

Tracking Database Schema Changes

Guidelines on database versioning using Devart tools

Table of Contents

Introduction.....	3
Typical Development Environments.....	3
Database versioning problems.....	4
Improving database schema development.....	4
Putting a database schema under a version control.....	4
Making changes to a database schema.....	6
Committing changes to a version control system.....	7
Conclusion.....	9
About Devart.....	10

Introduction

This white paper examines problems that any developer can meet when changing a database structure, integrating structure changes made by other developers, and moving the database from one environment to another.

The document also introduces a solution from Devart, a vendor of handy tools to provide native connectivity to popular databases, and facilitate their development and management. The recommendations, provided in the white paper, target any developer team that develop Microsoft SQL Server databases and wants to improve the work with efficient tools.

Typical Development Environments

Database development process typically involves four types of database environments shown on Figure 1. They are sandbox, development environment, UAT environment, and production.

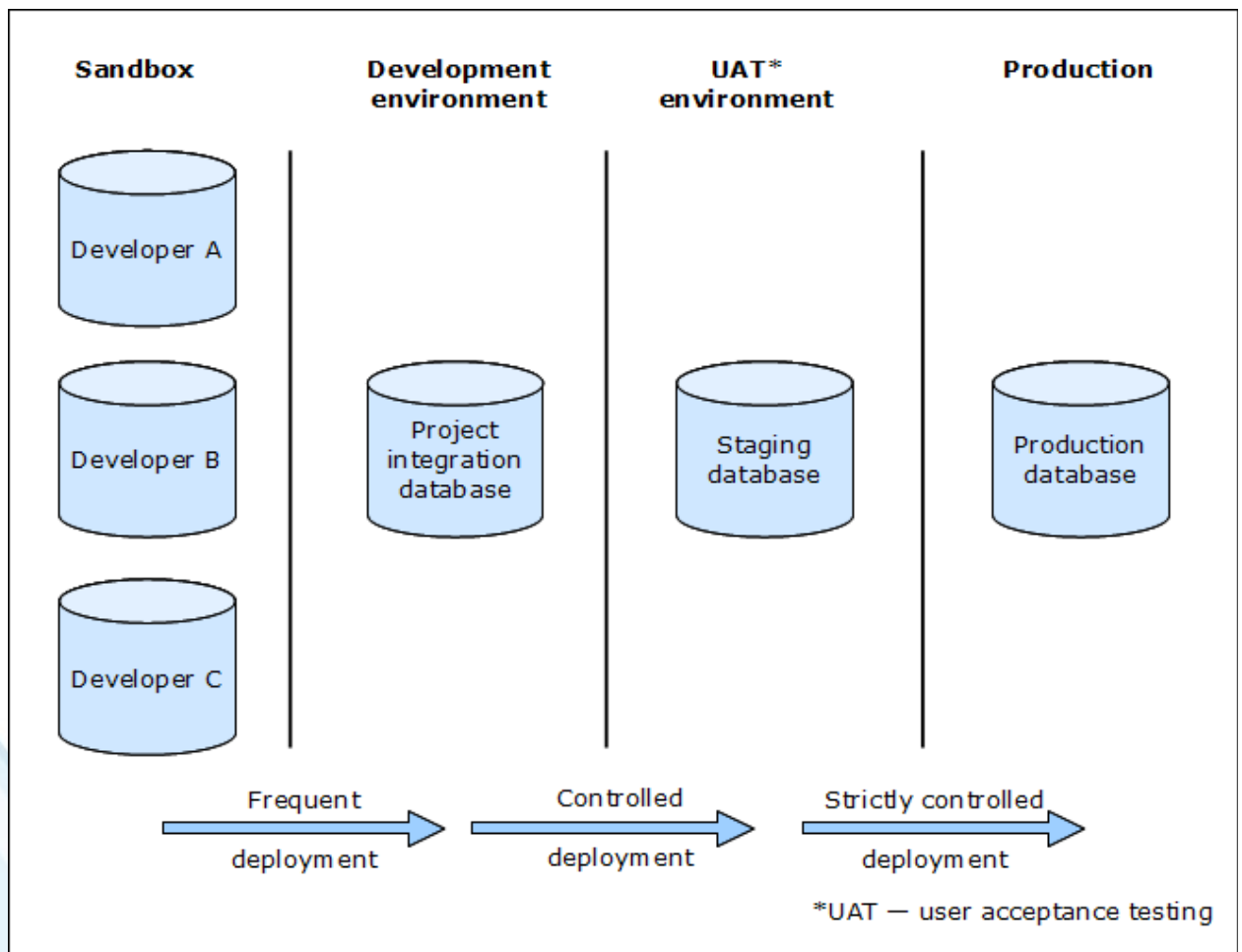


Figure 1 – Database development environments

Sandbox

This is a special-purpose type of development environment that is, in theory, developed for each developer separately. In the sandbox a developer can freely experiment with a database and make any modifications to complete the tasks.

Development environment

This special-purpose development environment integrates changes made by all the developers involved in the project. Big companies usually work on several projects and, therefore, several integrated development environments may exist. Deploying such environments is common, sometimes it happens several times a day.

User acceptance testing environment

This is a special-purpose type of development environment created on database pre-deployment stage. Providing real conditions to test the database functionality, this environment allows getting specific feedback results before deploying the database to the production server. Deploying such environments is controllable so a developer can check the correctness of deployment methods used.

Production

This is a final type of database development environment. Production database deployment is strictly controlled by an administrator who is in charge of deployment steps and the result.

Database versioning problems

Any of aforementioned database development environments has its database modification. This means that during project development at least four database modifications can exist. This brings significant complexity during change management and an additional load for developers.

Here are some common challenges following database change management:

1. Integrating changes made by different developers.
2. Rolling back changes selectively in case of any errors.
3. Detecting what database modification is in each development environment.
4. Matching feedback results with a database modification being tested.

The situation becomes worse when no version control system is used during database development. This is still a common thing, especially for small projects.

Improving database schema development

One of effective solutions to improve database schema development includes usage of **dbForge Schema Compare for SQL Server**, a tailored tool from Devart to facilitate comparison and synchronization of MS SQL Server databases.

The following recommendations on improving database schema development

Putting a database schema under a version control

The first thing required to improve database development is to put a database under a version control system. It takes making a database snapshot. Even if you have already added deployment scenarios into the version control system, follow this recommendation. In many cases, as it is more convenient to make changes in a database itself, developers change the developers do this and ignore deployment scripts.

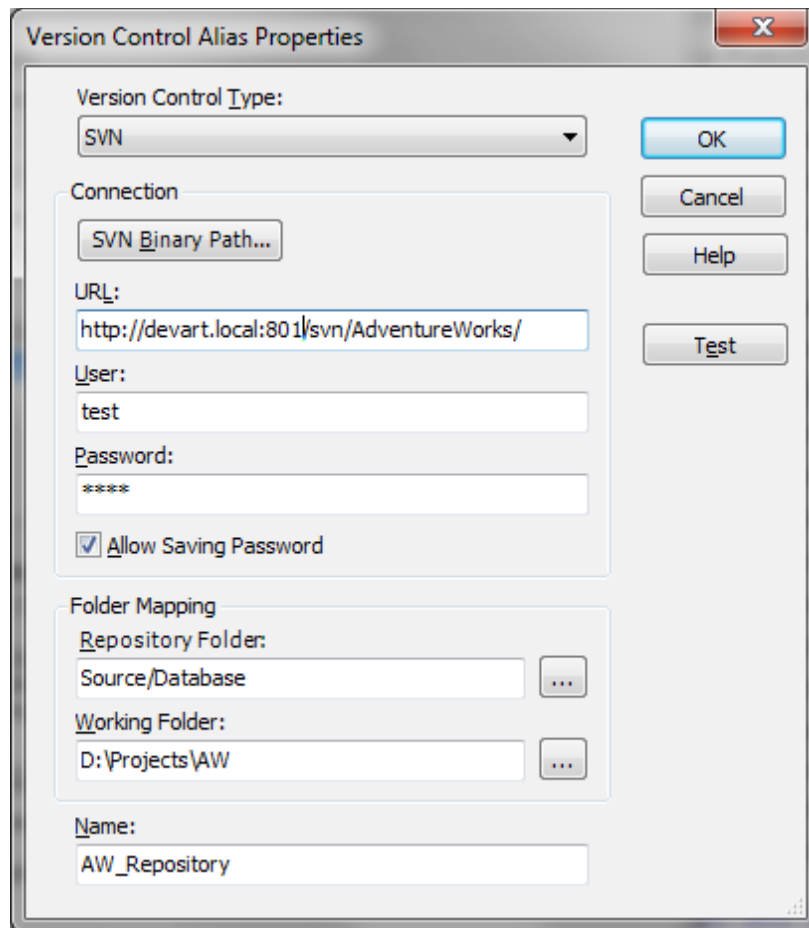


Figure 1 - Version control alias for Subversion

To put a database under a version control by using dbForge Schema Compare, three steps are required:

1. In the Version Control Alias Properties dialog box, set up a version control alias that points the working directory where a database schema is stored. (Figure 2 illustrates the created alias for Subversion.) Pay attention that you should specified a valid directory that exists on the disc and contains checked-out files. In the Create Schema Snapshot dialog box, select a database modification that will be a baseline and set up all necessary parameters to create a database snapshot (see Figure 3).

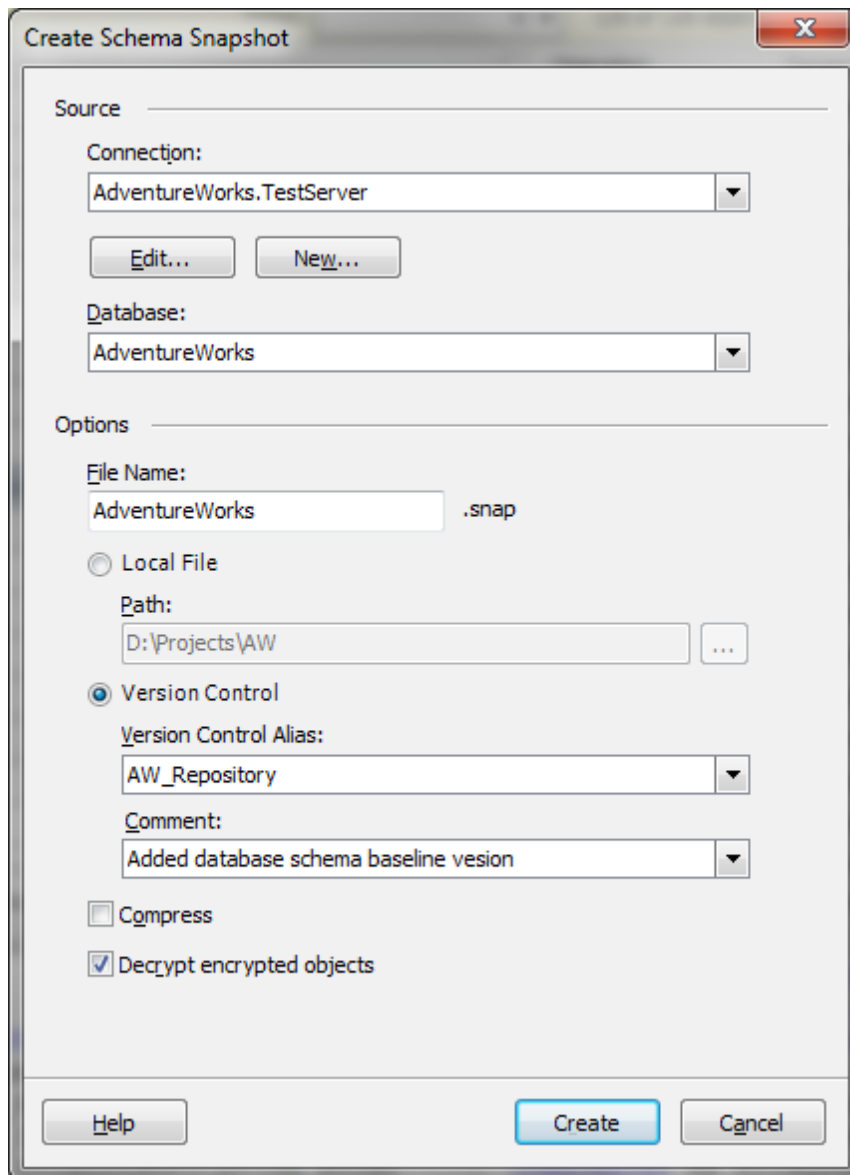


Figure 2 - Schema snapshot creation

2. Press Create and dbForge Schema Compare will create a database snapshot and automatically put it under a version control system based on the specified alias.

Making changes to a database schema

When the database is under a version control system, you can modify the database structure in 'the sandbox'. Database modifications are rarely separate, usually they consist of quite a few interconnected changes.

For example, DROP COLUMN operation may require the following:

- Redefining foreign keys
- Correcting broken views
- Correcting broken triggers and stored procedures

Mostly, after changing a database schema, a developer cannot just commit the changes into a version control system. The changes made by other developers, who are working on the same database, might conflict. That's why it is necessary to compare the modified copy of

the database with the latest revision in the version control repository before committing. It is a simple procedure with dbForge Schema Compare for SQL Server. It takes only to select required source and target and start a comparison (see Figure 4).

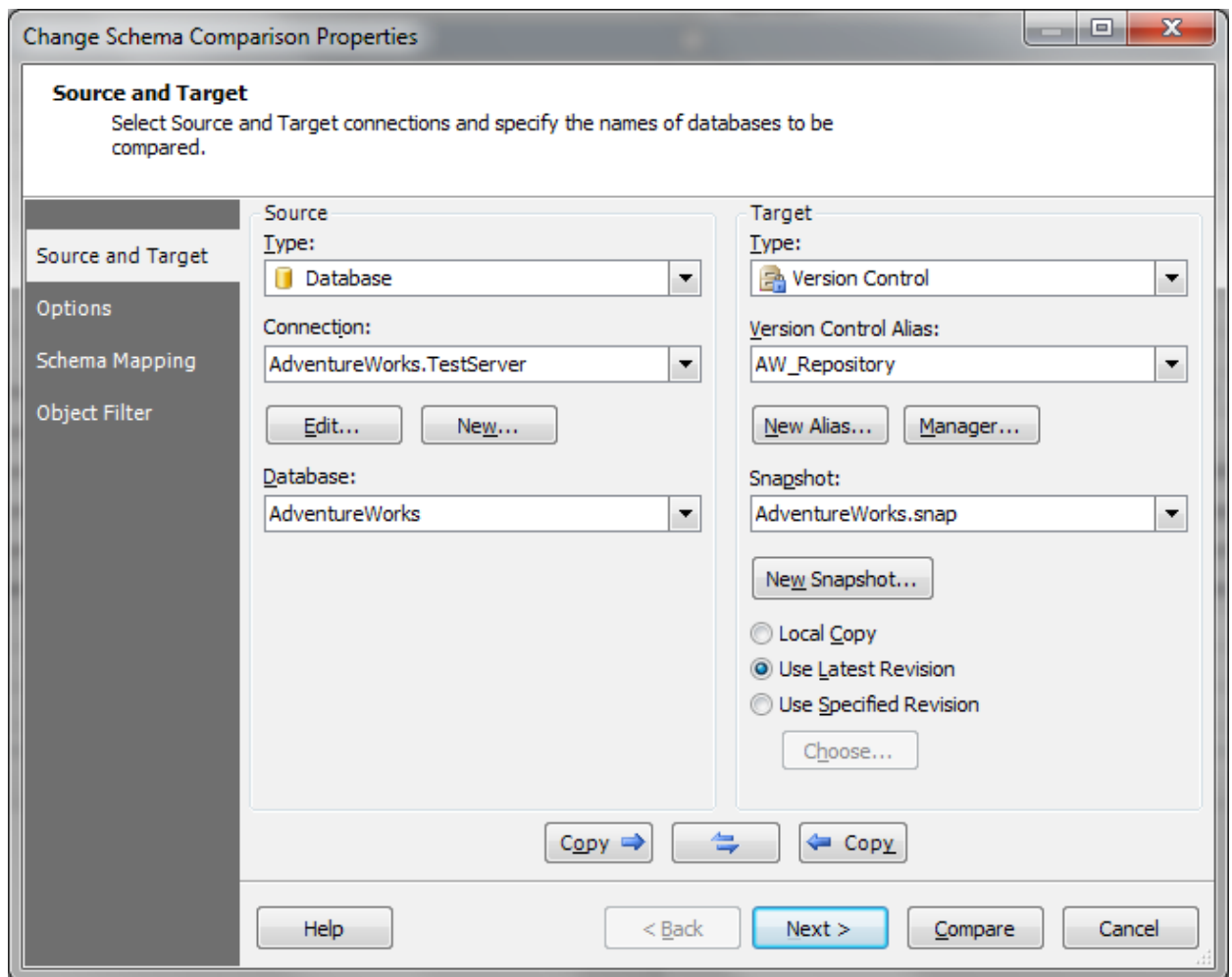


Figure 4 - Comparing developer changes with repository

Checking that there are no conflict changes or having solved the conflicts if any exist, the developer can commit the database changes to the version control system repository.

Committing changes to a version control system

Based on a project strategy, database deployment scenarios can be stored in a version control system in two ways:

1. The version control system repository stores deployment scenarios of the latest database revision.

This variant presupposes database deployment from scratch, i.e. while updating a database, you should a backup of the previous database revision, deploy a new database revision, then insert data.

Moving data from one database to another can be facilitated by **dbForge Data Compare for SQL Server**, a special-purpose tool for MySQL data migrating.

2. The version control system repository stores deployment scenarios for a baseline database version as well as for its revisions.

In this case you can deploy the database either from scratch or by using a certain revision, i.e. you can do incremental deployment.

Regardless of the selected project strategy, it is convenient to store a holistic database snapshot and use it to track a database revision. Especially it is beneficial when the version control system repository stores deployment scenarios for a baseline database version as well as for its revisions.

When committing database revisions, dbForge Schema Compare for SQL Server offers the Schema Synchronization Wizard (see Figure 5) to make this task in a quickly and accurate manner.

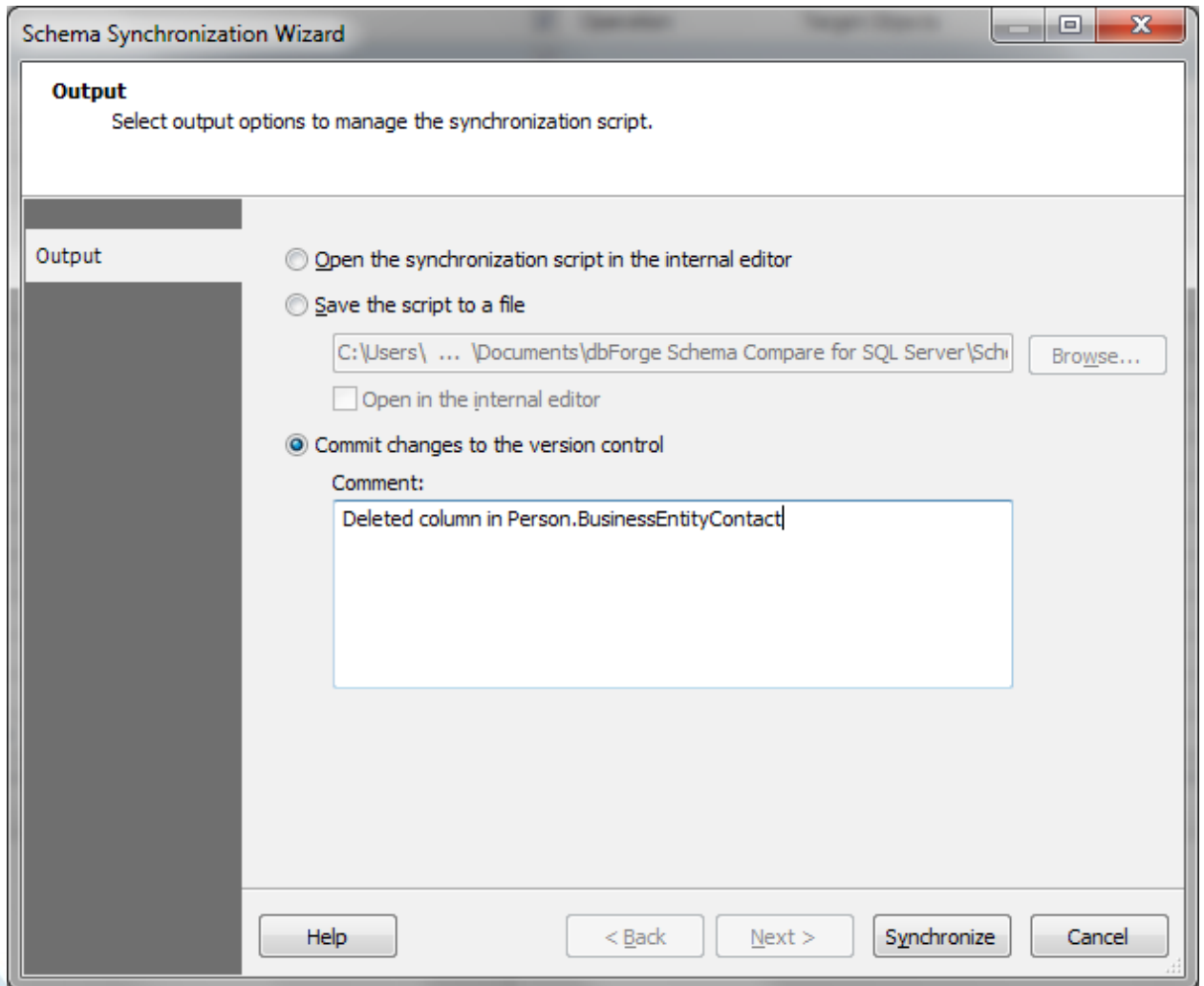


Figure 5 - Committing the new revision of schema snapshot

Conclusion

Devart's dbForge for SQL Server product line presents database development and change management tools for streamlining development processes. The tools boost productivity across the whole development cycle.

Free 30 day trials of the described tools are available for download:

Schema Compare for SQL Server - <http://www.devart.com/dbforge/sql/schemacompare/>

Data Compare for SQL Server - <http://www.devart.com/dbforge/sql/datacompare/>

About Devart



Devart (formerly known as Core Lab) is a software development company founded in 1998. It is a provider of native connectivity solutions, development and administration tools for Oracle, SQL Server, MySQL,

PostgreSQL, InterBase, Firebird, and SQLite databases. It is a partner of such software providers as Microsoft and CodeGear and participates in MySQL Network Program. Devart is dedicated to delivering the fastest available data access and the broadest database support to industry professionals.

Company Web Site: www.devart.com