>
<td>DataSize</td>
<td>Used to specify the size for allocating an Oracle object in the memory.</td>
</tr>
<tr>
<td>DataType (inherited from <strong>TObjectType</strong>)</td>
<td>Used to indicate the type of object dtObject, dtArray or dtTable.</td>
</tr>
<tr>
<td>IndicatorSize</td>
<td>Contains the size of the indicator structure for Oracle object.</td>
</tr>
<tr>
<td>RefCount (inherited from <strong>TSharedObject</strong>)</td>
<td>Used to return the count of</td>
</tr>
</tbody>
</table>
DataSize Property

Used to specify the size for allocating an Oracle object in the memory.

Class
TOraType

Syntax

```pascal
property DataSize: word;
```

Remarks

Use the DataSize property to determine the size for allocating an Oracle object in the memory.

See Also

• IndicatorSize

IndicatorSize Property

Contains the size of the indicator structure for Oracle object.

Class
TOraType
5.25.1.5.2.3 TDO Property

Used to retrieve the type descriptor object for type.

Class

TOraType

Syntax

property TDO: pOCType;

Remarks

Use the TDO property to retrieve the type descriptor object for type.

5.25.1.6 TOraXML Class

A class representing a value of the Oracle SYS.XMLTYPE type.

For a list of all members of this type, see TOraXML members.

Unit

oraObjects

Syntax

TOraXML = class(TOraObject);
Remarks

TOraXML represents a value of Oracle SYS.XMLTYPE type. Use TOraXML.AllocObject method to create an object. Use TOraXML.AsString and TOraXML.LoadFromStream to initialize XML value. You can get TOraXML object by TOraXMLField.AsXML method after fetching rows contained XMLTYPE field. Also you can get it by TOraParam.AsXML method. To manipulate obtained XML document use TOraXML.Extract, TOraXML.Exists, and TOraXML.Transform functions.

Example

```pascal
var
  XMLDoc : TOraXML;
begin
  XMLDoc := TOraXML.Create();
  XMLDoc.OCISvcCtx := OraSession1.OCISvcCtx;
  XMLDoc.AsString := '<root><node1>value</node1></root>';
  ...```

Inheritance Hierarchy

TSharedObject  
  TDBObject  
    TOraObject  
      TOraXML

See Also

- TOraType
- TOraXMLField

© 1997-2021 Devart. All Rights Reserved.

Request Support  
DAC Forum  
Provide Feedback

TOraXML class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsString</td>
<td>Used to get and set the XML document value.</td>
</tr>
<tr>
<td>AttrAsArray</td>
<td>(inherited from TOraObject) Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AttrAsDateTime</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsFloat</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsInteger</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsLob</code></td>
<td>(inherited from <code>TOraObject</code>) Used to get reference to the TOraLob object that represents the LOBAttribute value.</td>
</tr>
<tr>
<td><code>AttrAsObject</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsOCIDate</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsOCINumber</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsOCIString</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrAsString</code></td>
<td>(inherited from <code>TOraObject</code>) Used to read the attribute's data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td><code>AttrIsNull</code></td>
<td>(inherited from <code>TOraObject</code>) Used to indicate if the attribute value is NULL.</td>
</tr>
<tr>
<td><code>Indicator</code></td>
<td>(inherited from <code>TOraObject</code>) Used to get a pointer to the indicator structure of an object.</td>
</tr>
</tbody>
</table>
Instance (inherited from TOraObject) | Used to get a pointer to the internal representation of an object.
IsNull (inherited from TOraObject) | Used to verify if an object is empty.
ObjectType (inherited from TOraObject) | Used to indicate the object type.
OCISvcCtx (inherited from TOraObject) | Used to assign a service context handle.
RefCount (inherited from TSharedObject) | Used to return the count of reference to a TSharedObject object.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddRef (inherited from TSharedObject)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td>AllocObject</td>
<td>Overloaded. Allocates an object instance from an existing TOraLob object.</td>
</tr>
<tr>
<td>Assign (inherited from TOraObject)</td>
<td>Copies properties or other attributes from another object.</td>
</tr>
<tr>
<td>CreateObject (inherited from TOraObject)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td>Exists</td>
<td>Checks if the given set of nodes in the TOraXML exists.</td>
</tr>
<tr>
<td>Extract</td>
<td>Extracts the given set of nodes from the TOraXML.</td>
</tr>
<tr>
<td>Flush (inherited from TOraObject)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td>FreeObject (inherited from TOraObject)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td>GetSchema</td>
<td>Determines the based schema document, schema URL and root element used for the current XMLType creation.</td>
</tr>
<tr>
<td>IsDirty (inherited from TOraObject)</td>
<td>Verifies if an object instance</td>
</tr>
</tbody>
</table>
Properties of the **TOraXML** class.

For a complete list of the **TOraXML** class members, see the [**TOraXML Members**](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsString</td>
<td>Used to get and set the XML document value.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AttrAsArray</td>
<td>Used to get the TOraArray type attribute value.</td>
</tr>
<tr>
<td>AttrAsDateTime</td>
<td>Used to read the attribute's data value into TDateTime, or to assign a TDateTime value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsFloat</td>
<td>Used to read the attribute's data value into a double, or to assign a double value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsInteger</td>
<td>Used to read the attribute's data value as integer, or to assign an integer value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsLob</td>
<td>Used to get reference to the TOraLob object that represents the LOBAttribute value.</td>
</tr>
<tr>
<td>AttrAsObject</td>
<td>Used to read the attribute's data value as TOraObject, or to assign a TOraObject value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIDate</td>
<td>Used to read the attribute's data value as OCIDate, or to assign an OCIDate value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCINumber</td>
<td>Used to read the attribute's data value into OCINumber, or to assign an OCINumber value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsOCIString</td>
<td>Used to read the attribute's data value as OCIString, or to assign an OCIString value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrAsString</td>
<td>Used to read the attribute's data value as string, or to assign a string value to the contents of an attribute.</td>
</tr>
<tr>
<td>AttrIsNull</td>
<td>Used to indicate if the attribute value is NULL.</td>
</tr>
<tr>
<td>Indicator (inherited from TOraObject)</td>
<td>Used to get a pointer to the indicator structure of an object.</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Instance (inherited from TOraObject)</td>
<td>Used to get a pointer to the internal representation of an object.</td>
</tr>
<tr>
<td>IsNull (inherited from TOraObject)</td>
<td>Used to verify if an object is empty.</td>
</tr>
<tr>
<td>ObjectType (inherited from TOraObject)</td>
<td>Used to indicate the object type.</td>
</tr>
<tr>
<td>OCISvcCtx (inherited from TOraObject)</td>
<td>Used to assign a service context handle.</td>
</tr>
<tr>
<td>RefCount (inherited from TSharedObject)</td>
<td>Used to return the count of reference to a TSharedObject object.</td>
</tr>
</tbody>
</table>

See Also
- TOraXML Class
- TOraXML Class Members

5.25.1.6.2.1 AsString Property

Used to get and set the XML document value.

Class
TOraXML

Syntax

```
property AsString: string;
```

Remarks

Use the AsString property to get and set the XML document value. Reading AsString when TOraObject.IsNull is True returns empty string.

See Also
- TOraObject.IsNull

© 1997-2021 Request Support DAC Forum Provide Feedback

© 2021 Devart
Methods of the **TOraXML** class.

For a complete list of the **TOraXML** class members, see the [TOraXML Members](#) topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddRef</strong> (inherited from <strong>TSharedObject</strong>)</td>
<td>Increments the reference count for the number of references dependent on the TSharedObject object.</td>
</tr>
<tr>
<td><strong>AllocObject</strong></td>
<td>Overloaded. Allocates an object instance from an existing <strong>TOraLob</strong> object.</td>
</tr>
<tr>
<td><strong>Assign</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Copies properties or other attributes from another object.</td>
</tr>
<tr>
<td><strong>CreateObject</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Creates an object.</td>
</tr>
<tr>
<td><strong>Exists</strong></td>
<td>Checks if the given set of nodes in the TOraXML exists.</td>
</tr>
<tr>
<td><strong>Extract</strong></td>
<td>Extracts the given set of nodes from the TOraXML.</td>
</tr>
<tr>
<td><strong>Flush</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Places modifications made to the object to the database.</td>
</tr>
<tr>
<td><strong>FreeObject</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Deallocates and frees an object instance.</td>
</tr>
<tr>
<td><strong>GetSchema</strong></td>
<td>Determines the based schema document, schema URL and root element used for the current XMLType creation.</td>
</tr>
<tr>
<td><strong>IsDirty</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance is marked as dirty.</td>
</tr>
<tr>
<td><strong>IsLocked</strong> (inherited from <strong>TOraObject</strong>)</td>
<td>Verifies if an object instance is marked as locked.</td>
</tr>
<tr>
<td><strong>IsSchemaBased</strong></td>
<td>Determines if an XMLType instance is schema-based.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>LoadFromStream</strong></td>
<td>Copies the contents of a stream into the TOraXML object.</td>
</tr>
<tr>
<td><strong>Lock</strong> (inherited from TOraObject)</td>
<td>Marks an object as locked for update.</td>
</tr>
<tr>
<td><strong>MarkDelete</strong> (inherited from TOraObject)</td>
<td>Marks an object as being deleted.</td>
</tr>
<tr>
<td><strong>MarkUpdate</strong> (inherited from TOraObject)</td>
<td>Marks an object as being updated.</td>
</tr>
<tr>
<td><strong>Refresh</strong> (inherited from TOraObject)</td>
<td>Retrieves the latest database image for the object.</td>
</tr>
<tr>
<td><strong>Release</strong> (inherited from TSharedObject)</td>
<td>Decrements the reference count.</td>
</tr>
<tr>
<td><strong>SaveToStream</strong></td>
<td>Copies the contents of a TOraXML object to a stream.</td>
</tr>
<tr>
<td><strong>Transform</strong></td>
<td>Transforms and returns a TOraXML object.</td>
</tr>
<tr>
<td><strong>Unmark</strong> (inherited from TOraObject)</td>
<td>Marks an object as not being dirty.</td>
</tr>
<tr>
<td><strong>Validate</strong></td>
<td>Checks if the input instance conforms to a specified XML schema.</td>
</tr>
</tbody>
</table>

See Also
- TOraXML Class
- TOraXML Class Members

© 1997-2021 Devart. All Rights Reserved.

5.25.1.6.3.1 AllocObject Method

Allocates an object instance from an existing TOraLob object.
Allocates an object instance from an existing `TOraLob` object.

**Class**

`TOraXML`

**Syntax**

```pascal
procedure AllocObject; overload; override;
```

**Remarks**

Call the AllocObject method to allocate an object instance from an existing `TOraLob` object.

The procedure modifies the service context handle before allocating the object.

**See Also**

- `TOraObject.AllocObject`
- `TOraLob`

Allocates an object instance from an existing `TOraLob` object.

**Class**

`TOraXML`

**Syntax**

```pascal
procedure AllocObject(AOCISvcCtx: TOCISvcCtx; AOraLob: TOraLob);
```
### Parameters

**AOCISvcCtx**
- Holds the OCI service context.

**AOraLob**
- Holds a TOraLob object the data from which must be copied to an XML object.

### Remarks

Call the AllocObject method to allocate an object instance from an existing **TOraLob** object. The procedure modifies the service context handle before allocating the object.

© 1997-2021 Devart. All Rights Reserved.

Request Support | DAC Forum | Provide Feedback

Allocates an object instance from an existing **TOraLob** object.

### Class

**TOraXML**

### Syntax

```plaintext
procedure AllocObject(AOCISvcCtx: TOCISvcCtx; const TypeName: string); overload; override;
```

### Parameters

**AOCISvcCtx**
- Holds the OCI service context.

**TypeName**
- Holds the name of an Oracle object type that must be allocated.

### Remarks

Call the AllocObject method to allocate an object instance from an existing **TOraLob** object. The procedure modifies the service context handle before allocating the object.

© 1997-2021 Devart. All Rights Reserved.
Allocates an object instance from an existing TOraLob object.

Class

TOraXML

Syntax

```plaintext
procedure AllocObject(const TypeName: string); overload;
override;
```

Parameters

TypeName

Holds the name of an Oracle object type that must be allocated.

Remarks

Call the AllocObject method to allocate an object instance from an existing TOraLob object. The procedure modifies the service context handle before allocating the object.

© 1997-2021 Devart. All Rights Reserved.

5.25.1.6.3.2 Exists Method

Checks if the given set of nodes in the TOraXML exists.

Class

TOraXML

Syntax

```plaintext
function Exists(const XPathExpr: string; const NSmap: string = ''); boolean;
```

Parameters

XPathExpr

Holds the set of nodes in the TOraXML.

NSmap

Holds a namespace map of TOraXML.

Return Value

True if specified nodes exist in TOraXML, False otherwise.

Remarks
Call the Exists method to check if the given set of nodes in TOraXML exists. This set of nodes is specified by the XPathExpr parameter.

Returns True if specified nodes exist in the TOraXML, otherwise, returns False.

Example

```pascal
var
    RetDoc: TOraXML;
    Res: boolean;
begin
    ...
    Edit;
    TOraXMLField(FieldByName('XMLField')).AsXML.AsString :=
       '<root>
         <x xmlns:edi="http://ecommerce.org/schema"> '+
         '<b>32.18</b> '+
         '<edi:price units="Euro">32.18</edi:price> '+
         '</x> '+
       '</root>';   
    Post;
    ...
    with TOraXMLField(FieldByName('XMLField')).AsXML do begin
        Res := Exists('/root/node1'); //Res = False
    end;
end;
```

See Also
- Extract
- GetSchema
- IsSchemaBased
- Transform

Extracts the given set of nodes from the TOraXML.

Class
- TOraXML

Syntax

```pascal
procedure Extract(RetDoc: TOraXML; XPathExpr: string; NSmap: string; XPathResultType: string; XPathResultVal: string; XPathResult: string; XPathResultVal: string);  
```

© 2021 Devart

5.25.1.6.3.3 Extract Method
string = '');

Parameters

RetDoc
Holds a destination TOraXML object which holds the operation result.

XPathExpr
Holds an expression that specifies the set of nodes to extract.

NSmap
Holds a namespace map of TOraXML.

Remarks

Call the Extract method to extract the given set of nodes from the TOraXML. This set of nodes is specified by the XPathExpr expression. The original document remains unchanged. If no nodes match the specified expression, returns NULL document. RetDoc parameter is a destination TOraXML object which holds the operation result. RetDoc object must be created before passing it as a parameter. Use XPathExpr parameter to specify what nodes to search for. The NSmap parameter is a namespace that can be used to identify the mapping of prefix(es) specified in the XPath_string to the corresponding namespace(s). The format is "xmlns=a.com xmlns:b=b.com".

Example

```
var
  RetDoc: TOraXML;
  Str: string;
begin
  Edit;
  TOraXMLField(FieldByName('XMLField')).AsXML.AsString :=
    '<root> '+
    '<x xmlns:edi="http://ecommerce.org/schema"> '+
    '<edi:price units="Euro">32.18</edi:price> '+
    '</x> '+
    '</root>';
  Post;

  RetDoc := TOraXML.Create();
  RetDoc.OCISvcCtx := OraSession1.OCISvcCtx;
  try
    with TOraXMLField(FieldByName('XMLField')).AsXML do begin
      Extract(RetDoc, '//edi:price', 'xmlns:edi=http://ecommerce.org/schema');
      Str := RetDoc.AsString;
      Extract(RetDoc, '/root/x/b');
      Str := RetDoc.AsString; // Str = '<b>32.18</b>'
    end;
  finally
```
5.25.1.6.3.4  GetSchema Method

Determines the based schema document, schema URL and root element used for the current XMLType creation.

Class

TOraXML

Syntax

procedure GetSchema(SchemaDoc: TOraXML; var SchemaURL: string; var RootElem: string);

Parameters

SchemaDoc
  Holds a destination TOraXML object which holds the operation result.

SchemaURL
  Holds the schema URL.

RootElem
  Holds the root element declared in schema.

Remarks

Call the GetSchema method for schema based TOraXML objects to determine the based schema document, schema URL and root element used for the current XMLType creation. SchemaDoc parameter is a destination TOraXML object which holds the operation result. SchemaDoc object must be created before passing it as a parameter. Use SchemaURL parameter to return based schema URL used for creating XMLType object. Use RootElem parameter to return root element declared in schema that is represented by XML document.
Example

```pascal
var
  RetDoc: TOraXML;
  ResStr: boolean;
begin
  with OraQuery1 do begin
    RetDoc := TOraXML.Create();
    RetDoc.OCISvcCtx := OraSession1.OCISvcCtx;
    try
      with TOraXMLField(FieldByName('XMLField')).AsXML do begin
        GetSchema(RetDoc, SchemaURL, RootElem);
        ResStr := RetDoc.AsString;
      end;
    finally
      RetDoc.Free;
    end;
  end;
end;
```

See Also
- `Exists`
- `Extract`
- `IsSchemaBased`
- `Transform`

© 1997-2021 Devart. All Rights Reserved.

5.25.1.6.3.5 IsSchemaBased Method

Determines if an XMLType instance is schema-based.

Class

`TOraXML`

Syntax

```pascal
function IsSchemaBased: boolean;
```

Return Value

True, if the XML instance is schema-based. False otherwise.

Remarks

Call the `IsSchemaBased` method to determine whether the XMLType instance is schema-
based. Returns True or False depending on whether the XMLType instance is schema-based.

See Also
- Exists
- Extract
- GetSchema
- Transform

5.25.1.6.3.6 LoadFromStream Method

Copies the contents of a stream into the TOraXML object.

Class
TOraXML

Syntax

```pascal
procedure LoadFromStream(Stream: TStream);
```

Parameters
Stream
Holds the name of a stream from which the field's value is copied.

Remarks
Call the LoadFromStream method to copy the contents of a stream into the TOraXML object. Specify the stream from which the field's value is copied as the value of the Stream parameter.

See Also
- SaveToStream
5.25.1.6.3.7  SaveToStream Method

Copies the contents of a TOraXML object to a stream.

Class

TOraXML

Syntax

procedure SaveToStream(Stream: TStream);

Parameters

Stream

Hold the name of a stream to copy the contents of a TOraXML object to.

Remarks

Call the SaveToStream method to copy the contents of a TOraXML object to a stream. Specify the name of the stream to which the field's value is saved as the value of the Stream parameter.

See Also

• LoadFromStream

5.25.1.6.3.8  Transform Method

Transforms and returns a TOraXML object.

Class

TOraXML

Syntax

procedure Transform(XSLDoc: TOraXML; RetDoc: TOraXML);

Parameters

XSLDoc

Holds an XSL document.

RetDoc

Holds a destination TOraXML object which holds the new (transformed) XML document.
Remarks

Call the Transform method to transform and return TOraXML, using the given XSL document in XSLDoc parameter. The new (transformed) XML document is assigned to RetDoc. XSLDoc parameter is the XSL document to be applied to the XMLType. RetDoc parameter is a destination TOraXML object which holds the operation result. RetDoc object must be created before passing it as a parameter.

Example

```pascal
var
  RetDoc, XSLDoc: TOraXML;
begin
  RetDoc := TOraXML.Create();
  RetDoc.OCISvcCtx := OraSession1.OCISvcCtx;
  XSLDoc := TOraXML.Create();
  XSLDoc.OCISvcCtx := OraSession1.OCISvcCtx;
  try
    with TOraXMLField(FieldByName('XMLField')).AsXML do begin
      XSLDoc.AsString:=XSLDocument;
      Transform(XSLDoc, RetDoc);
      Str := RetDoc.AsString;
    end;
  finally
    RetDoc.Free;
    XSLDoc.Free;
  end;
end;
```

See Also

- Exists
- Extract
- GetSchema
- IsSchemaBased

© 1997-2021
Devart. All Rights Reserved.
Syntax

```pascal
function Validate(const SchemaURL: string): boolean;
```

Parameters

- `SchemaURL`: Holds the URL of the XML Schema against which to check the conformance.

Return Value

True, if the input instance conforms to a specified XML schema. False otherwise.

Remarks

Call the Validate method to check if the input instance conforms to a specified XML schema. SchemaURL parameter is the URL of the XML Schema against which to check the conformance.

Example

```pascal
Result := TOraXMLField(FieldByName('XMLField')).AsXML.Validate('http://www.oracle.com/PO.xsd');
```

See Also

- Extract
- GetSchema
- IsSchemaBased
- Transform

© 1997-2021 Devart. All Rights Reserved.

5.26 OraPackage

This unit contains implementation of the TOraPackage component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomOraPackage</td>
<td>A base class for components that provide access to packages stored in an Oracle database.</td>
</tr>
<tr>
<td>TOraPackage</td>
<td>A component providing access to packages stored</td>
</tr>
</tbody>
</table>
5.26.1 Classes

Classes in the OraPackage unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomOraPackage</td>
<td>A base class for components that provide access to packages stored in an Oracle database.</td>
</tr>
<tr>
<td>TOraPackage</td>
<td>A component providing access to packages stored in an Oracle database.</td>
</tr>
</tbody>
</table>

5.26.1.1 TCustomOraPackage Class

A base class for components that provide access to packages stored in an Oracle database.

For a list of all members of this type, see TCustomOraPackage members.

Unit
OraPackage

Syntax

```pascal
TCustomOraPackage = class(TCustomOraComponent);
```

Inheritance Hierarchy

TCustomOraComponent
  TCustomOraPackage
5.26.1.1 Members

TCustomOraPackage class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Params</td>
<td>Used to receive values that are the output parameters of package stored procedures.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session component which is associated with this TOraPackage object.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecProc</td>
<td>Calls the stored procedures defined for a given package.</td>
</tr>
<tr>
<td>ExecProcEx</td>
<td>Calls the stored procedures defined for a given package.</td>
</tr>
<tr>
<td>VariableByName</td>
<td>Provides access to package variables.</td>
</tr>
</tbody>
</table>

Properties of the TCustomOraPackage class.

For a complete list of the TCustomOraPackage class members, see the TCustomOraPackage Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Params</td>
<td>Used to receive values that are the output parameters of package stored procedures.</td>
</tr>
</tbody>
</table>
5.26.1.1.2.1  Params Property

Used to receive values that are the output parameters of package stored procedures.

Class

`TCustomOraPackage`

Syntax

```property` Params: `TOraParams;```

Remarks

Use the Params property to receive values that are the output parameters of package stored procedures.

5.26.1.1.2.2  Session Property

Used to specify the session component which is associated with this TOraPackage object.

Class

`TCustomOraPackage`
Syntax

```property
session: TOraSession;
```

Remarks

Use the Session property to specify the session component which is associated with this TOraPackage object.

At design-time select a session instance from a dropdown listbox.

Methods of the `TCustomOraPackage` class.

For a complete list of the `TCustomOraPackage` class members, see the `TCustomOraPackage Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecProc</td>
<td>Calls the stored procedures defined for a given package.</td>
</tr>
<tr>
<td>ExecProcEx</td>
<td>Calls the stored procedures defined for a given package.</td>
</tr>
<tr>
<td>VariableByName</td>
<td>Provides access to package variables.</td>
</tr>
</tbody>
</table>

See Also

- `TCustomOraPackage Class`
- `TCustomOraPackage Class Members`

Calls the stored procedures defined for a given package.
TCustomOraPackage

Syntax

```delphi
procedure ExecProc(const Name: string); overload;
function ExecProc(Name: string; const Params: array of variant): variant; overload;
```

Parameters

- **Name**
  - Holds the name of the stored procedure.

- **Params**
  - Holds the parameter values array.

Return Value

- A result, if a stored procedure is a function, Null otherwise.

Remarks

Call the ExecProc method to call the stored procedures defined for a given package. The Name parameter is a name of a stored procedure.

If a stored procedure accepts or returns parameters they must be supplied in the Params array in exactly the same order and number as they appear in the declaration of this stored procedure. If the value for an input parameter was not included in the Params array, the parameter default value is taken. Only the parameter values at the end of the list may be unincluded to the parameter values array. If the parameter has no default value, the NULL value is sent.

For example, the following is a stored procedure declaration in package DBMS_ALERT:

```sql
DBMS_ALERT.SIGNAL (    name       IN   VARCHAR2,    message   IN   VARCHAR2);
```

it may be called with this code:

```delphi
MyOraPackage.ExecProc('SIGNAL', ['MySignalName', 'MyMessage']);
```

This is different from the way ExecProcEx is used.

**Note:** Stored functions unlike stored procedures return result values. To understand the parameters usage see TCustomDAConnection.ExecProc.

See Also

- ExecProcEx
- TCustomDAConnection.ExecProc
5.26.1.3.2 ExecProcEx Method

Calls the stored procedures defined for a given package.

Class

TCustomOraPackage

Syntax

function ExecProcEx(Name: string; const Params: array of variant): variant;

Parameters

Name
Holds the name of the stored procedure.

Params
Holds the parameter values array.

Return Value
a result, if a stored procedure is a function, Null otherwise.

Remarks

Call the ExecProcEx method to call stored procedures defined for a given package. Name parameter is a name of a stored procedure.

If the stored procedure accepts or returns parameters, they must be supplied in the Params array as pairs of parameters' names and values so that every value would come immediately after its name. If the value for an input parameter was not included in the Params array, parameter default value is taken. If the parameter has no default value, NULL value is sent.

For example, the following is a stored procedure declaration in package DBMS_ALERT:

DBMS_ALERT.SIGNAL (name IN VARCHAR2,
message IN VARCHAR2);

it may be called by this code:

MyOraPackage.ExecProcEx("SIGNAL", ["name", "MySignalName", "message", "MyMes...
It is different from the way `ExecProc` is used.

See Also

- `ExecProc`
- `TCustomDACConnection.ExecProcEx`
- `TCustomDACConnection.ExecSQLEX`

5.26.1.3.3  VariableByName Method

Provides access to package variables.

Class

`TCustomOraPackage`

Syntax

```plaintext
function VariableByName(Name: string): TVariable;
```

Parameters

**Name**

Holds the name of a variable.

Return Value

A reference for a TVariable object.

Remarks

Call the VariableByName method to get access to package variables.

Reference for a variable with the name not defined yet will prepare a placeholder for it.

5.26.1.2  TOraPackage Class

A component providing access to packages stored in an Oracle database.

For a list of all members of this type, see `TOraPackage` members.

Unit
### Syntax

```delphi
tOraPackage = class(TCustomOraPackage);
```

### Remarks

Use the TOraPackage components to get access to packages stored in Oracle database.

Packages may encapsulate sets of procedures and functions along with related variables and constants. To get the list of Oracle supplied packages refer to the Oracle online documents.

### Inheritance Hierarchy

TCustomOraComponent

   TCustomOraPackage

       TOraPackage

© 1997-2021 Devart. All Rights Reserved.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PackageName</td>
<td>Used to supply an Oracle package name.</td>
</tr>
<tr>
<td>Params</td>
<td>(inherited from TCustomOraPackage) Used to receive values that are the output parameters of package stored procedures.</td>
</tr>
<tr>
<td>Session</td>
<td>(inherited from TCustomOraPackage) Used to specify the session component which is associated with this TOraPackage object.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecProc</td>
<td>(inherited from TCustomOraPackage) Calls the stored procedures defined for a given package.</td>
</tr>
</tbody>
</table>
Properties of the TOraPackage class.

For a complete list of the TOraPackage class members, see the TOraPackage Members topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Params</strong> (inherited from TCustomOraPackage)</td>
<td>Used to receive values that are the output parameters of package stored procedures.</td>
</tr>
</tbody>
</table>

**Published**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PackageName</strong></td>
<td>Used to supply an Oracle package name.</td>
</tr>
<tr>
<td><strong>Session</strong>  (inherited from TCustomOraPackage)</td>
<td>Used to specify the session component which is associated with this TOraPackage object.</td>
</tr>
</tbody>
</table>

See Also

- TOraPackage Class
- TOraPackage Class Members
5.26.1.2.2.1 PackageName Property

Used to supply an Oracle package name.

Class

T0raPackage

Syntax

property PackageName: string;

Remarks

Use the PackageName property to supply an Oracle package name. Subsequent accesses to package methods and variables will be associated with this name.

At design-time dropdown listbox provides all Oracle supplied and user-defined packages to select from.

© 1997-2021 Devart. All Rights Reserved.

5.27 OraProvider

This unit contains implementation of the TOraProvider component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraProvider</td>
<td>A component for loading data to and from a dataset.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback

5.27.1 Classes

Classes in the OraProvider unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraProvider</td>
<td>A component for loading</td>
</tr>
</tbody>
</table>
5.27.1.1 **TOraProvider Class**

A component for loading data to and from a dataset.

For a list of all members of this type, see [TOraProvider members](#).

**Unit**

OraProvider

**Syntax**

```
TOraProvider = class(TDataSetProvider);
```

**Remarks**

TOraProvider provides data to and applies updates from a client dataset. TOraProvider is derived from TDataSetProvider and has same methods and properties as TProvider component.

**Note:** TOraProvider component works with Delphi (C++Builder) Enterprise edition only. So to install it you need to compile and install OraProvider package `oraprovXX.bpk`.

© 1997-2021
Devart. All Rights Reserved.

5.28 **OraScript**

This unit contains implementation of the TOraScript component.

**Classes**
## Classes

Classes in the **OraScript** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraScript</td>
<td>A component for executing several SQL statements one by one.</td>
</tr>
<tr>
<td>TOraStatement</td>
<td>A class used for controlling single SQL statements of the script.</td>
</tr>
<tr>
<td>TOraStatements</td>
<td>A class for holding a collection of <strong>TOraStatement</strong> objects.</td>
</tr>
</tbody>
</table>

### 5.28.1.1 TOraScript Class

A component for executing several SQL statements one by one.

For a list of all members of this type, see **TOraScript** members.

**Unit**

**OraScript**
Syntax

```
TOraScript = class(TDAScript);
```

Remarks

Often it is necessary to execute several SQL statements one by one. Known way is using a lot of components such as TOraSQL. Usually it is not a good solution. Sometimes it can be performed by anonymous PL/SQL block. But sometimes it does not work. For example, DDL statements cannot be used in PL/SQL. With only one TOraScript component you can execute several SQL statements as one. This sequence of statements is named script. To separate single statements use semicolon (;), slash (/), and for PL/SQL - only slash . Note that slash must be the first character in line.

Errors that occur while execution can be processed in the TDAScript.OnError event handler. By default, on error TOraScript shows exception and continues execution.

Inheritance Hierarchy

```
TDAScript
   TOraScript
```

See Also

- TOraSQL

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>(inherited from TDAScript) Used to specify the connection in which the script will be executed.</td>
</tr>
<tr>
<td>DataSet</td>
<td>Used to assign a component that will be used by TOraScript to execute statements, and obtain results of execution.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Debug (inherited from TDAScript)</td>
<td>Used to display the script execution and all its parameter values.</td>
</tr>
<tr>
<td>Delimiter (inherited from TDAScript)</td>
<td>Used to set the delimiter string that separates script statements.</td>
</tr>
<tr>
<td>EndLine (inherited from TDAScript)</td>
<td>Used to get the current statement last line number in a script.</td>
</tr>
<tr>
<td>EndOffset (inherited from TDAScript)</td>
<td>Used to get the offset in the last line of the current statement.</td>
</tr>
<tr>
<td>EndPos (inherited from TDAScript)</td>
<td>Used to get the end position of the current statement.</td>
</tr>
<tr>
<td>Macros (inherited from TDAScript)</td>
<td>Used to change SQL script text in design- or run-time easily.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session in which the script will be executed.</td>
</tr>
<tr>
<td>SQL (inherited from TDAScript)</td>
<td>Used to get or set script text.</td>
</tr>
<tr>
<td>StartLine (inherited from TDAScript)</td>
<td>Used to get the current statement start line number in a script.</td>
</tr>
<tr>
<td>StartOffset (inherited from TDAScript)</td>
<td>Used to get the offset in the first line of the current statement.</td>
</tr>
<tr>
<td>StartPos (inherited from TDAScript)</td>
<td>Used to get the start position of the current statement in a script.</td>
</tr>
<tr>
<td>Statements</td>
<td>Contains the list of statements obtained from the SQL property.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BreakExec (inherited from TDAScript)</td>
<td>Stops script execution.</td>
</tr>
<tr>
<td>ErrorOffset (inherited from TDAScript)</td>
<td>Used to get the offset of the statement if the Execute method raised an exception.</td>
</tr>
<tr>
<td>Execute (inherited from TDAScript)</td>
<td>Executes a script.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ExecuteFile</td>
<td>Executes SQL statements contained in a file.</td>
</tr>
<tr>
<td>ExecuteNext</td>
<td>Executes the next statement in the script and then stops.</td>
</tr>
<tr>
<td>ExecuteStream</td>
<td>Executes SQL statements contained in a stream object.</td>
</tr>
<tr>
<td>FindMacro</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>MacroByName</td>
<td>Finds a macro with the specified name.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a SQL script execution.</td>
</tr>
<tr>
<td>BeforeExecute</td>
<td>Occurs when taking a specific action before executing the current SQL statement is needed.</td>
</tr>
<tr>
<td>OnError</td>
<td>Occurs when Oracle raises an error.</td>
</tr>
</tbody>
</table>

Properties of the **TOraScript** class.

For a complete list of the **TOraScript** class members, see the [TOraScript Members](#) topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>Used to specify the connection in which the script will be executed.</td>
</tr>
<tr>
<td>EndLine</td>
<td>Used to get the current statement last line number in a script.</td>
</tr>
<tr>
<td>EndOffset</td>
<td>Used to get the offset in the</td>
</tr>
</tbody>
</table>
**Table 1:** Properties of the TOraScript Class

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndPos</td>
<td>Used to get the end position of the current statement.</td>
</tr>
<tr>
<td>StartLine</td>
<td>Used to get the current statement start line number in a script.</td>
</tr>
<tr>
<td>StartOffset</td>
<td>Used to get the offset in the first line of the current statement.</td>
</tr>
<tr>
<td>StartPos</td>
<td>Used to get the start position of the current statement in a script.</td>
</tr>
<tr>
<td>Statements</td>
<td>Contains the list of statements obtained from the SQL property.</td>
</tr>
<tr>
<td>DataSet</td>
<td>Used to assign a component that will be used by TOraScript to execute statements, and obtain results of execution.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the script execution and all its parameter values.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Used to set the delimiter string that separates script statements.</td>
</tr>
<tr>
<td>Macros</td>
<td>Used to change SQL script text in design- or run-time easily.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session in which the script will be executed.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to get or set script text.</td>
</tr>
</tbody>
</table>

**Published**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>DataSet</td>
<td>Used to assign a component that will be used by TOraScript to execute statements, and obtain results of execution.</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the script execution and all its parameter values.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Used to set the delimiter string that separates script statements.</td>
</tr>
<tr>
<td>Macros</td>
<td>Used to change SQL script text in design- or run-time easily.</td>
</tr>
<tr>
<td>Session</td>
<td>Used to specify the session in which the script will be executed.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to get or set script text.</td>
</tr>
</tbody>
</table>

**See Also**

- TOraScript Class
- TOraScript Class Members
5.28.1.1.2.1  DataSet Property

Used to assign a component that will be used by TOraScript to execute statements, and obtain results of execution.

Class

TOraScript

Syntax

```pascal
property dataSet: TOraDataSet;
```

See Also

- TOraDataSet
- TDAScript.ExecuteNext

Remarks

Use the Session property to specify the session in which the script will be executed. If Session is not connected, Execute method calls Session.Connect.

See Also

- TOraSession
5.28.1.2.3 Statements Property

Contains the list of statements obtained from the SQL property.

Class
TOraScript

Syntax

```plaintext
property Statements: TOraStatements;
```

Remarks

Contains the list of statements that are obtained from the SQL property. Access the Statements property to view SQL statement, set parameters or execute the specified statement. Statements is a zero-based array of statement records. Index specifies the array element to access.

For example, consider the following script:

```sql
CREATE TABLE A (FIELD1 INTEGER);
INSERT INTO A VALUES(1);
INSERT INTO A VALUES(2);
INSERT INTO A VALUES(3);
CREATE TABLE B (FIELD1 INTEGER);
INSERT INTO B VALUES(1);
INSERT INTO B VALUES(2);
INSERT INTO B VALUES(3);
```

**Note:** The list of statements is created and filled when the value of the Statements property is requested for the first time. That's why the first access to the Statements property can take a long time.

Example

You can use the Statements property as presented below:

```delphi
procedure TForm1.Button1Click(Sender: TObject);

var
i: integer;
begin
  with Script do
  begin
    for i := 0 to Statements.Count - 1 do
      if Copy(Statements[i].SQL, 1, 6) <> 'CREATE' then
        Statements[i].Execute;
  end;
end;
```
5.28.1.2  TOraStatement Class

A class used for controlling single SQL statements of the script.

For a list of all members of this type, see TOraStatement members.

Unit
oraScript

Syntax

TOraStatement = class(TDAStatement);

Remarks

TOraScript contains SQL statements, represented as TOraStatement objects. TOraStatement class has attributes and methods for controlling single SQL statements of the script.

Inheritance Hierarchy

TDAStatement
  TOraStatement

See Also

• TOraScript
• TOraStatements
• TOraStatements

© 1997-2021 Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback

5.28.1.2.1  Members

TOraStatement class overview.
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndLine</td>
<td>Used to determine the number of the last statement line in a script.</td>
</tr>
<tr>
<td>EndOffset</td>
<td>Used to get the offset in the last line of the statement.</td>
</tr>
<tr>
<td>EndPos</td>
<td>Used to get the end position of the statement in a script.</td>
</tr>
<tr>
<td>Omit</td>
<td>Used to avoid execution of a statement.</td>
</tr>
<tr>
<td>Params</td>
<td>Contains parameters for an SQL statement.</td>
</tr>
<tr>
<td>Script</td>
<td>Used to determine the TDAScript object the SQL Statement belongs to.</td>
</tr>
<tr>
<td>SQL</td>
<td>Used to get or set the text of an SQL statement.</td>
</tr>
<tr>
<td>StartLine</td>
<td>Used to determine the number of the first statement line in a script.</td>
</tr>
<tr>
<td>StartOffset</td>
<td>Used to get the offset in the first line of a statement.</td>
</tr>
<tr>
<td>StartPos</td>
<td>Used to get the start position of the statement in a script.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>Executes a statement.</td>
</tr>
</tbody>
</table>

For a complete list of the TOraStatement class members, see the TOraStatement Members topic.
### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>EndLine</code></td>
<td>Used to determine the number of the last statement line in a script.</td>
</tr>
<tr>
<td><code>EndOffset</code></td>
<td>Used to get the offset in the last line of the statement.</td>
</tr>
<tr>
<td><code>EndPos</code></td>
<td>Used to get the end position of the statement in a script.</td>
</tr>
<tr>
<td><code>Omit</code></td>
<td>Used to avoid execution of a statement.</td>
</tr>
<tr>
<td><code>Params</code></td>
<td>Contains parameters for an SQL statement.</td>
</tr>
<tr>
<td><code>Script</code></td>
<td>Used to determine the TDAScript object the SQL Statement belongs to.</td>
</tr>
<tr>
<td><code>SQL</code></td>
<td>Used to get or set the text of an SQL statement.</td>
</tr>
<tr>
<td><code>StartLine</code></td>
<td>Used to determine the number of the first statement line in a script.</td>
</tr>
<tr>
<td><code>StartOffset</code></td>
<td>Used to get the offset in the first line of a statement.</td>
</tr>
<tr>
<td><code>StartPos</code></td>
<td>Used to get the start position of the statement in a script.</td>
</tr>
</tbody>
</table>

See Also
- [TOraStatement Class](#)
- [TOraStatement Class Members](#)

© 1997-2021 Devart. All Rights Reserved.

Contains parameters for an SQL statement.

**Class**

[TOraStatement](#)
Syntax

```
property Params: TOraParams;
```

Remarks

The `Params` property contains parameters for an SQL statement.

Access `Params` at runtime to view and set parameter names, values, and data types dynamically. `Params` is a zero-based array of parameter records. Index specifies the array element to access.

See Also

- `TOraParam`

© 1997-2021 Devart. All Rights Reserved.

5.28.1.3 TOraStatements Class

A class for holding a collection of `TOraStatement` objects.

For a list of all members of this type, see `TOraStatements` members.

Unit

`OraScript`

Syntax

```
TOraStatements = class(TDAStatements);
```

Remarks

Each `TOraStatements` holds a collection of `TOraStatement` objects. `TOraStatements` maintains an index of the statements in its `TOraStatements.Items` array. The `Count` property contains the number of statements in the collection. Use `TOraStatements` class to manipulate script SQL statements.

Inheritance Hierarchy

- `TDAStatements`
- `TOraStatements`
See Also
- TDAScript
- TDASTatement

© 1997-2021
Devart. All Rights Reserved.

5.28.1.3.1 Members

**TOraStatements** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access individual script statements.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

5.28.1.3.2 Properties

Properties of the **TOraStatements** class.

For a complete list of the **TOraStatements** class members, see the **TOraStatements Members** topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>Used to access individual script statements.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

See Also
- TOraStatements Class
- TOraStatements Class Members

© 1997-2021
Devart. All Rights Reserved.
5.28.1.3.2.1  Items Property (Indexer)

Used to access individual script statements.

**Class**

**TOraStatements**

**Syntax**

```delphi
property Items[Index: Integer]: TOraStatement; default;
```

**Parameters**

*Index*

Holds an array of the statements index.

**Remarks**

Use the Items property to access individual script statements. The value of the Index parameter corresponds to the Index property of **TOraStatement**.

**See Also**

- **TOraStatement**

© 1997-2021
Devart. All Rights Reserved.

5.29  **OraSmart**

This unit contains the TSmartQuery and TOraTable components.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomSmartQuery</td>
<td>A base class defining functionality for descendant classes that access database using dynamically generated SQL statements.</td>
</tr>
<tr>
<td>TOraTable</td>
<td>A component for retrieving and updating data in a single table without writing SQL statements.</td>
</tr>
<tr>
<td>TSmartQuery</td>
<td>A component providing</td>
</tr>
</tbody>
</table>
TCustomSmartQuery.Expanded fields feature, that lets all data controls be aware of all the fields belonging to updating table an not only those requested in SELECT clause, and TCustomSmartQuery.Smart Refresh feature (in Professional and Developer editions only).

**TSmartQueryOptions**

This class allows setting up the behaviour of the TSmartQuery class.

### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TSmartState</strong></td>
<td>Specifies if TCustomSmartQuery is in view mode.</td>
</tr>
</tbody>
</table>

### 5.29.1 Classes

Classes in the **OraSmart** unit.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomSmartQuery</td>
<td>A base class defining functionality for descendant classes that access database using dynamically generated SQL statements.</td>
</tr>
<tr>
<td>TOraTable</td>
<td>A component for retrieving and updating data in a single table without writing SQL statements.</td>
</tr>
<tr>
<td>TSmartQuery</td>
<td>A component providing TCustomSmartQuery.Expanded fields feature.</td>
</tr>
</tbody>
</table>
5.29.1.1 TCustomSmartQuery Class

A base class defining functionality for descendant classes that access database using dynamically generated SQL statements.

For a list of all members of this type, see TCustomSmartQuery members.

Unit

OraSmart

Syntax

TCustomSmartQuery = class(TCustomOraQuery);

Remarks

TCustomSmartQuery is a base class that defines functionality for descendant classes that access database using dynamically generated SQL statements. Applications never use TCustomSmartQuery objects directly. Instead they use descendants of TCustomSmartQuery, such as TSmartQuery and TOraTable.

TSmartQuery is an alternative to TOraQuery. It provides TCustomSmartQuery.Expand fields feature, that lets all data controls be aware of all the fields belonging to updating table and not only those requested in a SELECT clause, and TCustomSmartQuery.SmartRefresh feature (in Professional and Developer editions only).
Inheritance Hierarchy

TMemDataSet
  TCustomDADataSet
    TOraDataSet
      TCustomOraQuery
        TCustomSmartQuery

See Also
- TOraQuery
- TSmartQuery
- TOraStoredProc
- TOraTable

© 1997-2021 Devart. All Rights Reserved.

TCustomSmartQuery class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification (inherited from TOraDataSet)</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>CheckMode (inherited from TOraDataSet)</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td>CommandTimeout (inherited from TOraDataSet)</td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>TOracleDataSet</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>DependEvents</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong></td>
<td>TOracleDataSet</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>TOracleDataSet</td>
</tr>
<tr>
<td><strong>Expand</strong></td>
<td></td>
</tr>
<tr>
<td><strong>FetchAll</strong></td>
<td>TOracleDataSet</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FilterSQL</td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td>FinalSQL</td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>IsPLSQL</td>
<td>(inherited from <strong>TOraDataSet</strong>) Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td>IsQuery</td>
<td>(inherited from <strong>TOraDataSet</strong>) Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>(inherited from <strong>TMemDataSet</strong>) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>KeyFields</td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td>KeySequence</td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>LockMode</td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to define when to perform the locking of an editing record.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>MacroCount</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Macros</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>MasterFields</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>MasterSource</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Options</td>
<td></td>
</tr>
<tr>
<td>OptionsDS</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>ParamCount</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>Params</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Prepared</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Ranged</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>RefreshEvent</td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Inherited from</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>RefreshMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>RowsAffected</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Session</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>SmartFetch</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>SmartRefresh</td>
<td></td>
</tr>
<tr>
<td>SmartState</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQLLock</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>(inherited from TCustomDADataset) Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>(inherited from TOuraDataSet) Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>(inherited from TCustomDADataset) Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>(inherited from TOuraDataSet) Used for TOuraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>(inherited from TCustomDADataset) Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td>UpdateObject</td>
<td>(inherited from TOuraDataSet) Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>(inherited from TMemDataSet) Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>UpdatesPending</td>
<td>(inherited from TMemDataSet) Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere (inherited from TCustomDADataset)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange (inherited from TMemDataSet)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates (inherited from TMemDataSet)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>BreakExec (inherited from TCustomDADataset)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td>CancelRange (inherited from TMemDataSet)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates (inherited from TMemDataSet)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates (inherited from TMemDataSet)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>CreateBlobStream (inherited from TCustomDADataset)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td>CreateProcCall (inherited from TOraDataSet)</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td>DeferredPost (inherited from TMemDataSet)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>DeleteWhere (inherited from TCustomDADataset)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td>EditRangeEnd (inherited from TMemDataSet)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>EditRangeStart (inherited from TMemDataSet)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td>ErrorOffset (inherited from TOraDataSet)</td>
<td>Returns the parse error offset.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to find out whether TCustomDADataset has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to learn whether TCustomDADataset is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to learn whether TCustomDADataset is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>(inherited from <strong>TOraDataset</strong>) Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong></td>
<td>(inherited from <strong>TOraDataset</strong>) Retrieves a TOraArray object for a field when only its name is known.</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetDataType</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td><strong>GetErrorPos</strong></td>
<td>(inherited from <strong>TOraDataset</strong>) Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Returns a multireference</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetFieldPrecision</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td><strong>GetFieldScale</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td><strong>GetFile</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraDataFile object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetInterval</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetKeyFieldNames</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td><strong>GetKeyList</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the list of table primary key fields.</td>
</tr>
<tr>
<td><strong>GetLob</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetLobLocator</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetObject</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetOrderBy</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td><strong>GetRef</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraRef object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetTable</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieve a TOraNestTable object for a field with known name.</td>
</tr>
<tr>
<td><strong>GetTimeStamp</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Retrieves a TOraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td><strong>GotoCurrent</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td><strong>Locate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><strong>LocateEx</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Excludes features that don't need to be included to the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Lock</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td><strong>MacroByName</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>OpenNext</strong> (inherited from <code>TOraDataSet</code>)</td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td><strong>ParamByName</strong> (inherited from <code>TOraDataSet</code>)</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td><strong>Prepare</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td><strong>RefreshRecord</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td><strong>RestoreSQL</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong> (inherited from <code>TMemDataSet</code>)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td><strong>RevertRecord</strong> (inherited from <code>TMemDataSet</code>)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>SaveSQL</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td><strong>SaveToXML</strong> (inherited from <code>TMemDataSet</code>)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><strong>SetOrderBy</strong> (inherited from <code>TCustomDADataset</code>)</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
<tr>
<td><strong>SetRange</strong> (inherited from <code>TMemDataSet</code>)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><strong>SetRangeEnd</strong> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><strong>SetRangeStart</strong> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates that subsequent assignments to field values</td>
</tr>
</tbody>
</table>
**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AfterExecute</strong> (inherited from <code>TCustomDataDataSet</code>)</td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td><strong>AfterFetch</strong> (inherited from <code>TCustomDataDataSet</code>)</td>
<td>Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td><strong>AfterSmartRefresh</strong></td>
<td>Occurs after the Smart Refresh procedure was performed by <code>TCustomOraQuery</code>.</td>
</tr>
<tr>
<td><strong>AfterUpdateExecute</strong> (inherited from <code>TCustomDataDataSet</code>)</td>
<td>Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
<tr>
<td><strong>BeforeFetch</strong> (inherited from <code>TCustomDataDataSet</code>)</td>
<td>Occurs before dataset is going to fetch block of records from the server.</td>
</tr>
<tr>
<td><strong>BeforeUpdateExecute</strong> (inherited from)</td>
<td>Occurs before executing insert, delete, update, lock,</td>
</tr>
</tbody>
</table>
**Properties of the `TCustomSmartQuery` class.**

For a complete list of the `TCustomSmartQuery` class members, see the `TCustomSmartQuery Members` topic.

### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>BaseSQL</code> (inherited from <code>TCustomDADataSet</code>)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td><code>CachedUpdates</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><code>ChangeNotification</code> (inherited from <code>TOraDataSet</code>)</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td><code>CheckMode</code> (inherited from <code>TOraDataSet</code>)</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td><code>CommandTimeout</code> (inherited from <code>TOraDataSet</code>)</td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Conditions</td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td>Connection</td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td>Cursor</td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td>DataTypeMap</td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td>Debug</td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td>DependEvents</td>
<td>Used to get or set the names of the events that dataset will depend on in TCustomSmartQuery.Smart Refresh mode.</td>
</tr>
<tr>
<td>DetailFields</td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td>Disconnected</td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td>DMLRefresh</td>
<td>Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td>Encryption</td>
<td>Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td>Expand</td>
<td>Lets all data controls be aware of all the fields belonging to updating table.</td>
</tr>
<tr>
<td>FetchAll</td>
<td>Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td>FetchRows</td>
<td>Used to define the number of rows to be transferred across the network at the</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>FilterSQL</strong> (inherited from TCustomDADataSet)</td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong> (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong> (inherited from TMemDataSet)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>IsPLSQL</strong> (inherited from TOraDataSet)</td>
<td>Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td><strong>IsQuery</strong> (inherited from TOraDataSet)</td>
<td>Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong> (inherited from TMemDataSet)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>KeyFields</strong> (inherited from TOraDataSet)</td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td><strong>KeySequence</strong> (inherited from TOraDataSet)</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong> (inherited from TMemDataSet)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong> (inherited from TMemDataSet)</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>LockMode</strong> (inherited from TOraDataSet)</td>
<td>Used to define when to perform the locking of an editing record.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>MacroCount</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td><strong>Macros</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td><strong>MasterFields</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td><strong>MasterSource</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td><strong>NonBlocking</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to execute a SQL statement and fetch rows by a separate thread.</td>
</tr>
<tr>
<td><strong>Options</strong></td>
<td>Used to specify the behaviour of TCustomSmartQuery object.</td>
</tr>
<tr>
<td><strong>OptionsDS</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the behaviour of TOraDataSetObject.</td>
</tr>
<tr>
<td><strong>ParamCheck</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to indicate how many parameters are there in the Params property.</td>
</tr>
<tr>
<td><strong>Params</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Contains the parameters for a query's SQL statement.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>ReadOnly</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td><strong>RefreshEvent</strong></td>
<td>Used to get or set the name</td>
</tr>
</tbody>
</table>
of the event that dataset will be waiting for in TCustomSmartQuery.Smart Refresh mode.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Inherited From</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefreshMode</td>
<td>TOraDataSet</td>
<td>Used to specify when to refresh an editing record.</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>TCustomDADataSet</td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>TOraDataSet</td>
<td>Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td>RowsAffected</td>
<td>TCustomDADataSet</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>TOraDataSet</td>
<td>Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>TOraDataSet</td>
<td>Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td>Session</td>
<td>TOraDataSet</td>
<td>Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td>SmartFetch</td>
<td>TOraDataSet</td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td>SmartRefresh</td>
<td></td>
<td>Let TCustomSmartQuery components work in a concurrent environment.</td>
</tr>
<tr>
<td>SmartState</td>
<td></td>
<td>Defines if TCustomSmartQuery is in view mode.</td>
</tr>
<tr>
<td>SQL</td>
<td>TCustomDADataSet</td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>TCustomDADataSet</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>TCustomDADataSet</td>
<td>Used to specify the SQL statement that will be used</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>SQLLock</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify a SQL statement that will be used to perform a record lock.</td>
<td></td>
</tr>
<tr>
<td><strong>SQLRecCount</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
<td></td>
</tr>
<tr>
<td><strong>SQLRefresh</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify a SQL statement that will be used to refresh current record by calling the <strong>TCustomDADataset.RefreshRecord</strong> procedure.</td>
<td></td>
</tr>
<tr>
<td><strong>SQLType</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
<td></td>
</tr>
<tr>
<td><strong>SQLUpdate</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
<td></td>
</tr>
<tr>
<td><strong>StrictUpdate</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used for <strong>TOraDataSet</strong> to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
<td></td>
</tr>
<tr>
<td><strong>UniDirectional</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used if an application does not need bidirectional access to records in the result set.</td>
<td></td>
</tr>
<tr>
<td><strong>UpdateObject</strong></td>
<td>(inherited from <strong>TOraDataSet</strong>) Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
<td></td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to indicate the update status for the current record when cached updates are enabled.</td>
<td></td>
</tr>
<tr>
<td><strong>UpdatesPending</strong></td>
<td>(inherited from <strong>TMemDataSet</strong>) Used to check the status of the cached updates buffer.</td>
<td></td>
</tr>
</tbody>
</table>
See Also
- TCustomSmartQuery Class
- TCustomSmartQuery Class Members

5.29.1.1.2.1 DependEvents Property

Used to get or set the names of the events that dataset will depend on in SmartRefresh mode.

Class
TCustomSmartQuery

Syntax
```
property DependEvents: string;
```

Remarks
Use DependEvents property to get or set the names of the events that dataset will depend on in SmartRefresh mode. In the other words, when some TCustomSmartQuery object receives RefreshEvent, it causes an update of every dataset that has received event name in DependEvents list, i.e. all depending datasets are updated.

See Also
- SmartRefresh
- RefreshEvent

5.29.1.1.2.2 Expand Property

Lets all data controls be aware of all the fields belonging to updating table.

Class
TCustomSmartQuery

Syntax
**property** Expand: boolean **default** False;

Remarks

Set Expand property to True to let all data controls be aware of all the fields belonging to updating table an not only those requested in the SELECT clause. Those fields which are not requested explicitly will show no data until the dataset enters edit state for a particular record. At that time all fields for that record will be populated from the database.

Preferred design for an application may include DBGrid component which hides unwanted fields and other data components which reveal those fields only for the selected record. This way large tables with lots of fields are edited by the user only on demand during the course of a session. Thus network traffic may be lessened and memory usage lowered for transfers of the requested fields only.

If Expand property is True you can point in a SELECT statement the fields that will be displayed only in the grid. The rest of the fields from the updating table will be fetched before edit.

**Note:** This property is mutually exclusive with CachedUpdates property. Thus setting Expand to True leaves CachedUpdates property set to False. Also do not try to read blob fields when they are not expanded, this will cause an exception.

The default value is False.

© 1997-2021 Devart. All Rights Reserved.

5.29.1.1.2.3 Options Property

Used to specify the behaviour of TCustomSmartQuery object.

Class

**TCustomSmartQuery**

Syntax

```
**property** Options: **TSmartQueryOptions**;
```

Remarks

Set properties of Options to specify the behaviour of a TCustomSmartQuery object.
5.29.1.1.2.4 RefreshEvent Property

Used to get or set the name of the event that dataset will be waiting for in SmartRefresh mode.

Class

TCustomSmartQuery

Syntax

```property RefreshEvent: string;```

Remarks

Use RefreshEvent property to get or set the name of the event that dataset will be waiting for in the SmartRefresh mode. When this event occurs, dataset performs the Refresh procedure.

See Also

- SmartRefresh
- DependEvents

5.29.1.1.2.5 SmartRefresh Property

Let TCustomSmartQuery components work in a concurrent environment.

Class

TCustomSmartQuery

Syntax
property SmartRefresh: boolean default False;

Remarks
Set SmartRefresh property to True to let TCustomSmartQuery components work in a concurrent environment. Applications which instantiate TCustomSmartQuery descendants may notify each other about their activity on a shared database and be updated each time the database gets modified.

Setting SmartRefresh property to False indicates that concurrent sessions will refresh their datasets on their own.

Class
TCustomSmartQuery

Syntax
property SmartState: TSmartState;

Remarks
Check SmartState property to learn whether TCustomSmartQuery is in view mode.

TCustomSmartQuery is in view mode (SmartState is dsView) after calling View method.

See Also
• View

Methods of the TCustomSmartQuery class.

For a complete list of the TCustomSmartQuery class members, see the TCustomSmartQuery Members topic.
**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset’s pending cached updates to a database.</td>
</tr>
<tr>
<td>BreakExec (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td>CancelRange (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>CreateBlobStream (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td>CreateProcCall (inherited from <strong>TOraDataSet</strong>)</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td>DeferredPost (inherited from <strong>TMemDataSet</strong>)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>DeleteWhere (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td>EditRangeEnd (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>EditRangeStart (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td>ErrorOffset (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the parse error offset.</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>TOraDataSet</td>
</tr>
<tr>
<td><strong>GetArray</strong></td>
<td>TOraDataSet</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>TMemDataSet</td>
</tr>
<tr>
<td><strong>GetDataType</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td><strong>GetErrorPos</strong></td>
<td>TOraDataSet</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong></td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>GetFieldPrecision</code></td>
<td>(inherited from <code>TCustomDADataSet</code>) Retrieves the precision of a number field.</td>
</tr>
<tr>
<td><code>GetFieldScale</code></td>
<td>(inherited from <code>TCustomDADataSet</code>) Retrieves the scale of a number field.</td>
</tr>
<tr>
<td><code>GetFile</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td><code>GetInterval</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td><code>GetKeyFieldNames</code></td>
<td>(inherited from <code>TCustomDADataSet</code>) Provides a list of available key field names.</td>
</tr>
<tr>
<td><code>GetKeyList</code></td>
<td>(inherited from <code>TOraDataSet</code>) Returns the list of table primary key fields.</td>
</tr>
<tr>
<td><code>GetLob</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><code>GetLobLocator</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td><code>GetObject</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td><code>GetOrderBy</code></td>
<td>(inherited from <code>TCustomDADataSet</code>) Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td><code>GetRef</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraRef object for a field with known name.</td>
</tr>
<tr>
<td><code>GetTable</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieve a TOraNestTable object for a field with known name.</td>
</tr>
<tr>
<td><code>GetTimeStamp</code></td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td><code>GotoCurrent</code></td>
<td>(inherited from <code>TCustomDADataSet</code>) Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td><code>Locate</code></td>
<td>(inherited from <code>TMemDataSet</code>) Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><code>LocateEx</code></td>
<td>(inherited from <code>TMemDataSet</code>) Overloaded. Excludes features that don't need to be included to the</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td>Lock</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>MacroByName</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>OpenNext</td>
<td><code>TOraDataSet</code></td>
</tr>
<tr>
<td>ParamByName</td>
<td><code>TOraDataSet</code></td>
</tr>
<tr>
<td>Prepare</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>RefreshRecord</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>RestoreSQL</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td>RevertRecord</td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td>SaveSQL</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>SaveToXML</td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td>SetOrderBy</td>
<td><code>TCustomDADataset</code></td>
</tr>
<tr>
<td>SetRange</td>
<td><code>TMemDataSet</code></td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td><code>TMemDataSet</code></td>
</tr>
</tbody>
</table>
| SetRangeStart | `TMemDataSet` | Indicates that subsequent assignments to field values
### View Method

Allows viewing all fields of the updating table.

#### Class

**TCustomSmartQuery**

#### Syntax

```plaintext
procedure View;
```

#### Remarks

Specify the start of the range of rows to include in the dataset.

**SQLSaved** (inherited from **TCustomDADataset**)  
Determines if the **SQL** property value was saved to the **BaseSQL** property.

**UnLock** (inherited from **TCustomDADataset**)  
Releases a record lock.

**UnPrepare** (inherited from **TMemDataSet**)  
Frees the resources allocated for a previously prepared query on the server and client sides.

**UpdateResult** (inherited from **TMemDataSet**)  
Reads the status of the latest call to the **ApplyUpdates** method while cached updates are enabled.

**UpdateStatus** (inherited from **TMemDataSet**)  
Indicates the current update status for the dataset when cached updates are enabled.

**View**  
Allows viewing all fields of the updating table.
Useful when Expand is True. Lets view all fields of the updating table. Call Cancel method to return dataset to dsBrowse mode. To indicate view mode check SmartState. After calling View method SmartState is dsView.

See Also
- SmartState

© 1997-2021
Devart. All Rights Reserved.

5.29.1.1.4 Events

Events of the TCustomSmartQuery class.

For a complete list of the TCustomSmartQuery class members, see the TCustomSmartQuery Members topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute (inherited from TCustomDADataSet)</td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td>AfterFetch (inherited from TCustomDADataSet)</td>
<td>Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td>AfterSmartRefresh</td>
<td>Occurs after the Smart Refresh procedure was performed by TCustomOraQuery.</td>
</tr>
<tr>
<td>AfterUpdateExecute (inherited from TCustomDADataSet)</td>
<td>Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
<tr>
<td>BeforeFetch (inherited from TCustomDADataSet)</td>
<td>Occurs before dataset is going to fetch block of records from the server.</td>
</tr>
<tr>
<td>BeforeUpdateExecute (inherited from TCustomDADataSet)</td>
<td>Occurs before executing insert, delete, update, lock, and refresh operations.</td>
</tr>
<tr>
<td>OnUpdateError (inherited from TMemDataSet)</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
</tbody>
</table>
| OnUpdateRecord (inherited from TMemDataSet)    | Occurs when a single }
RefreshFields

Occures before an application posts changes for the current record to the database or cache.

See Also
- TCustomSmartQuery Class
- TCustomSmartQuery Class Members

5.29.1.1.4.1 AfterSmartRefresh Event

Occurs after the Smart Refresh procedure was performed by TCustomOraQuery.

Class
TCustomSmartQuery

Syntax

```
property AfterSmartRefresh: TDataSetNotifyEvent;
```

Remarks

Occurs every time after TCustomOraQuery performes Smart Refresh procedure.

See Also
- SmartRefresh

5.29.1.1.4.2 RefreshFields Event

Occures before an application posts changes for the current record to the database or cache.

Class

TCustomSmartQuery
Syntax

```pascal
property RefreshFields: TDataSetNotifyEvent;
```

Remarks

Occurs before an application posts changes for the current record to the database or cache.

When Expand is True, write RefreshFields event handler to assign a value to the nonupdatting fields.

Example

```pascal
procedure quEmpRefreshFields(DataSet: TDataSet);
begin
    DataSet.FieldByName('DNAME').AsString:= quDept.FieldByName('DNAME').AsString;
end;
```

5.29.1.2 TOraTable Class

A component for retrieving and updating data in a single table without writing SQL statements.

For a list of all members of this type, see TOraTable members.

Unit

OraSmart

Syntax

```pascal
TOraTable = class(TCustomSmartQuery);
```

Remarks

The TOraTable component allows retrieving and updating data in a single table without writing SQL statements. Use TOraTable to access data in a table. Use the TableName property to specify table name. TOraTable uses the KeyFields property to build SQL statements for updating table data. KeyFields is a string containing a semicolon-delimited list of the field names. If KeyFields is not defined before opening, TOraTable uses primary or unique key or ROWID pseudo field.

Inheritance Hierarchy
TMemDataSet
TCustomDADataSet
TOraDataSet
TCustomOraQuery
TCustomSmartQuery
TOraTable

See Also
• TSmartQuery
• Updating Data with ODAC Dataset Components
• Master/Detail Relationships

TOraTable class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification (inherited from TOraDataSet)</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>CheckMode (inherited from TOraDataSet)</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td>CommandTimeout (inherited from TOraDataSet)</td>
<td>Sets the wait time before a request is sent to the server to terminate the attempt to execute or fetch the current SQL statement.</td>
</tr>
<tr>
<td>Conditions (inherited from TCustomDADataSet)</td>
<td>Used to add WHERE</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>(inherited from <strong>TOradata</strong>set) Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to set data type mapping rules.</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DependEvents</strong></td>
<td>(inherited from <strong>TCustomSmartQuery</strong>) Used to get or set the names of the events that dataset will depend on in <strong>TCustomSmartQuery.Smart Refresh</strong> mode.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong></td>
<td>(inherited from <strong>TOradata</strong>set) Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>(inherited from <strong>TOradata</strong>set) Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>Expand</strong></td>
<td>(inherited from <strong>TCustomSmartQuery</strong>) Lets all data controls be aware of all the fields belonging to updating table.</td>
</tr>
<tr>
<td><strong>FetchAll</strong></td>
<td>Defines whether to request all records of the query from database server when the dataset is being opened.</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td><strong>FilterSQL</strong></td>
<td>(inherited from <strong>TCustomDADataset</strong>) Used to change the WHERE conditions to a query</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FinalSQL (inherited from TCustomDADataSet)</td>
<td>clause of SELECT statement and reopen a query. Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td>IndexFieldNames (inherited from TMemDataSet)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>IsPLSQL (inherited from TOraDataSet)</td>
<td>Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td>IsQuery (inherited from TOraDataSet)</td>
<td>Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td>KeyExclusive (inherited from TMemDataSet)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>KeyFields (inherited from TOraDataSet)</td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td>KeySequence (inherited from TOraDataSet)</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints (inherited from TMemDataSet)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate (inherited from TMemDataSet)</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>LockMode</td>
<td>Used to specify what kind of lock will be performed when editing a record.</td>
</tr>
<tr>
<td>MacroCount (inherited from TCustomDADataSet)</td>
<td>Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Macros</strong> (inherited from TCustomDADataSet)</td>
<td>Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td><strong>MasterFields</strong> (inherited from TCustomDADataSet)</td>
<td>Used to specify the names of one or more fields that are used as foreign keys for dataset when establishing detail/master relationship between it and the dataset specified in MasterSource.</td>
</tr>
<tr>
<td><strong>MasterSource</strong> (inherited from TCustomDADataSet)</td>
<td>Used to specify the data source component which binds current dataset to the master one.</td>
</tr>
<tr>
<td><strong>NonBlocking</strong> (inherited from TOracleDataSet)</td>
<td>Used to execute a SQL statement and fetch rows by a separate thread.</td>
</tr>
<tr>
<td><strong>Options</strong> (inherited from TCustomSmartQuery)</td>
<td>Used to specify the behaviour of TCustomSmartQuery object.</td>
</tr>
<tr>
<td><strong>OptionsDS</strong> (inherited from TOracleDataSet)</td>
<td>Used to specify the behaviour of TOracleDataSetObject.</td>
</tr>
<tr>
<td><strong>OrderFields</strong></td>
<td>Used to build ORDER BY clause of SQL statements.</td>
</tr>
<tr>
<td><strong>ParamCheck</strong> (inherited from TCustomDADataSet)</td>
<td>Used to specify whether parameters for the Params property are generated automatically after the SQL property was changed.</td>
</tr>
<tr>
<td><strong>ParamCount</strong> (inherited from TCustomDADataSet)</td>
<td>Used to indicate how many parameters are there in the Params property.</td>
</tr>
<tr>
<td><strong>Params</strong> (inherited from TOracleDataSet)</td>
<td>Contains the parameters for a query's SQL statement.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from TMemDataSet)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from TMemDataSet)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>ReadOnly</strong> (inherited from TCustomDADataSet)</td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
</tbody>
</table>
| **RefreshEvent** (inherited from TCustomSmartQuery) | Used to get or set the name of the event that dataset will
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RefreshMode</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify when to refresh an editing record.</td>
</tr>
<tr>
<td><strong>RefreshOptions</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>RowsProcessed</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td><strong>SequenceMode</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td><strong>Session</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td><strong>SmartFetch</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td><strong>SmartRefresh</strong> (inherited from <strong>TCustomSmartQuery</strong>)</td>
<td>Let TCustomSmartQuery components work in a concurrent environment.</td>
</tr>
<tr>
<td><strong>SmartState</strong> (inherited from <strong>TCustomSmartQuery</strong>)</td>
<td>Defines if TCustomSmartQuery is in view mode.</td>
</tr>
<tr>
<td><strong>SQL</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td><strong>SQLDelete</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td><strong>SQLInsert</strong> (inherited from <strong>TCustomDADataSet</strong>)</td>
<td>Used to specify the SQL statement that will be used when applying an insertion</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQLLock</td>
<td>(inherited from TCustomDADataSet) Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>(inherited from TCustomDADataSet) Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>(inherited from TCustomDADataSet) Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataSet.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>(inherited from TOraDataSet) Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>(inherited from TCustomDADataSet) Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>(inherited from TOraDataSet) Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td>TableName</td>
<td>Used to specify the name of the database table this component encapsulates.</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>(inherited from TCustomDADataSet) Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td>UpdateObject</td>
<td>(inherited from TOraDataSet) Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>(inherited from TMemDataSet) Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>UpdatesPending</td>
<td>(inherited from TMemDataSet) Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddWhere</strong> (inherited from TCustomDADataSet)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td><strong>ApplyRange</strong> (inherited from TMemDataSet)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from TMemDataSet)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong> (inherited from TCustomDADataSet)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from TMemDataSet)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from TMemDataSet)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from TMemDataSet)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong> (inherited from TCustomDADataSet)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong> (inherited from TOraDataSet)</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong> (inherited from TMemDataSet)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>DeleteWhere</strong> (inherited from TCustomDADataSet)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong> (inherited from TMemDataSet)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong> (inherited from TMemDataSet)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>EmptyTable</strong></td>
<td>Truncates the current table content on the server.</td>
</tr>
<tr>
<td>Method</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td>(inherited from TораDataSet)</td>
</tr>
<tr>
<td>Execute</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>Executing</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>Fetched</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>Fetching</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>FetchingAll</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>FindKey</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>FindMacro</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>FindNearest</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>FindParam</td>
<td>(inherited from ToraDataSet)</td>
</tr>
<tr>
<td>GetArray</td>
<td>(inherited from TораDataSet)</td>
</tr>
<tr>
<td>GetBlob</td>
<td>(inherited from TMemDataSet)</td>
</tr>
<tr>
<td>GetDataType</td>
<td>(inherited from TCustomDADataset)</td>
</tr>
<tr>
<td>GetErrorPos</td>
<td>(inherited from TораDataSet)</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetFieldObject</td>
<td>(inherited from TCustomDADataset) Returns a multireference shared object from field.</td>
</tr>
<tr>
<td>GetFieldPrecision</td>
<td>(inherited from TCustomDADataset) Retrieves the precision of a number field.</td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>(inherited from TCustomDADataset) Retrieves the scale of a number field.</td>
</tr>
<tr>
<td>GetFile</td>
<td>(inherited from TOraDataSet) Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td>GetInterval</td>
<td>(inherited from TOraDataSet) Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>(inherited from TCustomDADataset) Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetKeyList</td>
<td>(inherited from TOraDataSet) Returns the list of table primary key fields.</td>
</tr>
<tr>
<td>GetLob</td>
<td>(inherited from TOraDataSet) Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td>GetLobLocator</td>
<td>(inherited from TOraDataSet) Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td>GetObject</td>
<td>(inherited from TOraDataSet) Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>(inherited from TCustomDADataset) Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td>GetRef</td>
<td>(inherited from TOraDataSet) Retrieves a TOraRef object for a field with known name.</td>
</tr>
<tr>
<td>GetTable</td>
<td>(inherited from TOraDataSet) Retrieve a TOraNestTable object for a field with known name.</td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td>(inherited from TOraDataSet) Retrieves a TOraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>(inherited from TCustomDADataset) Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td>Locate</td>
<td>(inherited from TMemDataSet) Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx</td>
<td>(inherited from TMemDataSet) Overloaded. Excludes</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Lock (inherited from TCustomDADataSet)</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td>MacroByName (inherited from TCustomDADataSet)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>OpenNext (inherited from TOraDataSet)</td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td>ParamByName (inherited from TOraDataSet)</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td>Prepare (inherited from TCustomDADataSet)</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>PrepareSQL</td>
<td>Determines KeyFields and builds query of TOraTable.</td>
</tr>
<tr>
<td>RefreshRecord (inherited from TCustomDADataSet)</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td>RestoreSQL (inherited from TCustomDADataSet)</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td>RestoreUpdates (inherited from TMemDataSet)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord (inherited from TMemDataSet)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveSQL (inherited from TCustomDADataSet)</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td>SaveToXML (inherited from TMemDataSet)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetOrderBy (inherited from TCustomDADataSet)</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
<tr>
<td>SetRange (inherited from TMemDataSet)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td>SetRangeEnd (inherited from TMemDataSet)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range.</td>
</tr>
</tbody>
</table>

Features that don’t need to be included to the TMemDataSet Locate method of TDataSet.
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetRangeStart</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SQLSaved</td>
<td>Determines if the SQL property value was saved to the BaseSQL property.</td>
</tr>
<tr>
<td>UnLock</td>
<td>Releases a record lock.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>UpdateResult</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
<tr>
<td>View</td>
<td>Allows viewing all fields of the updating table.</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td>AfterFetch</td>
<td>Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td>AfterSmartRefresh</td>
<td>Occurs after the Smart Refresh procedure was performed by TCustomOraQuery.</td>
</tr>
<tr>
<td>AfterUpdateExecute</td>
<td>Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
</tbody>
</table>
| BeforeFetch | Occurs before dataset is
BeforeUpdateExecute (inherited from TCustomDADataSet)  
Occurs before executing insert, delete, update, lock, and refresh operations.

OnUpdateError (inherited from TMemDataSet)  
Occurs when an exception is generated while cached updates are applied to a database.

OnUpdateRecord (inherited from TMemDataSet)  
Occurs when a single update component can not handle the updates.

RefreshFields (inherited from TCustomSmartQuery)  
Occures before an application posts changes for the current record to the database or cache.

### 5.29.1.2.2 Properties

Properties of the TOraTable class.

For a complete list of the TOraTable class members, see the TOraTable Members topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification (inherited from TOraDataSet)</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>CheckMode (inherited from TOraDataSet)</td>
<td>Used to define the check mode before editing a record.</td>
</tr>
<tr>
<td>CommandTimeout (inherited from TOraDataSet)</td>
<td>Sets the wait time before a</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Conditions</strong></td>
<td>Used to add WHERE conditions to a query</td>
</tr>
<tr>
<td><strong>Connection</strong></td>
<td>Used to specify a connection object to use to connect to a data store.</td>
</tr>
<tr>
<td><strong>Cursor</strong></td>
<td>Used to fetch data from the cursor parameter and cursor field in Oracle 8.</td>
</tr>
<tr>
<td><strong>DataTypeMap</strong></td>
<td>Used to set data type mapping rules</td>
</tr>
<tr>
<td><strong>Debug</strong></td>
<td>Used to display the statement that is being executed and the values and types of its parameters.</td>
</tr>
<tr>
<td><strong>DependEvents</strong></td>
<td>Used to get or set the names of the events that dataset will depend on in TCustomSmartQuery.Smart Refresh mode.</td>
</tr>
<tr>
<td><strong>DetailFields</strong></td>
<td>Used to specify the fields that correspond to the foreign key fields from MasterFields when building master/detail relationship.</td>
</tr>
<tr>
<td><strong>Disconnected</strong></td>
<td>Used to keep dataset opened after connection is closed.</td>
</tr>
<tr>
<td><strong>DMLRefresh</strong></td>
<td>Used to refresh record by RETURNING clause when insert or update is performed.</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>Used to specify encryption options in a dataset.</td>
</tr>
<tr>
<td><strong>Expand</strong></td>
<td>Lets all data controls be aware of all the fields belonging to updating table.</td>
</tr>
<tr>
<td><strong>FetchRows</strong></td>
<td>Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>FilterSQL</strong></td>
<td>(inherited from TCustomDADataSet) Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td><strong>FinalSQL</strong></td>
<td>(inherited from TCustomDADataSet) Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong></td>
<td>(inherited from TMemDataSet) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>IsPLSQL</strong></td>
<td>(inherited from TOraDataSet) Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td><strong>IsQuery</strong></td>
<td>(inherited from TOraDataSet) Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong></td>
<td>(inherited from TMemDataSet) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>KeyFields</strong></td>
<td>(inherited from TOraDataSet) Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td><strong>KeySequence</strong></td>
<td>(inherited from TOraDataSet) Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong></td>
<td>(inherited from TMemDataSet) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong></td>
<td>(inherited from TMemDataSet) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>MacroCount</strong></td>
<td>(inherited from TCustomDADataSet) Used to get the number of macros associated with the Macros property.</td>
</tr>
<tr>
<td><strong>Macros</strong></td>
<td>(inherited from TCustomDADataSet) Makes it possible to change SQL queries easily.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>MasterFields</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>MasterSource</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Options</td>
<td>TCustomSmartQuery</td>
</tr>
<tr>
<td>OptionsDS</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>ParamCount</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Params</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Prepared</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Ranged</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>RefreshEvent</td>
<td>TCustomSmartQuery</td>
</tr>
<tr>
<td>RefreshMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RefreshOptions</td>
<td>(inherited from TCustomDADataSet) refresh an editing record.</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>(inherited from TOraDataSet) Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td>RowsAffected</td>
<td>(inherited from TCustomDADataSet) Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td>RowsProcessed</td>
<td>(inherited from TOraDataSet) Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td>SequenceMode</td>
<td>(inherited from TOraDataSet) Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td>Session</td>
<td>(inherited from TOraDataSet) Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td>SmartFetch</td>
<td>(inherited from TOraDataSet) The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td>SmartRefresh</td>
<td>(inherited from TCustomSmartQuery) Let TCustomSmartQuery components work in a concurrent environment.</td>
</tr>
<tr>
<td>SmartState</td>
<td>(inherited from TCustomSmartQuery) Defines if TCustomSmartQuery is in view mode.</td>
</tr>
<tr>
<td>SQL</td>
<td>(inherited from TCustomDADataSet) Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>(inherited from TCustomDADataSet) Used to specify a SQL statement that will be used when applying a deletion to a record.</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>(inherited from TCustomDADataSet) Used to specify the SQL statement that will be used when applying an insertion to a dataset.</td>
</tr>
<tr>
<td>SQLLock</td>
<td>(inherited from TCustomDADataSet) Used to specify a SQL statement that will be used to perform a record lock.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>Used to specify the SQL statement that is used to get the record count when opening a dataset.</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>Used to specify a SQL statement that will be used to refresh current record by calling the TCustomDADataset.RefreshRecord procedure.</td>
</tr>
<tr>
<td>SQLType</td>
<td>Used to get the typecode of the SQL statement being processed by Oracle database server.</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>Used to specify a SQL statement that will be used when applying an update to a dataset.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>Used for TOraDataSet to raise the 'Update failed' exception when the number of updated or deleted records are not equal to 1.</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>Used if an application does not need bidirectional access to records in the result set.</td>
</tr>
<tr>
<td>UpdateObject</td>
<td>Used to specify an update object component which provides SQL statements that perform updates of the read-only datasets.</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>UpdatesPending</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
<tr>
<td>FetchAll</td>
<td>Defines whether to request all records of the query from the database.</td>
</tr>
</tbody>
</table>
5.29.1.2.2.1 FetchAll Property

Defines whether to request all records of the query from database server when the dataset is being opened.

Class

T0raTable

Syntax

property FetchAll: boolean;

Remarks

When set to True, all records of the query are requested from database server when the dataset is being opened. When set to False, records are retrieved when a data-aware component or a program requests it. If a query can return a lot of records, set this property to False if initial response time is important.

When the FetchAll property is False, the first call to TMemDataSet.Locate and TMemDataSet.LocateEx methods may take a lot of time to retrieve additional records to the client side.
5.29.1.2.2.2 LockMode Property

Used to specify what kind of lock will be performed when editing a record.

Class

TOraTable

Syntax

(property LockMode: TLockMode default lmLockImmediate;)

Remarks

Use the LockMode property to define what kind of lock will be performed when editing a record. Locking a record is useful in creating multi-user applications. It prevents modification of a record by several users at the same time.

Locking is performed by the RefreshRecord method.

The default value is lmNone.

To set pessimistic locking use LockMode = lmLockImmediate, TораDataSet.CheckMode = cmException. To set optimistic locking use LockMode = lmLockDelayed, CheckMode = cmException.

See Also

- TOraStoredProc.LockMode
- TOraQuery.LockMode

5.29.1.2.2.3 OrderFields Property

Used to build ORDER BY clause of SQL statements.

Class

TOraTable

Syntax

(property OrderFields: string;)

© 1997-2021 Devart. All Rights Reserved.
Remarks

TOraTable uses the OrderFields property to build ORDER BY clause of SQL statements. To set several field names to this property separate them with commas.

TOraTable is reopened when OrderFields is being changed.

See Also
- TOraTable

5.29.1.2.2.4 TableName Property

Used to specify the name of the database table this component encapsulates.

Class

TOraTable

Syntax

```plaintext
property TableName: string;
```

Remarks

Use the TableName property to specify the name of the database table this component encapsulates. If TCustomDADataset.Connection is assigned If Session is set at design time, select a valid table name from the TableName drop-down list in Object Inspector.

See Also
- TOraTable

5.29.1.2.3 Methods

Methods of the TOraTable class.

For a complete list of the TOraTable class members, see the TOraTable Members topic.

Public
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddWhere</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td><strong>ApplyRange</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>BreakExec</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>CancelRange</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>CreateBlobStream</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td><strong>CreateProcCall</strong></td>
<td>(inherited from <code>TOraDataSet</code>) Generates the stored procedure call.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>DeleteWhere</strong></td>
<td>(inherited from <code>TCustomDADataSet</code>) Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>(inherited from <code>TMemDataSet</code>) Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>EmptyTable</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ErrorOffset</strong></td>
<td>(inherited from <code>TOraDataSet</code>) Returns the parse error offset.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Execute</strong></td>
<td>Overloaded. Executes a SQL statement on the server.</td>
</tr>
<tr>
<td><strong>Executing</strong></td>
<td>Indicates whether SQL statement is still being executed.</td>
</tr>
<tr>
<td><strong>Fetched</strong></td>
<td>Used to find out whether TCustomDADataSet has fetched all rows.</td>
</tr>
<tr>
<td><strong>Fetching</strong></td>
<td>Used to learn whether TCustomDADataSet is still fetching rows.</td>
</tr>
<tr>
<td><strong>FetchingAll</strong></td>
<td>Used to learn whether TCustomDADataSet is fetching all rows to the end.</td>
</tr>
<tr>
<td><strong>FindKey</strong></td>
<td>Searches for a record which contains specified field values.</td>
</tr>
<tr>
<td><strong>FindMacro</strong></td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>FindNearest</strong></td>
<td>Moves the cursor to a specific record or to the first record in the dataset that matches or is greater than the values specified in the KeyValues parameter.</td>
</tr>
<tr>
<td><strong>FindParam</strong></td>
<td>Determines whether a parameter with the specified name exists in a dataset.</td>
</tr>
<tr>
<td><strong>GetArray</strong></td>
<td>Retrieves a TOraArray object for a field when only its name is known.</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>GetDataType</strong></td>
<td>Returns internal field types defined in the MemData and accompanying modules.</td>
</tr>
<tr>
<td><strong>GetErrorPos</strong></td>
<td>Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td><strong>GetFieldObject</strong></td>
<td>Returns a multireference</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetFieldPrecision</td>
<td>Retrieves the precision of a number field.</td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>Retrieves the scale of a number field.</td>
</tr>
<tr>
<td>GetFile</td>
<td>Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td>GetInterval</td>
<td>Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetKeyList</td>
<td>Returns the list of table primary key fields.</td>
</tr>
<tr>
<td>GetLob</td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td>GetLobLocator</td>
<td>Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td>GetObject</td>
<td>Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td>GetRef</td>
<td>Retrieves a TOraRef object for a field with known name.</td>
</tr>
<tr>
<td>GetTable</td>
<td>Retrieve a TOraNestTable object for a field with known name.</td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td>Retrieves a TOraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td>Locate</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx</td>
<td>Overloaded. Excludes features that don't need to be included to the dataset.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Lock</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td><strong>MacroByName</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td><strong>OpenNext</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td><strong>ParamByName</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td><strong>Prepare</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td><strong>PrepareSQL</strong></td>
<td>Determines KeyFields and builds query of TOraData.</td>
</tr>
<tr>
<td><strong>RefreshRecord</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td><strong>RestoreSQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td><strong>RevertRecord</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>SaveSQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td><strong>SaveToXML</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><strong>SetOrderBy</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
<tr>
<td><strong>SetRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><strong>SetRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
</tbody>
</table>

© 2021 Devart
<table>
<thead>
<tr>
<th>Method</th>
<th>(inherited from)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetRangeStart</td>
<td><strong>TMemDataSet</strong></td>
<td>Indicates that subsequent assignments to field values specify the start of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SQLSaved</td>
<td><strong>TCustomDADataSet</strong></td>
<td>Determines if the SQL property value was saved to the BaseSQL property.</td>
</tr>
<tr>
<td>UnLock</td>
<td><strong>TCustomDADataSet</strong></td>
<td>Releases a record lock.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td><strong>TMemDataSet</strong></td>
<td>Frees the resources allocated for a previously prepared query on the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>server and client sides.</td>
</tr>
<tr>
<td>UpdateResult</td>
<td><strong>TMemDataSet</strong></td>
<td>Reads the status of the latest call to the ApplyUpdates method while</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus</td>
<td><strong>TMemDataSet</strong></td>
<td>Indicates the current update status for the dataset when cached updates are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enabled.</td>
</tr>
<tr>
<td>View</td>
<td><strong>TCustomSmartQuery</strong></td>
<td>Allows viewing all fields of the updating table.</td>
</tr>
</tbody>
</table>

See Also
- **TOraTable Class**
- **TOraTable Class Members**

Truncates the current table content on the server.

Class
**TOraTable**

Syntax
```pascal
procedure EmptyTable;
```
Remarks
Call the EmptyTable method to truncate the current table content on the server.

5.29.1.2.3.2 PrepareSQL Method

Determines KeyFields and builds query of TOraTable.

Class
TOraTable

Syntax
procedure PrepareSQL;

Remarks
Call the PrepareSQL method to determine KeyFields and build query of TOraTable. PrepareSQL is called implicitly when TOraTable is being opened.

5.29.1.3 TSmartQuery Class

A component providing TCustomSmartQuery.Expand fields feature, that lets all data controls be aware of all the fields belonging to updating table an not only those requested in SELECT clause, and TCustomSmartQuery.SmartRefresh feature (in Professional and Developer editions only).

For a list of all members of this type, see TSmartQuery members.

Unit
OraSmart

Syntax
TSmartQuery = class(TCustomSmartQuery);

Remarks
TSmartQuery component is a direct descendant of the TOraDataSet class.

TSmartQuery is an alternative to TOraQuery. It provides TCustomSmartQuery.Expand fields feature, that lets all data controls be aware of all the fields belonging to updating table an not only those requested in SELECT clause, and TCustomSmartQuery.SmartRefresh feature (in Professional and Developer editions only).

Inheritance Hierarchy

TMemDataSet
  TCustomDADataSet
    TOraDataSet
      TCustomOraQuery
        TCustomSmartQuery
          TSmartQuery

See Also
- TOraQuery
- TOraTable
- Updating Data with ODAC Dataset Components

© 1997-2021 Devart. All Rights Reserved.

Request Support DAC Forum Provide Feedback

5.29.1.3.1 Members

TSmartQuery class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseSQL (inherited from TCustomDADataSet)</td>
<td>Used to return SQL text without any changes performed by AddWhere, SetOrderBy, and FilterSQL.</td>
</tr>
<tr>
<td>CachedUpdates (inherited from TMemDataSet)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>ChangeNotification (inherited from TOraDataSet)</td>
<td>Used to receive database change notification messages to refresh dataset when required.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>CheckMode</td>
<td>TORADataSet</td>
</tr>
<tr>
<td>CommandTimeout</td>
<td>TORADataSet</td>
</tr>
<tr>
<td>Conditions</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Connection</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Cursor</td>
<td>TORADataSet</td>
</tr>
<tr>
<td>DataTypeMap</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Debug</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>DependEvents</td>
<td>TCustomSmartQuery</td>
</tr>
<tr>
<td>DetailFields</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Disconnected</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>DMLRefresh</td>
<td>TORADataSet</td>
</tr>
<tr>
<td>Encryption</td>
<td>TORADataSet</td>
</tr>
<tr>
<td>Expand</td>
<td>TCustomSmartQuery</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FetchAll</td>
<td>Used to request all records of the query from database server when a dataset is being opened.</td>
</tr>
<tr>
<td>FetchRows</td>
<td>Used to define the number of rows to be transferred across the network at the same time.</td>
</tr>
<tr>
<td>FilterSQL</td>
<td>Used to change the WHERE clause of SELECT statement and reopen a query.</td>
</tr>
<tr>
<td>FinalSQL</td>
<td>Used to return SQL text with all changes performed by AddWhere, SetOrderBy, and FilterSQL, and with expanded macros.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>IsPLSQL</td>
<td>Indicates whether a SQL statement is a PL/SQL block.</td>
</tr>
<tr>
<td>IsQuery</td>
<td>Indicates whether SQL statement returns rows or not.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>KeyFields</td>
<td>Used to build SQL statements for the SQLDelete, SQLInsert, and SQLUpdate properties if they were empty before updating a database.</td>
</tr>
<tr>
<td>KeySequence</td>
<td>Used to specify the name of a sequence that will be used to fill in a key field after a new record is inserted or posted to a database.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>Property</td>
<td>Inherited From</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>LockMode</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>MacroCount</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Macros</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>MasterFields</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>MasterSource</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>NonBlocking</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Options</td>
<td>TCustomSmartQuery</td>
</tr>
<tr>
<td>OptionsDS</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>ParamCheck</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>ParamCount</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Params</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Prepared</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>ReadOnly</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to prevent users from updating, inserting, or deleting data in the dataset.</td>
</tr>
<tr>
<td><strong>RefreshEvent</strong> (inherited from <strong>TCustomSmartQuery</strong>)</td>
<td>Used to get or set the name of the event that dataset will be waiting for in <strong>TCustomSmartQuery.SmartRefresh</strong> mode.</td>
</tr>
<tr>
<td><strong>RefreshMode</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify when to refresh an editing record.</td>
</tr>
<tr>
<td><strong>RefreshOptions</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to indicate when the editing record is refreshed.</td>
</tr>
<tr>
<td><strong>ReturnParams</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to return a new fields value to dataset after insert or update.</td>
</tr>
<tr>
<td><strong>RowsAffected</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to indicate the number of rows which were inserted, updated, or deleted during the last query operation.</td>
</tr>
<tr>
<td><strong>RowsProcessed</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Returns the number of rows processed by a query.</td>
</tr>
<tr>
<td><strong>SequenceMode</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the methods used internally to generate a sequenced field.</td>
</tr>
<tr>
<td><strong>Session</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>Used to specify the session in which dataset will be executed.</td>
</tr>
<tr>
<td><strong>SmartFetch</strong> (inherited from <strong>TOraDataSet</strong>)</td>
<td>The SmartFetch mode is used for fast navigation through a huge amount of records and to minimize memory consumption.</td>
</tr>
<tr>
<td><strong>SmartRefresh</strong> (inherited from <strong>TCustomSmartQuery</strong>)</td>
<td>Let TCustomSmartQuery components work in a concurrent environment.</td>
</tr>
<tr>
<td><strong>SmartState</strong> (inherited from <strong>TCustomSmartQuery</strong>)</td>
<td>Defines if TCustomSmartQuery is in view mode.</td>
</tr>
<tr>
<td><strong>SQL</strong> (inherited from <strong>TCustomDADataset</strong>)</td>
<td>Used to provide a SQL statement that a query component executes when its Open method is called.</td>
</tr>
<tr>
<td>Method</td>
<td>Inherited From</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>SQLDelete</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>SQLInsert</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>SQLLock</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>SQLRecCount</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>SQLRefresh</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>SQLType</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>SQLUpdate</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>UniDirectional</td>
<td>TCustomDADataSet</td>
</tr>
<tr>
<td>UpdateObject</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>UpdateRecordTypes</td>
<td>TMemDataSet</td>
</tr>
</tbody>
</table>
### UpdatesPending (inherited from TMemDataSet)

Used to check the status of the cached updates buffer.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddWhere (inherited from TCustomDADataset)</td>
<td>Adds condition to the WHERE clause of SELECT statement in the SQL property.</td>
</tr>
<tr>
<td>ApplyRange (inherited from TMemDataSet)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates (inherited from TMemDataSet)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>BreakExec (inherited from TCustomDADataset)</td>
<td>Breaks execution of a SQL statement on the server.</td>
</tr>
<tr>
<td>CancelRange (inherited from TMemDataSet)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates (inherited from TMemDataSet)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates (inherited from TMemDataSet)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>CreateBlobStream (inherited from TCustomDADataset)</td>
<td>Used to obtain a stream for reading data from or writing data to a BLOB field, specified by the Field parameter.</td>
</tr>
<tr>
<td>CreateProcCall (inherited from TOraDataSet)</td>
<td>Generates the stored procedure call.</td>
</tr>
<tr>
<td>DeferredPost (inherited from TMemDataSet)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>DeleteWhere (inherited from TCustomDADataset)</td>
<td>Removes WHERE clause from the SQL property and assigns the BaseSQL property.</td>
</tr>
<tr>
<td>EditRangeEnd (inherited from TMemDataSet)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>Function</td>
<td>Inherited From</td>
</tr>
<tr>
<td>---------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>ErrorOffset</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>Execute</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Executing</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Fetched</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Fetching</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>FetchingAll</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>FindKey</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>FindMacro</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>FindNearest</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>FindParam</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetArray</td>
<td>TOraDataSet</td>
</tr>
<tr>
<td>GetBlob</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>GetDataType</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetErrorPos</td>
<td>(inherited from <code>TOraDataSet</code>) Returns a row and column of parse error for a SQL statement.</td>
</tr>
<tr>
<td>GetFieldObject</td>
<td>(inherited from <code>TCustomDADataSet</code>) Returns a multireference shared object from field.</td>
</tr>
<tr>
<td>GetFieldPrecision</td>
<td>(inherited from <code>TCustomDADataSet</code>) Retrieves the precision of a number field.</td>
</tr>
<tr>
<td>GetFieldScale</td>
<td>(inherited from <code>TCustomDADataSet</code>) Retrieves the scale of a number field.</td>
</tr>
<tr>
<td>GetFile</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraFile object for a field with known name.</td>
</tr>
<tr>
<td>GetInterval</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraInterval object for a field with known name.</td>
</tr>
<tr>
<td>GetKeyFieldNames</td>
<td>(inherited from <code>TCustomDADataSet</code>) Provides a list of available key field names.</td>
</tr>
<tr>
<td>GetKeyList</td>
<td>(inherited from <code>TOraDataSet</code>) Returns the list of table primary key fields.</td>
</tr>
<tr>
<td>GetLob</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td>GetLobLocator</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraLob object for a field with known name.</td>
</tr>
<tr>
<td>GetObject</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraObject object for a field with known name.</td>
</tr>
<tr>
<td>GetOrderBy</td>
<td>(inherited from <code>TCustomDADataSet</code>) Retrieves an ORDER BY clause from a SQL statement.</td>
</tr>
<tr>
<td>GetRef</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraRef object for a field with known name.</td>
</tr>
<tr>
<td>GetTable</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieve a TOraNestTable object for a field with known name.</td>
</tr>
<tr>
<td>GetTimeStamp</td>
<td>(inherited from <code>TOraDataSet</code>) Retrieves a TOraTimeStamp object for a field with known name.</td>
</tr>
<tr>
<td>GotoCurrent</td>
<td>(inherited from <code>TCustomDADataSet</code>) Sets the current record in this dataset similar to the current record in another dataset.</td>
</tr>
<tr>
<td>Locate</td>
<td>(inherited from <code>TMemDataSet</code>) Overloaded. Searches a dataset for a specific record.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LocateEx (inherited from TMemDataSet)</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Lock (inherited from TCustomDADataset)</td>
<td>Locks the current record.</td>
</tr>
<tr>
<td>MacroByName (inherited from TCustomDADataset)</td>
<td>Finds a macro with the specified name.</td>
</tr>
<tr>
<td>OpenNext (inherited from TOraDataSet)</td>
<td>Opens next cursor or rowset in the statement.</td>
</tr>
<tr>
<td>ParamByName (inherited from TOraDataSet)</td>
<td>Sets or uses parameter information for a specific parameter based on its name.</td>
</tr>
<tr>
<td>Prepare (inherited from TCustomDADataset)</td>
<td>Allocates, opens, and parses cursor for a query.</td>
</tr>
<tr>
<td>RefreshRecord (inherited from TCustomDADataset)</td>
<td>Actualizes field values for the current record.</td>
</tr>
<tr>
<td>RestoreSQL (inherited from TCustomDADataset)</td>
<td>Restores the SQL property modified by AddWhere and SetOrderBy.</td>
</tr>
<tr>
<td>RestoreUpdates (inherited from TMemDataSet)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord (inherited from TMemDataSet)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveSQL (inherited from TCustomDADataset)</td>
<td>Saves the SQL property value to BaseSQL.</td>
</tr>
<tr>
<td>SaveToXML (inherited from TMemDataSet)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetOrderBy (inherited from TCustomDADataset)</td>
<td>Builds an ORDER BY clause of a SELECT statement.</td>
</tr>
<tr>
<td>SetRange (inherited from TMemDataSet)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td>SetRangeEnd (inherited from TMemDataSet)</td>
<td>Indicates that subsequent assignments to field values and positions the cursor on it.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>SetRangeStart</td>
<td>(inherited from TMemDataSet) Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SQLSaved</td>
<td>(inherited from TCustomDADataset) Determines if the SQL property value was saved to the BaseSQL property.</td>
</tr>
<tr>
<td>UnLock</td>
<td>(inherited from TCustomDADataset) Releases a record lock.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>(inherited from TMemDataSet) Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td>UpdateResult</td>
<td>(inherited from TMemDataSet) Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td>UpdateStatus</td>
<td>(inherited from TMemDataSet) Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
<tr>
<td>View</td>
<td>(inherited from TCustomSmartQuery) Allows viewing all fields of the updating table.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AfterExecute</td>
<td>(inherited from TCustomDADataset) Occurs after a component has executed a query to database.</td>
</tr>
<tr>
<td>AfterFetch</td>
<td>(inherited from TCustomDADataset) Occurs after dataset finishes fetching data from server.</td>
</tr>
<tr>
<td>AfterSmartRefresh</td>
<td>(inherited from TCustomSmartQuery) Occurs after the Smart Refresh procedure was performed by TCustomOraQuery.</td>
</tr>
<tr>
<td>AfterUpdateExecute</td>
<td>(inherited from TCustomDADataset) Occurs after executing insert, delete, update, lock and refresh operations.</td>
</tr>
<tr>
<td>Event Name</td>
<td>Inherited From</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------------</td>
</tr>
<tr>
<td>BeforeFetch</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>BeforeUpdateExecute</td>
<td>TCustomDADataset</td>
</tr>
<tr>
<td>OnUpdateError</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>OnUpdateRecord</td>
<td>TMemDataSet</td>
</tr>
<tr>
<td>RefreshFields</td>
<td>TCustomSmartQuery</td>
</tr>
</tbody>
</table>

### 5.29.1.4 TSmartQueryOptions Class

This class allows setting up the behaviour of the TSmartQuery class.

For a list of all members of this type, see TSmartQueryOptions members.

**Unit**

oraSmart

**Syntax**

```plaintext
TSmartQueryOptions = class(TOraDataSetOptions);
```

**Inheritance Hierarchy**

TDADatasetOptions

TOraDataSetOptionsDS

TOraDataSetOptions

TSmartQueryOptions
**TSmartQueryOptions** class overview.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoClose</td>
<td>(inherited from TOraDataSetOptions) Used to close OCI cursor after fetching all rows.</td>
</tr>
<tr>
<td>AutoPrepare</td>
<td>(inherited from TDADatasetOptions) Used to execute automatic TCustomDADataset.Prepare on the query execution.</td>
</tr>
<tr>
<td>CacheCalcFields</td>
<td>(inherited from TDADatasetOptions) Used to enable caching of the TField.Calculated and TField.Lookup fields.</td>
</tr>
<tr>
<td>CacheLobs</td>
<td>(inherited from TOraDataSetOptions) Used to allocate local memory buffer to hold a copy of the Lob content.</td>
</tr>
<tr>
<td>CompressBlobMode</td>
<td>(inherited from TDADatasetOptions) Used to store values of the BLOB fields in compressed form.</td>
</tr>
<tr>
<td>DefaultValue</td>
<td>(inherited from TOraDataSetOptions) Used for TOraDataSet to fill the DefaultExpression property of TField objects by appropriate value.</td>
</tr>
<tr>
<td>DeferredLobRead</td>
<td>(inherited from TOraDataSetOptions) Used to fetch all Oracle 8 Lob values when they are explicitly requested.</td>
</tr>
<tr>
<td>DetailDelay</td>
<td>(inherited from TDADatasetOptions) Used to get or set a delay in milliseconds before refreshing detail dataset while navigating master dataset.</td>
</tr>
<tr>
<td>EnableBCD</td>
<td>(inherited from TOraDataSetOptions) Used to enable currency type. Default value of this option is False.</td>
</tr>
<tr>
<td>EnableFMTBCD</td>
<td>(inherited from TOraDataSetOptions) Used to enable using FMTBCD instead of float for large integer numbers to keep precision.</td>
</tr>
<tr>
<td>ExtendedFieldsInfo</td>
<td>(inherited from TOraDataSetOptions) Used to perform an additional query to get information about returned fields and the tables they belong to.</td>
</tr>
<tr>
<td>Field</td>
<td>Inherited From</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>FieldsAsString</strong> (inherited from <strong>TOraDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>FieldsOrigin</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>FlatBuffers</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>FullRefresh</strong> (inherited from <strong>TOraDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>HideRowId</strong> (inherited from <strong>TOraDataSetOptionsDS</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>InsertAllSetFields</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>KeepPrepared</strong> (inherited from <strong>TOraDataSetOptionsDS</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>LocalMasterDetail</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>LongStrings</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>MasterFieldsNullable</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td><strong>NumberRange</strong> (inherited from <strong>TDADeDataSetOptions</strong>)</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>PrefetchLobSize</strong></td>
<td>(inherited from <strong>TOraDataSetOptions</strong>) Used to retrieve the LOB length and the LOB data beginning during regular fetch.</td>
</tr>
<tr>
<td><strong>PrefetchRows</strong></td>
<td>(inherited from <strong>TOraDataSetOptions</strong>) Used to get or set the number of rows that OCI prefetched when executing a query.</td>
</tr>
<tr>
<td><strong>PrepareUpdateSQL</strong></td>
<td>(inherited from <strong>TOraDataSetOptions</strong>) Used to automatically prepare update queries before execution.</td>
</tr>
<tr>
<td><strong>ProcNamedParams</strong></td>
<td>(inherited from <strong>TOraDataSetOptions</strong>) Used to specify a notation method of passing parameter values to the stored PL/SQL object.</td>
</tr>
<tr>
<td><strong>QueryRecCount</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for <strong>TCustomDADataset</strong> to perform additional query to get the record count for this SELECT, so the RecordCount property reflects the actual number of records.</td>
</tr>
<tr>
<td><strong>QuoteNames</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for <strong>TCustomDADataset</strong> to quote all database object names in autogenerated SQL statements such as update SQL.</td>
</tr>
<tr>
<td><strong>RawAsString</strong></td>
<td>(inherited from <strong>TOraDataSetOptions</strong>) Used to treat all RAW fields as being of string datatype.</td>
</tr>
<tr>
<td><strong>ReflectChangeNotify</strong></td>
<td>(inherited from <strong>TOraDataSetOptions</strong>) Used for a dataset component to refresh its data when it gets database change notification messages in response to DML or DDL changes on the objects associated with the dataset query.</td>
</tr>
<tr>
<td><strong>RemoveOnRefresh</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for a dataset to locally remove a record that can not be found on the server.</td>
</tr>
<tr>
<td><strong>RequiredFields</strong></td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for <strong>TCustomDADataset</strong> to set</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReturnParams</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to return the new value of fields to dataset after insert or update.</td>
</tr>
<tr>
<td>ScrollableCursor</td>
<td>(inherited from <strong>TOradataOptions</strong>) Used for TOraDataSet to use scrollable server cursor (available since Oracle 9 only) instead of caching data on the client side.</td>
</tr>
<tr>
<td>SetFieldsReadOnly</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for a dataset to set the ReadOnly property to True for all fields that do not belong to UpdatingTable or can not be updated.</td>
</tr>
<tr>
<td>StatementCache</td>
<td>(inherited from <strong>TOradataOptions</strong>) Used to get a value indicating whether Oracle resources associated with the current statement will be cached inside a session.</td>
</tr>
<tr>
<td>StrictUpdate</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used for TCustomDADataset to raise an exception when the number of updated or deleted records is not equal 1.</td>
</tr>
<tr>
<td>TemporaryLobUpdate</td>
<td>(inherited from <strong>TOradataOptions</strong>) Temporary LOBs are used to write input and input/output LOB parameters into database when executing dataset's SQL statements.</td>
</tr>
<tr>
<td>TrimFixedChar</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Specifies whether to discard all trailing spaces in the string fields of a dataset.</td>
</tr>
<tr>
<td>UpdateAllFields</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to include all dataset fields in the generated UPDATE and INSERT statements.</td>
</tr>
<tr>
<td>UpdateBatchSize</td>
<td>(inherited from <strong>TDADatasetOptions</strong>) Used to get or set a value that enables or disables batch processing support, and specifies the number of commands that can be executed in a batch.</td>
</tr>
</tbody>
</table>
5.29.2 Enumerations

Enumerations in the **OraSmart** unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TSmartState</strong></td>
<td>Specifies if TCustomSmartQuery is in view mode.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

5.29.2.1 **TSmartState** Enumeration

Specifies if TCustomSmartQuery is in view mode.

**Unit**

**OraSmart**

**Syntax**

```
TSmartState = (dsView);
```

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>dsView</strong></td>
<td>TCustomSmartQuery is in view mode.</td>
</tr>
</tbody>
</table>

**Remarks**

Check the SmartState property to learn whether TCustomSmartQuery is in view mode.

TCustomSmartQuery is in view mode (SmartState is dsView) after calling View method.
5.30 **OraSQLMonitor**

This unit contains implementation of the TOraSQLMonitor component.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOraSQLMonitor</strong></td>
<td>This component serves for monitoring dynamic SQL execution in ODAC-based applications.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

**5.30.1 Classes**

Classes in the **OraSQLMonitor** unit.

**Classes**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TOraSQLMonitor</strong></td>
<td>This component serves for monitoring dynamic SQL execution in ODAC-based applications.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

**5.30.1.1 TOraSQLMonitor Class**

This component serves for monitoring dynamic SQL execution in ODAC-based applications.

For a list of all members of this type, see **TOraSQLMonitor** members.

**Unit**

**OraSQLMonitor**

**Syntax**

```
TOraSQLMonitor = class(TCustomDASQLMonitor);
```
Remarks

Use TOraSQLMonitor to monitor dynamic SQL execution in ODAC-based applications. TOraSQLMonitor provides two ways of displaying debug information: with dialog window, DBMonitor or Borland SQL Monitor. Furthermore to receive debug information the TCustomDASQLMonitor.OnSQL event can be used. Also it is possible to use all these ways at the same time, though an application may have only one TOraSQLMonitor object. If an application has no TOraSQLMonitor instance, the Debug window is available to display SQL statements to be sent.

Inheritance Hierarchy

TCustomDASQLMonitor
   TOraSQLMonitor

See Also

- TCustomDADataset.Debug
- TCustomDASQL.Debug
- DBMonitor

© 1997-2021 Devart. All Rights Reserved.

Members

**TOraSQLMonitor class overview.**

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong> (inherited from TCustomDASQLMonitor)</td>
<td>Used to activate monitoring of SQL.</td>
</tr>
<tr>
<td><strong>DBMonitorOptions</strong> (inherited from TCustomDASQLMonitor)</td>
<td>Used to set options for dbMonitor.</td>
</tr>
<tr>
<td><strong>Options</strong> (inherited from TCustomDASQLMonitor)</td>
<td>Used to include the desired properties for TCustomDASQLMonitor.</td>
</tr>
<tr>
<td><strong>TraceFlags</strong> (inherited from TCustomDASQLMonitor)</td>
<td>Used to specify which database operations the monitor should track in an application at runtime.</td>
</tr>
</tbody>
</table>
Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnSQL (inherited from TCustomDASQLMonitor)</td>
<td>Occurs when tracing of SQL activity on database components is needed.</td>
</tr>
</tbody>
</table>

5.31 OraTransaction

This unit contains implementation of the TOraTransaction component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraTransaction</td>
<td>A component for managing transactions in an application.</td>
</tr>
</tbody>
</table>

Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TGlobalCoordinator</td>
<td>Specifies with what distributed transaction, perform two-phase commit or rollback on all sessions will be coordinated.</td>
</tr>
</tbody>
</table>

5.31.1 Classes

Classes in the OraTransaction unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOraTransaction</td>
<td>A component for managing transactions in an application.</td>
</tr>
</tbody>
</table>
5.31.1.1 TOraTransaction Class

A component for managing transactions in an application.

For a list of all members of this type, see TOraTransaction members.

Unit

OraTransaction

Syntax

TOraTransaction = class(TDATransaction);

Remarks

The TOraTransaction component is used to provide discrete transaction control over connection. It can be used for manipulating simple local and global transactions.

Inheritance Hierarchy

TDATransaction

TOraTransaction

See Also

- TOraTransaction Component
- TCustomDACConnection.StartTransaction
- TCustomDACConnection.Commit
- TCustomDACConnection.Rollback
- TOraTransaction Component

© 1997-2021 Devart. All Rights Reserved.
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Used to indicate whether the transaction is active or not.</td>
</tr>
<tr>
<td>BranchQualifiers</td>
<td>Used to represent branch qualifier part of XID for transaction branches.</td>
</tr>
<tr>
<td>DefaultCloseAction</td>
<td>(inherited from <strong>TDATransaction</strong>) Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.</td>
</tr>
<tr>
<td>DefaultSession</td>
<td>Used to specify the session for performing the transaction.</td>
</tr>
<tr>
<td>GlobalCoordinator</td>
<td>Used to determine with what distributed transaction, perform two-phase commit or rollback on all sessions will be coordinated.</td>
</tr>
<tr>
<td>InactiveTimeOut</td>
<td>Used to specify the waiting time before deleting inactive transaction branch.</td>
</tr>
<tr>
<td>IsolationLevel</td>
<td>Used to specify how the transactions containing database modifications are handled.</td>
</tr>
<tr>
<td>ResumeTimeOut</td>
<td>Used to set the wait time to resume the transaction branch if it is used by another session.</td>
</tr>
<tr>
<td>Sessions</td>
<td>Used to specify a session for the given index.</td>
</tr>
<tr>
<td>SessionsCount</td>
<td>Used to provide the number of sessions associated with the transaction component.</td>
</tr>
<tr>
<td>TransactionId</td>
<td>Used to represent global transaction identifier.</td>
</tr>
<tr>
<td>TransactionName</td>
<td>Used to assign a name to the current transaction.</td>
</tr>
</tbody>
</table>
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AddSession</strong></td>
<td>Overloaded. Associates a TOraSession component with the transaction component.</td>
</tr>
<tr>
<td><strong>ClearSessions</strong></td>
<td>Disassociates the transaction component from all its session components.</td>
</tr>
<tr>
<td><strong>Commit</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Commits the current transaction.</td>
</tr>
<tr>
<td><strong>Detach</strong></td>
<td>Deactivates a transaction.</td>
</tr>
<tr>
<td><strong>RemoveSession</strong></td>
<td>Disassociates the specified session from the transaction.</td>
</tr>
<tr>
<td><strong>Resume</strong></td>
<td>Resumes a detached transaction.</td>
</tr>
<tr>
<td><strong>Rollback</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Discards all modifications of data associated with the current transaction and ends the transaction.</td>
</tr>
<tr>
<td><strong>RollbackToSavepoint</strong></td>
<td>Discards all modifications made during the current transaction and restores its state to the moment of the savepoint.</td>
</tr>
<tr>
<td><strong>Savepoint</strong></td>
<td>Defines a savepoint in the transaction.</td>
</tr>
<tr>
<td><strong>StartTransaction</strong></td>
<td>Overloaded. Begins a new user transaction against the database server.</td>
</tr>
</tbody>
</table>

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnCommit</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Occurs after the transaction has been successfully committed.</td>
</tr>
<tr>
<td><strong>OnCommitRetaining</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Occurs after CommitRetaining has been executed.</td>
</tr>
<tr>
<td><strong>OnError</strong></td>
<td>Occurs for processing errors.</td>
</tr>
</tbody>
</table>
that can be arised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnRollback</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Occurs after the transaction has been successfully rolled back.</td>
</tr>
<tr>
<td><strong>OnRollbackRetaining</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Occurs after RollbackRetaining has been executed.</td>
</tr>
</tbody>
</table>

### 5.31.1.2 Properties

Properties of the **TOraTransaction** class.

For a complete list of the **TOraTransaction** class members, see the **TOraTransaction Members** topic.

#### Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Active</strong></td>
<td>Used to indicate whether the transaction is active or not.</td>
</tr>
<tr>
<td><strong>BranchQualifiers</strong></td>
<td>Used to represent branch qualifier part of XID for transaction branches.</td>
</tr>
<tr>
<td><strong>DefaultCloseAction</strong> (inherited from <strong>TDATransaction</strong>)</td>
<td>Used to specify the transaction behaviour when it is destroyed while being active, or when one of its connections is closed with the active transaction.</td>
</tr>
<tr>
<td><strong>Sessions</strong></td>
<td>Used to specify a session for the given index.</td>
</tr>
<tr>
<td><strong>SessionsCount</strong></td>
<td>Used to provide the number of sessions associated with the transaction component.</td>
</tr>
<tr>
<td><strong>TransactionId</strong></td>
<td>Used to represent global transaction id.</td>
</tr>
</tbody>
</table>
Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultSession</td>
<td>Used to specify the session for performing the transaction.</td>
</tr>
<tr>
<td>GlobalCoordinator</td>
<td>Used to determine with what distributed transaction, perform two-phase commit or rollback on all sessions will be coordinated.</td>
</tr>
<tr>
<td>InactiveTimeOut</td>
<td>Used to specify the waiting time before deleting inactive transaction branch.</td>
</tr>
<tr>
<td>IsolationLevel</td>
<td>Used to specify how the transactions containing database modifications are handled.</td>
</tr>
<tr>
<td>ResumeTimeOut</td>
<td>Used to set the wait time to resume the transaction branch if it is used by another session.</td>
</tr>
<tr>
<td>TransactionName</td>
<td>Used to assign a name to the current transaction.</td>
</tr>
</tbody>
</table>

See Also
- **TOraTransaction Class**
- **TOraTransaction Class Members**

Used to indicate whether the transaction is active or not.

Class
- **TOraTransaction**

Syntax
### Property `Active` (boolean)

Remarks

Use the `Active` property to indicate whether the transaction is active or not.

---

### `BranchQualifiers` Property

**Used to represent branch qualifier part of XID for transaction branches.**

**Class**

`TOraTransaction`

**Syntax**

```plaintext
property BranchQualifiers[Index: integer]: TBytes;
```

**Parameters**

- **Index**
  
  Holds a branch qualifier index.

**Remarks**

Use the `BranchQualifiers` property to represent a branch qualifier part of XID for transaction branches.

---

### DefaultSession Property

**Used to specify the session for performing the transaction.**

**Class**

`TOraTransaction`

**Syntax**

```plaintext
property DefaultSession: TOraSession;
```

**Remarks**
Use the DefaultSession property to specify the session which is used to perform the transaction. For distributed transactions use the \texttt{TOraTransaction.AddSession} method instead.

See Also

- \texttt{TOraTransaction.AddSession}

Class

\texttt{TOraTransaction}

Syntax

\begin{verbatim}
property GlobalCoordinator: \texttt{TGlobalCoordinator} default gcInternal;
\end{verbatim}

Remarks

Use the GlobalCoordinator property to determine with what distributed transaction, perform two-phase commit or rollback on all sessions will be coordinated.

Class

\texttt{TOraTransaction}

Syntax

\begin{verbatim}
property InactiveTimeOut: integer default 0;
\end{verbatim}

Used to specify the waiting time before deleting inactive transaction branch.
Remarks

Use the InactiveTimeOut property to set for server the time to wait before deleting inactive transaction branch.

Syntax

```
property IsolationLevel: T0raIsolationLevel default ilReadCommitted;
```

Remarks

Use the IsolationLevel property to specify how the transactions containing database modifications are handled.

Class

T0raTransaction

Syntax

```
property ResumeTimeOut: integer default 0;
```

Remarks

Use the ResumeTimeOut property for setting the wait time to resume the transaction branch if it is used by another session.
5.31.1.1.2.8 Sessions Property (Indexer)

Used to specify a session for the given index.

Class

TOraTransaction

Syntax

```property
Sessions[Index: integer]: TOraSession;
```

Parameters

**Index**

Holds the index to specify a session for.

Remarks

Use the Sessions property to specify a session for the given index.

See Also

- SessionsCount
- RemoveSession
- TOraTransaction.AddSession

5.31.1.1.2.9 SessionsCount Property

Used to provide the number of sessions associated with the transaction component.

Class

TOraTransaction

Syntax
property SessionsCount: integer;

Remarks
Use the SessionsCount property to get the number of sessions associated with the transaction component.

See Also
• Sessions
• RemoveSession
• TOraTransaction.AddSession

5.31.1.2.10 TransactionId Property

Used to represent global transaction identifier.

Class
TOraTransaction

Syntax
property TransactionId: TBytes;

Remarks
Use the TransactionId property to represent global transaction identifier which is the part of XID. Server associates it with local transaction.
**property** TransactionName: *string*;

Remarks

Use the TransactionName property to assign a name to the current transaction. This parameter is useful in distributed database environments when you have to identify and resolve in-doubt transactions. The text string is limited to 255 bytes.

© 1997-2021 Devart. All Rights Reserved.

5.31.1.3 Methods

Methods of the **TOraTransaction** class.

For a complete list of the **TOraTransaction** class members, see the [TOraTransaction Members](#) topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSession</td>
<td>Overloaded. Associates a TOraSession component with the transaction component.</td>
</tr>
<tr>
<td>ClearSessions</td>
<td>Disassociates the transaction component from all its session components.</td>
</tr>
<tr>
<td>Commit (inherited from <strong>TDATransaction</strong>)</td>
<td>Commits the current transaction.</td>
</tr>
<tr>
<td>Detach</td>
<td>Deactivates a transaction.</td>
</tr>
<tr>
<td>RemoveSession</td>
<td>Disassociates the specified session from the transaction.</td>
</tr>
<tr>
<td>Resume</td>
<td>Resumes a detached transaction.</td>
</tr>
<tr>
<td>Rollback (inherited from <strong>TDATransaction</strong>)</td>
<td>Discards all modifications of data associated with the current transaction and ends the transaction.</td>
</tr>
<tr>
<td>RollbackToSavepoint</td>
<td>Discards all modifications made during the current transaction and restores its state to the moment of the savepoint.</td>
</tr>
</tbody>
</table>
Savepoint

Defines a savepoint in the transaction.

StartTransaction

Overloaded. Begins a new user transaction against the database server.

See Also

- **TOraTransaction Class**
- **TOraTransaction Class Members**

### 5.31.1.3.1 AddSession Method

Associates a TOraSession component with the transaction component.

**Class**

**TOraTransaction**

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddSession(Session: TOraSession)</td>
<td>Associates a TOraSession component with the transaction component.</td>
</tr>
<tr>
<td>AddSession(Session: TOraSession; BranchQualifier: TBytes)</td>
<td>Associates a TOraSession component with the transaction component.</td>
</tr>
</tbody>
</table>

Associates a TOraSession component with the transaction component.

**Class**

**TOraTransaction**

**Syntax**

```pascal
procedure AddSession(Session: TOraSession; overload; 
```

**Parameters**
Session
   Holds a TOraSession component

Remarks
Call the AddSession method to associate a TOraSession component with the transaction component.

Associates a TOraSession component with the transaction component.

Class
TOraTransaction

Syntax

```
procedure AddSession(Session: TOraSession; BranchQualifier: TBytes); overload;
```

Parameters

**Session**
   Holds a TOraSession component

**BranchQualifier**
   Holds a branch qualifier.

See Also
- TOraTransaction.Sessions
- TOraTransaction.RemoveSession
- TOraTransaction.ClearSessions

Disassociates the transaction component from all its session components.

Class
TOraTransaction
### 5.31.1.3.3 Detach Method

Deactivates a transaction.

**Class**

**T0raTransaction**

**Syntax**

```plaintext
procedure Detach;
```

**Remarks**

Call the Detach method to make a transaction inactive.

**See Also**

- [Resume](#)

© 1997-2021 Devart. All Rights Reserved.
TOraTransaction

Syntax

```plaintext
procedure RemoveSession(Session: TOraSession);
```

**Parameters**

`Session`  
Holds a TOraSession object.

**Remarks**

Call the RemoveSession method to disassociate the specified session from the transaction.

**See Also**

- Sessions
- TOraTransaction.AddSession
- ClearSessions

Resume Method

Resumes a detached transaction.

Class

TOraTransaction

Syntax

```plaintext
procedure Resume;
```

**Remarks**

Call the Resume method to resume a detached transaction.

**See Also**

- ResumeTimeOut
- Detach

© 1997-2021
Devart. All Rights Reserved.  
Request Support  
DAC Forum  
Provide Feedback
5.31.1.3.6 RollbackToSavepoint Method

Discards all modifications made during the current transaction and restores its state to the moment of the savepoint.

Class

T0raTransaction

Syntax

procedure RollbackToSavepoint(const Savepoint: string);

Parameters

Savepoint

Holds the name of the savepoint.

Remarks

Call the RollbackToSavepoint method to cancel all updates for the current transaction and restore its state up to the moment of the last defined savepoint.

See Also

- Savepoint
- TDATransaction.Rollback

5.31.1.3.7 Savepoint Method

Defines a savepoint in the transaction.

Class

T0raTransaction

Syntax

procedure Savepoint(const Savepoint: string);

Parameters

Savepoint

Holds the savepoint name that identifies the savepoint.
Remarks

Call the Savepoint method to define a point in the transaction to which you can roll back later. As the parameter, you can pass any valid name to identify the savepoint.

To roll back to the last savepoint call `RollbackToSavepoint`.

See Also

- `RollbackToSavepoint`

### StartTransaction Method

Begins a new user transaction against the database server.

**Class**

`TOraTransaction`

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>StartTransaction</code></td>
<td>Begins a new user transaction against the database server.</td>
</tr>
<tr>
<td><code>StartTransaction(Resume: boolean)</code></td>
<td>Begins a new user transaction against the database server.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.
Reserved.

Begins a new user transaction against the database server.

Class

**TOraTransaction**

Syntax

```plaintext
procedure StartTransaction(Resume: boolean); reintroduce;
overload;
```

Parameters

*Resume*

True, if detached transaction branches will be resumed on sessions. False otherwise.

Remarks

Call the StartTransaction method to begin a new user transaction against the database server. Before calling StartTransaction, an application should check the status of the **TOraTransaction.Active** property. If **TOraTransaction.Active** is True, it indicates that a transaction is already in progress, a subsequent call to StartTransaction without first calling **TDATransaction.Commit** or **TDATransaction.Rollback** to end the current transaction raises **EDatabaseError**. Calling StartTransaction when connection is closed also raises **EDatabaseError**.

Updates, insertions, and deletions that take place after a call to StartTransaction are held by the server until an application calls Commit to save the changes or Rollback to cancel them.

Setting the Resume parameter to True means detached transaction branches will be resumed on sessions.

© 1997-2021 Devart. All Rights Reserved.

5.31.1.1.4 Events

Events of the **TOraTransaction** class.

For a complete list of the **TOraTransaction** class members, see the **TOraTransaction Members** topic.

Public
### OnCommit Event

**Name**: OnCommit (inherited from **TDATransaction**)

**Description**: Occurs after the transaction has been successfully committed.

### OnCommitRetaining Event

**Name**: OnCommitRetaining (inherited from **TDATransaction**)

**Description**: Occurs after CommitRetaining has been executed.

### OnRollback Event

**Name**: OnRollback (inherited from **TDATransaction**)

**Description**: Occurs after the transaction has been successfully rolled back.

### OnRollbackRetaining Event

**Name**: OnRollbackRetaining (inherited from **TDATransaction**)

**Description**: Occurs after RollbackRetaining has been executed.

### Published

**Name**: OnError

**Description**: Occurs for processing errors that can be arised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

### See Also
- **TOraTransaction Class**
- **TOraTransaction Class Members**

Occurs for processing errors that can be arised during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others.

**Class**

**TOraTransaction**
Syntax

```property
OnError: TDATransactionErrorEvent;
```

Remarks

Write the `OnError` event handler to process errors that occur during executing transaction and savepoint control statements such as COMMIT, ROLLBACK, SAVEPOINT, RELEASE SAVEPOINT and others. Check the `E` parameter to get an error code.

### 5.31.2 Enumerations

Enumerations in the `OraTransaction` unit.

#### Enumerations

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TGlobalCoordinator</code></td>
<td>Specifies with what distributed transaction, perform two-phase commit or rollback on all sessions will be coordinated.</td>
</tr>
</tbody>
</table>

© 1997-2021
Devart. All Rights Reserved.

### 5.31.2.1 TGlobalCoordinator Enumeration

Specifies with what distributed transaction, perform two-phase commit or rollback on all sessions will be coordinated.

Unit

`OraTransaction`

Syntax

```
TGlobalCoordinator = (gcInternal, gcMTS);
```

Values
### 5.32 VirtualDataSet

This unit contains implementation of the TVirtualDataSet component.

#### Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomVirtualDataSet</td>
<td>A base class for representation of arbitrary data in tabular form.</td>
</tr>
<tr>
<td>TVirtualDataSet</td>
<td>Dataset that processes arbitrary non-tabular data.</td>
</tr>
</tbody>
</table>

#### Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOnDeleteRecordEvent</td>
<td>This type is used for the E:Devart.Dac.TVirtualDataSet.OnDeleteRecord event.</td>
</tr>
<tr>
<td>TOnGetFieldValueEvent</td>
<td>This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetFieldValue event.</td>
</tr>
<tr>
<td>TOnGetRecordCountEvent</td>
<td>This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetRecordCount event.</td>
</tr>
<tr>
<td>TOnModifyRecordEvent</td>
<td>This type is used for E:Devart.Dac.TVirtualDataSet.OnInsertRecord and E:Devart.Dac.TVirtualDataSet.OnModifyRecord events.</td>
</tr>
</tbody>
</table>
5.32.1 Classes

Classes in the VirtualDataSet unit.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCustomVirtualDataSet</td>
<td>A base class for representation of arbitrary data in tabular form.</td>
</tr>
<tr>
<td>TVirtualDataSet</td>
<td>Dataset that processes arbitrary non-tabular data.</td>
</tr>
</tbody>
</table>

5.32.1.1 TCustomVirtualDataSet Class

A base class for representation of arbitrary data in tabular form.

For a list of all members of this type, see TCustomVirtualDataSet members.

Unit

VirtualDataSet

Syntax

TCustomVirtualDataSet = class(TMemDataSet);

Inheritance Hierarchy

TMemDataSet
  TCustomVirtualDataSet

Properties

TCustomVirtualDataSet class overview.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CachedUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>Prepared</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong></td>
<td>(inherited from TMemDataSet) Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong></td>
<td>(inherited from TMemDataSet) Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong></td>
<td>(inherited from TMemDataSet) Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong></td>
<td>(inherited from TMemDataSet) Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>GetBlob</strong></td>
<td>(inherited from TMemDataSet) Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>Locate</strong></td>
<td>(inherited from TMemDataSet) Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><strong>LocateEx</strong></td>
<td>(inherited from TMemDataSet) Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td><strong>Prepare</strong></td>
<td>(inherited from TMemDataSet) Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong></td>
<td>(inherited from TMemDataSet) Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td><strong>RevertRecord</strong></td>
<td>(inherited from TMemDataSet) Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>SaveToXML</strong></td>
<td>(inherited from TMemDataSet) Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><strong>SetRange</strong></td>
<td>(inherited from TMemDataSet) Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><strong>SetRangeEnd</strong></td>
<td>(inherited from TMemDataSet) Indicates that subsequent state.</td>
</tr>
</tbody>
</table>
assignments to field values specify the end of the range of rows to include in the dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><strong>UnPrepare</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><strong>UpdateResult</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdateStatus</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnUpdateError</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td><strong>OnUpdateRecord</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Occurs when a single update component can not handle the updates.</td>
</tr>
</tbody>
</table>

5.32.1.2 **TVirtualDataSet Class**

Dataset that processes arbitrary non-tabular data.

For a list of all members of this type, see **TVirtualDataSet** members.
**virtualDataSet**

**Syntax**

```pascal
TVirtualDataSet = class(TCustomVirtualDataSet);
```

**Inheritance Hierarchy**

- TMemDataSet
  - TCustomVirtualDataSet
    - TVirtualDataSet

5.32.1.2.1 Members

**TVirtualDataSet** class overview.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CachedUpdates</td>
<td>(inherited from TMemDataSet) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td>IndexFieldNames</td>
<td>(inherited from TMemDataSet) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td>KeyExclusive</td>
<td>(inherited from TMemDataSet) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td>LocalConstraints</td>
<td>(inherited from TMemDataSet) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td>LocalUpdate</td>
<td>(inherited from TMemDataSet) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td>Prepared</td>
<td>(inherited from TMemDataSet) Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td>Ranged</td>
<td>(inherited from TMemDataSet) Indicates whether a range is applied to a dataset.</td>
</tr>
</tbody>
</table>
**UpdateRecordTypes** (inherited from **TMemDataSet**)  
Used to indicate the update status for the current record when cached updates are enabled.

**UpdatesPending** (inherited from **TMemDataSet**)  
Used to check the status of the cached updates buffer.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>Locate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><strong>LocateEx</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Excludes features that don't need to be included to the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>Prepare</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td><code>RestoreUpdates</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td><code>RevertRecord</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><code>SaveToXML</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><code>SetRange</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><code>SetRangeEnd</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><code>SetRangeStart</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><code>UnPrepare</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
<tr>
<td><code>UpdateResult</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.</td>
</tr>
<tr>
<td><code>UpdateStatus</code> (inherited from <code>TMemDataSet</code>)</td>
<td>Indicates the current update status for the dataset when cached updates are enabled.</td>
</tr>
</tbody>
</table>
Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnUpdateError (inherited from TMemDataSet)</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td>OnUpdateRecord (inherited from TMemDataSet)</td>
<td>Occurs when a single update component can not handle the updates.</td>
</tr>
</tbody>
</table>

5.32.2 Types

Types in the VirtualDataSet unit.

Types

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOnDeleteRecordEvent</td>
<td>This type is used for the E:Devart.Dac.TVirtualDataSet.OnDeleteRecord event.</td>
</tr>
<tr>
<td>TOnGetFieldValueEvent</td>
<td>This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetFieldValue event.</td>
</tr>
<tr>
<td>TOnGetRecordCountEvent</td>
<td>This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetRecordCount event.</td>
</tr>
<tr>
<td>TOnModifyRecordEvent</td>
<td>This type is used for E:Devart.Dac.TVirtualDataSet.OnInsertRecord and E:Devart.Dac.TVirtualDataSet.OnModifyRecord events.</td>
</tr>
</tbody>
</table>
5.32.2.1 TOnDeleteRecordEvent Procedure Reference

This type is used for the E:Devart.Dac.TVirtualDataSet.OnDeleteRecord event.

Unit

VirtualDataSet

Syntax

TOnDeleteRecordEvent = procedure (Sender: TObject; RecNo: Integer) of object;

Parameters

Sender
An object that raised the event.

RecNo
Number of the record being deleted.

5.32.2.2 TOnGetFieldValueEvent Procedure Reference

This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetFieldValue event.

Unit

VirtualDataSet

Syntax

TOnGetFieldValueEvent = procedure (Sender: TObject; Field: TField; RecNo: Integer; out Value: Variant) of object;

Parameters

Sender
An object that raised the event.

Field
The field, which data has to be returned.

RecNo
The number of the record, which data has to be returned.

Value
Requested field value.
5.32.2.3  TOnGetRecordCountEvent Procedure Reference

This type is used for the E:Devart.Dac.TVirtualDataSet.OnGetRecordCount event.

Unit

VirtualDataSet

Syntax

TOnGetRecordCountEvent = procedure (Sender: TObject; out Count: Integer) of object;

Parameters

Sender
An object that raised the event.

Count
The number of records that the virtual dataset will contain.

5.32.2.4  TOnModifyRecordEvent Procedure Reference

This type is used for E:Devart.Dac.TVirtualDataSet.OnInsertRecord and E:Devart.Dac.TVirtualDataSet.OnModifyRecord events.

Unit

VirtualDataSet

Syntax

TOnModifyRecordEvent = procedure (Sender: TObject; var RecNo: Integer) of object;

Parameters

Sender
An object that raised the event.

RecNo
Number of the record being inserted or modified.

© 1997-2021 Devart. All Rights Reserved.
5.33 VirtualTable

This unit contains implementation of the TVirtualTable component.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVirtualTable</td>
<td>Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.</td>
</tr>
</tbody>
</table>

5.33.1 Classes

Classes in the VirtualTable unit.

Classes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TVirtualTable</td>
<td>Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.</td>
</tr>
</tbody>
</table>

5.33.1.1 TVirtualTable Class

Dataset that stores data in memory. This component is placed on the Data Access page of the Component palette.

For a list of all members of this type, see TVirtualTable members.

Unit
VirtualTable

Syntax

```pascal
TVirtualTable = class(TMemDataSet);
```

Inheritance Hierarchy

```
TMemDataSet
 TVirtualTable
```

© 1997-2021 Devart. All Rights Reserved.

5.3.1.1.1 Members

**TVirtualTable** class overview.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CachedUpdates</strong></td>
<td>(inherited from TMemDataSet) Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>DefaultSortType</strong></td>
<td>Used to determine the default type of local sorting for string fields.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong></td>
<td>(inherited from TMemDataSet) Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong></td>
<td>(inherited from TMemDataSet) Specifies the upper and lower boundaries for a range.</td>
</tr>
<tr>
<td><strong>LocalConstraints</strong></td>
<td>(inherited from TMemDataSet) Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.</td>
</tr>
<tr>
<td><strong>LocalUpdate</strong></td>
<td>(inherited from TMemDataSet) Used to prevent implicit update of rows on database server.</td>
</tr>
<tr>
<td><strong>Prepared</strong></td>
<td>(inherited from TMemDataSet) Determines whether a query is prepared for execution or not.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Ranged</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates whether a range is applied to a dataset.</td>
</tr>
<tr>
<td><strong>UpdateRecordTypes</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to indicate the update status for the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>UpdatesPending</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Used to check the status of the cached updates buffer.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ApplyRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td><strong>ApplyUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td><strong>Assign</strong></td>
<td>Copies fields and data from another TDataSet component.</td>
</tr>
<tr>
<td><strong>CancelRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td><strong>CancelUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td><strong>CommitUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td><strong>DeferredPost</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td><strong>EditRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td><strong>EditRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>LoadFromFile</strong></td>
<td>Loads data from a file into a TVirtualTable component.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LoadFromStream</td>
<td>Copies data from a stream into a TVirtualTable component.</td>
</tr>
<tr>
<td>Locate</td>
<td>(inherited from TMemDataSet) Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td>LocateEx</td>
<td>(inherited from TMemDataSet) Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td>Prepare</td>
<td>(inherited from TMemDataSet) Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td>RestoreUpdates</td>
<td>(inherited from TMemDataSet) Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td>RevertRecord</td>
<td>(inherited from TMemDataSet) Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td>SaveToXML</td>
<td>(inherited from TMemDataSet) Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td>SetRange</td>
<td>(inherited from TMemDataSet) Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td>SetRangeEnd</td>
<td>(inherited from TMemDataSet) Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>SetRangeStart</td>
<td>(inherited from TMemDataSet) Indicates that subsequent assignments to field values specify the start of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td>UnPrepare</td>
<td>(inherited from TMemDataSet) Frees the resources allocated for a previously prepared query on the server and client sides.</td>
</tr>
</tbody>
</table>
**UpdateResult** (inherited from *TMemDataSet*)

Reads the status of the latest call to the ApplyUpdates method while cached updates are enabled.

**UpdateStatus** (inherited from *TMemDataSet*)

Indicates the current update status for the dataset when cached updates are enabled.

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnUpdateError</strong> (inherited from <em>TMemDataSet</em>)</td>
<td>Occurs when an exception is generated while cached updates are applied to a database.</td>
</tr>
<tr>
<td><strong>OnUpdateRecord</strong> (inherited from <em>TMemDataSet</em>)</td>
<td>Occurs when a single update component cannot handle the updates.</td>
</tr>
</tbody>
</table>

© 1997-2021 Devart. All Rights Reserved.

5.33.1.1.2 Properties

Properties of the *TVirtualTable* class.

For a complete list of the *TVirtualTable* class members, see the *TVirtualTable Members* topic.

**Public**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CachedUpdates</strong> (inherited from <em>TMemDataSet</em>)</td>
<td>Used to enable or disable the use of cached updates for a dataset.</td>
</tr>
<tr>
<td><strong>IndexFieldNames</strong> (inherited from <em>TMemDataSet</em>)</td>
<td>Used to get or set the list of fields on which the recordset is sorted.</td>
</tr>
<tr>
<td><strong>KeyExclusive</strong> (inherited from <em>TMemDataSet</em>)</td>
<td>Specifies the upper and lower boundaries for a range.</td>
</tr>
</tbody>
</table>
LocalConstraints (inherited from TMemDataSet)  
Used to avoid setting the Required property of a TField component for NOT NULL fields at the time of opening TMemDataSet.

LocalUpdate (inherited from TMemDataSet)  
Used to prevent implicit update of rows on database server.

Prepared (inherited from TMemDataSet)  
Determines whether a query is prepared for execution or not.

Ranged (inherited from TMemDataSet)  
Indicates whether a range is applied to a dataset.

UpdateRecordTypes (inherited from TMemDataSet)  
Used to indicate the update status for the current record when cached updates are enabled.

UpdatesPending (inherited from TMemDataSet)  
Used to check the status of the cached updates buffer.

Published

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultSortType</td>
<td>Used to determine the default type of local sorting for string fields.</td>
</tr>
</tbody>
</table>

See Also
- TVirtualTable Class
- TVirtualTable Class Members

© 1997-2021 Devart. All Rights Reserved.

5.33.1.1.2.1 DefaultSortType Property

Used to determine the default type of local sorting for string fields.

Class
- TVirtualTable

Syntax
property DefaultSortType: TSortType default stCaseSensitive;

Remarks

The DefaultSortType property is used when a sort type is not specified explicitly after the field name in the `TMemDataSet.IndexFieldNames` property of a dataset.

Methods

Methods of the `TVirtualTable` class.

For a complete list of the `TVirtualTable` class members, see the `TVirtualTable Members` topic.

Public

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyRange</td>
<td>Applies a range to the dataset.</td>
</tr>
<tr>
<td>ApplyUpdates</td>
<td>Overloaded. Writes dataset's pending cached updates to a database.</td>
</tr>
<tr>
<td>Assign</td>
<td>Copies fields and data from another TDataSet component.</td>
</tr>
<tr>
<td>CancelRange</td>
<td>Removes any ranges currently in effect for a dataset.</td>
</tr>
<tr>
<td>CancelUpdates</td>
<td>Clears all pending cached updates from cache and restores dataset in its prior state.</td>
</tr>
<tr>
<td>CommitUpdates</td>
<td>Clears the cached updates buffer.</td>
</tr>
<tr>
<td>DeferredPost</td>
<td>Makes permanent changes to the database server.</td>
</tr>
<tr>
<td>EditRangeEnd</td>
<td>Enables changing the ending value for an existing range.</td>
</tr>
<tr>
<td>EditRangeStart</td>
<td>Enables changing the starting value for an existing range.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetBlob</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Retrieves TBlob object for a field or current record when only its name or the field itself is known.</td>
</tr>
<tr>
<td><strong>LoadFromFile</strong></td>
<td>Loads data from a file into a TVirtualTable component.</td>
</tr>
<tr>
<td><strong>LoadFromStream</strong></td>
<td>Copies data from a stream into a TVirtualTable component.</td>
</tr>
<tr>
<td><strong>Locate</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Searches a dataset for a specific record and positions the cursor on it.</td>
</tr>
<tr>
<td><strong>LocateEx</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Excludes features that don't need to be included to the TMemDataSet.Locate method of TDataSet.</td>
</tr>
<tr>
<td><strong>Prepare</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Allocates resources and creates field components for a dataset.</td>
</tr>
<tr>
<td><strong>RestoreUpdates</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Marks all records in the cache of updates as unapplied.</td>
</tr>
<tr>
<td><strong>RevertRecord</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Cancels changes made to the current record when cached updates are enabled.</td>
</tr>
<tr>
<td><strong>SaveToXML</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Overloaded. Saves the current dataset data to a file or a stream in the XML format compatible with ADO format.</td>
</tr>
<tr>
<td><strong>SetRange</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Sets the starting and ending values of a range, and applies it.</td>
</tr>
<tr>
<td><strong>SetRangeEnd</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates that subsequent assignments to field values specify the end of the range of rows to include in the dataset.</td>
</tr>
<tr>
<td><strong>SetRangeStart</strong> (inherited from <strong>TMemDataSet</strong>)</td>
<td>Indicates that subsequent</td>
</tr>
</tbody>
</table>
Assign Method

Copies fields and data from another TDataSet component.

Class

TVirtualTable

Syntax

```pascal
procedure Assign(Source: TPersistent); override;
```

Parameters

Source

Holds the TDataSet component to copy fields and data from.

Remarks

Call the Assign method to copy fields and data from another TDataSet component.
Note: Unsupported field types are skipped (i.e. destination dataset will contain less fields than the source one). This may happen when Source is not a TVirtualTable component but some server-oriented dataset.

Example
Query1.SQL.Text := 'SELECT * FROM DEPT';
Query1.Active := True;
VirtualTable1.Assign(Query1);
VirtualTable1.Active := True;

OraQuery1.SQL.Text := 'SELECT * FROM DEPT';
OraQuery1.Active := True;
VirtualTable1.Assign(OraQuery1);
VirtualTable1.Active := True;

See Also
• TVirtualTable

LoadFromFile Method

Loads data from a file into a TVirtualTable component.

Class
TVirtualTable

Syntax

procedure LoadFromFile(const FileName: string; LoadFields: boolean = True; DecodeHTMLEntities: boolean = True);

Parameters
FileName
  Holds the name of the file to load data from.

LoadFields
  Indicates whether to load fields from the file.

DecodeHTMLEntities
  Indicates whether to decode HTML entities from the file.

Remarks
Call the LoadFromFile method to load data from a file into a TVirtualTable component. Specify
the name of the file to load into the field as the value of the FileName parameter. This file may be an XML document in ADO-compatible format or in virtual table data format. The file format is detected automatically.

5.33.1.3.3 LoadFromStream Method

Copies data from a stream into a TVirtualTable component.

Class

­TVirtualTable

Syntax

procedure LoadFromStream(Stream: TStream; LoadFields: boolean = True; DecodeHTMLEntities: boolean = True);

Parameters

Stream
Holds the stream from which the field's value is copied.

LoadFields
Indicates whether to load fields from the stream.

DecodeHTMLEntities
Indicates whether to decode HTML entities from the stream.

Remarks

Call the LoadFromStream method to copy data from a stream into a TVirtualTable component. Specify the stream from which the field's value is copied as the value of the Stream parameter. Data in the stream may be in ADO-compatible format or in virtual table data format. The data format is detected automatically.