



# Choosing the Right Way of Migrating MySQL Databases

**Devart White Paper**

June, 2010

# Table of Contents

Introduction.....	3
Common Cases and Challenges of Migrating Databases.....	3
Moving to a New MySQL Server Version.....	3
Moving to Another Server Machine.....	4
Moving a Local MySQL Database to a Remote Hosting Provider Server.....	4
Creating a Copy of a Live Database and Its Further Support.....	4
Standard Ways to Migrate Databases.....	5
Meeting Challenges with Devart Schema and Data Compare Tools.....	6
Conclusion.....	18
About Devart.....	20

# Introduction

Every day on web development or server-related forums someone asks a question about how to move databases across different servers or server versions. The answers are not straight and simple, but rather diverse and dependent on each particular case. While working with databases, almost every database developer or DBA faces the task to migrate database structure and data. It is very common in database development life cycle, but still remains challenging. This white paper examines the use cases of database migration and challenges that accompany it. The main attention is devoted to solutions used to remove the complexity and bottlenecks of database migration. The white paper introduces Devart's Schema and Data Compare tools tailored to facilitate any database migration tasks and reduce time and effort.

## Common Cases and Challenges of Migrating Databases

The reasons why one would want to move MySQL databases are different. Here are several most frequent ones:

- Moving to a new MySQL Server version
- Moving to another server machine (changing a web hosting provider or replacing server equipment)
- Moving a local MySQL database to a remote hosting provider server
- Creating a copy of a live database and its further support (required for testing new technologies and a new architecture)

### Moving to a New MySQL Server Version

This task can be considered in two aspects: moving to a newer server version (upgrading) or some prior version (downgrading). In the first case, the problems may happen as a new server version does not support prior structures or handles them differently. In the second case, the trouble worsens as the prior version may lack not only new structures, but the whole object types (for example, MySQL 4.0 does not support procedures, neither MySQL 5.1.9. supports partitioning) and data types. As a result, simple statements execution may fail or a MySQL server may execute them in an unexpected way. Either outcome is unfavorable. Suppose, your database consists of few objects and you can fix queries to execute them on a new server, but what if you are moving data, and some data types mismatch? It is usually followed by hours and hours of manual and intensive values converting.

## Moving to Another Server Machine

This commonly happens when changing a web hosting provider or replacing server environment. In the simplest cases, it takes to backup a database and restore it on a new server machine. However, as it happens in the life, what is originally planned as a backup/restore turns into a disaster. The usual situations during changing a web hosting provider is that a new server is configured in a different mode as the old server, has different default values for some parameters (for example, by default it has InnoDB engine for tables instead of MyISAM one) or lacks the necessary functionality (for example, partitioning support is turned off). To illustrate the case, suppose a backup file was created without a table engine specified, then the default engine will be used while creating the tables, meanwhile all the settings of the prior table engine will be lost. And, much manual edition follows, for example, you will have to remove foreign keys indications from each table description. (Например, указание внешних ключей придется удалять из каждого описания таблицы.) Another tough time can come due to encoding differences. You may try to upload data with utf-8 charset to the server with some other default charset, as the result, data will be damaged and its restoring will become heavy toil.

## Moving a Local MySQL Database to a Remote Hosting Provider Server

Every web developer has to move fresh changes, tested and verified on the local database, to a remote database. The problems mentioned in case of moving to another server machine is applicable to this task. Besides, additional problems relate to metadata and data changes, such as:

- Adding new objects, deleting old ones. It refers both to top level objects (tables, views, procedures) and sub-objects (columns, indexes, etc)
- Changing texts of views, procedures, triggers or events
- Changing table structures (for example, adding partitions)
- Adding, deleting or changing records in tables (taking into an account metadata changes)

Often, you need to move all the changes from one database to another. One of possible solutions is to fully drop an existing target database, generate a script that contains the structure and the data of the source database and execute it. However, in most cases, it is prohibited to drop the existing database that plays an important role and delivers valuable information.

## Creating a Copy of a Live Database and Its Further Support

The need to use copies of a live database is an integral part of developing and testing any database. You can run tests, change the structure, add data, check and change again until your database is ready for moving to the remote server. Until this moment, the server is not affected.

The bottleneck is that you want to accurately move the ready changes, but save the testing stuff like specific data for functional, regression or load testing.

All the aforementioned challenges become even tougher when databases, you need to move, have complex structures and relations, great number of objects and records in the tables. The unfortunate truth is database migration becomes not only an error-prone time-consuming routine, but leaves you with almost no efficient way to complete the task.

## Standard Ways to Migrate Databases

There are several ways commonly used to move MySQL databases from one server to another.

They include:

### 1. Manual

Manual migrating is the most time-consuming and hard to use way. It perfectly suits migrating small databases. The advantage of manual migration is that you can fully control the process. Completing each step, you have time to evaluate the situation and decide how to handle this or that change. However, the value of this way is diminished by such shortcomings as huge amount of time and effort to spend and impossibility to synchronize databases of large sizes. As modern databases gain more complex structure and larger amounts of data, manual migrating of databases proves its disadvantage.

### 2. Semiautomatic

To partially automate database migrating, you can write scripts and use them to compare and synchronize databases. This method is quite accessible to many database specialists and many leverage it. You can retrieve metadata by using famous queries to `information_schema` or with the help of `SHOW` command. Data synchronization can be done through `INSERT..SELECT` statements. A most common example of semiautomatic database migration consists of three steps: first is to prepare a synchronization script with all the metadata and data of the source database, then dropping the target database and at last restoring it from the script. However, there are considerable complications that follow the method:

- In most cases, a special text comparison tool is required to compare metadata. A simple review of two texts requires titanic efforts to find the differences and not to miss a thing or two. Imagine comparing results of executed `SHOW CREATE TABLE` statement or any other common clauses barehanded. On the other hand, tools themselves require additional time and insight to use them efficiently. Usage of several tools to complete migration may bring some inconveniences.
- Data comparison is not a snap, and such tasks as replacing or updating parts of data practically cannot be done with the help of synchronization scripts.
- Unless you are lucky to have scripts written for all the cases, each particular database

migrating requires a new synchronization script. Even if you update the same database on a regular basis, you may need different scripts due to the type of changes made in the database.

### **3. Automatic**

There are many specific tools designed to replace manual comparison and synchronization of databases. Their capabilities, performance and efficiency are diverse. They usually compare source and target databases and display the results in the graphical interface convenient for analysis. You can easily exclude database objects from synchronization and apply multiple options to tune the process.

Among solid benefits such as canceling manual work, eliminating errors due to a human factor, reducing time and providing a correct result, the disadvantages are a high product price, a lack of required functionality, and much time and effort to use the tools efficiently.

## **Meeting Challenges with Devart Schema and Data Compare Tools**

Devart Schema and Data Compare tools are tailored to not only facilitate comparison and synchronization of database but also make each case of database migration a simple and predicted thing. Let us see how Devart tools can facilitate database migrating tasks in web site developing.

Suppose, during developing an internet shop (small\_shop) for some client, you should update the first version of the small\_shop web site with the new functionality and data. The first thing you should do before developing the second version of the web site is making a local copy of a staging database. You need to fully backup the database to a sql file. dbForge Studio for MySQL offers a quick and convenient way of doing backup files with Database Backup Wizard (see Figure 1) and then helps to restore databases via Database Restore Wizard.

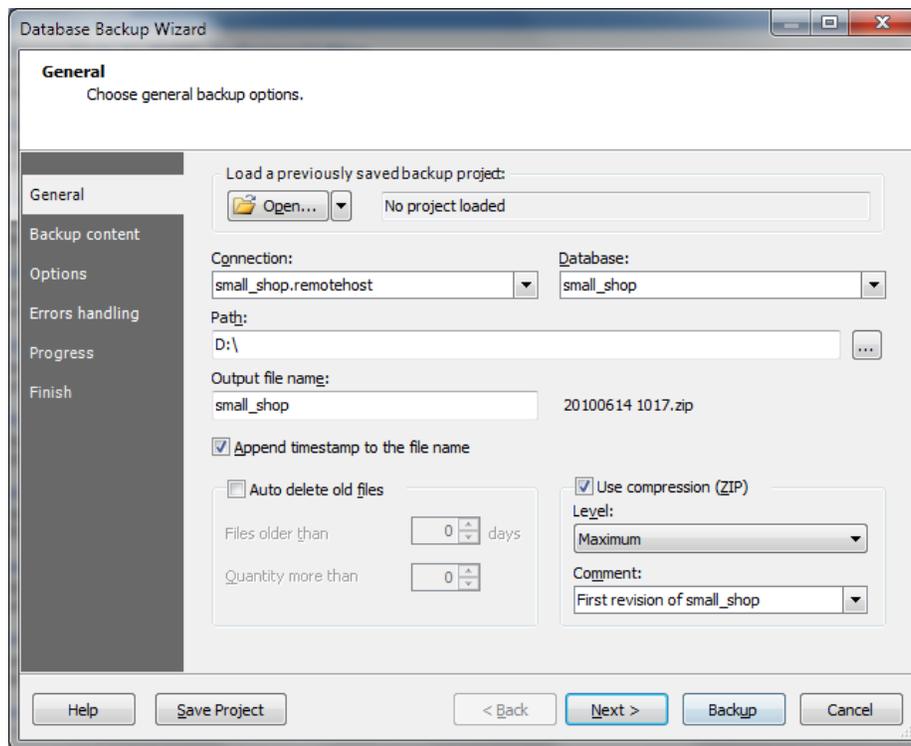


Figure 1: Database Backup Wizard

The first version of the small\_shop web site contains the following:

- customers – customers data
- navigation – a table describing the menu located in the left block on the web site
- pages – data about web site pages and their content
- products – products the e-shop sells
- orders – orders made by customers.

The changes in the second version include:

- *The news* table was added to store the e-shop news.
- The *navigation\_types* table was added to provide the navigation using not only the left menu, but the top one and the breadcrumbs. The table contains the *type\_id* column and the *navigation* table has a foreign key referring to the *navigation\_types* table
- Foreign keys referring to *customers* and *products* tables were added to the *orders* table.
- A new *address* field appeared in the *customers* table.
- The *customers\_products* view appeared to provide the list of customers and the products

they bought. Besides, the *get\_best\_customer* was added to let us choose the best customer in the e-shop.

- Some changes were made in the data of *navigation*, *news* and *pages* tables, the latter got *news* and *news\_detail* pages.

Now the task is to synchronize the structure and data changes in local and remote databases. While the second version of the web site was under development, the initial web site copy got new data: the list of products were extended, new clients as well as new orders were added. The bottleneck here is how to save the new data in the remote database and update it with the latest changes from the staging database. Simple restoring from the backup script with dropping the target database is no help in this case.

Before comparing data, the identical database structures are required. Schema Comparison Wizard (Figure 2), provided in dbForge Studio for MySQL, can synchronize database structures of any complexity in double-quick time. It takes only to select source and target databases for comparison and click the Compare button to get comparison with default settings. In our case the selected source and target are *small\_shop.localhost* (the local copy of the database) and *small\_shop.remotehost* (the database on the remote server). The comparison results appear in the neat grid (Figure 3).

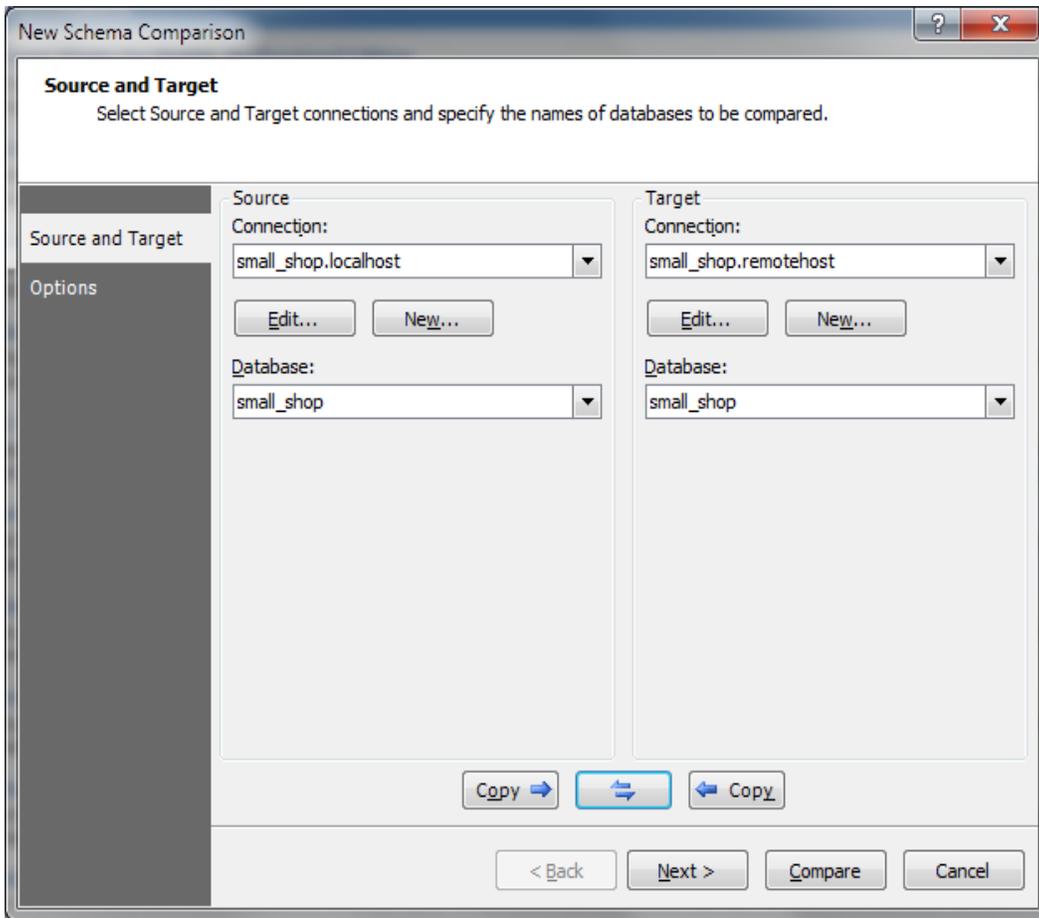


Figure 2: Schema Comparison Wizard with selected Source and Target

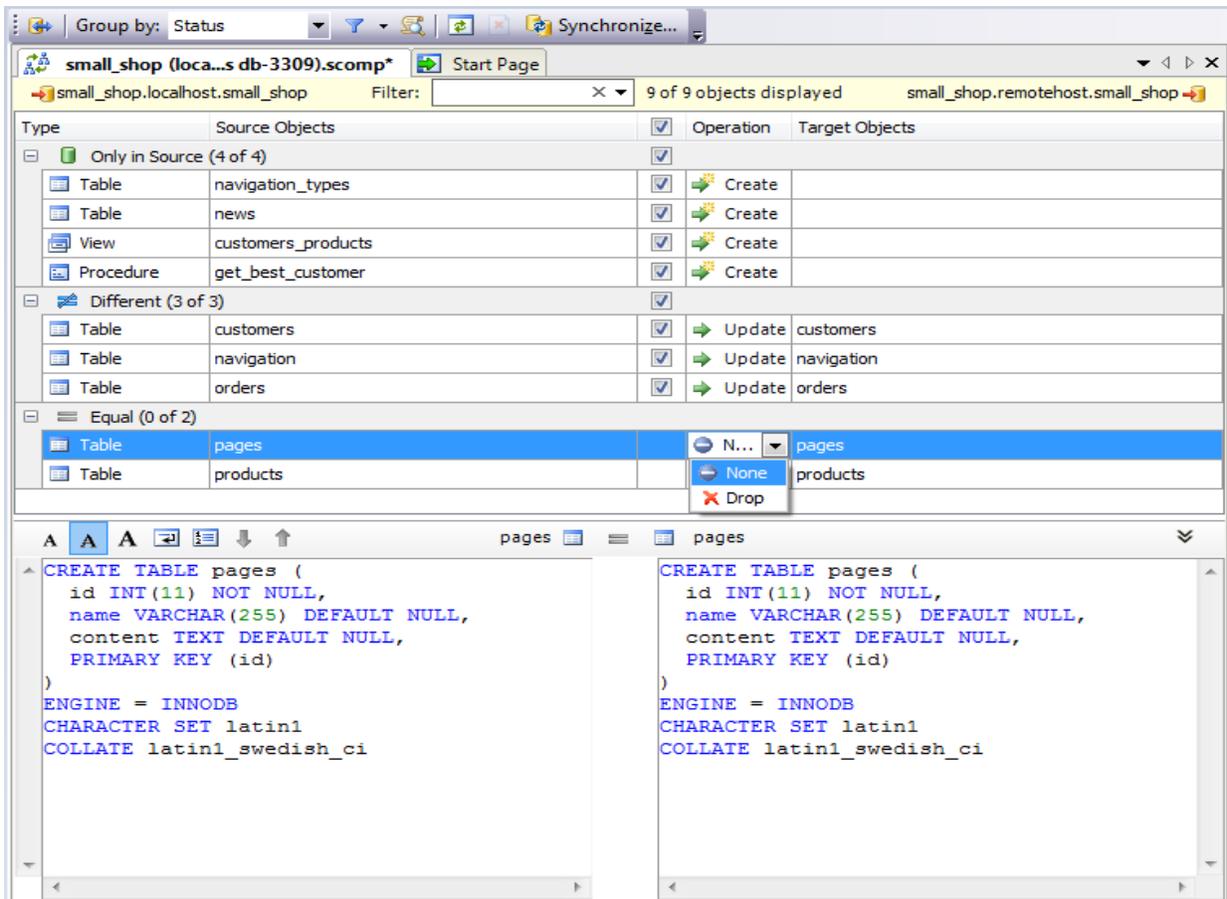


Figure 3: Comparison results of database schemas

The comparison results show four new objects, three different objects, which differences can be thoroughly reviewed in the text editors under the grid, and two objects with identical metadata. For partial synchronization, it is easy to exclude unnecessary objects by unselecting check boxes next to them. You can select required synchronization operation for each pair of objects by using the Operation column. For example, to drop an unnecessary object from the target database, you can select the Drop operation.

When you analyzed the results and made necessary operations based on your needs, it is time to press the Synchronize button and set up the synchronization in Schema Synchronization Wizard (Figure 4).

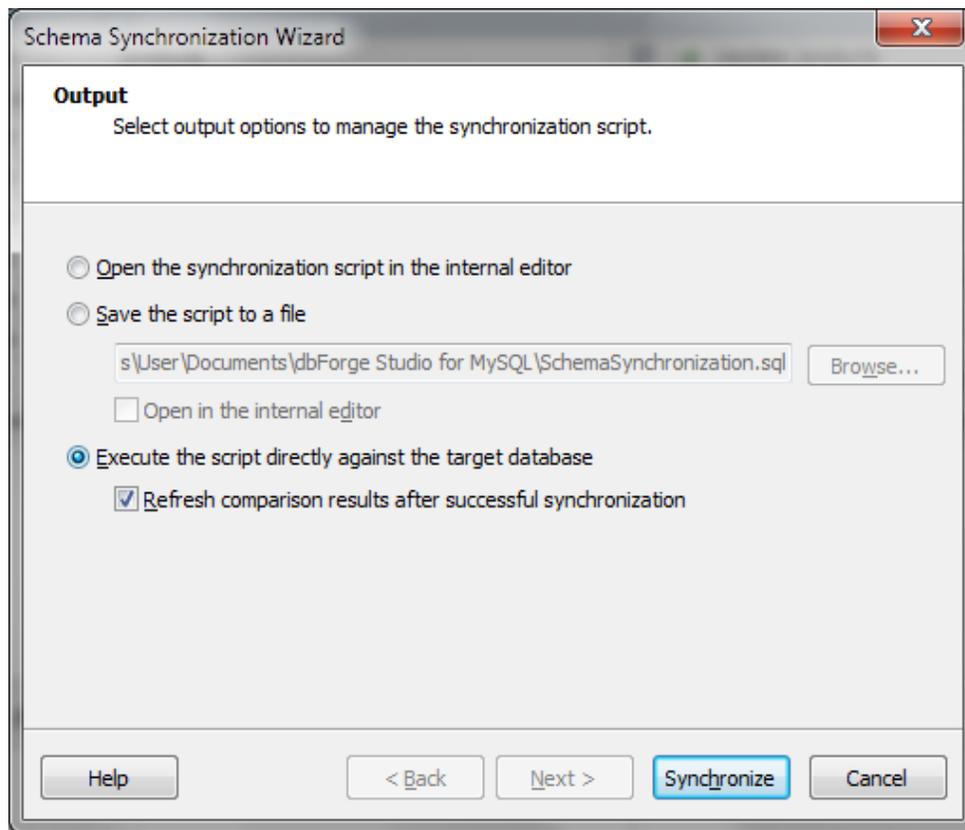


Figure 4: Schema Synchronization Wizard

It provides several types of synchronization. You can open the generated synchronization script in the script editor, save the script as a file for further review and execution, or just execute the script immediately after it is generated. Let us select the immediate synchronization and click Synchronize. Upon the synchronization, the schemas are re-compared and we can see the identical schemas (Figure 5).

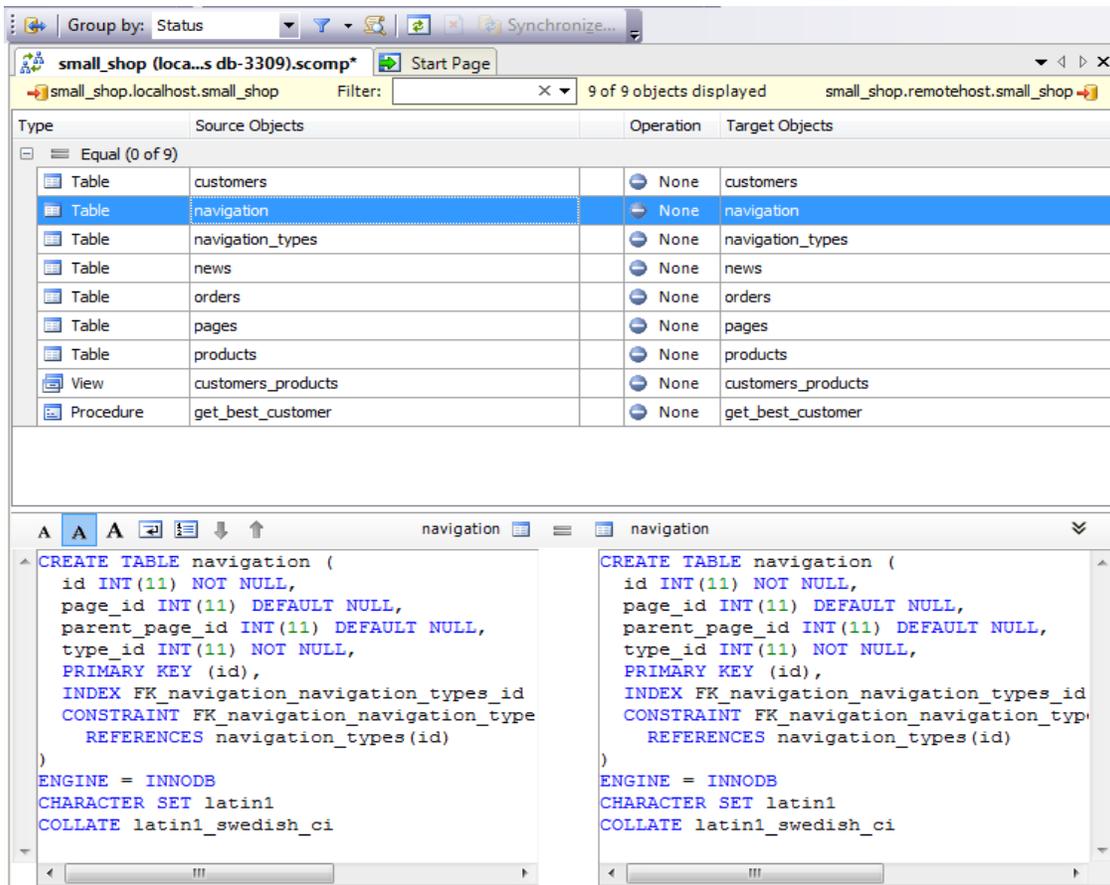


Figure 5: Schemas re-comparing after synchronization

Now it is time for convenient data synchronization with Data Comparison Wizard (Figure 6). It offers three groups of various settings to help you tune the following:

- Automatic mapping of objects – you can select whether to take into account case, spaces or ignore some properties.
- Comparing objects – you can define what objects to be compared and what should be ignored.
- Displaying comparison results – these options definitely simplify your work while analyzing the differences and show only that information you need.

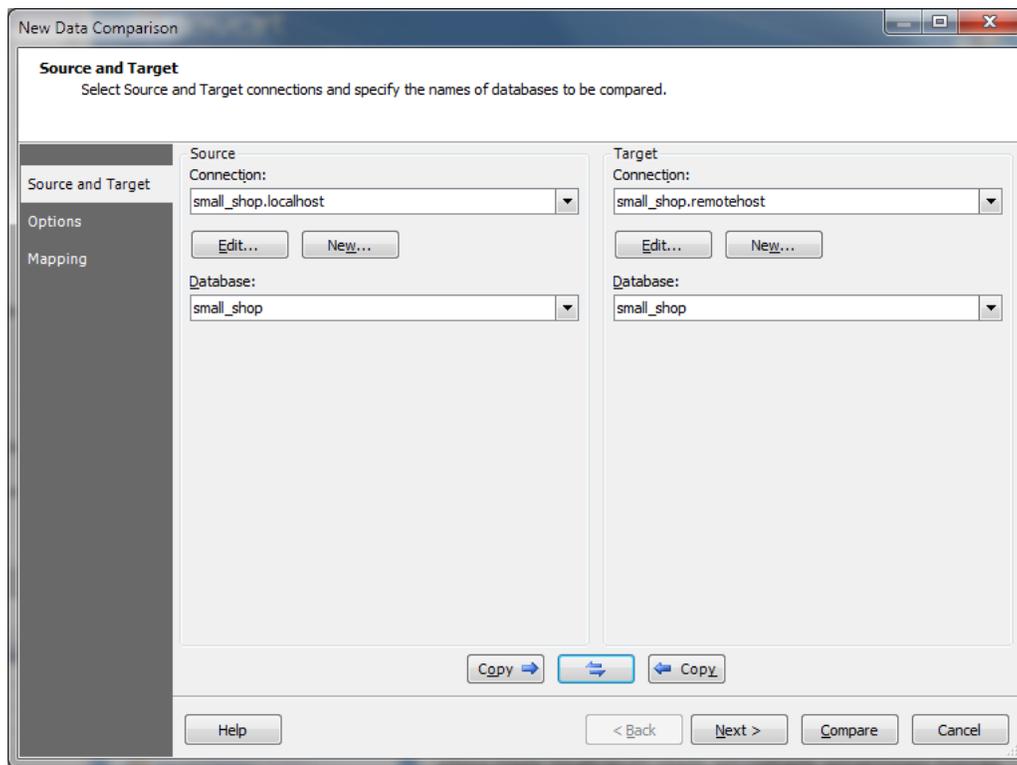


Figure 6: Data Comparison Wizard

As we do not want to see identical records in our case, let us just exclude them from displaying. Besides automatic mapping of objects, the wizard lets you do some manual mapping. It is very essential for cases when automatic mapping does not suit your needs, for example, you want to map schemas with different names. The Mapping wizard page (Figure 7) gives you freedom to set comparison keys by selecting existing or custom ones; select columns for comparison, apply SQL filter for each object, exclude unnecessary objects from comparison, and map those objects that were not automatically mapped.

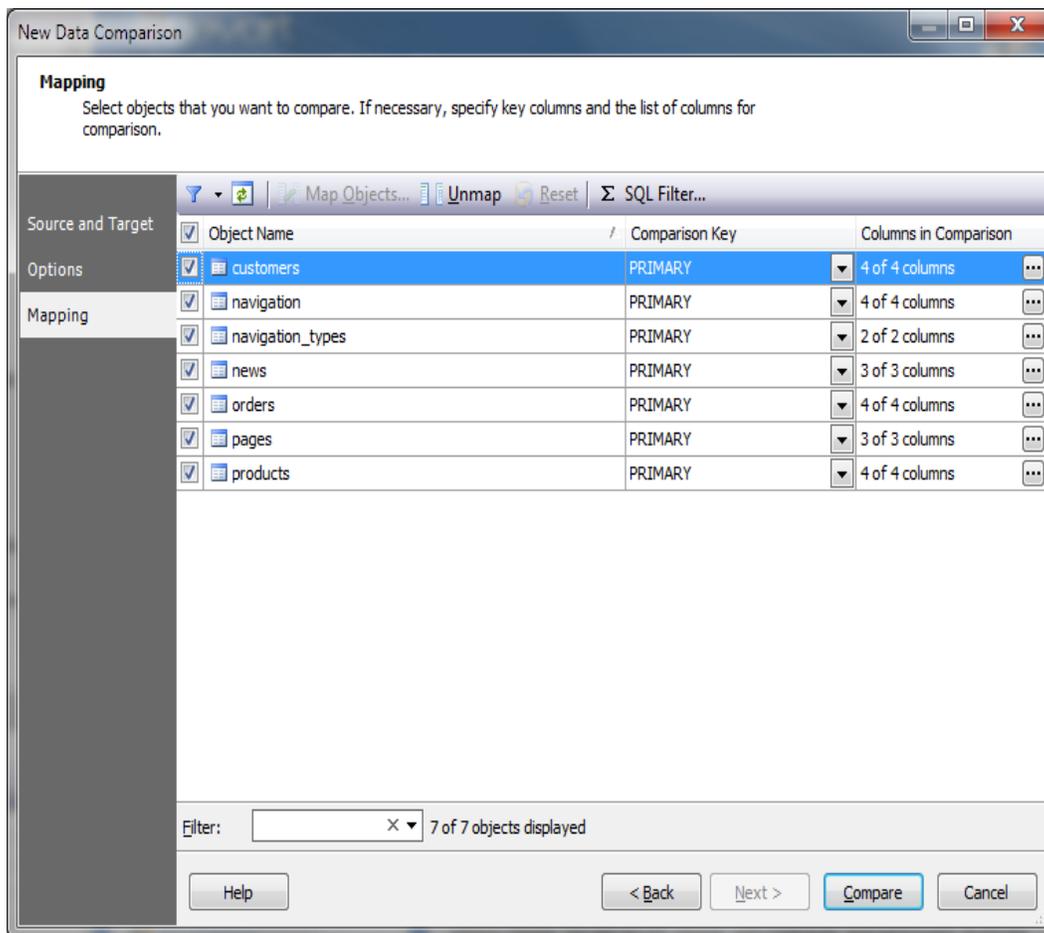


Figure 7: Mapping in Data Comparison Wizard

When the comparison is set up, we press the Compare button and get the data comparison results in the convenient grid (Figure 8).

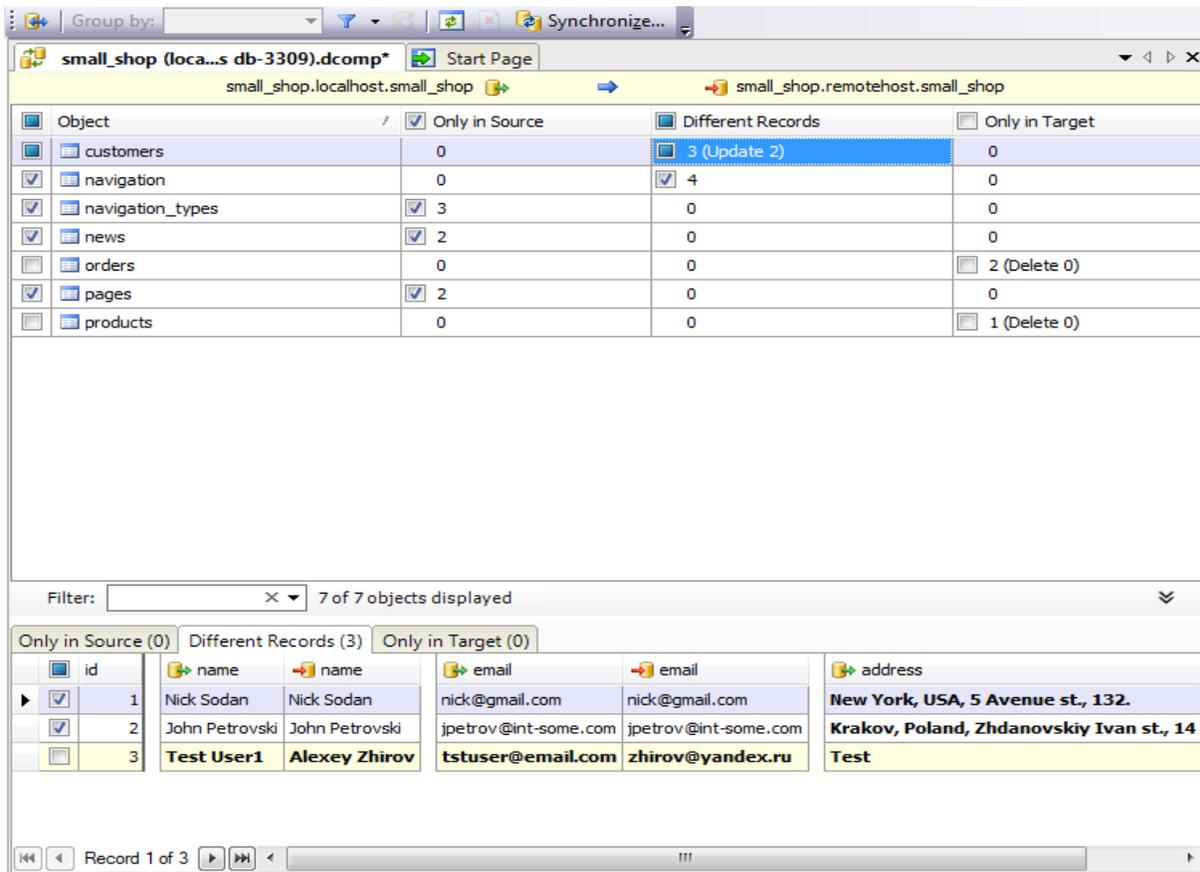


Figure 8: Data comparison results

The results include:

- New data (Only in Source) that exists only in the local copy of the database.
- Data existing in the remote database (Only In Target). It can be new data or some data deleted in the local database. To preserve the new data on the remote server, we can exclude it from the synchronization.
- Modified data in *customers* and *navigation* tables. As it is shown, the changes in record with id 3 record contradict each other. The record in the remote database contains real data of a new customer, meanwhile in the local database the record has some working data used for testing new functionality. To keep the real data, we exclude the record with test data from synchronization, but all the rest records will be synchronized as a new *address* column has been added.

When the comparison results are ready for synchronization, we open Data Synchronization Wizard (Figure 9) by selecting the Synchronize button on the toolbar.

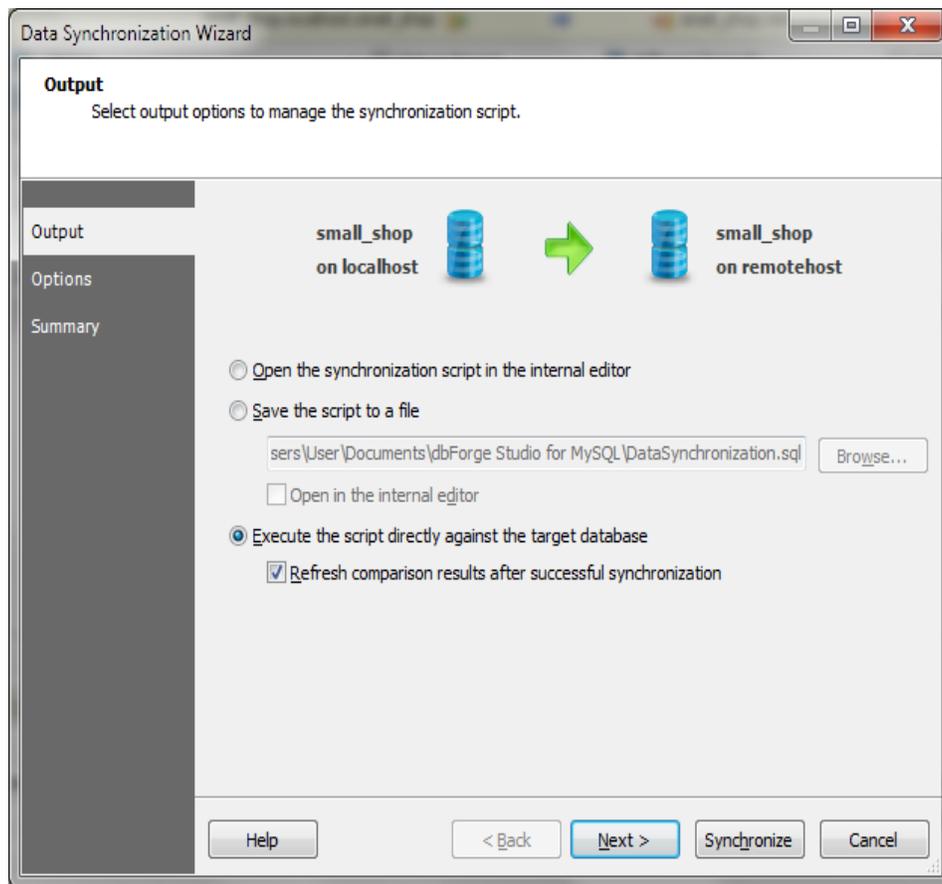


Figure 9: Data Synchronization Wizard

With the wizard, you have a choice of what to do with the synchronization script (whether to see it now, save for further review or execute immediately), apply synchronization options, e.g., to turn off foreign keys, and see the execution plan along with the list of warnings and notifications if any are generated. You have the complete control and insight what the result you will get.

To synchronize the data, it takes only to press Synchronize in the wizard and check the re-compared data (Figure 10) to ensure we have the desired result. We have all the selected records successfully synchronized. The records selected to preserve in the remote database are preserved. The result is the same as expected, with little effort, time and errors.

The screenshot shows a synchronization tool interface with the following data:

Object	Only in Source	Different Records	Only in Target
customers	0	1	0
navigation	0	0	0
navigation_types	0	0	0
news	0	0	0
orders	0	0	2
pages	0	0	0
products	0	0	1

id	name	name	email	email	address	address
3	Test User1	Alexey Zhiron	tstuser@email.com	zhiron@yandex.ru	Test	(null)

Figure 10: Data re-comparing after synchronization

Imagine the benefits you may get while dealing with more complex database structures, larger data and specific requirements of your database migrating cases.

## Conclusion

In the fast-developing world of databases, database migrating is a business necessity. As databases grow in size and gain more complex structures, migrating failures become costly and put the companies' prosperity at a big risk. Without reliable and speedy tools, it is hard to meet new business challenges, be efficient and safe. The modern market offers enough tools, varying in functionality, efficiency and price.

As each company faces the choice, the rational way to find the perfect tool is to define what database migration tasks the tool should complete, what kind of databases and data will be involved, and what critical functionality the company wants to see. This will narrow the selection and indicate the very tool that will suit the specific needs.

## Additional Resources

Articles on forward engineering with MySQL Workbench:

<http://dev.mysql.com/doc/workbench/en/wb-forward-engineering.html>

Schema and Data Compare Tools by Devart

<http://www.devart.com/dbforge/mysql/schemacompare/>

<http://www.devart.com/dbforge/mysql/datacompare/>

Automated MySQL Data Comparison and Synchronization: How It Works:

<http://www.devart.com/blogs/dbforge/?p=541>

Data Comparison Methods Overview

<http://www.devart.com/blogs/dbforge/?p=1056>

Guide to Schema Synchronization with MySQL Workbench:

<http://wb.mysql.com/?p=116>

## 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](http://www.devart.com)