MySQL Workbench

Abstract

This is the MySQL™ Workbench Reference Manual. It documents the MySQL Workbench Community and MySQL Workbench commercial editions 6.3 through 6.3.7.

If you have not yet installed MySQL Workbench Community please download your free copy from the download site. MySQL Workbench Community is available for Windows, OS X, and Linux.

For notes detailing the changes in each release, see the MySQL Workbench Release Notes.

For legal information, see the Legal Notices.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists, where you can discuss your issues with other MySQL users.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Document generated on: 2016-11-04 (revision: 49755)

Table of Contents

Preface and Legal Notices

This is the User Manual for the MySQL Workbench.

Licensing information. This product may include third-party software, used under license. If you are using MySQL Workbench commercial, see this document for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using MySQL Workbench Community, see this document for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Chapter 1 General Information

Table of Contents

This chapter provides general information about MySQL Workbench and how it has changed.

MySQL Workbench is a graphical tool for working with MySQL Servers and databases. MySQL Workbench fully supports MySQL Server versions 5.1 and above. It is also compatible with MySQL Server 5.0, but not every feature of 5.0 may be supported. It does not support MySQL Server versions 4.x.

MySQL Workbench functionality covers five main topics:

- **SQL Development**: Enables you to create and manage connections to database servers. Along with enabling you to configure connection parameters, MySQL Workbench provides the capability to execute SQL queries on the database connections using the built-in SQL Editor.
- **Data Modeling (Design)**: Enables you to create models of your database schema graphically, reverse and forward engineer between a schema and a live database, and edit all aspects of your database using the comprehensive Table Editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.
- **Server Administration**: Enables you to administer MySQL server instances by administering users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the MySQL server performance.
- **Data Migration**: Allows you to migrate from Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL, and other RDBMS tables, objects and data to MySQL. Migration also supports migrating from earlier versions of MySQL to the latest releases.
- **MySQL Enterprise Support**: Support for Enterprise products such as MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit.

MySQL Workbench is available in two editions, the Community Edition and the Commercial Edition. The Community Edition is available free of charge. The Commercial Edition provides additional Enterprise features, such as access to MySQL Enterprise Backup, MySQL Firewall, and MySQL Audit. For a complete comparison, see http://www.mysql.com/products/workbench/features.html

For notes detailing the changes in each release, see the MySQL Workbench Release Notes.

1.1 What Is New in MySQL Workbench 6

This section summarizes how the MySQL Workbench 6 series progressed with each minor release.

For notes detailing the changes in each point release, see the MySQL Workbench Release Notes.

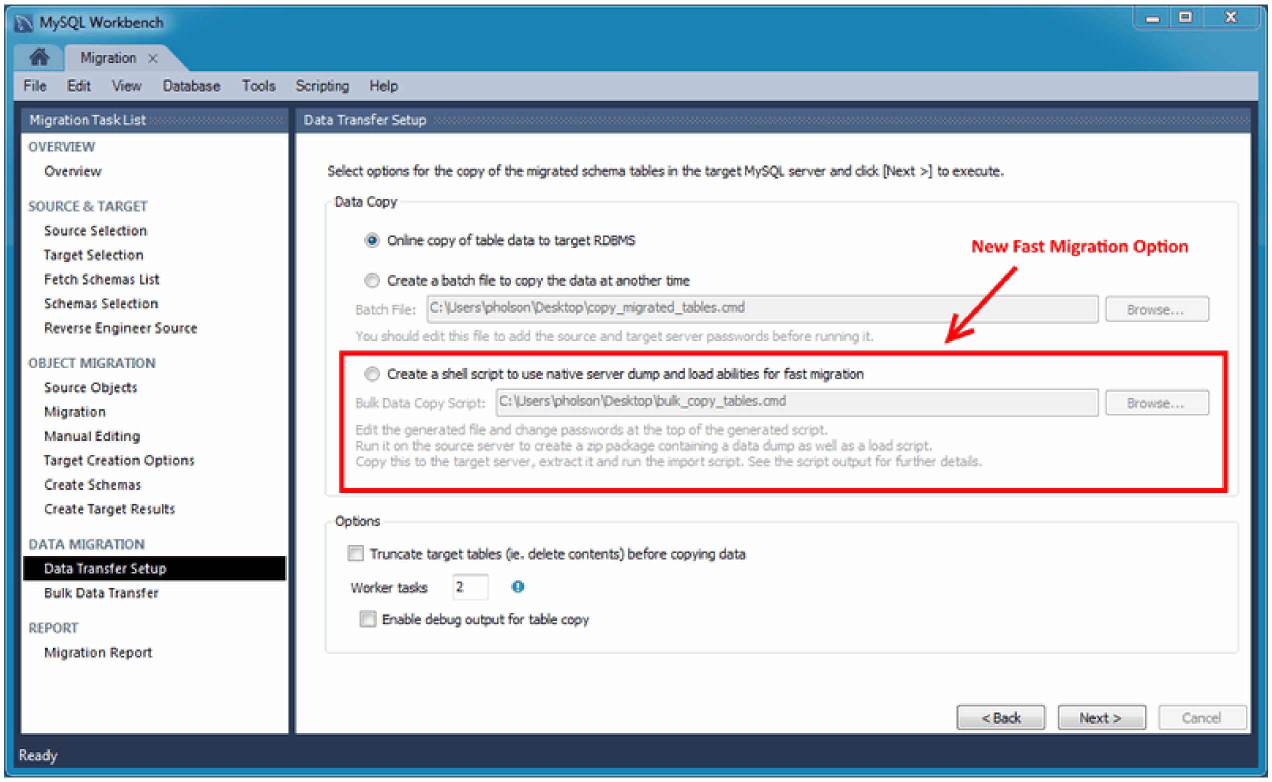
1.1.1 New in MySQL Workbench 6.3

This section summarizes many of the new features added to MySQL Workbench 6.3.x, in relation to MySQL Workbench 6.2.x;.

Fast Data Migration

A new "fast migration" option was added to the migration wizard. This is another way to transfer data from one MySQL server to another while performing a migration, and it complements the existing solutions. The premise is to use a generated script on the source server to create a dump that you move to the target machine to perform the import there. This avoids the need to traffic all data through MySQL Workbench, or to have a permanent network connection between the servers. Instead, the dump and restore is performed at maximum speed by using the LOAD DATA call for the MySQL import. The migration wizard automatically creates all necessary scripts for all supported platforms and servers. The generated script creates a self-contained Zip file that must be copied to the target server. You unzip it and execute the provided script to perform the data import.

Figure 1.1 Data Transfer Setup: New Fast Migration Option



Select options for the copy of the migrated schema tables in the target MySQL server and click [Next >] to execute.

Data Copy

- [✓] Online copy of table data to target RDBMS
- Create a batch file to copy the data at another time

Batch File: C:\Users\pholson\Desktop\copy_migrated_tables.cmd Browse...

You should edit this file to add the source and target server passwords before running it.

- Create a shell script to use native server dump and load abilities for fast migration

Bulk Data Copy Script: C:\Users\pholson\Desktop\bulk_copy_tables.cmd Browse...

Edit the generated file and change passwords at the top of the generated script.
Run it on the source server to create a zip package containing a data dump as well as a load script.
Copy this to the target server, extract it and run the import script. See the script output for further details.

Options

- Truncate target tables (ie. delete contents) before copying data

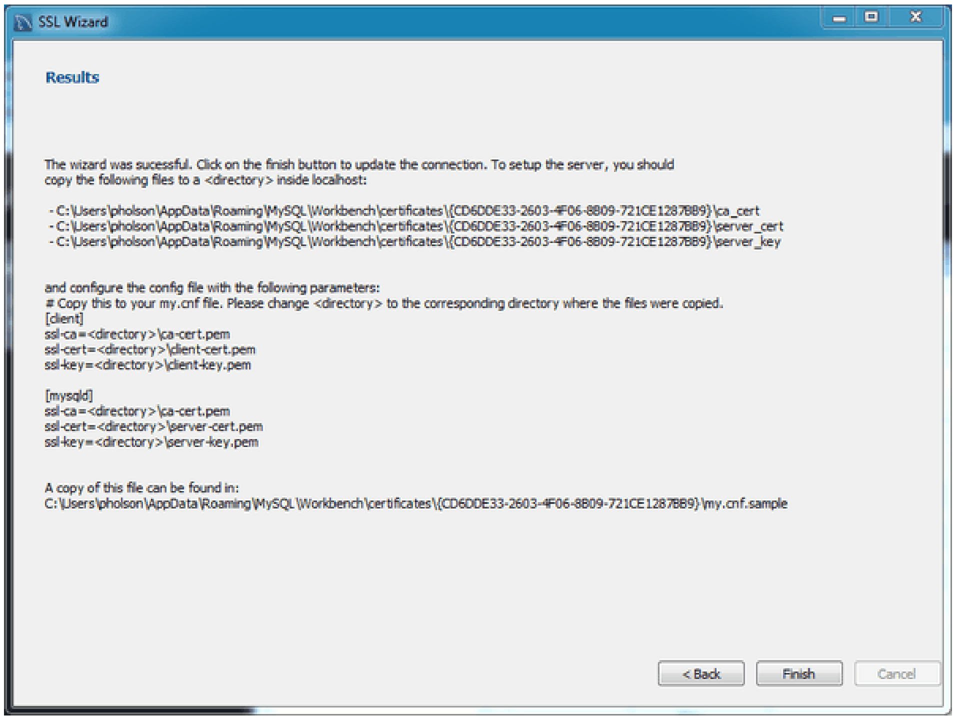
Worker tasks 2

- Enable debug output for table copy

SSL Certificate Generator

A new SSL certificate generation wizard was added. This new wizard helps create proper SSL certificates for both MySQL clients and MySQL servers. Connections in MySQL Workbench are updated with the certificates by the wizard. This wizard requires OpenSSL to create the certificates. An example `my.cnf` / `my.ini` file is also generated that utilizes the generated certificates.

Figure 1.2 SSL Certificate Wizard



```
The wizard was sucessful. Click on the finish button to update the connection. To setup the server, you should
copy the following files to a <directory> inside localhost:

 - C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\ca_cert
 - C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\server_cert
 - C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\server_key

and configure the config file with the following parameters:
# Copy this to your my.cnf file. Please change <directory> to the corresponding directory where the files were copied.
[client]
ssl-ca=<directory>\ca-cert.pem
ssl-cert=<directory>\client-cert.pem
ssl-key=<directory>\client-key.pem

[mysqld]
ssl-ca=<directory>\ca-cert.pem
ssl-cert=<directory>\server-cert.pem
ssl-key=<directory>\server-key.pem

A copy of this file can be found in:
C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\my.cnf.sample
```

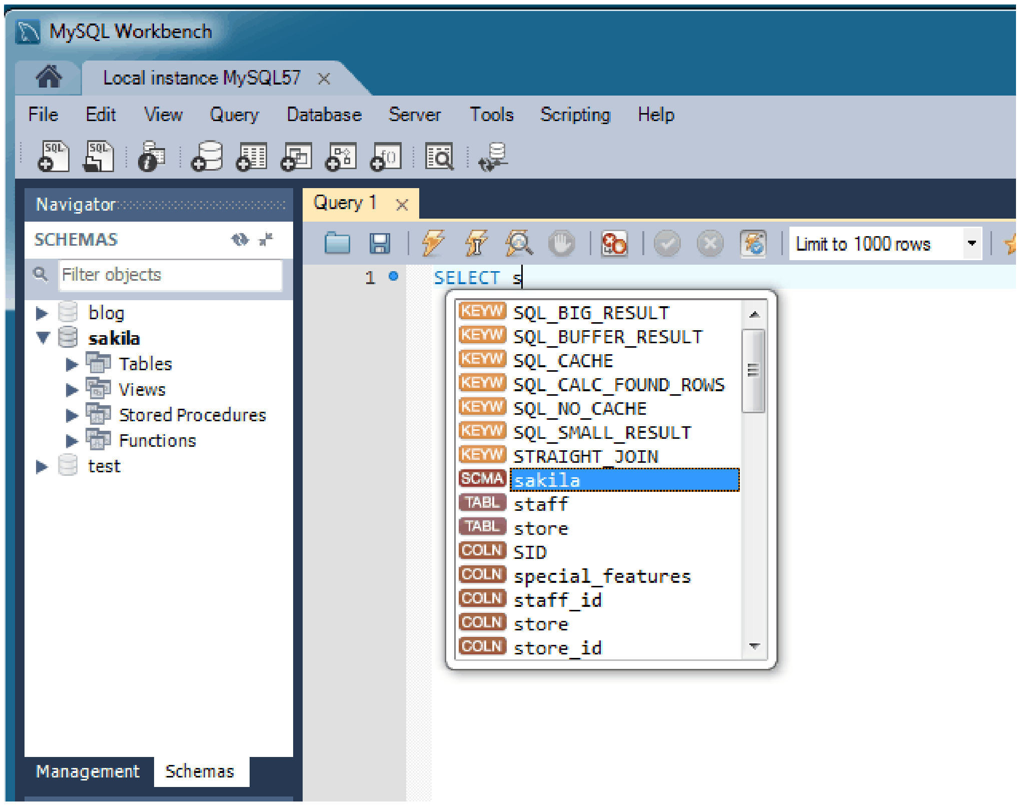
For additional details, see Section 5.3.4, “SSL Wizard (Certificates)”.

SQL Editor Auto-Completion

The SQL editor auto-completion improvements include the following changes:

- It now functions with all statement types, when before only SELECT statements were fully supported.
- It now minds the MySQL server version. For example, it now only shows the engines available from the server.
- Additional suggestions are now available, such as system variables, engines, table spaces, logfile groups, and more.
- New graphics including color coded (and tagged) entries.
- It is context aware, as for example it only shows available keywords, columns, and tables.
- Improved MySQL 5.7 syntax support.

Figure 1.3 SQL Editor Auto-Completion



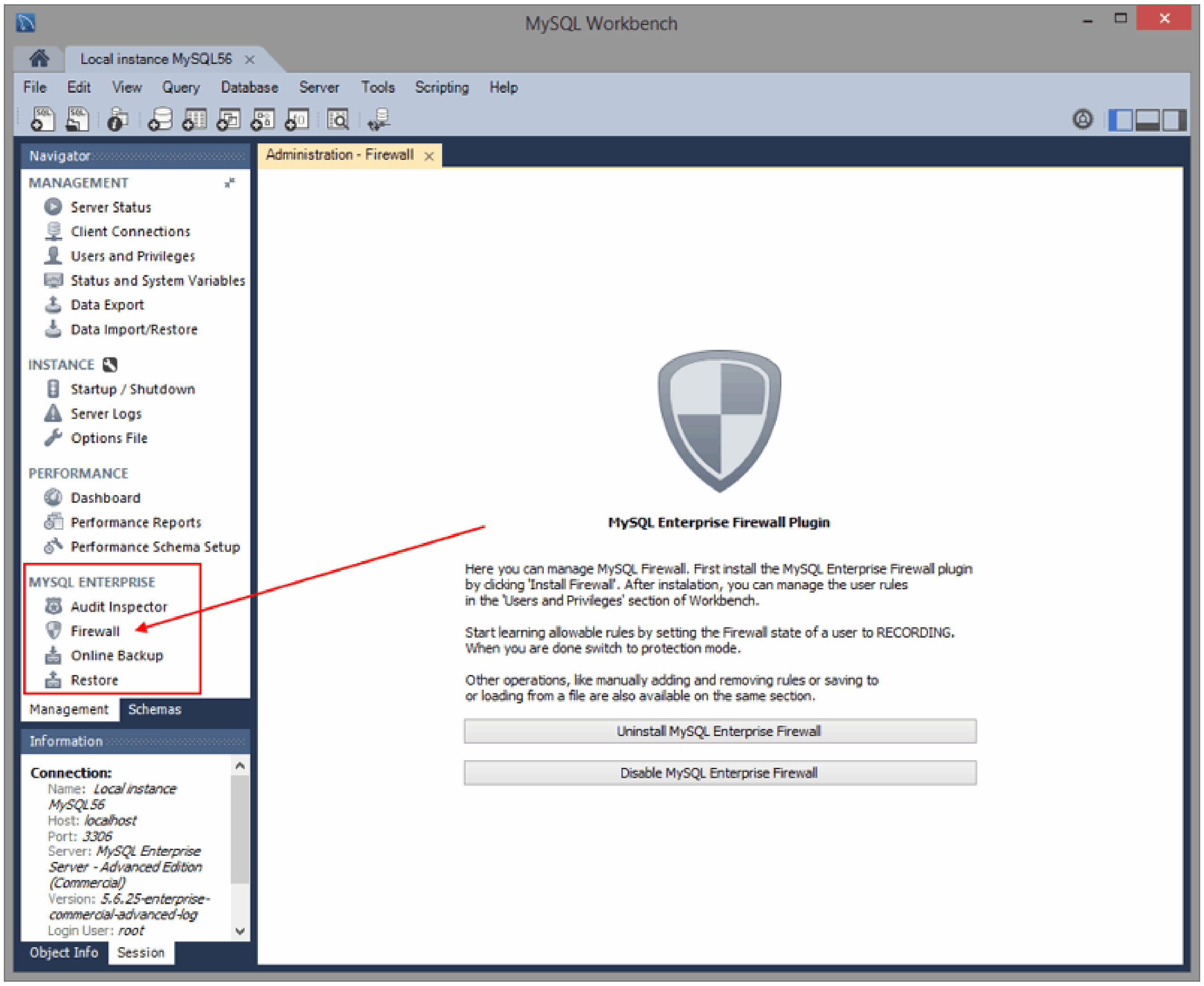
```
SELECT s
```

```
KEYW SQL_BIG_RESULT
KEYW SQL_BUFFER_RESULT
KEYW SQL_CACHE
KEYW SQL_CALC_FOUND_ROWS
KEYW SQL_NO_CACHE
KEYW SQL_SMALL_RESULT
KEYW STRAIGHT_JOIN
SCMA sakila
TABL staff
TABL store
COLN SID
COLN special_features
COLN staff_id
COLN store
COLN store_id
```

MySQL Enterprise Firewall

MySQL Enterprise Firewall support was added in MySQL Workbench 6.3.4. Use MySQL Workbench to install and enable MySQL Enterprise Firewall, and manage the MySQL Enterprise Firewall rules and variables. For additional information, see Section 6.8, “MySQL Enterprise Firewall Interface”.

Figure 1.4 MySQL Enterprise Firewall: Install / Enable



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*

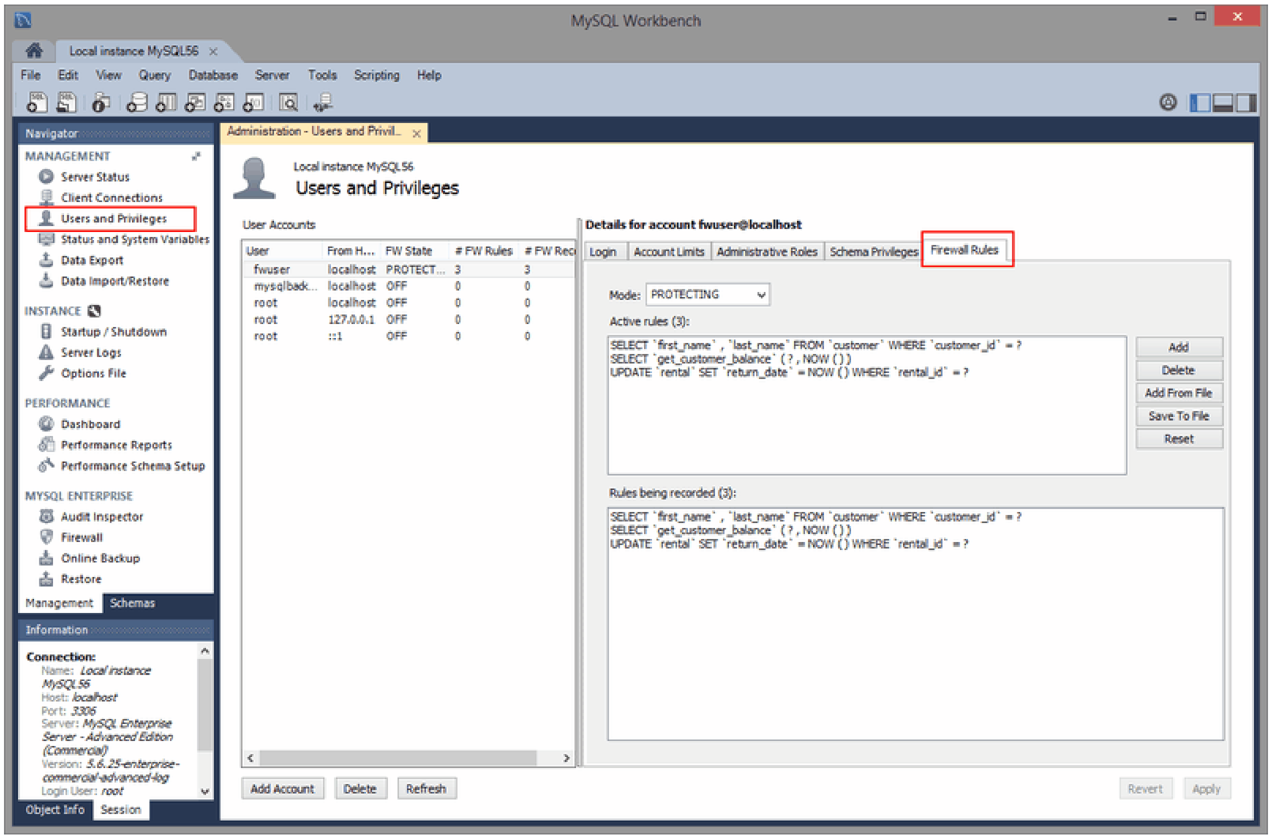
MySQL Enterprise Firewall Plugin

Here you can manage MySQL Firewall. First install the MySQL Enterprise Firewall plugin by clicking 'Install Firewall'. After instalation, you can manage the user rules in the 'Users and Privileges' section of Workbench.

Start learning allowable rules by setting the Firewall state of a user to RECORDING. When you are done switch to protection mode.

Other operations, like manually adding and removing rules or saving to or loading from a file are also available on the same section.

Figure 1.5 MySQL Enterprise Firewall Rules



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*

User Accounts

User	From H...	FW State	# FW Rules	# FW Rec
fwuser	localhost	PROTECT...	3	3
mysqlback...	localhost	OFF	0	0
root	localhost	OFF	0	0
root	127.0.0.1	OFF	0	0
root	::1	OFF	0	0

Details for account fwuser@localhost

Mode: PROTECTING

Active rules (3):

```
SELECT `first_name` , `last_name` FROM `customer` WHERE `customer_id` = ?
SELECT `get_customer_balance` ( ? , NOW ( ) )
UPDATE `rental` SET `return_date` = NOW ( ) WHERE `rental_id` = ?
```

Rules being recorded (3):

```
SELECT `first_name` , `last_name` FROM `customer` WHERE `customer_id` = ?
SELECT `get_customer_balance` ( ? , NOW ( ) )
UPDATE `rental` SET `return_date` = NOW ( ) WHERE `rental_id` = ?
```

MySQL Enterprise Backup

Profile handling now detects mismatches between MySQL Enterprise Backup executables and corresponding profiles.

Improved scheduling logic

Table Data Export and Import Wizard

A new table data import/export wizard was added. This feature enhances the current CSV import and export feature found in the SQL editor's result set viewer. It supports import and export of CSV and JSON files, and allows a more flexible configuration (separators, column selection, encoding selection, and more). This new wizard does not require an executed statement on a table for a result set to be operated on, as it can now work directly on tables. The wizard can be performed against either a local or remotely connected MySQL server. The import action includes table, column, and type mapping. For additional information, see Section 6.5.1, “Table Data Export and Import Wizard”.

The wizard is accessible from the object browser's context menu.

Figure 1.6 Table Data Import/Export Wizard Menu

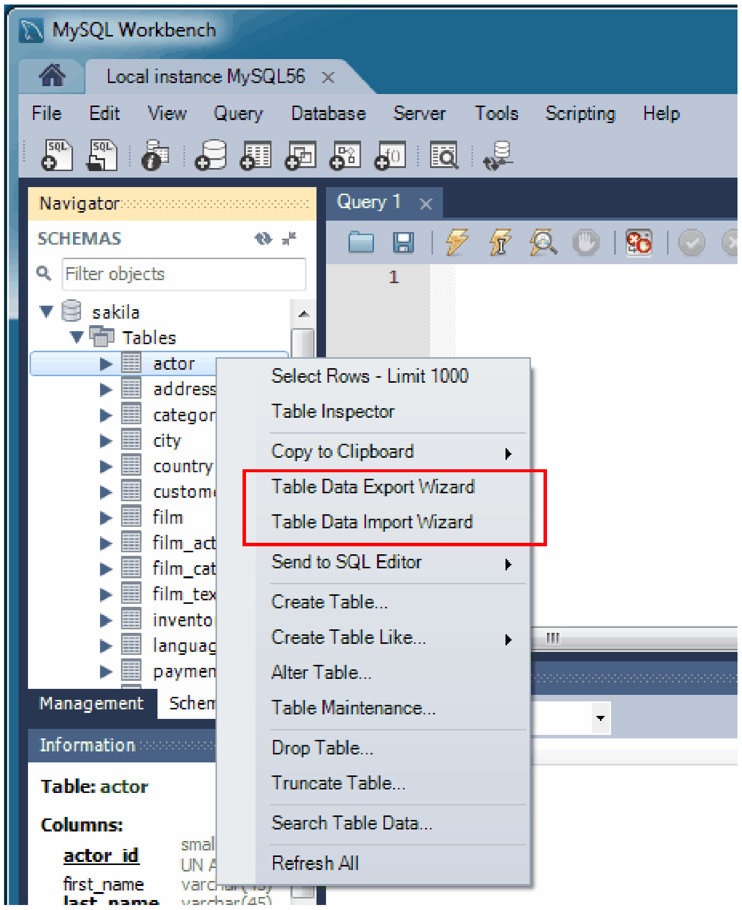
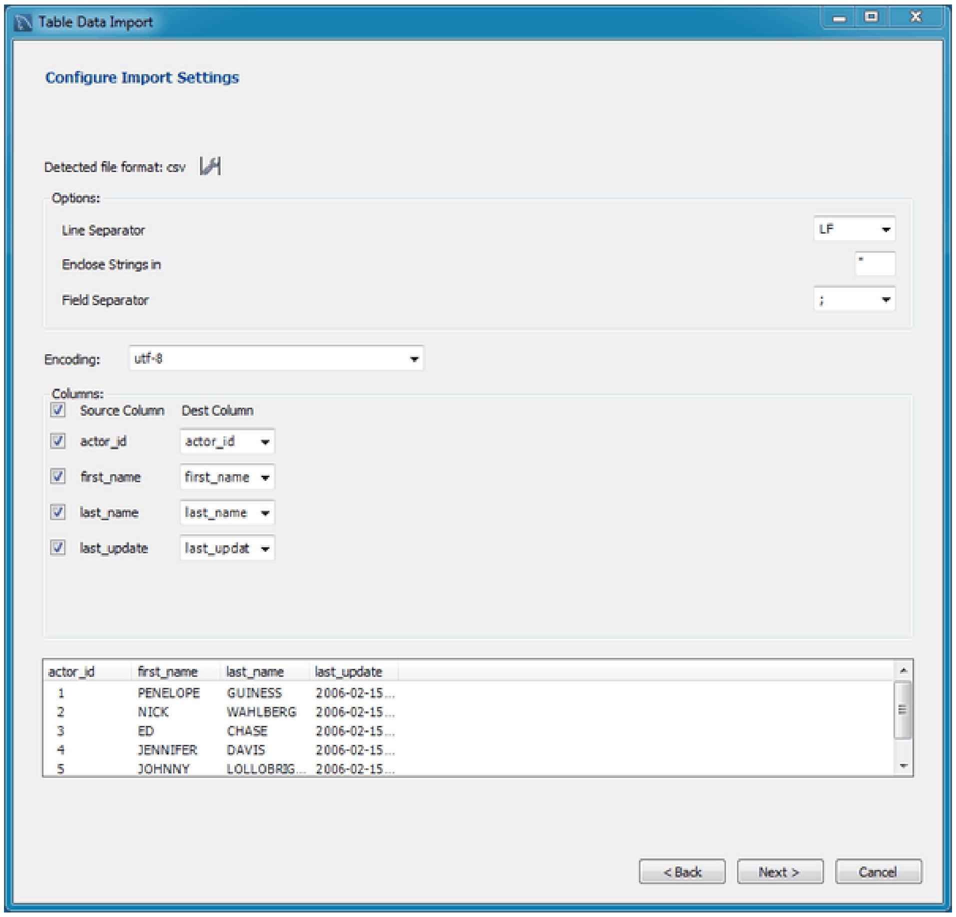


Figure 1.7 Table Data Import/Export Wizard CSV Configure



Detected file format: csv

Options:

Line Separator: LF

Enclose Strings in: "

Field Separator: ;

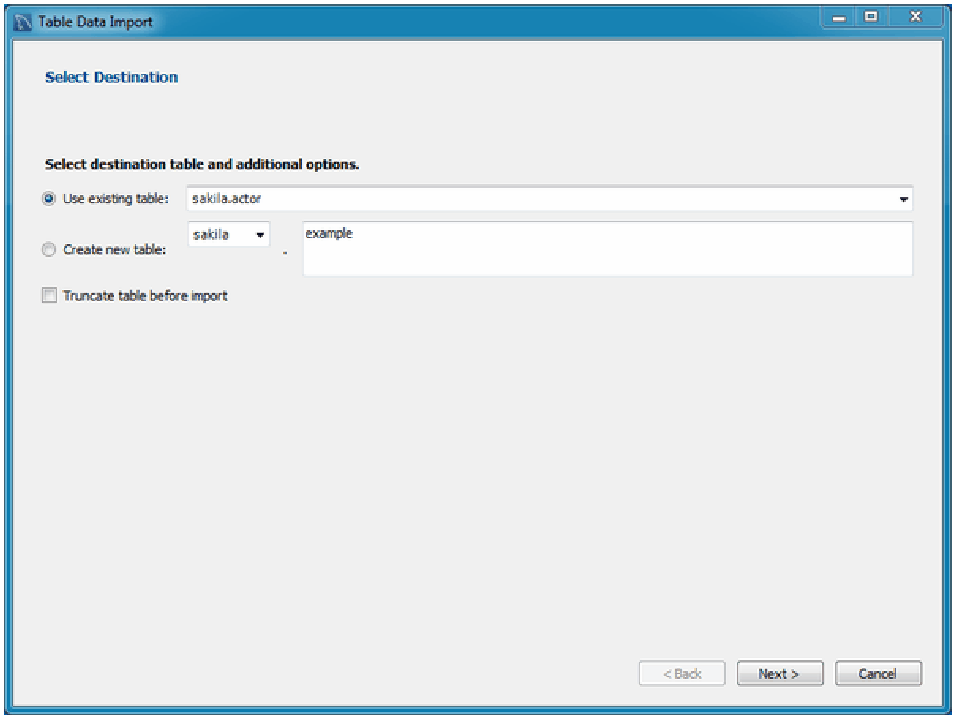
Encoding: utf-8

Columns:

Source Column	Dest Column
✓ actor_id	actor_id
✓ first_name	first_name
✓ last_name	last_name
✓ last_update	last_updat

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15...
2	NICK	WAHLBERG	2006-02-15...
3	ED	CHASE	2006-02-15...
4	JENNIFER	DAVIS	2006-02-15...
5	JOHNNY	LOLLOBRIG...	2006-02-15...

Figure 1.8 Table Data Import/Export Wizard Options



Additional changes

MySQL Fabric 1.5 is now supported. Older versions of Fabric are no longer supported due to incompatible protocol changes.

OS X builds were switched from 32-bit to 64-bit.

Platforms support changes: 6.3.0: Fedora 21 and Ubuntu 14.10 support was added, Ubuntu 12.10 support was dropped. 6.3.4: Fedora 22 and Ubuntu 15.04 support was added, Ubuntu 14.10 support was dropped.

1.1.2 New in MySQL Workbench 6.2

This section summarizes many of the new features added to MySQL Workbench 6.2.x, in relation to MySQL Workbench 6.1.x;.

SQL Editor

Most of the changes and improvements were made to the SQL editor.

Overlay Icons in the Object Viewer

The schema navigator now includes shortcut buttons for common operations such as table data view, the table editor, and the table/schema inspector.

Figure 1.9 Object Viewer Overlay Icons

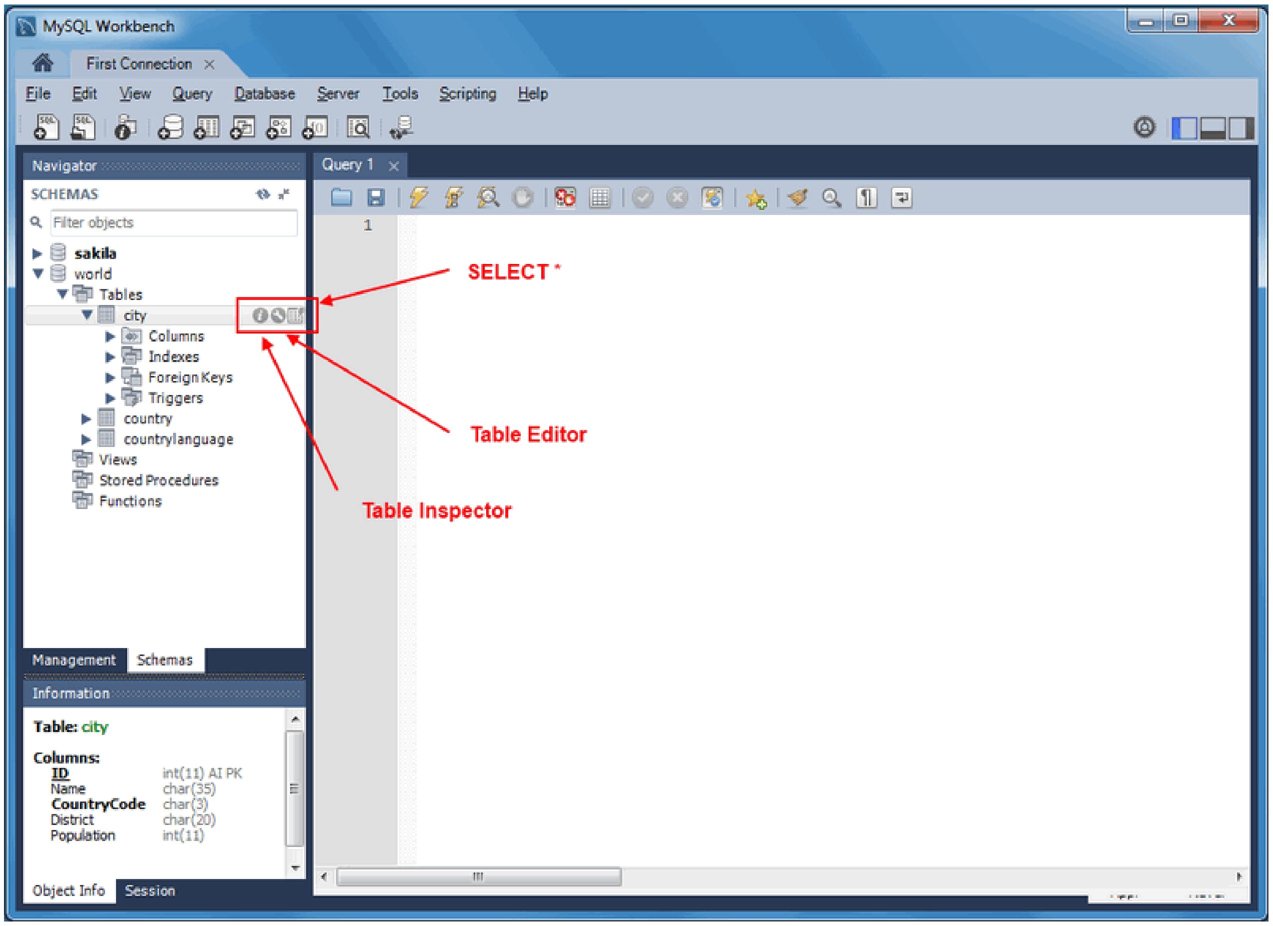


Table: city

Columns:

ID	int(11) AI PK
Name	char(35)
CountryCode	char(3)
District	char(20)
Population	int(11)

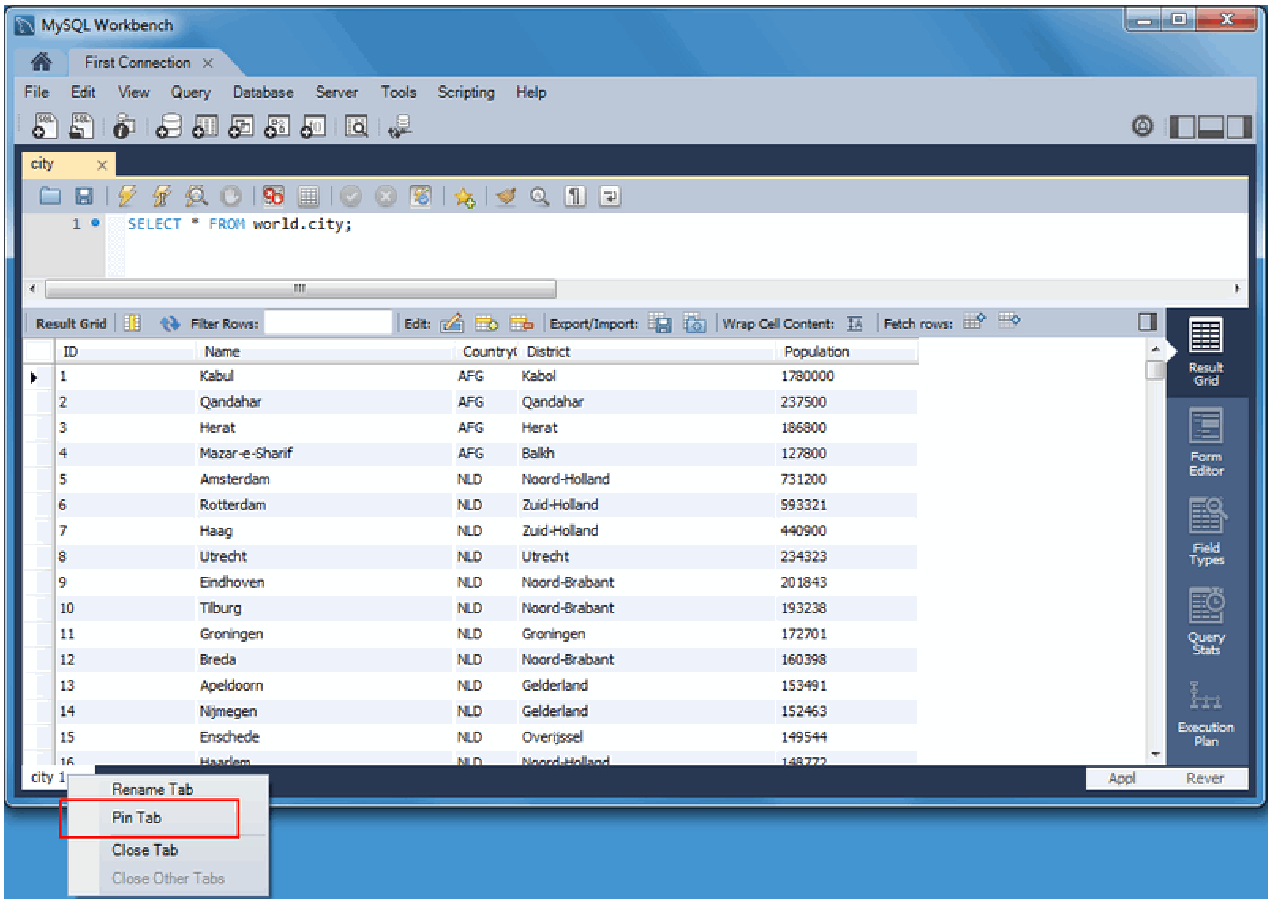
A "Pin Tab" Results Option

Result tabs can now be "pinned" to your result set window.

Note

The "Rename Tab" context menu option is also new. New names are preserved (and remembered) in your Workbench's `cache/` directory.

Figure 1.10 Pin Tab



```
SELECT * FROM world.city;
```

ID	Name	Country(	District	Population
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772

Microsoft Access to MySQL Migration

The migration wizard now supports Microsoft Access migration. Select "Microsoft Access" as your source database in the wizard, use MySQL as your target source database, and then execute. For additional information, see Section 10.4, “Microsoft Access Migration”.

MySQL Fabric Integration

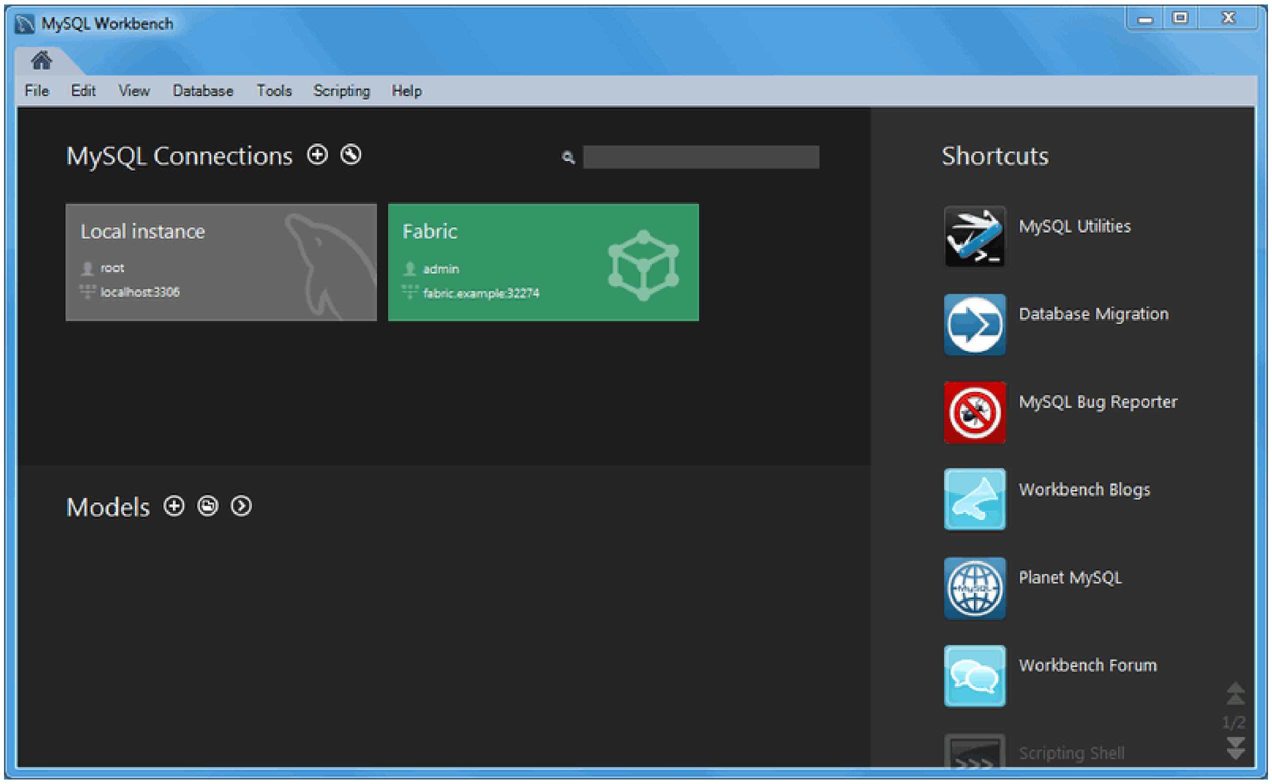
MySQL Fabric cluster connectivity was added: Browse, view status, and connect to any MySQL instance in a Fabric Cluster.

Note

This requires Connector/Python and MySQL Utilities 1.4.3+ installed, including the Python module.

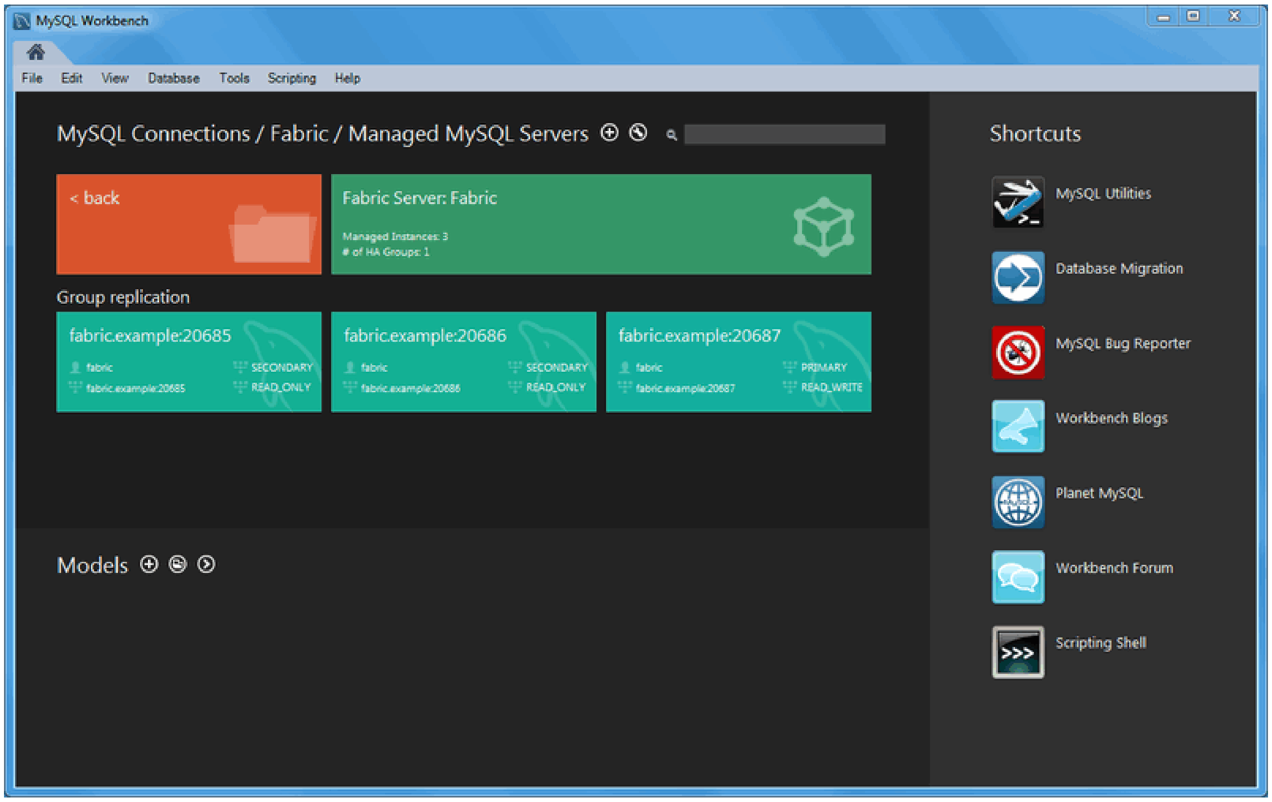
To set up a managed Fabric connection, create a new MySQL connection with the new **MySQL Fabric Management Node** connection method. The connection tiles have a different look:

Figure 1.11 Fabric Connection Group Tile



Clicking the new fabric group tile shows the managed connections:

Figure 1.12 Fabric Connection Group Tiles

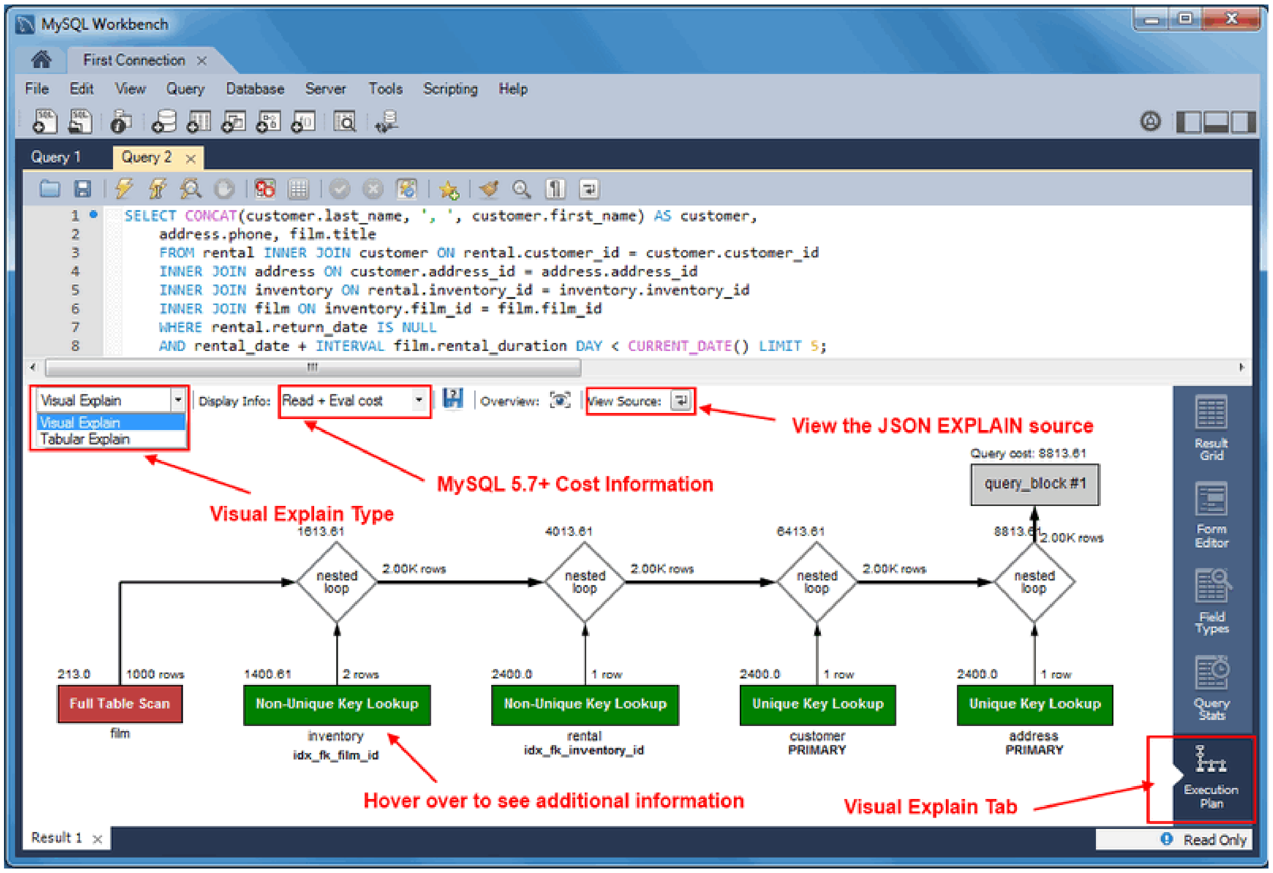


Visual Explain / Execution Plan Improvements

The Visual Explain **Execution Plan** feature was improved. A list of changes includes:

- An "Execution Plan" tab was added to the results view
- All statements now offer a "Visual Explain" execution plan
- The layout changed, and was improved to allow easier navigation in large query plans

Figure 1.13 Execution Plan Explained



```
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
    address.phone, film.title
    FROM rental INNER JOIN customer ON rental.customer_id = customer.customer_id
    INNER JOIN address ON customer.address_id = address.address_id
    INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
    INNER JOIN film ON inventory.film_id = film.film_id
    WHERE rental.return_date IS NULL
    AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE() LIMIT 5;
```

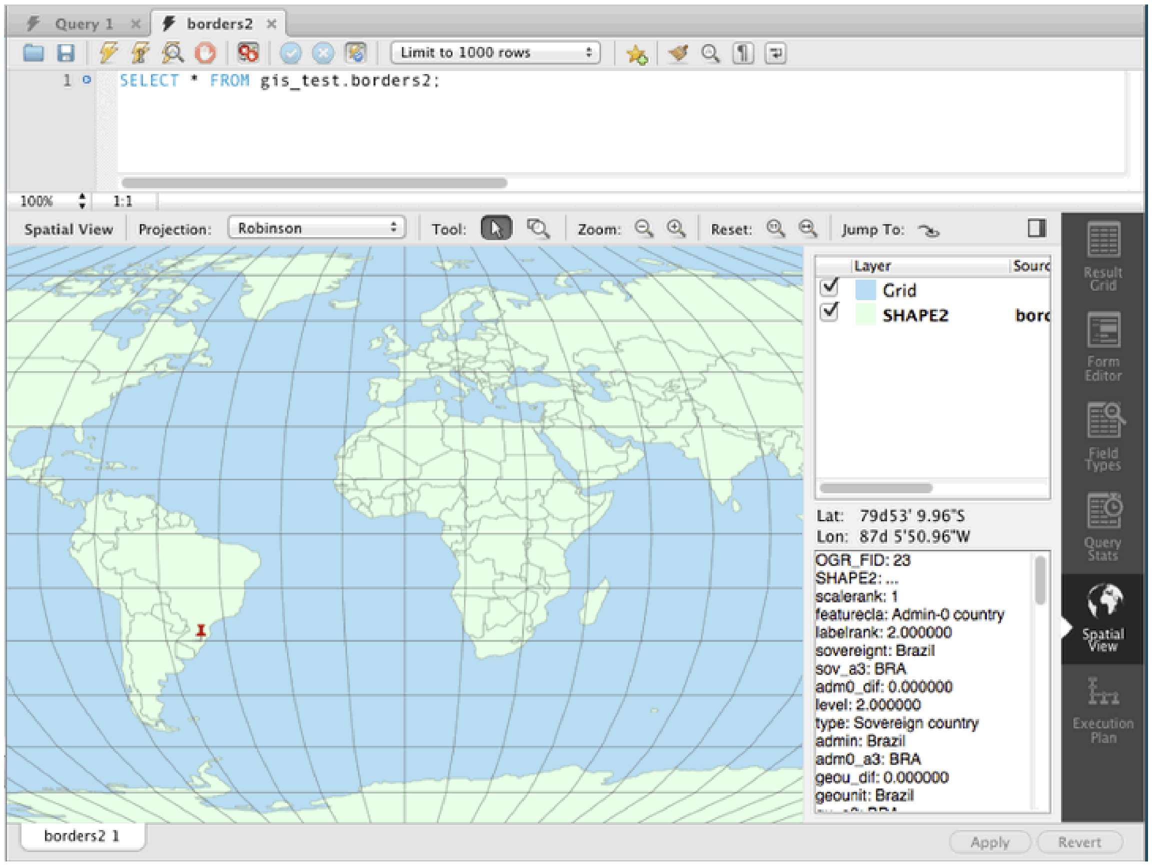
Spatial View Panel

GIS support for InnoDB tables is now supported to make it easier to visualize spatial and geometry data in a geographic context. The new spatial view panel renders data from each row into a separate and selectable element. When clicked, you can view the rest of the data from that row in the textbox. If you have multiple queries with geometry data, you can overlay them onto the same map. View options include the Robinson, Mercator, Equirectangular, and Bonne projection methods.

Note

GIS support for InnoDB tables was added in MySQL server 5.7.

Figure 1.14 Spatial View Example



```
SELECT * FROM gis_test.borders2;
```

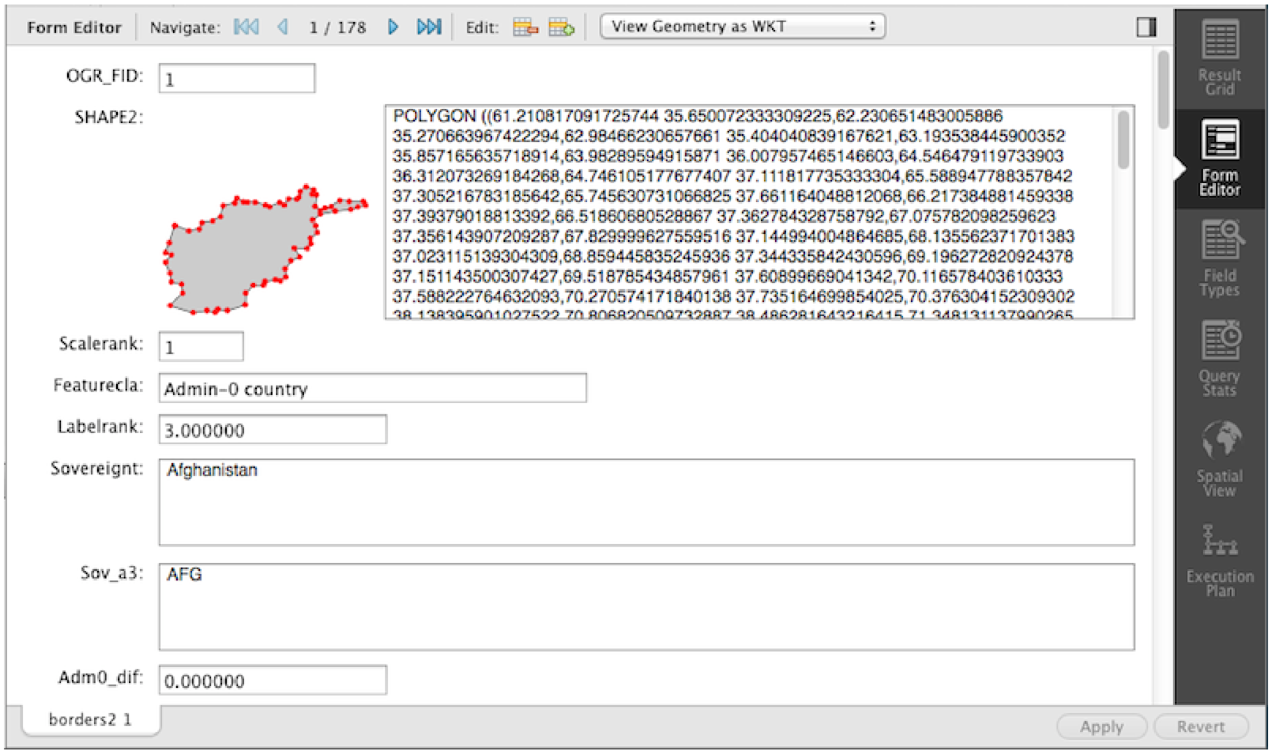
```
Lat:  79d53' 9.96"S
Lon:  87d 5'50.96"W
```

```
OGR_FID: 23
SHAPE2: ...
scalerank: 1
featurecla: Admin-0 country
labelrank: 2.000000
sovereignt: Brazil
sov_a3: BRA
adm0_dif: 0.000000
level: 2.000000
type: Sovereign country
admin: Brazil
adm0_a3: BRA
geou_dif: 0.000000
geounit: Brazil
```

Geometry Data Viewer

The SQL field and form editors were updated to support the `GEOMETRY` datatype. You can view geometry data, such as polygons, from a single row as an image or as text. The available formats include WKT, GeoJSON, GML, and KML.

Figure 1.15 Geometry Data Viewer



OGR_FID: 1

SHAPE2:

```
POLYGON ((61.210817091725744 35.650072333309225,62.230651483005886
35.270663967422294,62.98466230657661 35.404040839167621,63.193538445900352
35.857165635718914,63.98289594915871 36.007957465146603,64.546479119733903
36.312073269184268,64.746105177677407 37.111817735333304,65.588947788357842
37.305216783185642,65.745630731066825 37.661164048812068,66.217384881459338
37.39379018813392,66.51860680528867 37.362784328758792,67.075782098259623
37.356143907209287,67.829999627559516 37.144994004864685,68.135562371701383
37.023115139304309,68.859445835245936 37.344335842430596,69.196272820924378
37.151143500307427,69.518785434857961 37.60899669041342,70.116578403610333
37.588222764632093,70.270574171840138 37.735164699854025,70.376304152309302
38.138395901027522,70.806820509732887 38.486281643216415,71.348131137990265
```

Scalerank: 1

Featurecla: Admin-0 country

Labelrank: 3.000000

Sovereignt: Afghanistan

Sov_a3: AFG

Adm0_dif: 0.000000

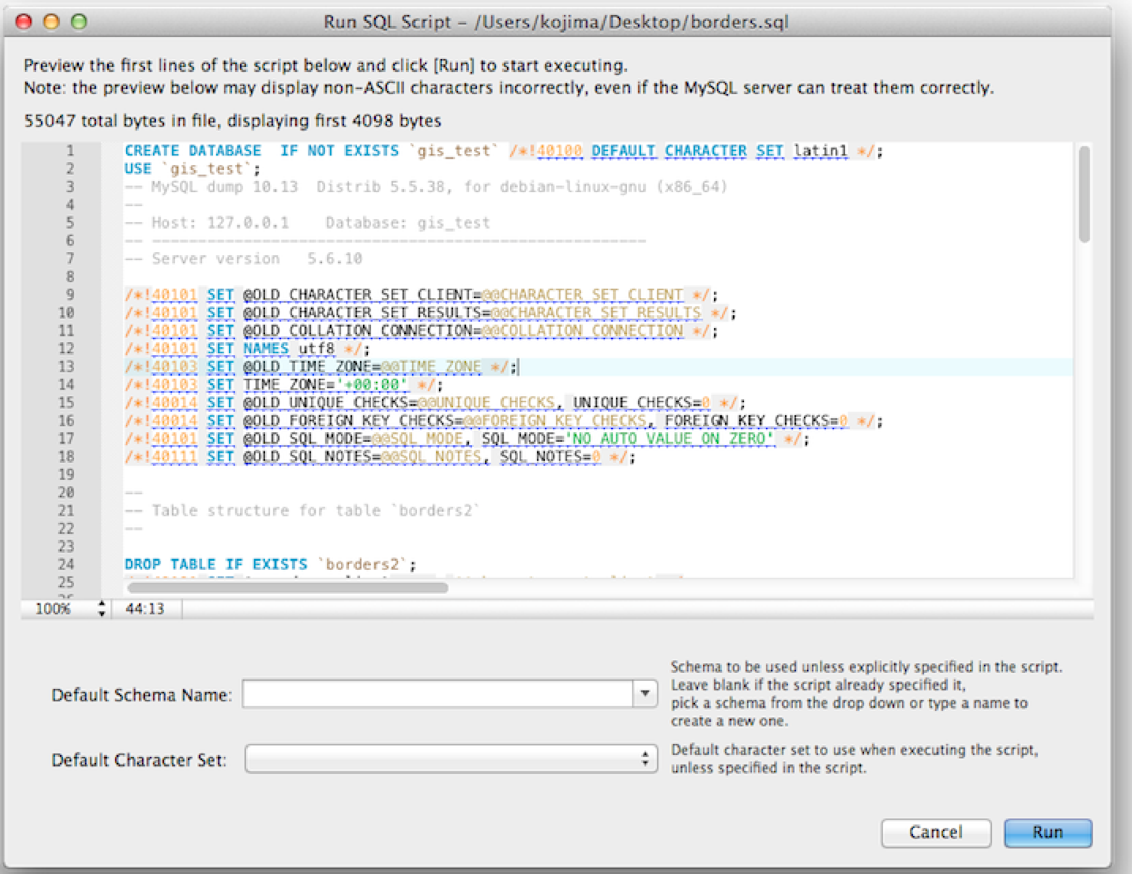
Additional New SQL Editor Features

- **Result Set Widths**: resized result set column widths are now preserved and remembered. This data is saved under Workbench's `cache/` directory using the *schema.table.column* format.
- Opened, closed, and reordered SQL editor tabs are now properly saved and restored. The scroll position and cursor locations are also remembered.
- **Shared Snippets**: these allow multiple users to share SQL code across a shared MySQL connection. They are stored in a schema named *.mysqlworkbench* on the connected MySQL server. by storing the snippets in a shared MySQL instance. For additional information, see Section 8.1.5, “SQL Snippets tab”.
- The full SQL syntax error is now viewable by hovering over the error response message.
- The **Query Status** tab was improved to include graphs and additional information.

Execute SQL Scripts

The new **Run SQL Script** dialog executes an SQL script without loading it into the SQL editor. This is useful because loading large scripts for editing can cause performance problems related to increased memory usage and required processing for editor features such as syntax highlighting, syntax checking, and code-folding. The dialog lets you preview a part of the script, optionally specify a default schema, and optionally set the default character set to use for the imported data. The output window shows warnings, messages, and an execution progression bar. Select **Run SQL Script** from the **File** menu to execute this wizard.

Figure 1.16 Run SQL Script



Preview the first lines of the script below and click [Run] to start executing.
Note: the preview below may display non-ASCII characters incorrectly, even if the MySQL server can treat them correctly.

55047 total bytes in file, displaying first 4098 bytes

```
CREATE DATABASE  IF NOT EXISTS `gis_test` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `gis_test`;
-- MySQL dump 10.13  Distrib 5.5.38, for debian-linux-gnu (x86_64)
--
-- Host: 127.0.0.1    Database: gis_test
-- ------------------------------------------------------
-- Server version	5.6.10

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `borders2`
--

DROP TABLE IF EXISTS `borders2`;
```

Default Schema Name:

Schema to be used unless explicitly specified in the script. Leave blank if the script already specified it, pick a schema from the drop down or type a name to create a new one.

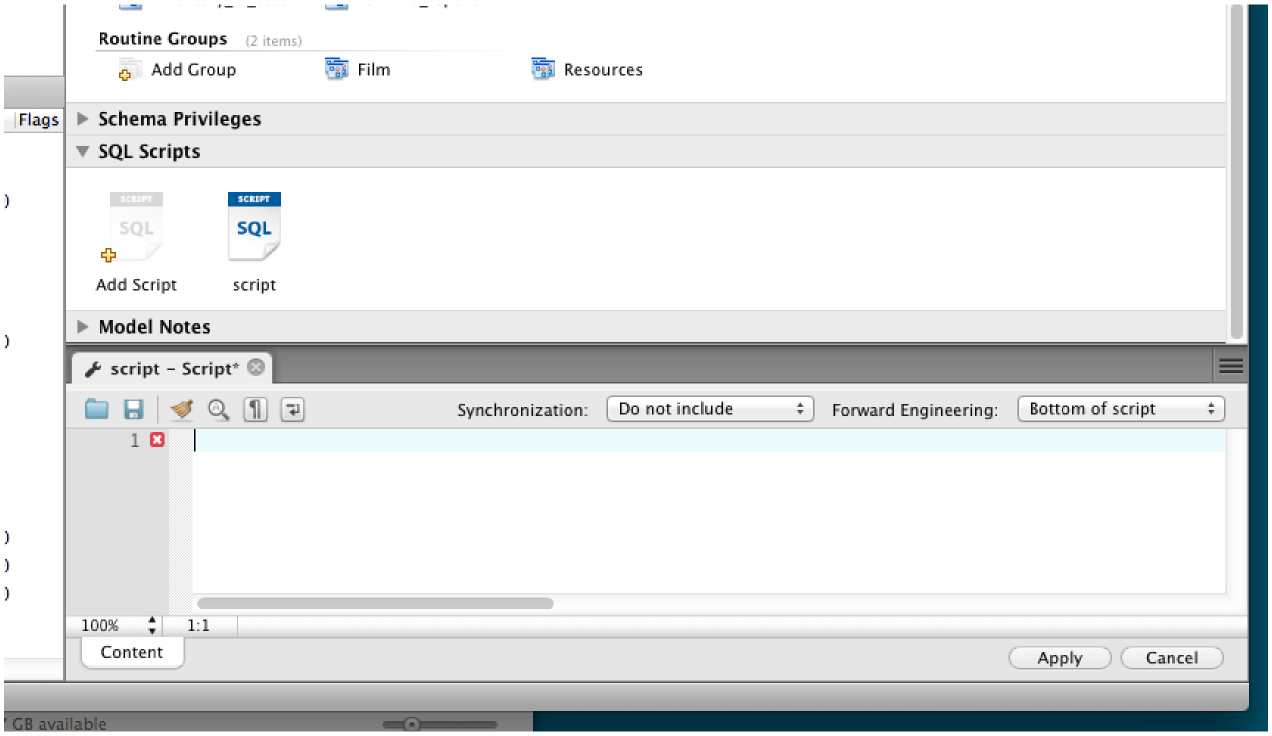
Default Character Set:

Default character set to use when executing the script, unless specified in the script.

Model Script Attachments

Previously, MySQL Workbench modeling supported attaching SQL script files to models, usually for documentation and organization purposes. You can now include attached SQL files to the output script when performing forward engineering or synchronization operations.

Figure 1.17 Data Modeling Script Attachments

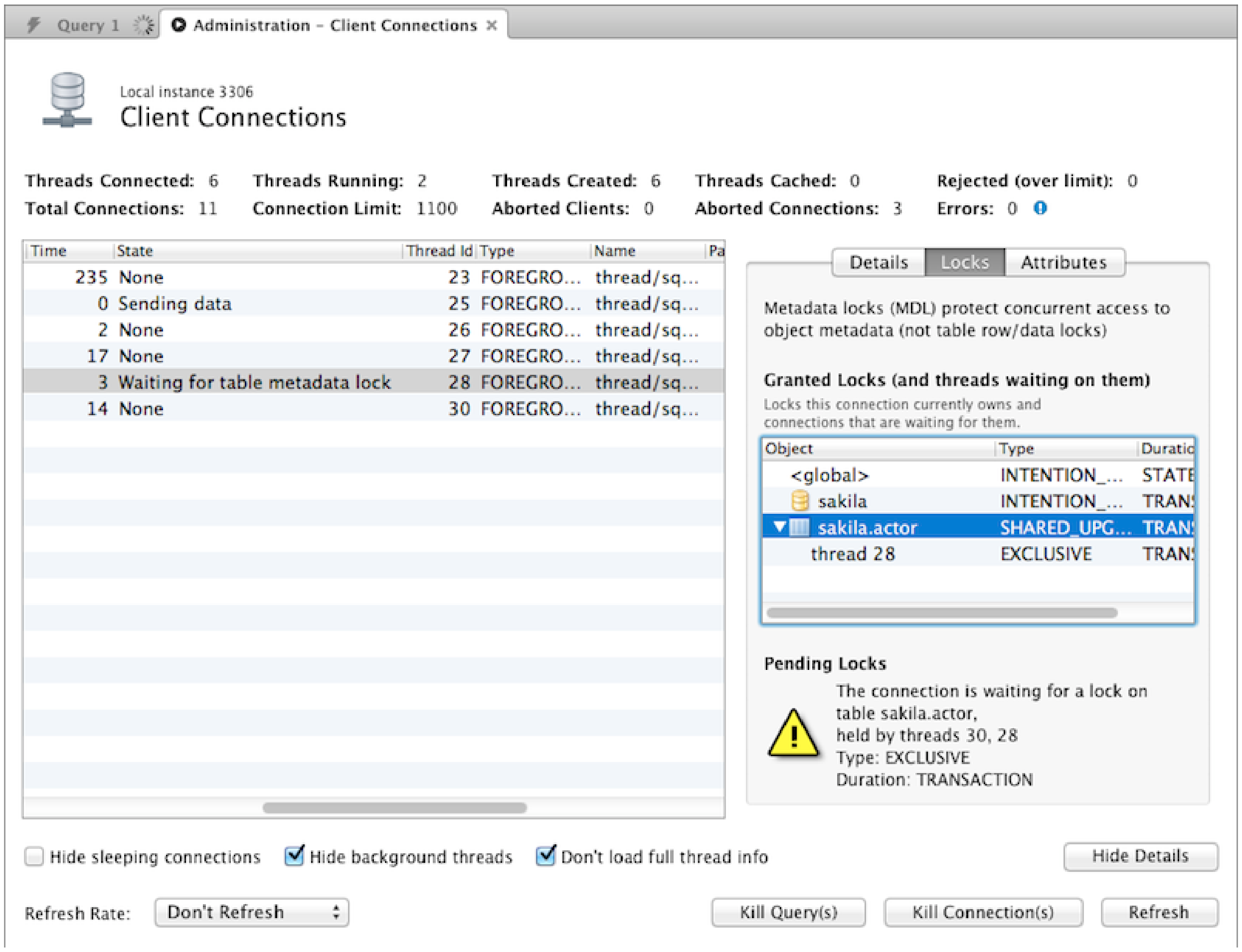


Client Connections and Metadata locks

The **Client Connections** management window has a new **Show Details** window. This window's three tabs are:

- **Details**: connection details such as Process ID, Type, User, Host, Instrumented, and additional information.
- **Locks**: MySQL uses metadata locking to manage access to objects such as tables and triggers. Sometimes a query might be blocked while being manipulated by another connection from another user. The **Locks** feature utilizes these MySQL metadata locks (MDL) to show the locked connections that are blocked or being waiting on, and shows information about the locks, what they are waiting for, and what they hold.

Figure 1.18 Metadata Locks Browser



Threads Connected: 6 Threads Running: 2 Threads Created: 6 Threads Cached: 0 Rejected (over limit): 0
Total Connections: 11 Connection Limit: 1100 Aborted Clients: 0 Aborted Connections: 3 Errors: 0

Time	State	Thread Id	Type	Name	Pa
235	None	23	FOREGRO...	thread/sq...	
0	Sending data	25	FOREGRO...	thread/sq...	
2	None	26	FOREGRO...	thread/sq...	
17	None	27	FOREGRO...	thread/sq...	
3	Waiting for table metadata lock	28	FOREGRO...	thread/sq...	
14	None	30	FOREGRO...	thread/sq...	

Metadata locks (MDL) protect concurrent access to object metadata (not table row/data locks)

Granted Locks (and threads waiting on them)

Locks this connection currently owns and connections that are waiting for them.

Object	Type	Duratio
<global>	INTENTION_...	STATE
sakila	INTENTION_...	TRAN
sakila.actor	SHARED_UPG...	TRAN
thread 28	EXCLUSIVE	TRAN

Pending Locks

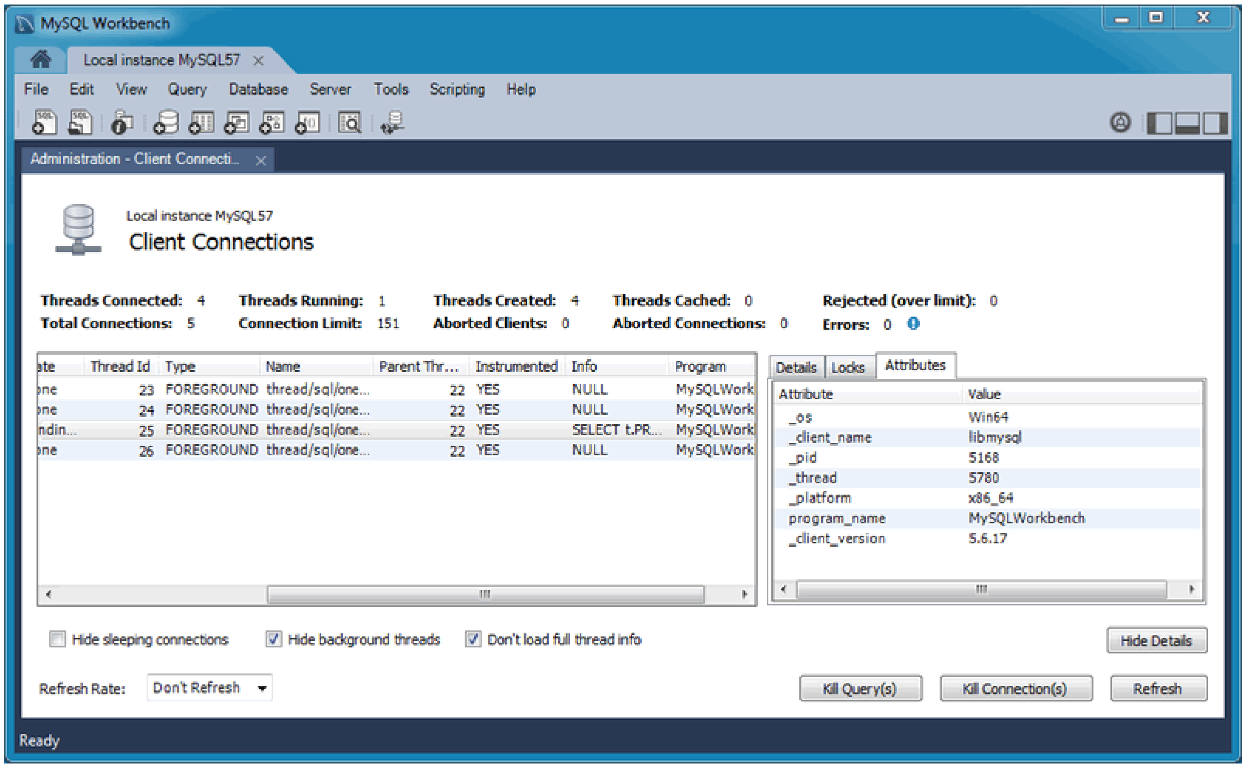
The connection is waiting for a lock on table sakila.actor,
held by threads 30, 28
Type: EXCLUSIVE
Duration: TRANSACTION

Note

The metadata lock information is provided in the performance schema as of MySQL server 5.7.3.

- **Attributes**: these are connection attributes such as OS, Client Name, Client Version, and Platform.

Figure 1.19 Client Connection Attributes


MySQL Workbench
Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Administration - Client Connecti...
Local instance MySQL57
Client Connections
Details Locks Attributes
Hide Details
Kill Query(s) Kill Connection(s) Refresh
Ready

Threads Connected: 4 Threads Running: 1 Threads Created: 4 Threads Cached: 0 Rejected (over limit): 0
Total Connections: 5 Connection Limit: 151 Aborted Clients: 0 Aborted Connections: 0 Errors: 0

ate	Thread Id	Type	Name	Parent Thr...	Instrumented	Info	Program
one	23	FOREGROUND	thread/sql/one...	22	YES	NULL	MySQLWork
one	24	FOREGROUND	thread/sql/one...	22	YES	NULL	MySQLWork
ndin...	25	FOREGROUND	thread/sql/one...	22	YES	SELECT t.PR...	MySQLWork
one	26	FOREGROUND	thread/sql/one...	22	YES	NULL	MySQLWork

Attribute	Value
_os	Win64
_client_name	libmysql
_pid	5168
_thread	5780
_platform	x86_64
program_name	MySQLWorkbench
_client_version	5.6.17

☐ Hide sleeping connections ☑ Hide background threads ☑ Don't load full thread info

Refresh Rate: Don't Refresh

Note

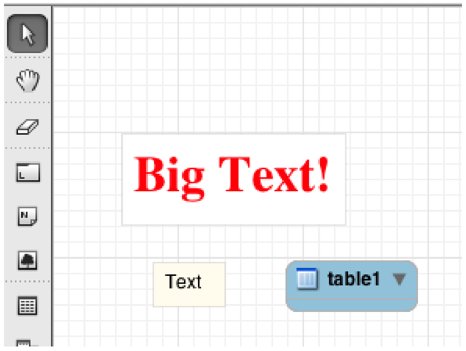
This feature uses performance schema details from MySQL server 5.7 and above.

For additional information, see Section 5.5, "Client Connections".

Additional New Features

- Performance columns (that display sizes) now have an option to alter the value units. They can be set to KB, MB, or GB. Right-click on a column header and choose **Set Display Unit**.
- The migration wizard can now resume operation if a data copy failed during a database migration from, for example, a timeout or network failure. Click **Resume** retry the data copy, and MySQL Workbench locates the last row that was copied successfully and attempts to restart the copy from that row.
- The MySQL connection password is now remembered across the MySQL Workbench session, even if it not stored in the keychain. This is so you do not need to re-enter it whenever a new MySQL connection is needed.
- Under Modeling, the Role Editor now has "Add Everything" and "Check All Privileges" options.
- The **Preferences** layout changed. The tabs were replaced by a list using a horizontal sidebar, and additional category names were added. For additional information, see Section 3.2, "Workbench Preferences".
- Keyboard shortcuts now function in the *Scripting Shell*.
- **Model diagram notes** can now be resized and automatically rearranged. You can also change the style attributes such as the font, background color, and text color.

Figure 1.20 Model Diagram Note Formatting



1.1.3 New in MySQL Workbench 6.1

This section summarizes many of the new features added to MySQL Workbench 6.1.x, in relation to MySQL Workbench 6.0.x;.

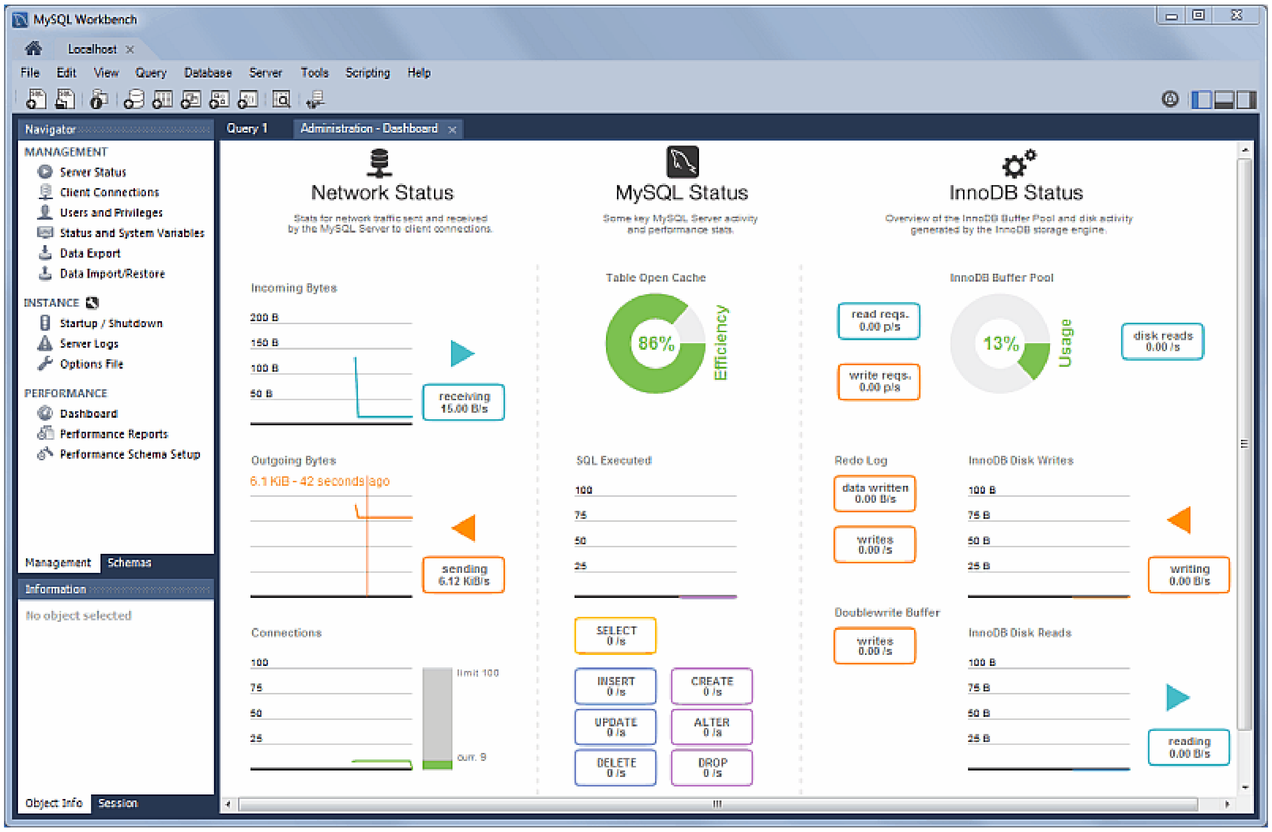
New Navigator PERFORMANCE Section

The new **PERFORMANCE** section includes **Dashboard**, **Performance Reports**, and **Performance Schema Setup** pages. Generally, this new performance reporting feature provides a graphical representation of key statistics from the MySQL server status, and provides an overview of the MySQL server subsystems.

Dashboard

View server performance statistics in a graphical dashboard.

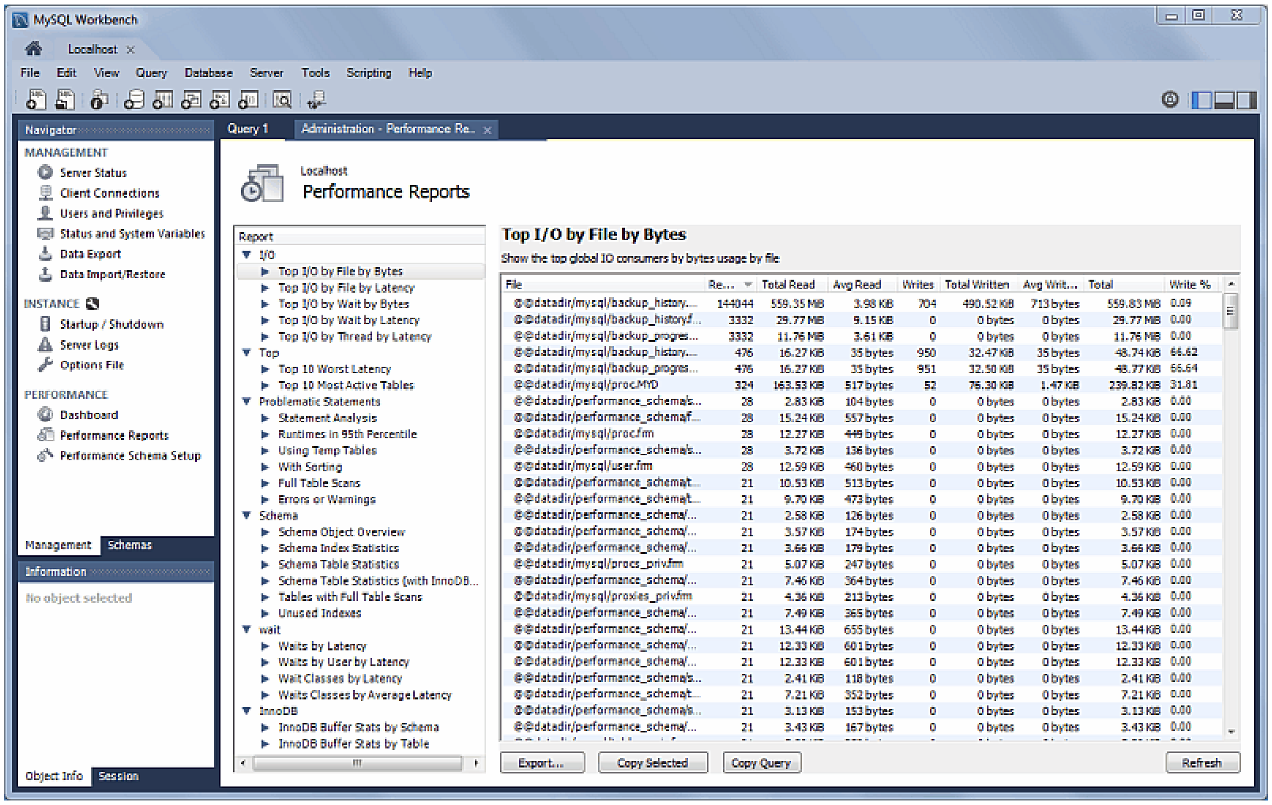
Figure 1.21 Performance Dashboard



Performance Reports

Performance schema based reports that provide insight into the operation of the MySQL server through many high-level reports.

Figure 1.22 Performance Reports: Top I/0 By Bytes



Top I/O by File by Bytes

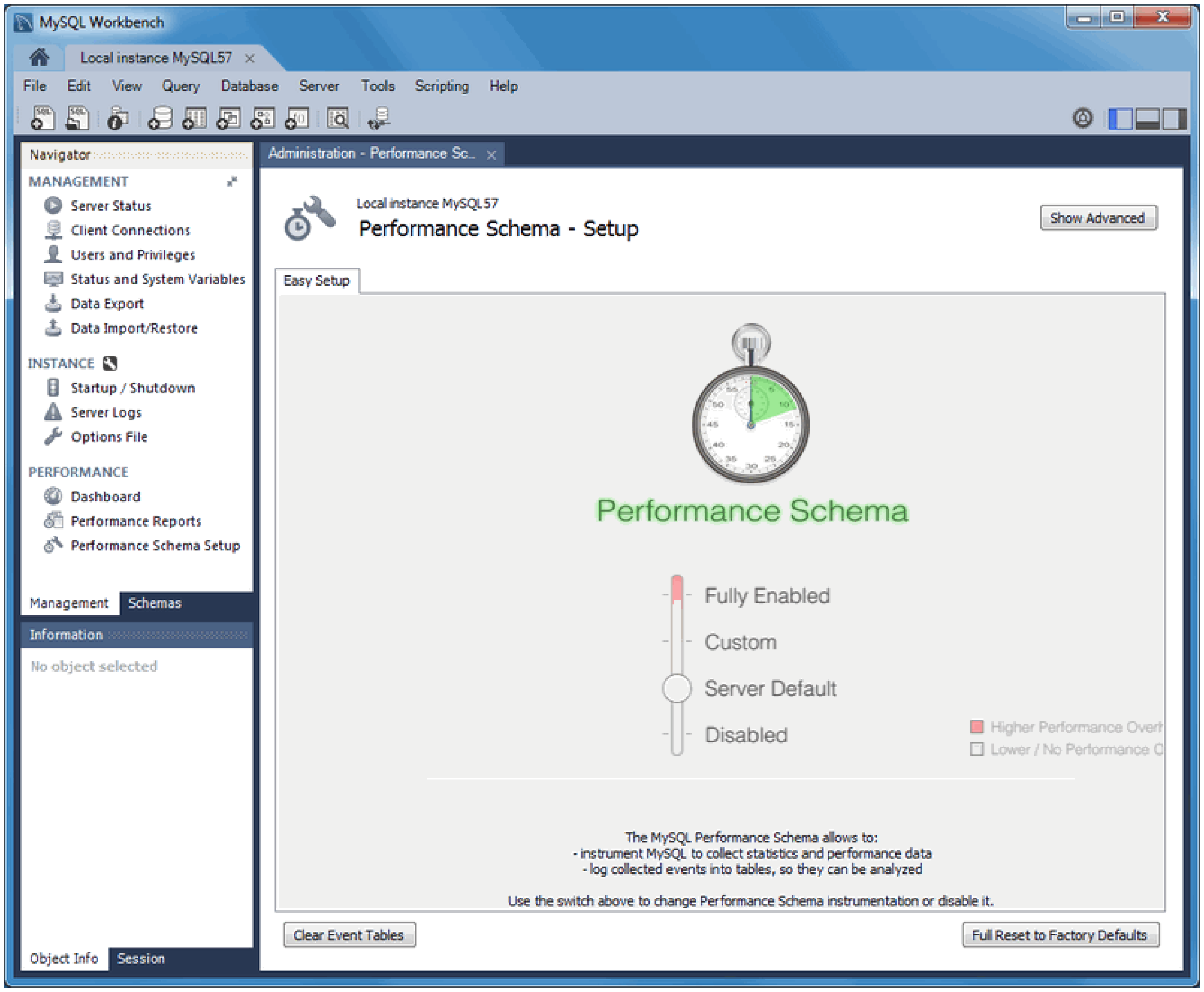
Show the top global IO consumers by bytes usage by file

File	Re...	Total Read	Avg Read	Writes	Total Written	Avg Writ...	Total	Write %
@@datadir/mysql/backup_history....	144044	559.35 MB	3.98 KB	704	490.52 KB	713 bytes	559.83 MB	0.09
@@datadir/mysql/backup_historyf...	3332	29.77 MB	9.15 KB	0	0 bytes	0 bytes	29.77 MB	0.00
@@datadir/mysql/backup_progres...	3332	11.76 MB	3.61 KB	0	0 bytes	0 bytes	11.76 MB	0.00
@@datadir/mysql/backup_history....	476	16.27 KB	35 bytes	950	32.47 KB	35 bytes	48.74 KB	66.62
@@datadir/mysql/backup_progres...	476	16.27 KB	35 bytes	951	32.50 KB	35 bytes	48.77 KB	66.64
@@datadir/mysql/proc.MYD	324	163.53 KB	517 bytes	52	76.30 KB	1.47 KB	239.82 KB	31.81
@@datadir/performance_schema/s...	28	2.83 KB	104 bytes	0	0 bytes	0 bytes	2.83 KB	0.00
@@datadir/performance_schema/f...	28	15.24 KB	557 bytes	0	0 bytes	0 bytes	15.24 KB	0.00
@@datadir/mysql/proc.frm	28	12.27 KB	449 bytes	0	0 bytes	0 bytes	12.27 KB	0.00
@@datadir/performance_schema/s...	28	3.72 KB	136 bytes	0	0 bytes	0 bytes	3.72 KB	0.00
@@datadir/mysql/user.frm	28	12.59 KB	460 bytes	0	0 bytes	0 bytes	12.59 KB	0.00
@@datadir/performance_schema/t...	21	10.53 KB	513 bytes	0	0 bytes	0 bytes	10.53 KB	0.00
@@datadir/performance_schema/t...	21	9.70 KB	473 bytes	0	0 bytes	0 bytes	9.70 KB	0.00
@@datadir/performance_schema/...	21	2.58 KB	126 bytes	0	0 bytes	0 bytes	2.58 KB	0.00
@@datadir/performance_schema/...	21	3.57 KB	174 bytes	0	0 bytes	0 bytes	3.57 KB	0.00
@@datadir/performance_schema/...	21	3.66 KB	179 bytes	0	0 bytes	0 bytes	3.66 KB	0.00
@@datadir/mysql/procs_priv.frm	21	5.07 KB	247 bytes	0	0 bytes	0 bytes	5.07 KB	0.00
@@datadir/performance_schema/...	21	7.46 KB	364 bytes	0	0 bytes	0 bytes	7.46 KB	0.00
@@datadir/mysql/proxies_priv.frm	21	4.36 KB	213 bytes	0	0 bytes	0 bytes	4.36 KB	0.00
@@datadir/performance_schema/...	21	7.49 KB	365 bytes	0	0 bytes	0 bytes	7.49 KB	0.00
@@datadir/performance_schema/...	21	13.44 KB	655 bytes	0	0 bytes	0 bytes	13.44 KB	0.00
@@datadir/performance_schema/...	21	12.33 KB	601 bytes	0	0 bytes	0 bytes	12.33 KB	0.00
@@datadir/performance_schema/...	21	12.33 KB	601 bytes	0	0 bytes	0 bytes	12.33 KB	0.00
@@datadir/performance_schema/s...	21	2.41 KB	118 bytes	0	0 bytes	0 bytes	2.41 KB	0.00
@@datadir/performance_schema/t...	21	7.21 KB	352 bytes	0	0 bytes	0 bytes	7.21 KB	0.00
@@datadir/performance_schema/s...	21	3.13 KB	153 bytes	0	0 bytes	0 bytes	3.13 KB	0.00
@@datadir/performance_schema/...	21	3.43 KB	167 bytes	0	0 bytes	0 bytes	3.43 KB	0.00

Performance Schema Setup

A GUI for configuring and fine tuning the Performance Schema instrumentation. Initially, this loads an "Easy Setup" page that is enough for most users. Slide the "Performance Schema Full Enabled" slider to **YES** to enable all available Performance Schema instruments.

Figure 1.23 Performance Schema Setup: Easy Setup

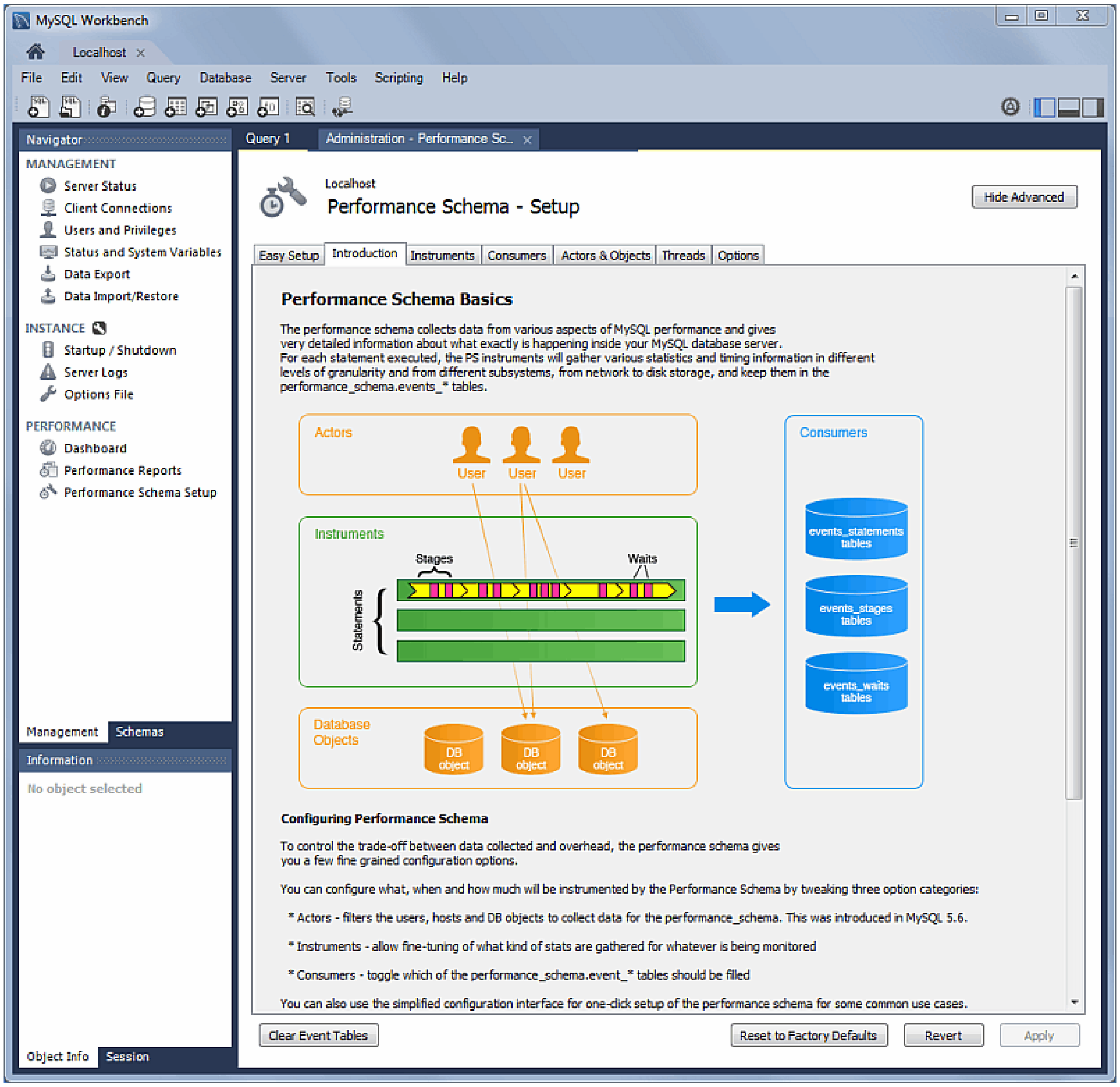


The MySQL Performance Schema allows to:
- instrument MySQL to collect statistics and performance data
- log collected events into tables, so they can be analyzed

Use the switch above to change Performance Schema instrumentation or disable it.

Clicking **Show Advanced** provides methods to fine tune the Performance Schema instrumentation.

Figure 1.24 Performance Schema Setup: Introduction



Performance Schema Basics

The performance schema collects data from various aspects of MySQL performance and gives very detailed information about what exactly is happening inside your MySQL database server. For each statement executed, the PS instruments will gather various statistics and timing information in different levels of granularity and from different subsystems, from network to disk storage, and keep them in the performance_schema.events_* tables.

Configuring Performance Schema

To control the trade-off between data collected and overhead, the performance schema gives you a few fine grained configuration options.

You can configure what, when and how much will be instrumented by the Performance Schema by tweaking three option categories:

* Actors - filters the users, hosts and DB objects to collect data for the performance_schema. This was introduced in MySQL 5.6.

* Instruments - allow fine-tuning of what kind of stats are gathered for whatever is being monitored

* Consumers - toggle which of the performance_schema.event_* tables should be filled

You can also use the simplified configuration interface for one-click setup of the performance schema for some common use cases.

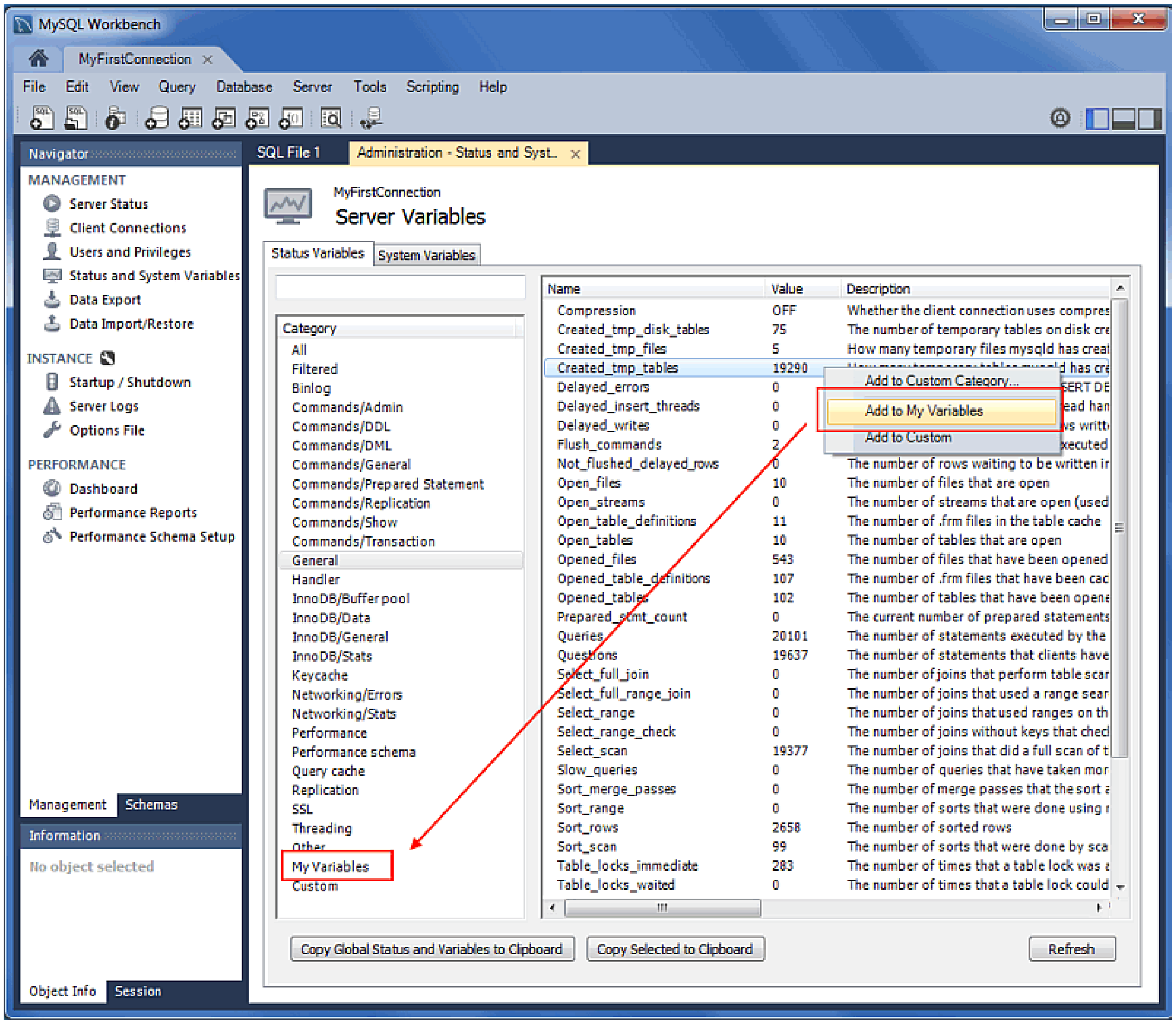
For additional information, see Chapter 7, *Performance Tools*.

Server Variable Groupings

Variables can now be organized using custom groupings in the **Status and System Variables** Management tab.

To create a custom group, right-click on a variable and choose either **Add to Custom Category** (to create a new category), or an existing custom category. For additional information, see Section 6.4, “Status and System Variables”.

Figure 1.25 Status And System Variables: Custom



Name	Value	Description
Compression	OFF	Whether the client connection uses compres
Created_tmp_disk_tables	75	The number of temporary tables on disk cre
Created_tmp_files	5	How many temporary files mysqld has creat
Created_tmp_tables	19290	d has cre
Delayed_errors	0	SERT DE
Delayed_insert_threads	0	ead han
Delayed_writes	0	ws writt
Flush_commands	2	xecuted
Not_flushed_delayed_rows	0	The number of rows waiting to be written ir
Open_files	10	The number of files that are open
Open_streams	0	The number of streams that are open (used
Open_table_definitions	11	The number of .frm files in the table cache
Open_tables	10	The number of tables that are open
Opened_files	543	The number of files that have been opened
Opened_table_definitions	107	The number of .frm files that have been cac
Opened_tables	102	The number of tables that have been opene
Prepared_stmt_count	0	The current number of prepared statements
Queries	20101	The number of statements executed by the
Questions	19637	The number of statements that clients have
Select_full_join	0	The number of joins that perform table scar
Select_full_range_join	0	The number of joins that used a range sear
Select_range	0	The number of joins that used ranges on th
Select_range_check	0	The number of joins without keys that checl
Select_scan	19377	The number of joins that did a full scan of t
Slow_queries	0	The number of queries that have taken mor
Sort_merge_passes	0	The number of merge passes that the sort a
Sort_range	0	The number of sorts that were done using r
Sort_rows	2658	The number of sorted rows
Sort_scan	99	The number of sorts that were done by sca
Table_locks_immediate	283	The number of times that a table lock was a
Table_locks_waited	0	The number of times that a table lock could

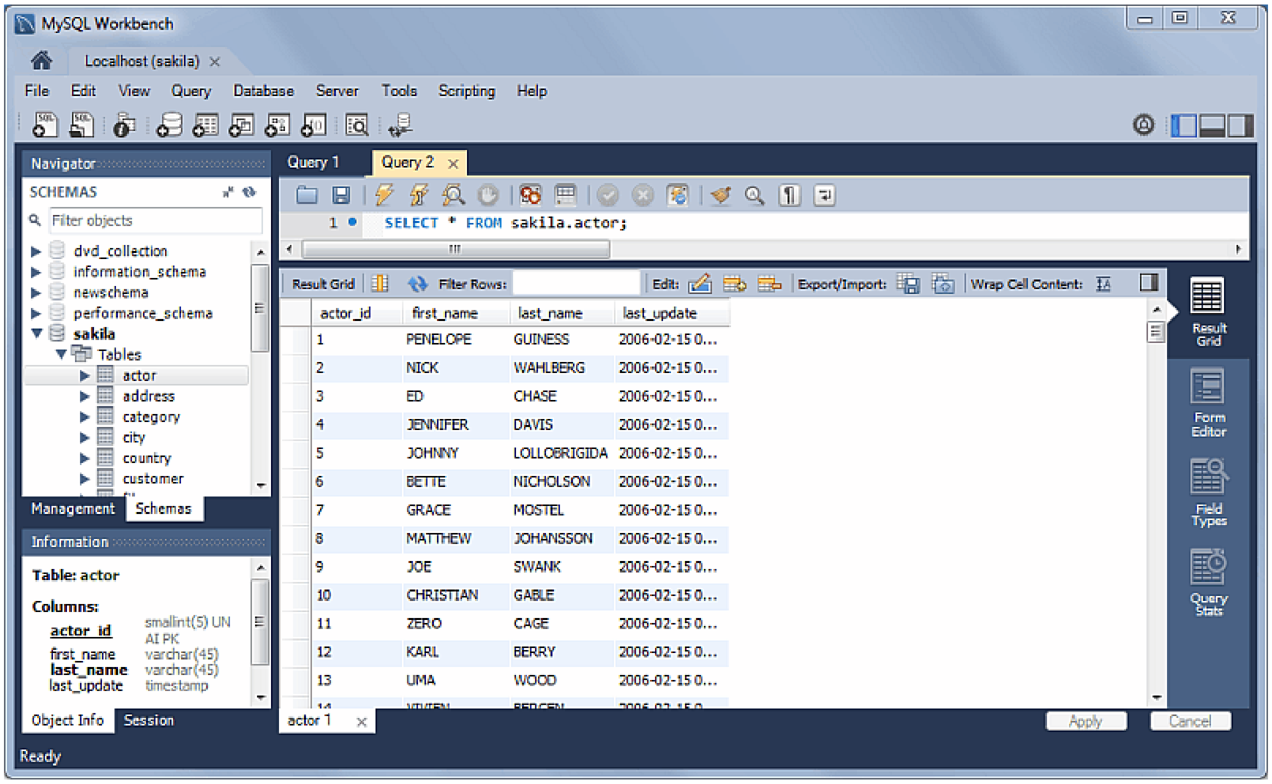
SQL Editor Views

Additional viewing options were added for executed statements:

Result Grid

Available previously, and it remains the default view.

Figure 1.26 SQL Editor: Result Grid



```
SELECT * FROM sakila.actor;
```

Table: actor

Columns:

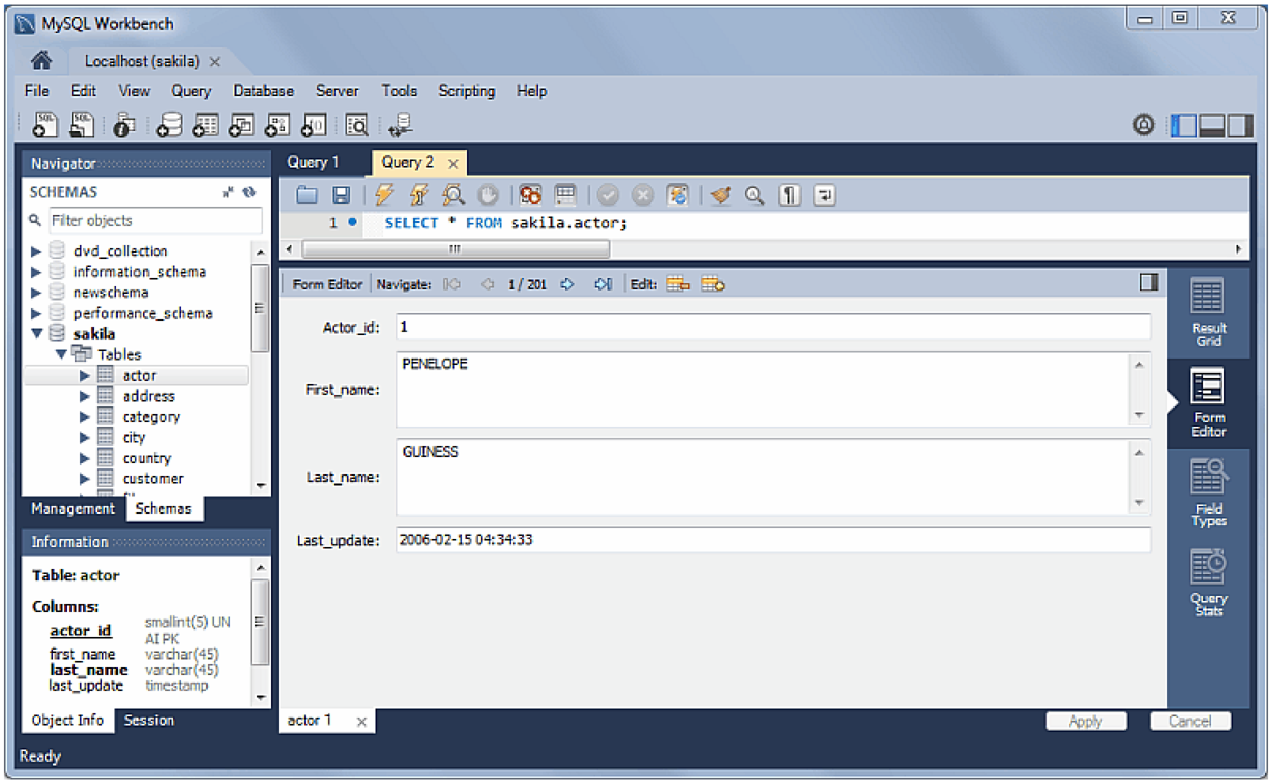
actor_id	smallint(5) UN AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15 0...
2	NICK	WAHLBERG	2006-02-15 0...
3	ED	CHASE	2006-02-15 0...
4	JENNIFER	DAVIS	2006-02-15 0...
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 0...
6	BETTE	NICHOLSON	2006-02-15 0...
7	GRACE	MOSTEL	2006-02-15 0...
8	MATTHEW	JOHANSSON	2006-02-15 0...
9	JOE	SWANK	2006-02-15 0...
10	CHRISTIAN	GABLE	2006-02-15 0...
11	ZERO	CAGE	2006-02-15 0...
12	KARL	BERRY	2006-02-15 0...
13	UMA	WOOD	2006-02-15 0...

Form Editor

You can now edit records row by row in a form style editor.

Figure 1.27 SQL Editor: Form Editor



```
SELECT * FROM sakila.actor;
```

Actor_id:	1
First_name:	PENELOPE
Last_name:	GUINESS
Last_update:	2006-02-15 04:34:33

Table: actor

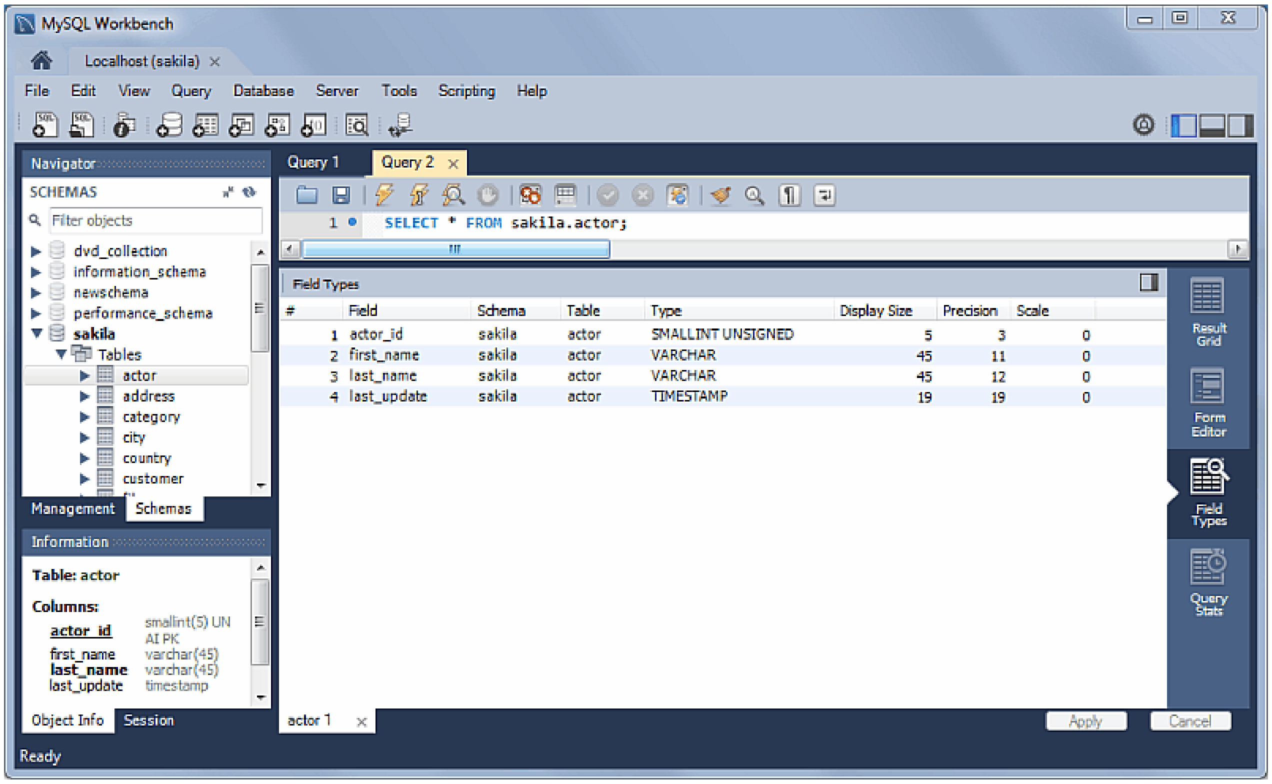
Columns:

actor_id	smallint(5) UN AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

Field Types

Displays information about the selected fields, similar to passing in `--column-type-info` from the command line client.

Figure 1.28 SQL Editor: Field Types



```
SELECT * FROM sakila.actor;
```

Field Types

#	Field	Schema	Table	Type	Display Size	Precision	Scale
1	actor_id	sakila	actor	SMALLINT UNSIGNED	5	3	0
2	first_name	sakila	actor	VARCHAR	45	11	0
3	last_name	sakila	actor	VARCHAR	45	12	0
4	last_update	sakila	actor	TIMESTAMP	19	19	0

Table: actor

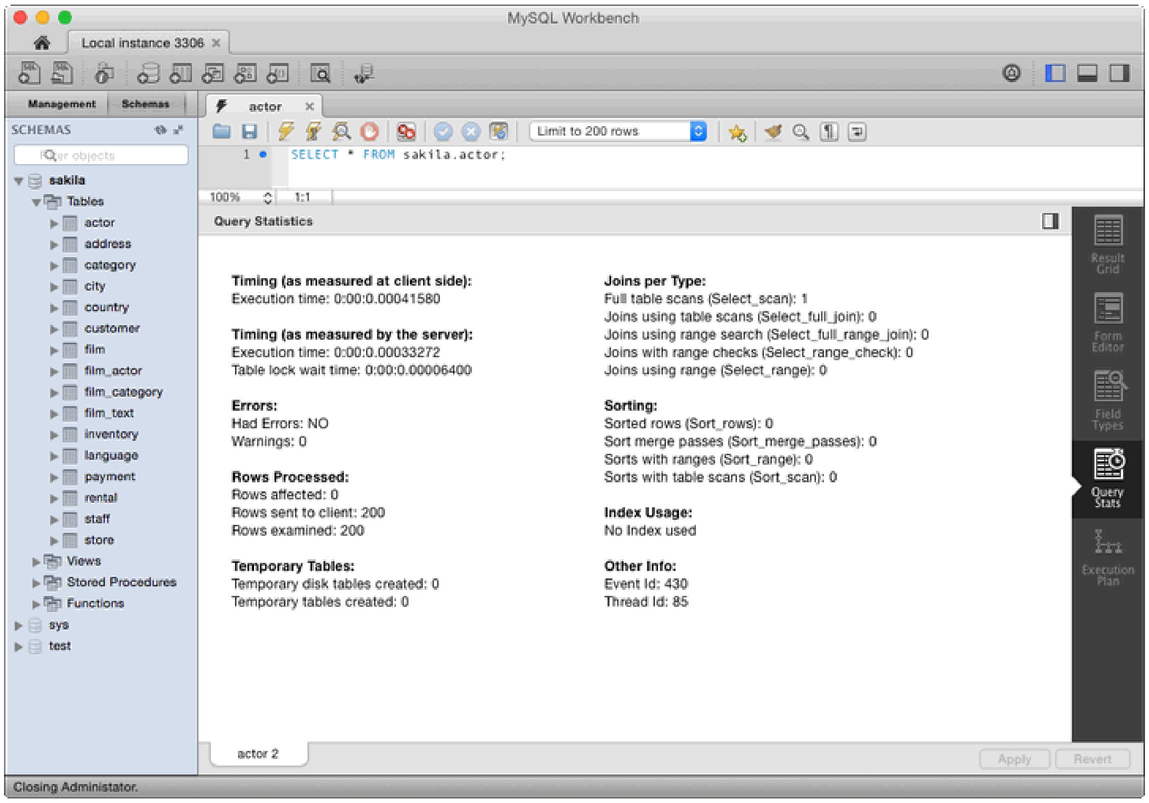
Columns:

actor_id	smallint(5) UN AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

Query Stats

Query statistics are taken from the Performance Schema, and includes information about timing, temporary tables, indexes, joins, and more.

Figure 1.29 SQL Editor: Query Stats



```
SELECT * FROM sakila.actor;
```

Query Statistics

```
Timing (as measured at client side):
Execution time: 0:00:0.00041580

Timing (as measured by the server):
Execution time: 0:00:0.00033272
Table lock wait time: 0:00:0.00006400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 200
Rows examined: 200

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 430
Thread Id: 85
```

Home Screen Features

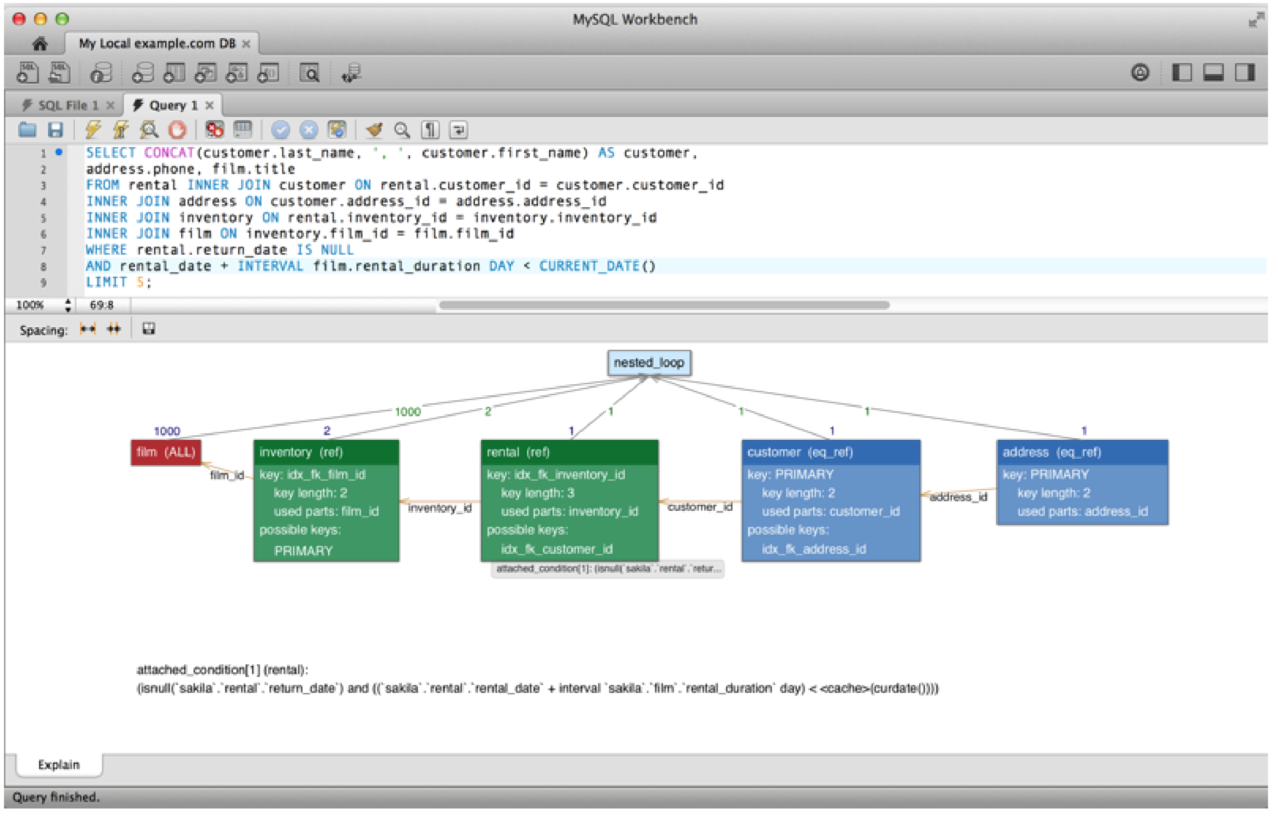
Several behavioral improvements were made to the MySQL Workbench Home screen, including:

- Connection tiles can now be repositioned by using drag and drop
- A script or model file can be dragged into a MySQL connection tile
- The following right-click options were added to the connection tiles: **Copy JDBC Connection String** and **Copy Connection String**
- Right-clicking a blank area in the **MySQL Connections** area now offers an option to create a **New Connection From Clipboard**

Visual Explain

The layout changed, and additional information is now viewable by hovering over the fields. It also displays traditional `EXPLAIN` output in a separate tab, and the **Raw Explain Data** (as JSON) in another. For MySQL server 5.7+, the new "cost information" (such as "query_cost" and "sort_cost) is also utilized.

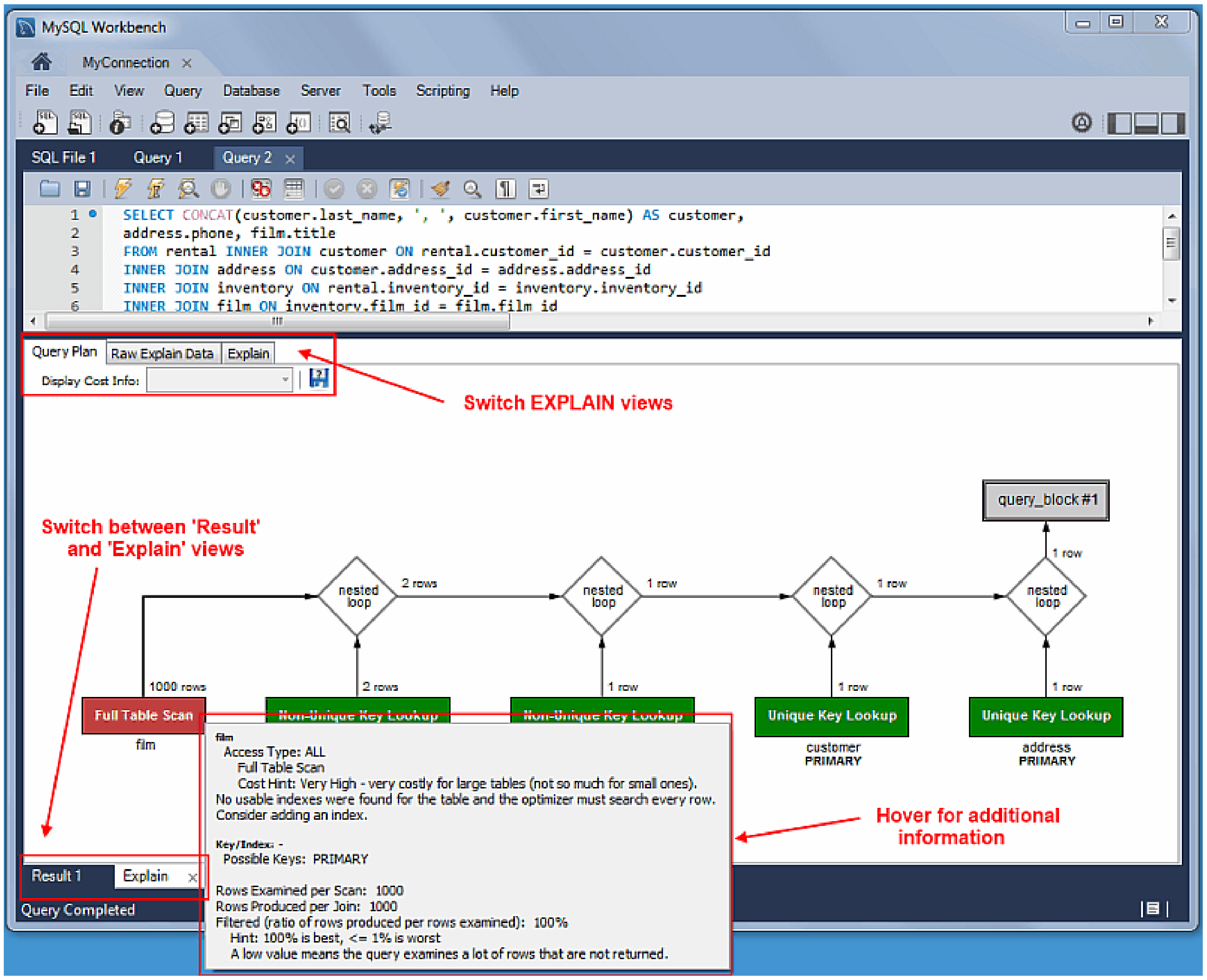
Figure 1.30 Visual Explain: Workbench 6.0



```sql
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
address.phone, film.title
FROM rental INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
LIMIT 5;
```

```
attached_condition[1] (rental):
(isnull(`sakila`.`rental`.`return_date`) and ((`sakila`.`rental`.`rental_date` + interval `sakila`.`film`.`rental_duration` day) < <cache>(curdate())))
```

Figure 1.31 Visual Explain: Workbench 6.1



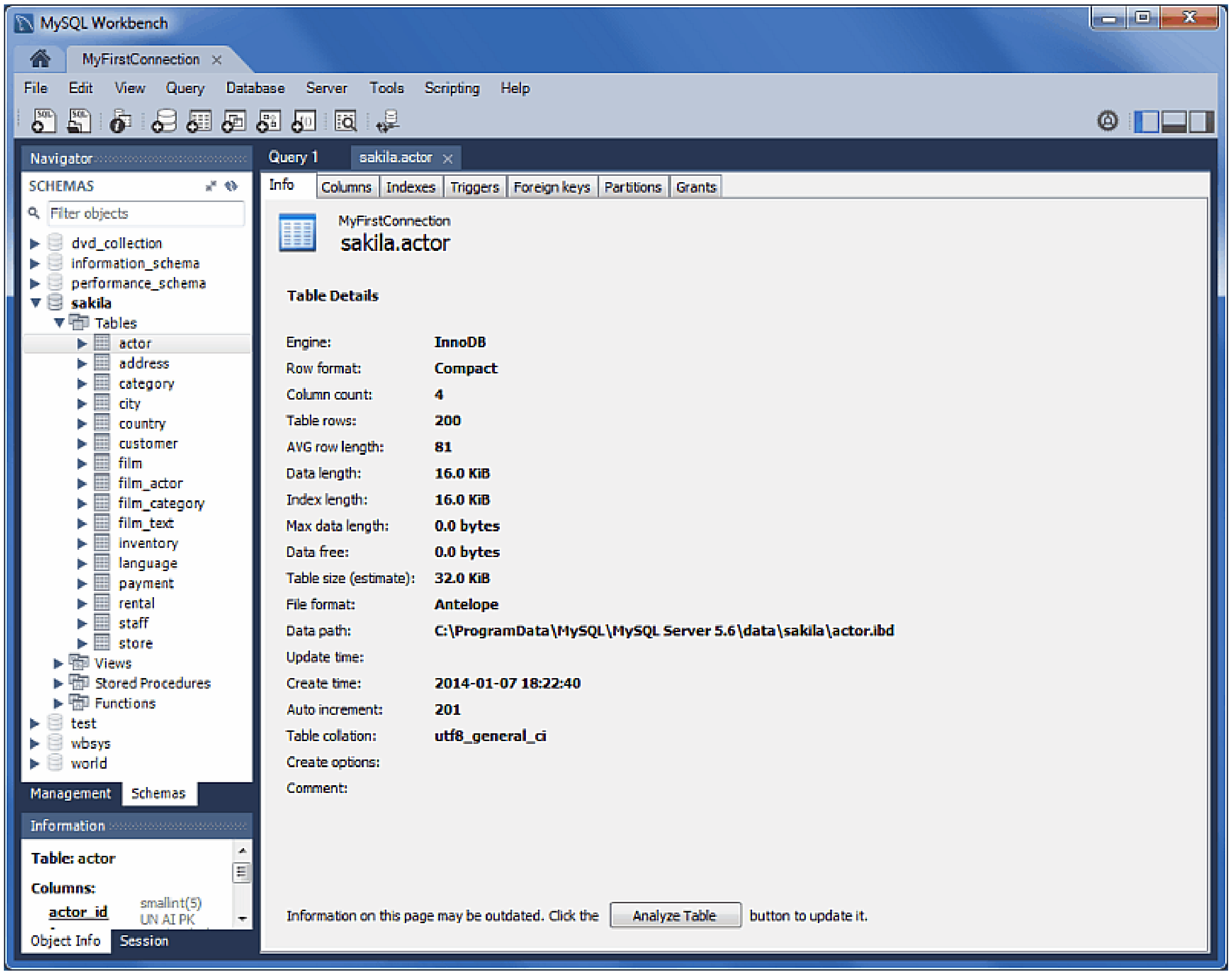
```
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
address.phone, film.title
FROM rental INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
```

```
film
  Access Type: ALL
     Full Table Scan
     Cost Hint: Very High - very costly for large tables (not so much for small ones).
No usable indexes were found for the table and the optimizer must search every row.
Consider adding an index.

Key/Index: -
  Possible Keys: PRIMARY

Rows Examined per Scan: 1000
Rows Produced per Join: 1000
Filtered (ratio of rows produced per rows examined): 100%
   Hint: 100% is best, <= 1% is worst
   A low value means the query examines a lot of rows that are not returned.
```

Table Inspector

View table information, similar to the Schema Inspector. This also has a simpler and easier to use interface for analyzing and creating indexes for your tables.

Figure 1.32 Table Inspector



MyFirstConnection
sakila.actor

Table Details

Engine:	InnoDB
Row format:	Compact
Column count:	4
Table rows:	200
AVG row length:	81
Data length:	16.0 KiB
Index length:	16.0 KiB
Max data length:	0.0 bytes
Data free:	0.0 bytes
Table size (estimate):	32.0 KiB
File format:	Antelope
Data path:	C:\ProgramData\MySQL\MySQL Server 5.6\data\sakila\actor.ibd
Update time:	
Create time:	2014-01-07 18:22:40
Auto increment:	201
Table collation:	utf8_general_ci
Create options:	
Comment:	

Information on this page may be outdated. Click the Analyze Table button to update it.

Table: actor

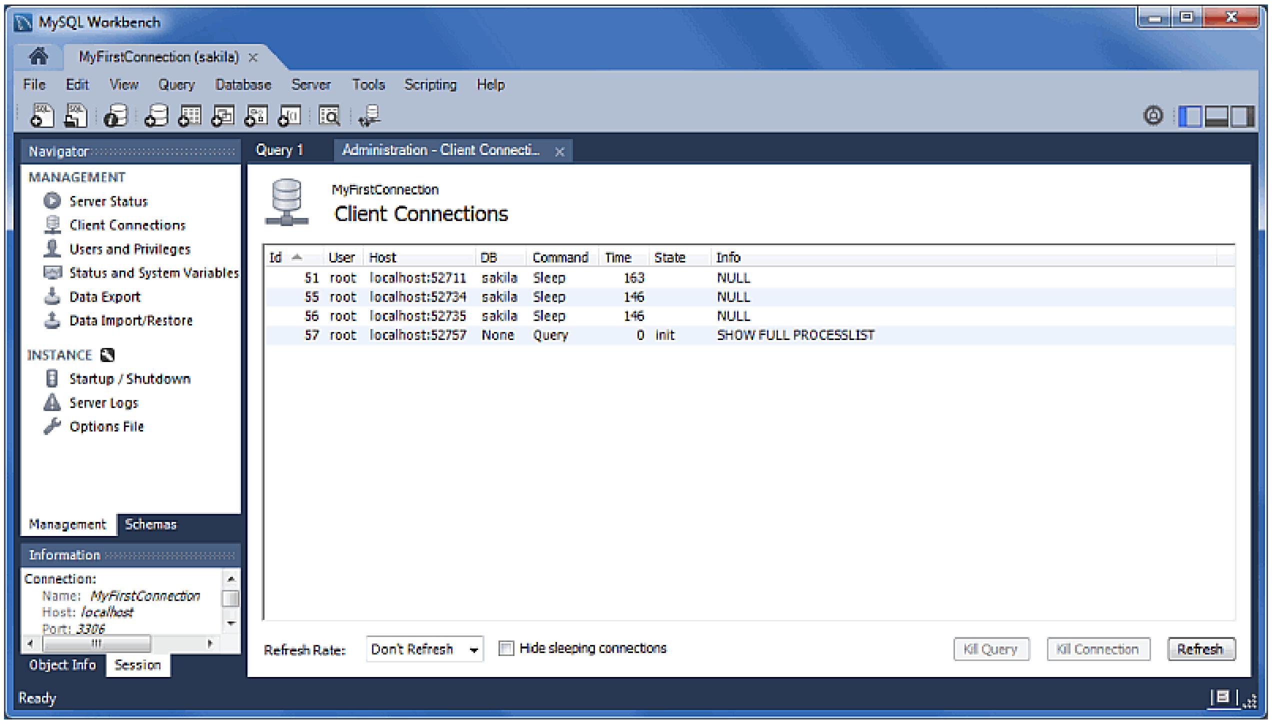
Columns:

actor_id smallint(5) UN AI PK

Additional Client Connection Information

Additional information was added to the **Client Connections** tab, such as Thread ID, Parent Thread, Instrumented, and Type.

Figure 1.33 Client Connections: MySQL Workbench 6.0



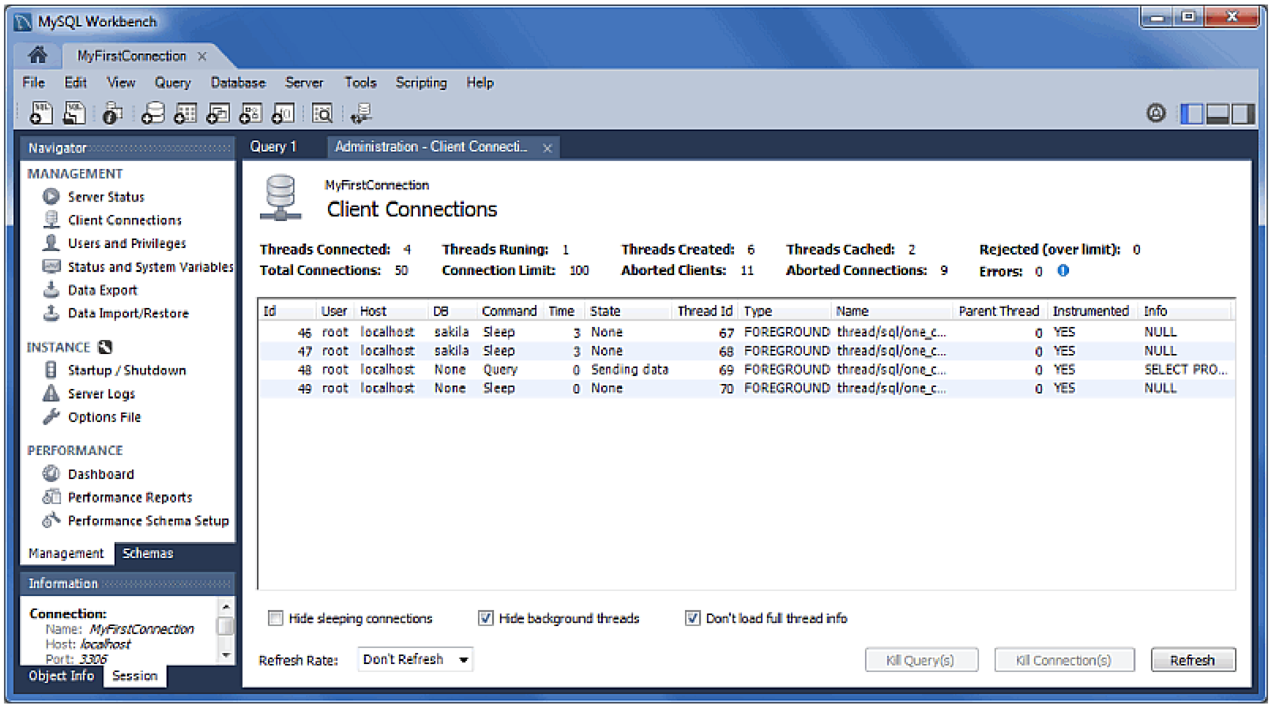
MyFirstConnection
Client Connections

Id	User	Host	DB	Command	Time	State	Info
51	root	localhost:52711	sakila	Sleep	163		NULL
55	root	localhost:52734	sakila	Sleep	146		NULL
56	root	localhost:52735	sakila	Sleep	146		NULL
57	root	localhost:52757	None	Query	0	init	SHOW FULL PROCESSLIST

Refresh Rate: Don't Refresh
Hide sleeping connections

Connection:
Name: *MyFirstConnection*
Host: *localhost*
Port: *3306*

Figure 1.34 Client Connections: MySQL Workbench 6.1



MyFirstConnection
Client Connections

Threads Connected: 4 **Threads Runing:** 1 **Threads Created:** 6 **Threads Cached:** 2 **Rejected (over limit):** 0
Total Connections: 50 **Connection Limit:** 100 **Aborted Clients:** 11 **Aborted Connections:** 9 **Errors:** 0

Id	User	Host	DB	Command	Time	State	Thread Id	Type	Name	Parent Thread	Instrumented	Info
46	root	localhost	sakila	Sleep	3	None	67	FOREGROUND	thread/sql/one_c...	0	YES	NULL
47	root	localhost	sakila	Sleep	3	None	68	FOREGROUND	thread/sql/one_c...	0	YES	NULL
48	root	localhost	None	Query	0	Sending data	69	FOREGROUND	thread/sql/one_c...	0	YES	SELECT PRO...
49	root	localhost	None	Sleep	0	None	70	FOREGROUND	thread/sql/one_c...	0	YES	NULL

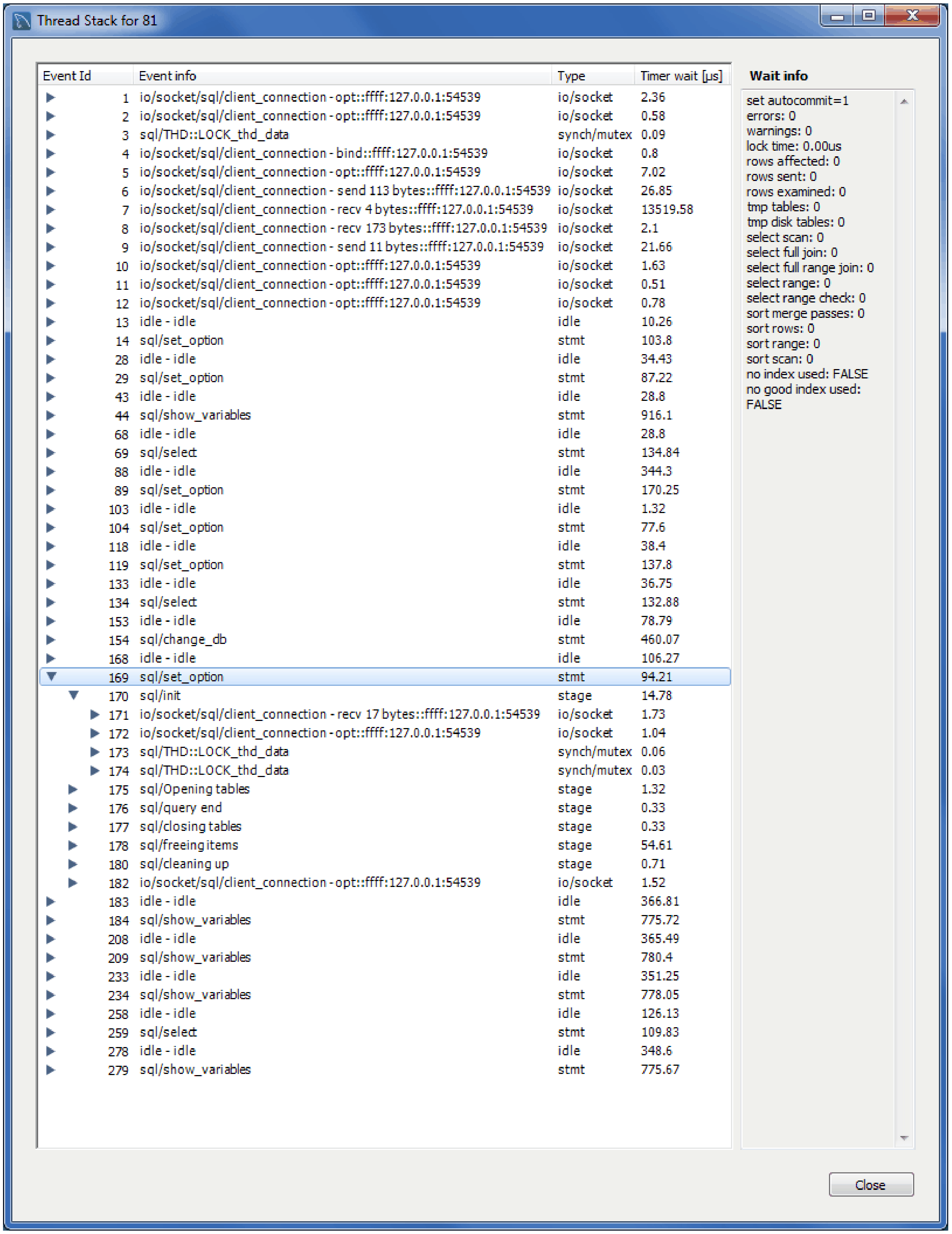
Hide sleeping connections [✓ Hide background threads] [✓ Don't load full thread info]

Refresh Rate: Don't Refresh

Connection:
Name: *MyFirstConnection*
Host: *localhost*
Port: *3306*

Also, a **Thread Stack** view option was added by right-clicking a connection entry in the **Client Connections** tab and choosing **View Thread Stack**.

Figure 1.35 Client Connections: View Thread Stack



Event Id	Event info	Type	Timer wait [µs]
1	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	2.36
2	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	0.58
3	sql/THD::LOCK_thd_data	synch/mutex	0.09
4	io/socket/sql/client_connection - bind::ffff:127.0.0.1:54539	io/socket	0.8
5	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	7.02
6	io/socket/sql/client_connection - send 113 bytes::ffff:127.0.0.1:54539	io/socket	26.85
7	io/socket/sql/client_connection - recv 4 bytes::ffff:127.0.0.1:54539	io/socket	13519.58
8	io/socket/sql/client_connection - recv 173 bytes::ffff:127.0.0.1:54539	io/socket	2.1
9	io/socket/sql/client_connection - send 11 bytes::ffff:127.0.0.1:54539	io/socket	21.66
10	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	1.63
11	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	0.51
12	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	0.78
13	idle - idle	idle	10.26
14	sql/set_option	stmt	103.8
28	idle - idle	idle	34.43
29	sql/set_option	stmt	87.22
43	idle - idle	idle	28.8
44	sql/show_variables	stmt	916.1
68	idle - idle	idle	28.8
69	sql/select	stmt	134.84
88	idle - idle	idle	344.3
89	sql/set_option	stmt	170.25
103	idle - idle	idle	1.32
104	sql/set_option	stmt	77.6
118	idle - idle	idle	38.4
119	sql/set_option	stmt	137.8
133	idle - idle	idle	36.75
134	sql/select	stmt	132.88
153	idle - idle	idle	78.79
154	sql/change_db	stmt	460.07
168	idle - idle	idle	106.27
169	sql/set_option	stmt	94.21
170	sql/init	stage	14.78
171	io/socket/sql/client_connection - recv 17 bytes::ffff:127.0.0.1:54539	io/socket	1.73
172	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	1.04
173	sql/THD::LOCK_thd_data	synch/mutex	0.06
174	sql/THD::LOCK_thd_data	synch/mutex	0.03
175	sql/Opening tables	stage	1.32
176	sql/query end	stage	0.33
177	sql/closing tables	stage	0.33
178	sql/freeing items	stage	54.61
180	sql/cleaning up	stage	0.71
182	io/socket/sql/client_connection - opt::ffff:127.0.0.1:54539	io/socket	1.52
183	idle - idle	idle	366.81
184	sql/show_variables	stmt	775.72
208	idle - idle	idle	365.49
209	sql/show_variables	stmt	780.4
233	idle - idle	idle	351.25
234	sql/show_variables	stmt	778.05
258	idle - idle	idle	126.13
259	sql/select	stmt	109.83
278	idle - idle	idle	348.6
279	sql/show_variables	stmt	775.67

Wait info

set autocommit=1
errors: 0
warnings: 0
lock time: 0.00us
rows affected: 0
rows sent: 0
rows examined: 0
tmp tables: 0
tmp disk tables: 0
select scan: 0
select full join: 0
select full range join: 0
select range: 0
select range check: 0
sort merge passes: 0
sort rows: 0
sort range: 0
sort scan: 0
no index used: FALSE
no good index used: FALSE

Additional Miscellaneous Additions

- MSAA (Windows Accessibility API) support and High contrast color theme in Microsoft Windows
- MySQL Enterprise Backup improvements
- Improvements with general performance and overall stability

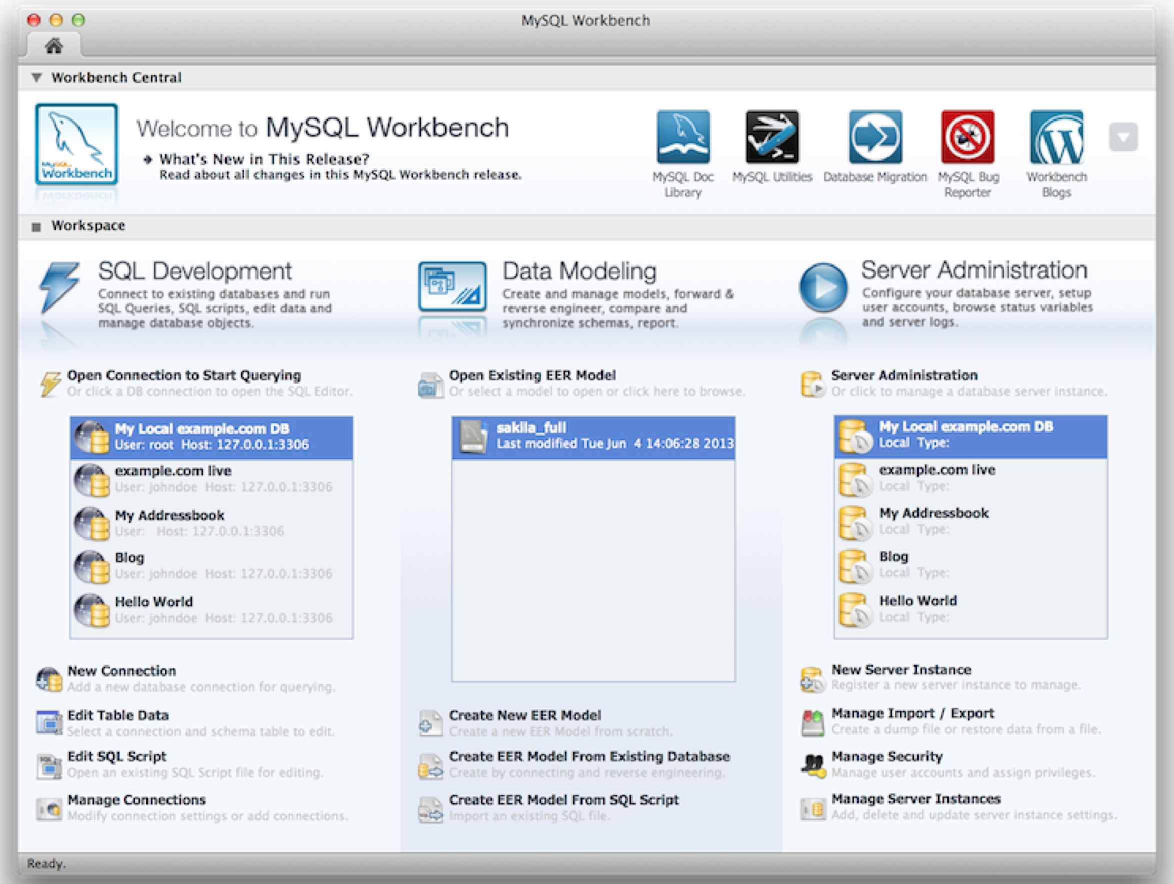
1.1.4 New in MySQL Workbench 6.0

This section summarizes many of the new features added to MySQL Workbench 6.0.0, in relation to MySQL Workbench 5.2.x;.

A new home screen

A new, modernized Home screen where major functionality of MySQL Workbench can be accessed, including connections to MySQL servers, modeling, migration, and the command-line utilities.

Figure 1.36 Home Screen: Workbench 5.2



MySQL Workbench

Workbench Central

Welcome to MySQL Workbench

What's New in This Release?
Read about all changes in this MySQL Workbench release.

MySQL Doc Library · MySQL Utilities · Database Migration · MySQL Bug Reporter · Workbench Blogs

Workspace

SQL Development
Connect to existing databases and run SQL Queries, SQL scripts, edit data and manage database objects.

Open Connection to Start Querying
Or click a DB connection to open the SQL Editor.

My Local example.com DB
User: root Host: 127.0.0.1:3306
example.com live
User: johndoe Host: 127.0.0.1:3306
My Addressbook
User: Host: 127.0.0.1:3306
Blog
User: johndoe Host: 127.0.0.1:3306
Hello World
User: johndoe Host: 127.0.0.1:3306

New Connection
Add a new database connection for querying.
Edit Table Data
Select a connection and schema table to edit.
Edit SQL Script
Open an existing SQL Script file for editing.
Manage Connections
Modify connection settings or add connections.

Data Modeling
Create and manage models, forward & reverse engineer, compare and synchronize schemas, report.

Open Existing EER Model
Or select a model to open or click here to browse.

sakila_full
Last modified Tue Jun 4 14:06:28 2013

Create New EER Model
Create a new EER Model from scratch.
Create EER Model From Existing Database
Create by connecting and reverse engineering.
Create EER Model From SQL Script
Import an existing SQL file.

Server Administration
Configure your database server, setup user accounts, browse status variables and server logs.

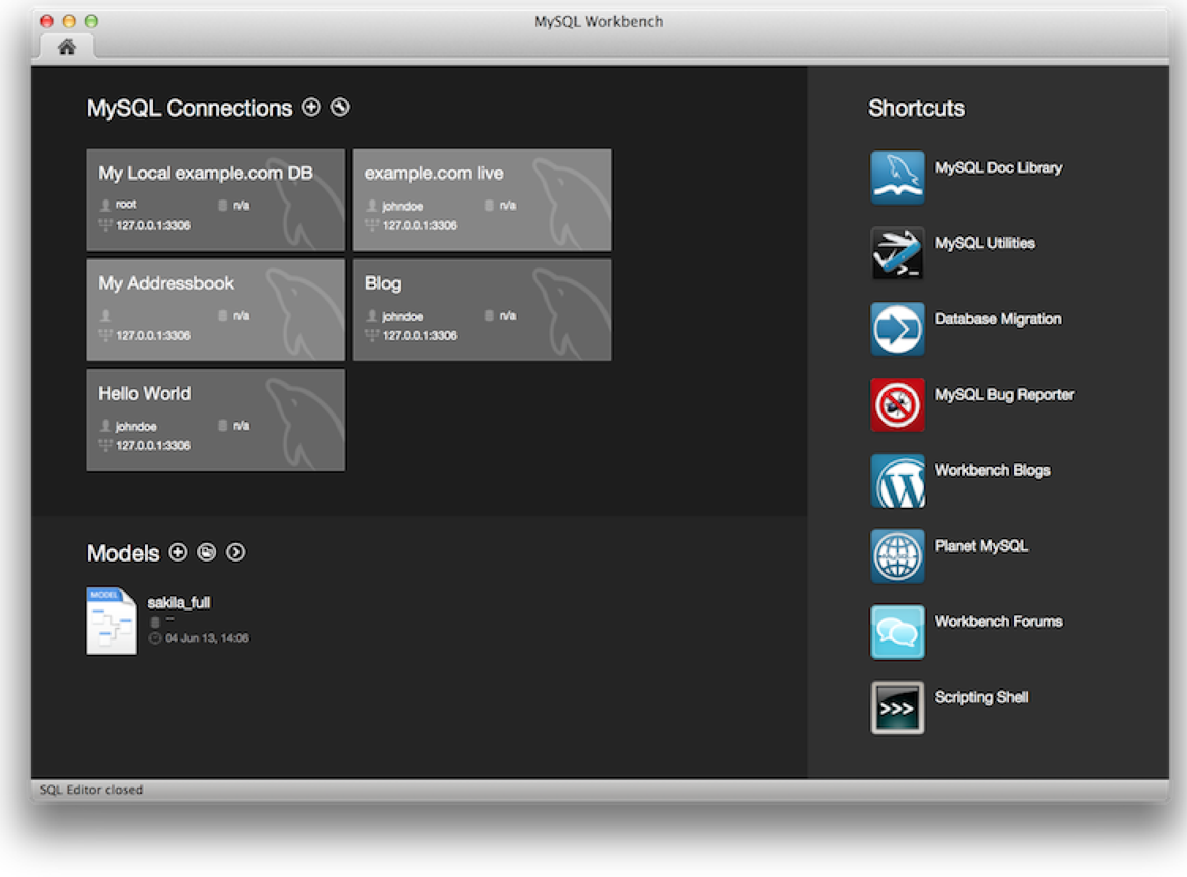
Server Administration
Or click to manage a database server instance.

My Local example.com DB
Local Type:
example.com live
Local Type:
My Addressbook
Local Type:
Blog
Local Type:
Hello World
Local Type:

New Server Instance
Register a new server instance to manage.
Manage Import / Export
Create a dump file or restore data from a file.
Manage Security
Manage user accounts and assign privileges.
Manage Server Instances
Add, delete and update server instance settings.

Ready.

Figure 1.37 Home Screen: Workbench 6.0



Unified SQL Editor and Administration interface

In the new user interface, the Server Administration functionality (such as start/stop server, managing user accounts etc) is now accessible directly from the SQL Editor interface, located near where the schema information can be browsed and queries executed.

The image below contains three screenshots of the Schema window in the SQL Editor. The first is from MySQL Workbench 5.2, the second is MySQL Workbench 6.0 with the management tab collapsed, and the third shows what the merged management tab looks like. Toggle the merged and tabbed views by clicking the new merge button next to the refresh button.

Figure 1.38 Comparing the SQL Editor interface for Workbench 5.2 and 6.0

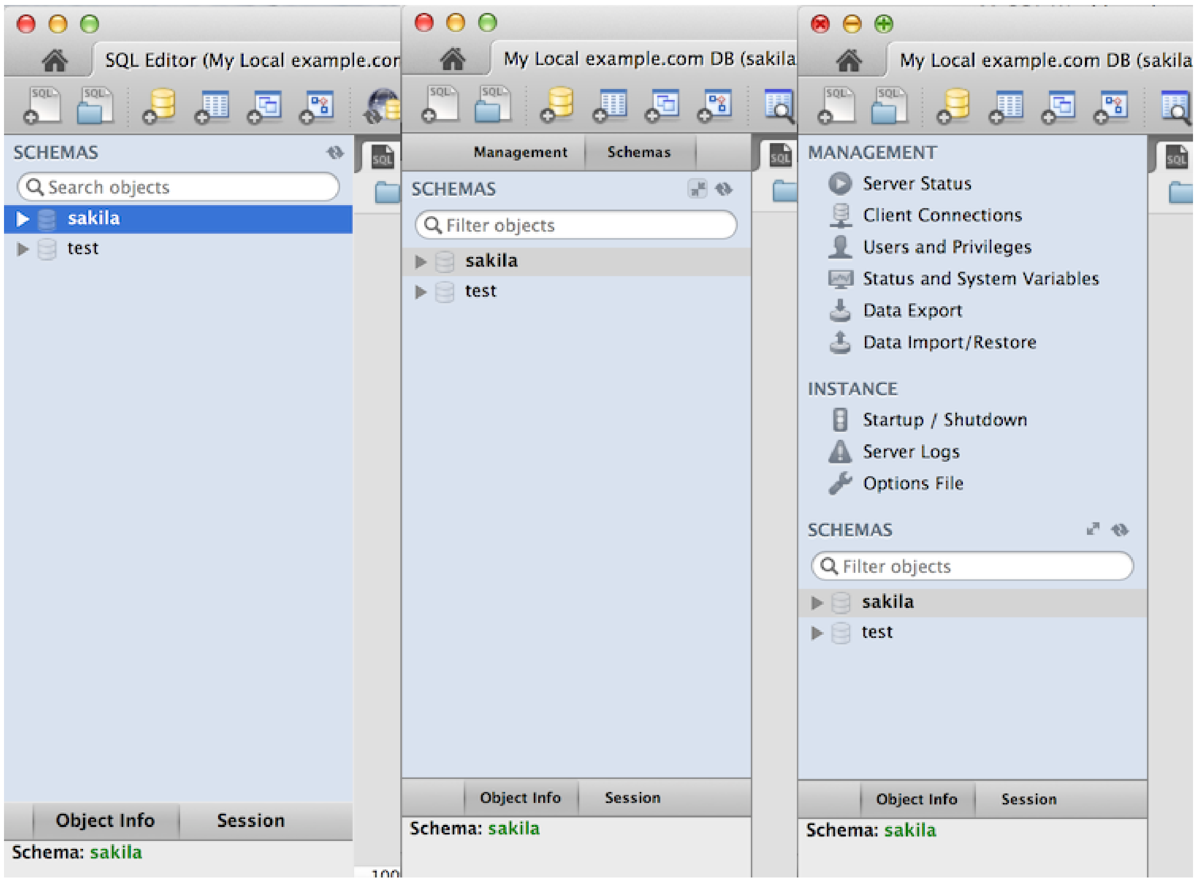
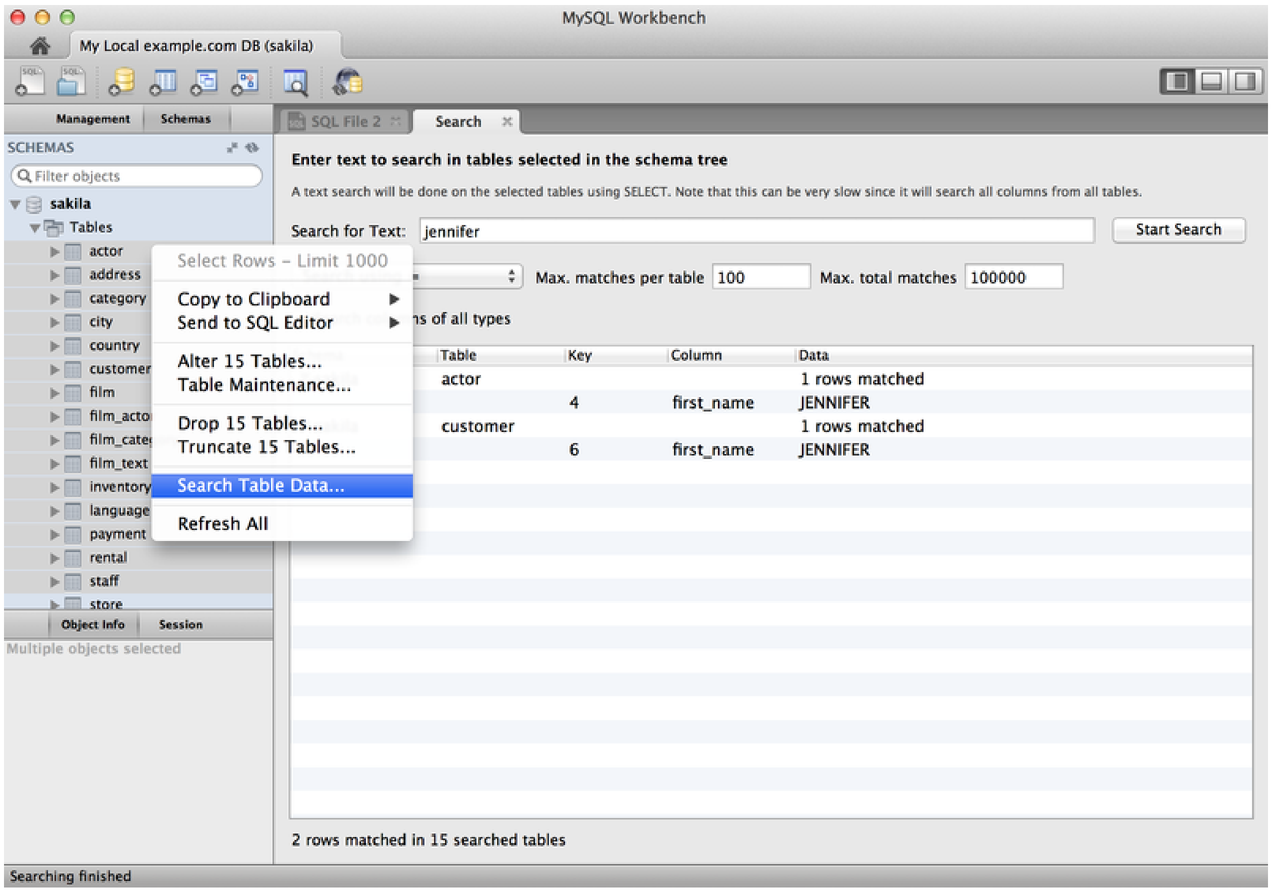


Table data search

You can select schemas and/or tables to perform client-side searches for user specified strings and patterns. To access this new search feature, right click select a schema or a table in the left sidebar and select **Search Table Data...**.

This screenshot demonstrates the search feature, along with an example search. Multiple tables were selected and searched in this example:

Figure 1.39 Table search functionality



Enter text to search in tables selected in the schema tree

A text search will be done on the selected tables using SELECT. Note that this can be very slow since it will search all columns from all tables.

Search for Text: jennifer

Max. matches per table 100 Max. total matches 100000

...ns of all types

Table	Key	Column	Data
actor			1 rows matched
	4	first_name	JENNIFER
customer			1 rows matched
	6	first_name	JENNIFER

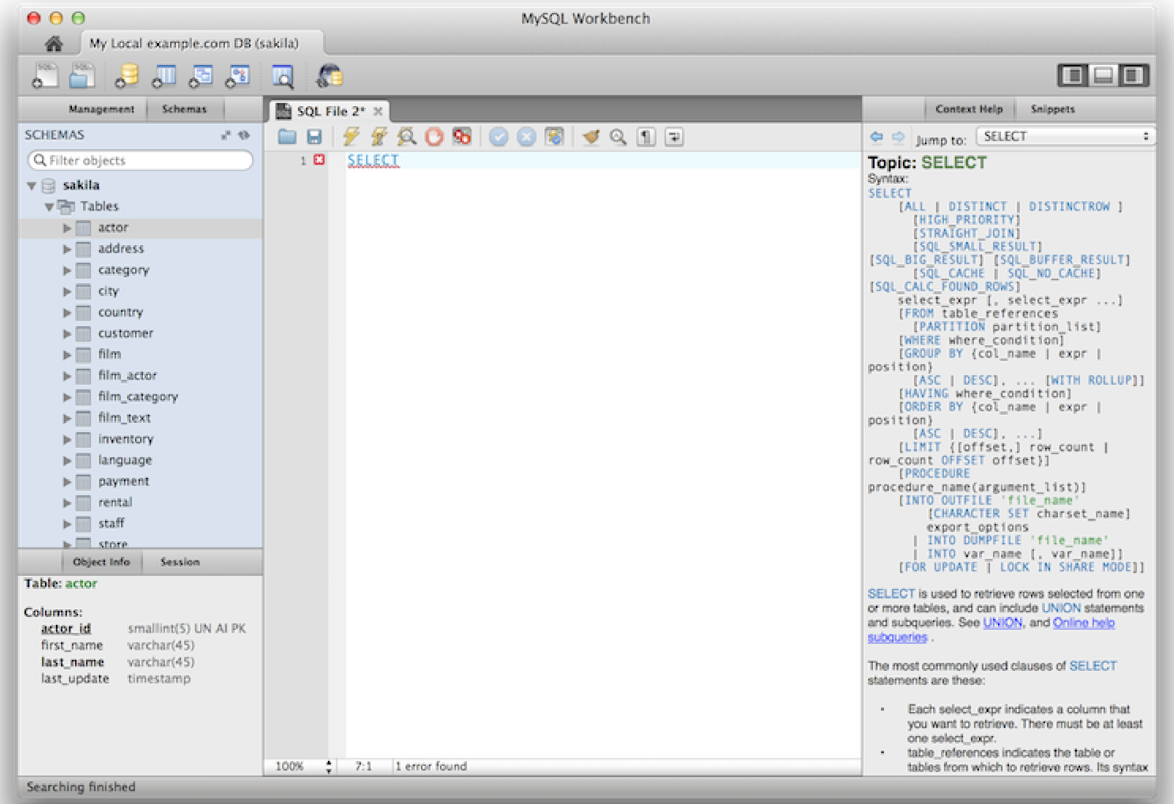
2 rows matched in 15 searched tables

For additional information, see Section 8.1.8, “Table Data Search Panel”.

Context Sensitive help for the SQL Editor

Select a keyword or function in your query and after a delay it will show formatted help information from the MySQL Server (equivalent to using the help command from the command-line MySQL Client).

Figure 1.40 Context Sensitive Help



```
SELECT
```

Table: actor

Columns:

actor_id	smallint(5) UN AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

Topic: SELECT

Syntax:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT]
[SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE]
[SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr |
position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr |
position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count |
row_count OFFSET offset}]
    [PROCEDURE
procedure_name(argument_list)]
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT is used to retrieve rows selected from one or more tables, and can include UNION statements and subqueries. See UNION, and Online help subqueries .

The most commonly used clauses of SELECT statements are these:

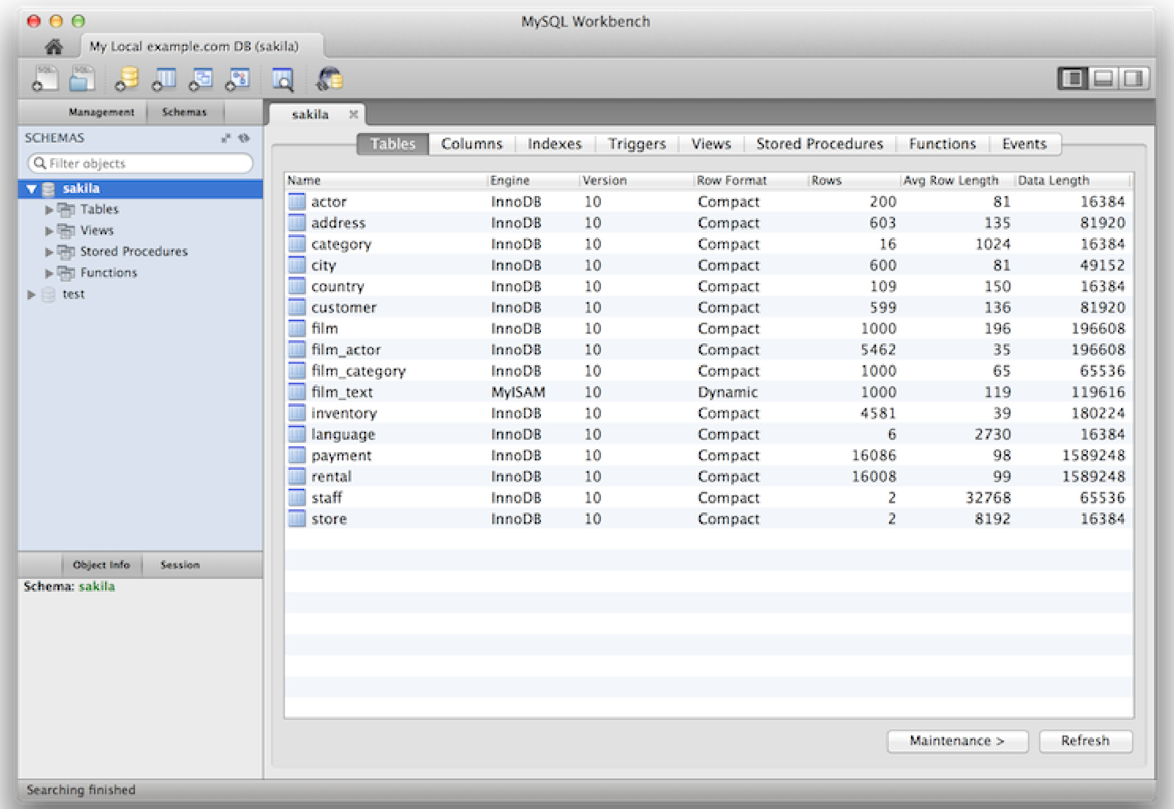
- Each select_expr indicates a column that you want to retrieve. There must be at least one select_expr.
- table_references indicates the table or tables from which to retrieve rows. Its syntax

For additional information, see Section 8.1.6, “Context Sensitive Help”.

Schema Inspector

New Schema Inspector feature allows you to browse general information from schema objects. For tables, it's also possible to perform maintenance tasks such as `ANALYZE`, `OPTIMIZE`, `CHECK`, and `CHECKSUM TABLE`. To access the inspector, right-click a schema and select the **Schema Inspector**

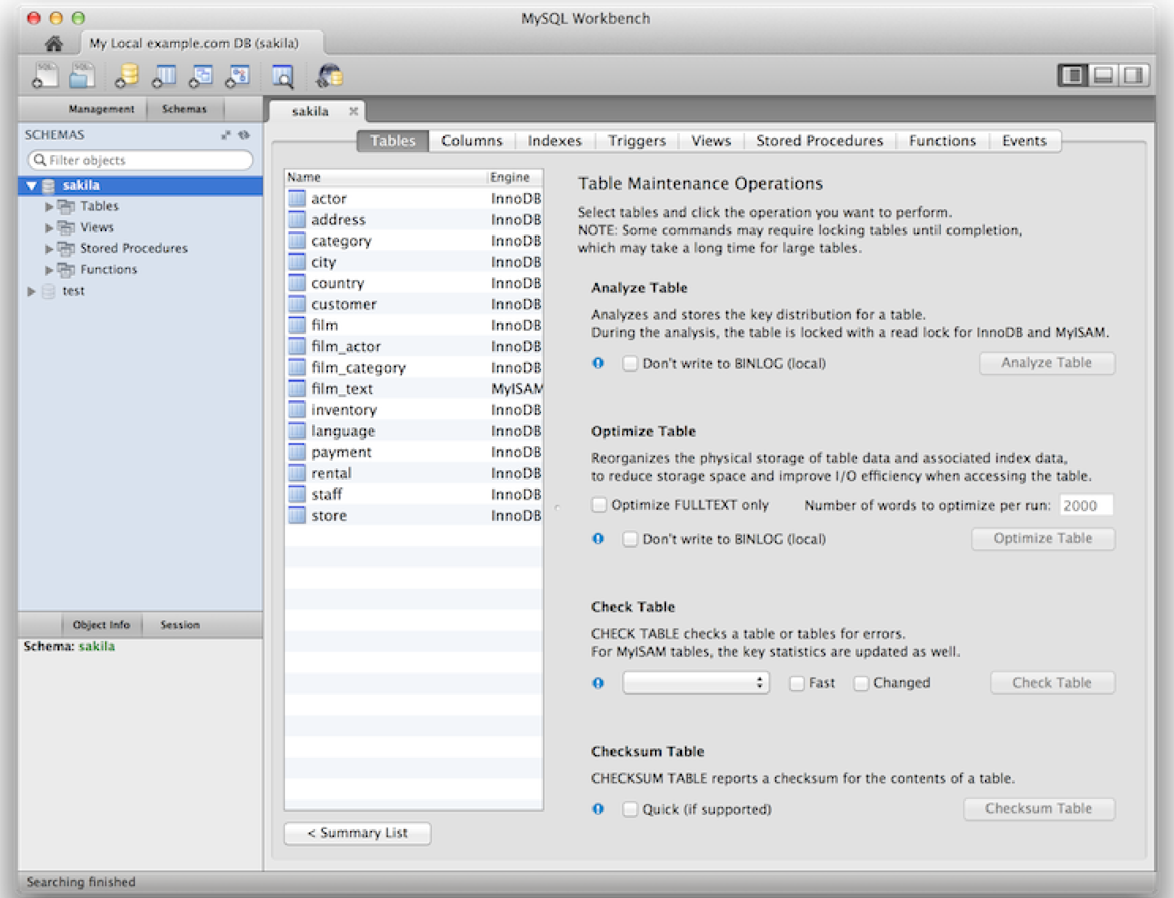
Figure 1.41 Schema Inspector



Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length
actor	InnoDB	10	Compact	200	81	16384
address	InnoDB	10	Compact	603	135	81920
category	InnoDB	10	Compact	16	1024	16384
city	InnoDB	10	Compact	600	81	49152
country	InnoDB	10	Compact	109	150	16384
customer	InnoDB	10	Compact	599	136	81920
film	InnoDB	10	Compact	1000	196	196608
film_actor	InnoDB	10	Compact	5462	35	196608
film_category	InnoDB	10	Compact	1000	65	65536
film_text	MyISAM	10	Dynamic	1000	119	119616
inventory	InnoDB	10	Compact	4581	39	180224
language	InnoDB	10	Compact	6	2730	16384
payment	InnoDB	10	Compact	16086	98	1589248
rental	InnoDB	10	Compact	16008	99	1589248
staff	InnoDB	10	Compact	2	32768	65536
store	InnoDB	10	Compact	2	8192	16384

And choosing **Maintenance** for a table:

Figure 1.42 Schema Inspector: Maintenance



Schema: sakila

Name	Engine
actor	InnoDB
address	InnoDB
category	InnoDB
city	InnoDB
country	InnoDB
customer	InnoDB
film	InnoDB
film_actor	InnoDB
film_category	InnoDB
film_text	MyISAM
inventory	InnoDB
language	InnoDB
payment	InnoDB
rental	InnoDB
staff	InnoDB
store	InnoDB

Table Maintenance Operations

Select tables and click the operation you want to perform.
NOTE: Some commands may require locking tables until completion,
which may take a long time for large tables.

Analyze Table

Analyzes and stores the key distribution for a table.
During the analysis, the table is locked with a read lock for InnoDB and MyISAM.

☐ Don't write to BINLOG (local)

Optimize Table

Reorganizes the physical storage of table data and associated index data,
to reduce storage space and improve I/O efficiency when accessing the table.

☐ Optimize FULLTEXT only   Number of words to optimize per run: 2000

☐ Don't write to BINLOG (local)

Check Table

CHECK TABLE checks a table or tables for errors.
For MyISAM tables, the key statistics are updated as well.

☐ Fast ☐ Changed

Checksum Table

CHECKSUM TABLE reports a checksum for the contents of a table.

☐ Quick (if supported)

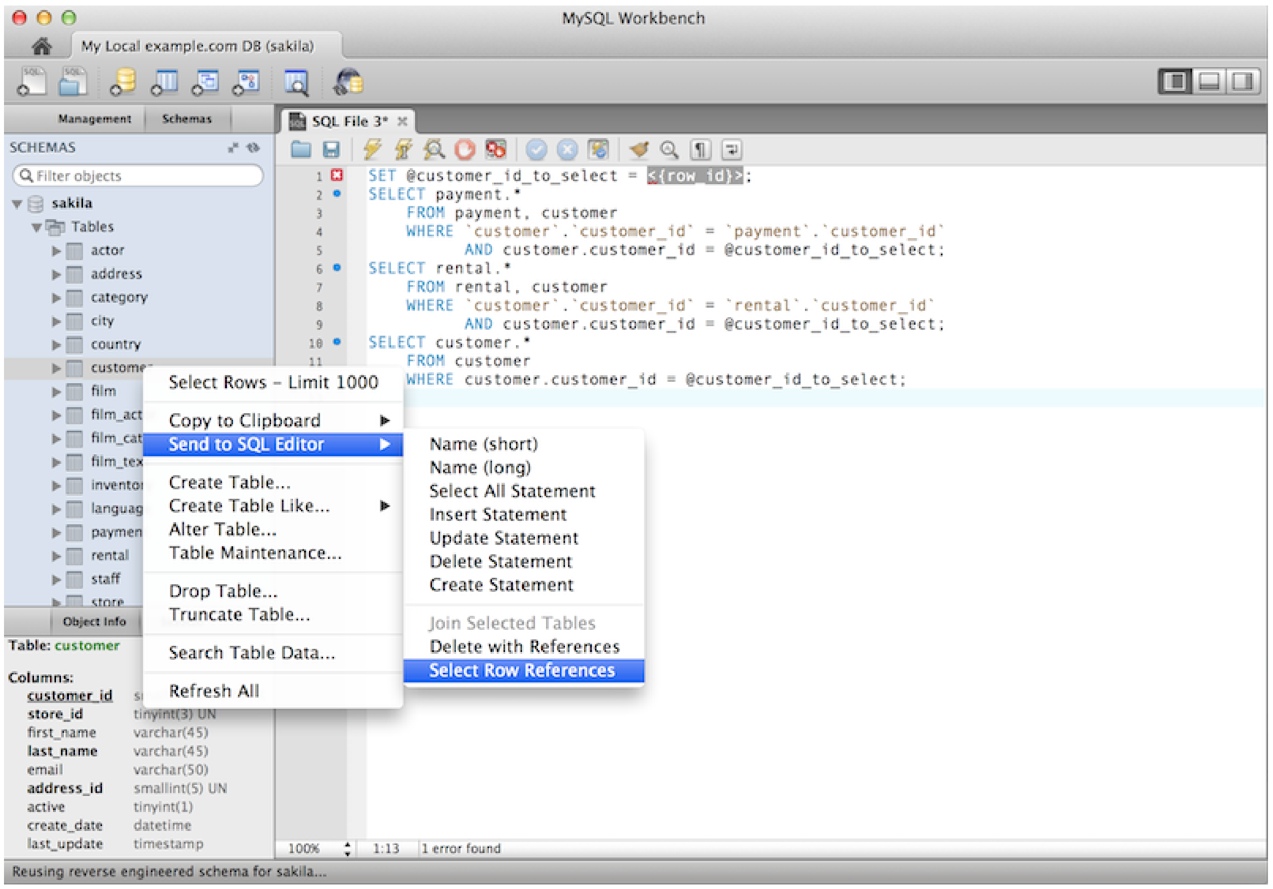
Searching finished

For additional information, see Schema Inspector.

Cascaded DELETE statements generator

You can generate a series of `DELETE` statements needed to delete a row from that table, which includes rows from other tables that reference it, recursively. The `SELECT` version allows you to preview what rows would be deleted. Right click a table and select **Copy to Clipboard**, **Delete with References**.

Figure 1.43 Cascading SELECT



```
SET @customer_id_to_select = ${row_id};
SELECT payment.*
    FROM payment, customer
    WHERE `customer`.`customer_id` = `payment`.`customer_id`
        AND customer.customer_id = @customer_id_to_select;
SELECT rental.*
    FROM rental, customer
    WHERE `customer`.`customer_id` = `rental`.`customer_id`
        AND customer.customer_id = @customer_id_to_select;
SELECT customer.*
    FROM customer
    WHERE customer.customer_id = @customer_id_to_select;
```

Table: customer

Columns:

customer_id	s
store_id	tinyint(3) UN
first_name	varchar(45)
last_name	varchar(45)
email	varchar(50)
address_id	smallint(5) UN
active	tinyint(1)
create_date	datetime
last_update	timestamp

1 error found

Reusing reverse engineered schema for sakila...

Table templates

Define templates of tables with commonly used columns, to be used to create new tables in a live connection or in an EER model. In the SQL Editor, choose **Create Table Like...**, or in Modeling, use the right sidebar. For additional information, see Section 9.6, “Table Templates”.

Vertical Text

A Vertical Text output option for queries (equivalent to \G from the command-line Client) was added. To execute, choose **Query**, **Execute Current Statement (Vertical Text Output)**.

Figure 1.44 Vertical Text (\G)

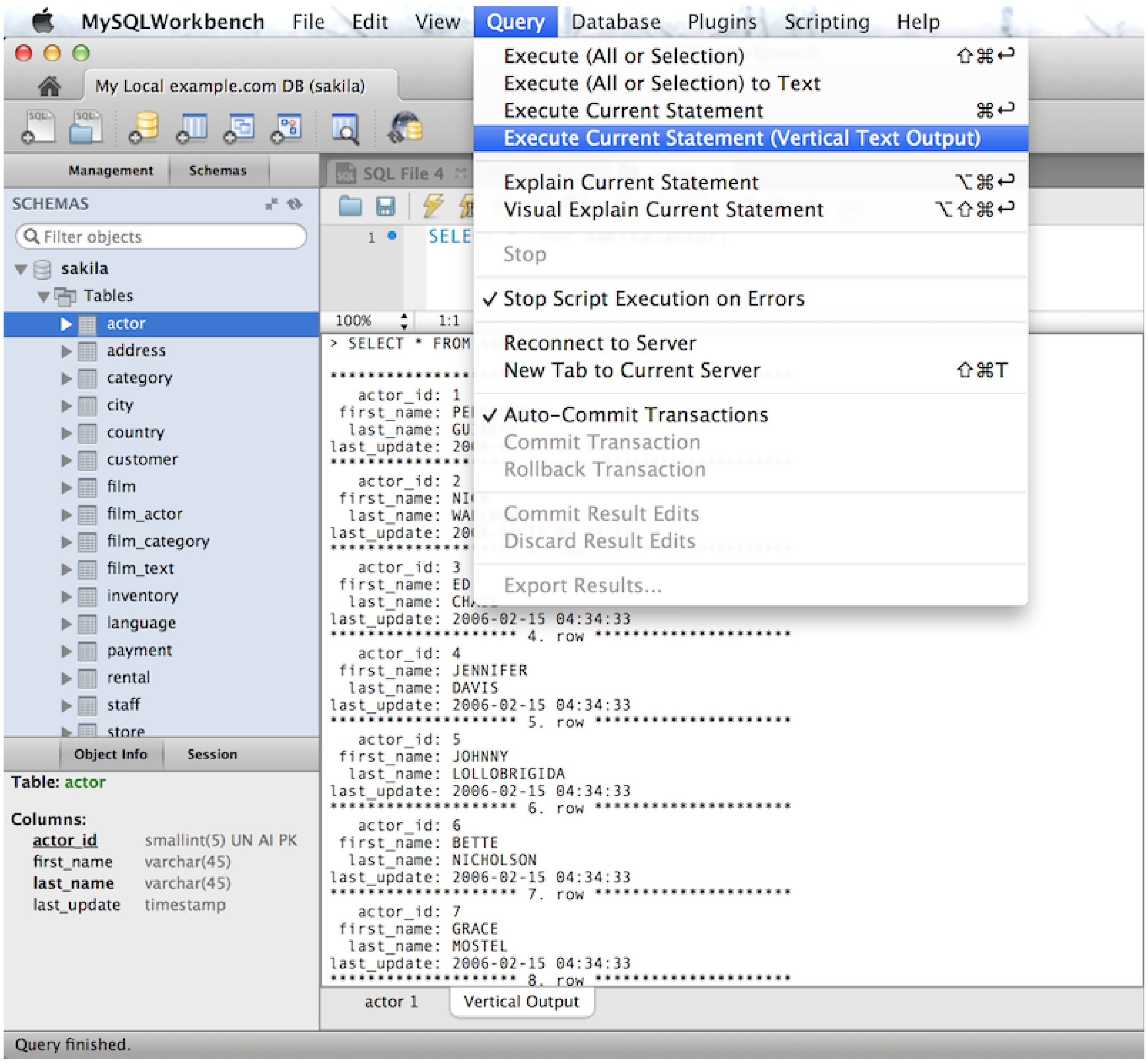


Table: actor

Columns:

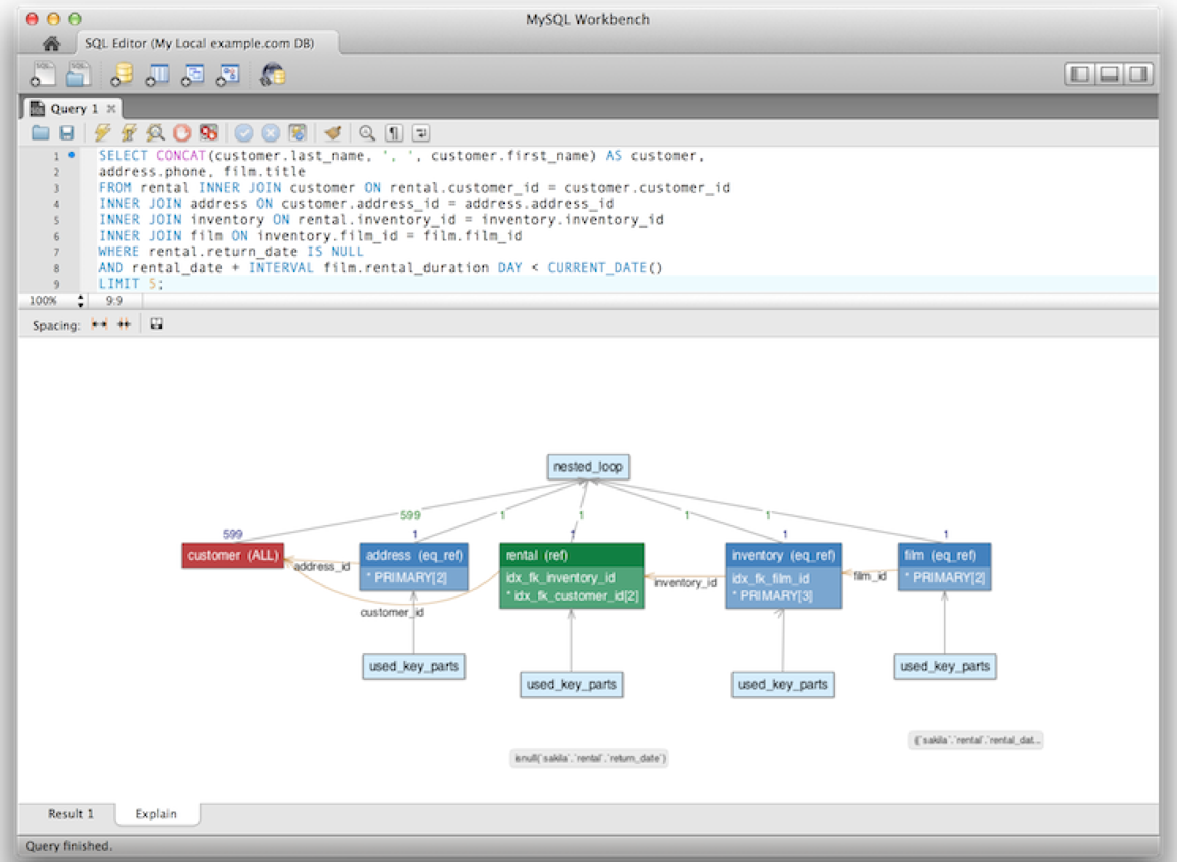
actor_id	smallint(5) UN AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

```
> SELECT * FROM
***************
   actor_id: 1
 first_name: PE
  last_name: GU
last_update: 20
***************
   actor_id: 2
 first_name: NI
  last_name: WA
last_update: 20
***************
   actor_id: 3
 first_name: ED
  last_name: CH
last_update: 2006-02-15 04:34:33
******************** 4. row ********************
   actor_id: 4
 first_name: JENNIFER
  last_name: DAVIS
last_update: 2006-02-15 04:34:33
******************** 5. row ********************
   actor_id: 5
 first_name: JOHNNY
  last_name: LOLLOBRIGIDA
last_update: 2006-02-15 04:34:33
******************** 6. row ********************
   actor_id: 6
 first_name: BETTE
  last_name: NICHOLSON
last_update: 2006-02-15 04:34:33
******************** 7. row ********************
   actor_id: 7
 first_name: GRACE
  last_name: MOSTEL
last_update: 2006-02-15 04:34:33
******************** 8. row ********************
```

Improved Visual Explain

The Visual Explain output was improved.

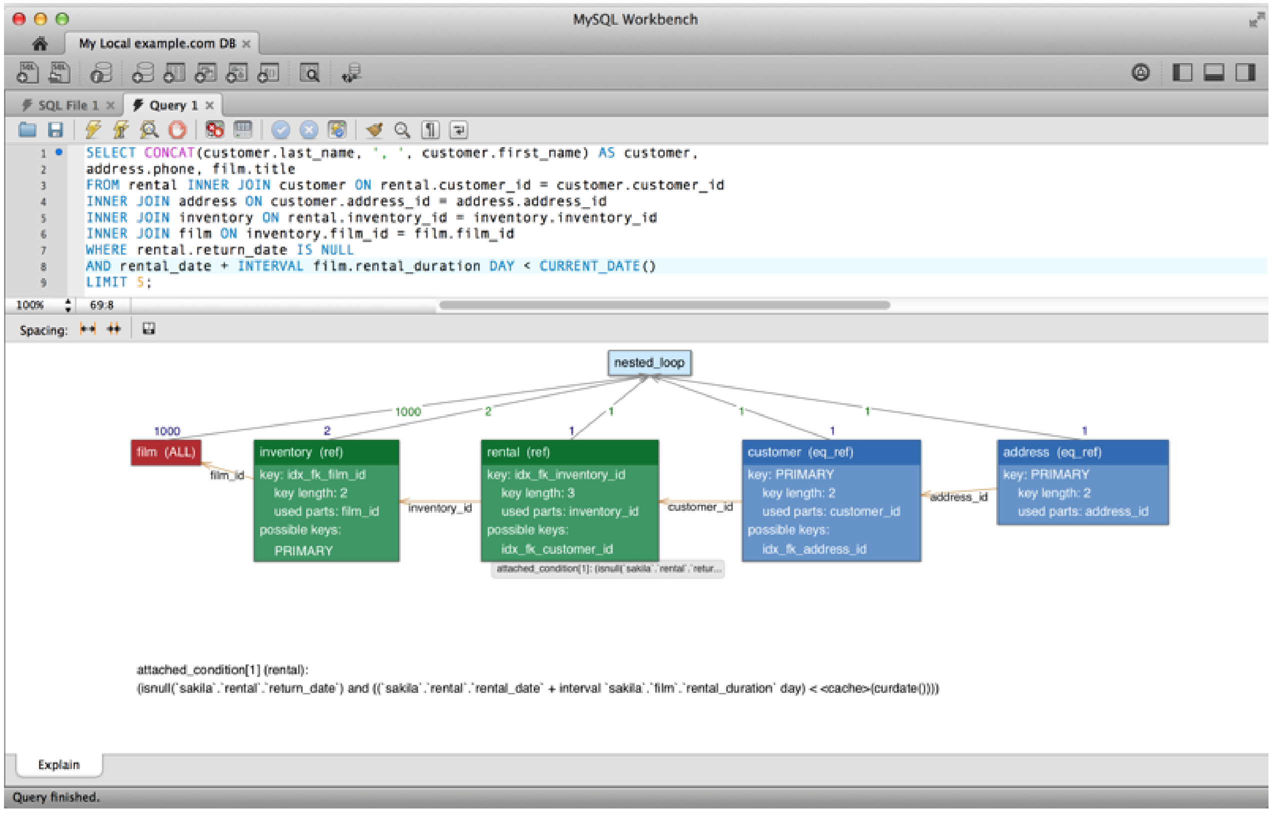
Figure 1.45 Visual Explain: Workbench 5.2



```
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
address.phone, film.title
FROM rental INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
LIMIT 5;
```

Query finished.

Figure 1.46 Visual Explain: Workbench 6.0



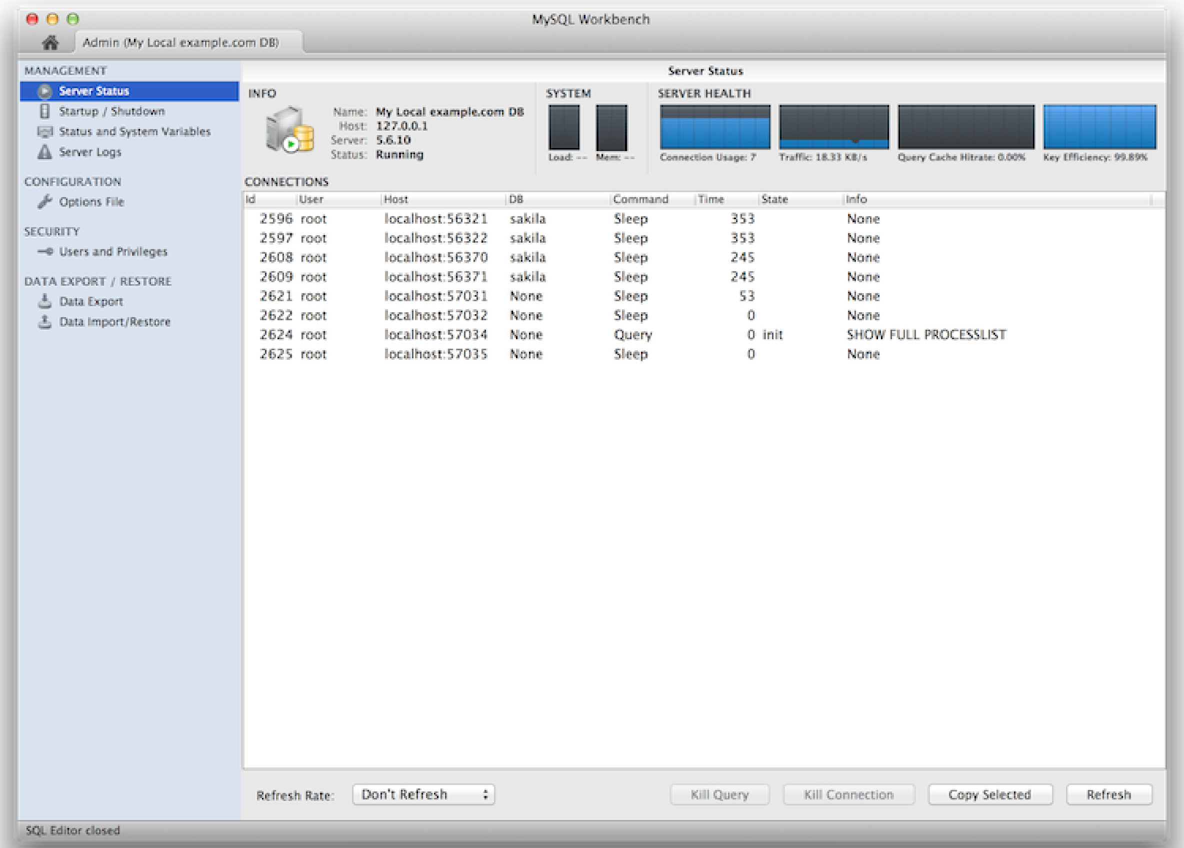
```
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer,
address.phone, film.title
FROM rental INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
LIMIT 5;
```

```
attached_condition[1] (rental):
(isnull(`sakila`.`rental`.`return_date`) and ((`sakila`.`rental`.`rental_date` + interval `sakila`.`film`.`rental_duration` day) < <cache>(curdate())))
```

Improved Server Status

Additional server status information was added, and the user interface was improved. Select **Server Status** from the **Management** tab to open this window.

Figure 1.47 Server Status: Workbench 5.2



Name: My Local example.com DB
Host: 127.0.0.1
Server: 5.6.10
Status: Running

CONNECTIONS

Id	User	Host	DB	Command	Time	State	Info
2596	root	localhost:56321	sakila	Sleep	353		None
2597	root	localhost:56322	sakila	Sleep	353		None
2608	root	localhost:56370	sakila	Sleep	245		None
2609	root	localhost:56371	sakila	Sleep	245		None
2621	root	localhost:57031	None	Sleep	53		None
2622	root	localhost:57032	None	Sleep	0		None
2624	root	localhost:57034	None	Query	0	init	SHOW FULL PROCESSLIST
2625	root	localhost:57035	None	Sleep	0		None

Refresh Rate: Don't Refresh

Figure 1.48 Server Status: Workbench 6.0

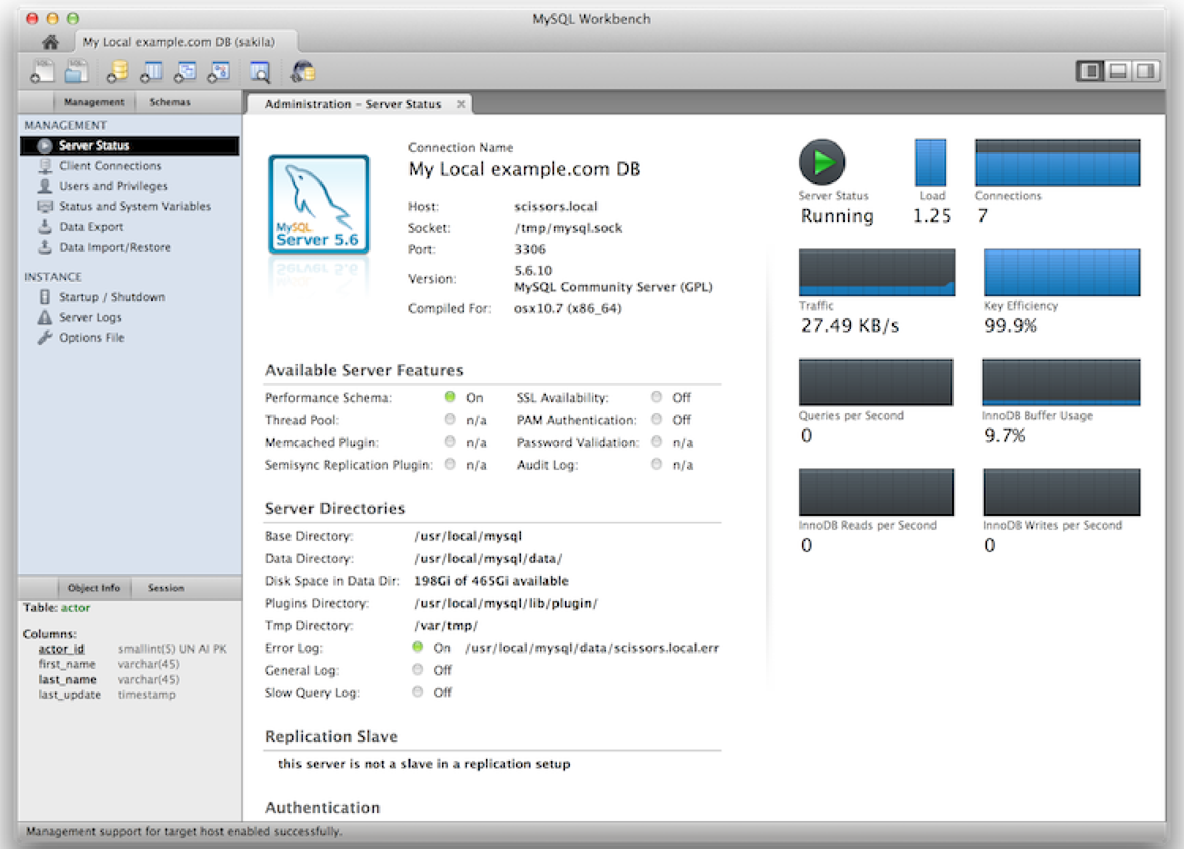


Table: actor

Columns:

actor_id	smallint(5) UN AI PK
first_name	varchar(45)
last_name	varchar(45)
last_update	timestamp

Connection Name
My Local example.com DB

Host:	scissors.local
Socket:	/tmp/mysql.sock
Port:	3306
Version:	5.6.10 MySQL Community Server (GPL)
Compiled For:	osx10.7 (x86_64)

Server Status	Running
Load	1.25
Connections	7
Traffic	27.49 KB/s
Key Efficiency	99.9%
Queries per Second	0
InnoDB Buffer Usage	9.7%
InnoDB Reads per Second	0
InnoDB Writes per Second	0

Available Server Features

Performance Schema:	On	SSL Availability:	Off
Thread Pool:	n/a	PAM Authentication:	Off
Memcached Plugin:	n/a	Password Validation:	n/a
Semisync Replication Plugin:	n/a	Audit Log:	n/a

Server Directories

Base Directory:	/usr/local/mysql
Data Directory:	/usr/local/mysql/data/
Disk Space in Data Dir:	198Gi of 465Gi available
Plugins Directory:	/usr/local/mysql/lib/plugin/
Tmp Directory:	/var/tmp/
Error Log:	On /usr/local/mysql/data/scissors.local.err
General Log:	Off
Slow Query Log:	Off

Replication Slave

this server is not a slave in a replication setup

Authentication

Management support for target host enabled successfully.

Enterprise Features

Support for MySQL Enterprise features in the Commercial edition of MySQL Workbench was added. From within the **Management** tab for an open connection, look for the following products under the heading **MySQL Enterprise**:

`MySQL Enterprise Backup (MEB)`: A GUI front end for the MEB tool. After installing a commercial version of MySQL Workbench and MySQL Enterprise Backup, MySQL Workbench will check for and handle the pre-requisites. Backup recovery is also supported. This plugin supports MEB with local and remote installations of Linux and OS X, and locally for MySQL Windows.

`MySQL Audit Log Inspector`: A GUI for browsing the contents of generated logs by the commercial Audit Log Plugin. Powerful filtering and search capabilities are available. Fast browsing is provided by caching the log data locally in an encrypted file. This plugin supports MEB with local and remote installations of Linux and OS X, and locally for MySQL Windows.

Database Migration Features

SQL Anywhere and SQLite are now supported.

1.2 MySQL Workbench Editions

MySQL Workbench is available in the following editions:

- `Community Edition` (Open Source, GPL) -- This is the foundation for all other editions
- `Standard Edition` (Commercial)
- `Enterprise Edition` (Commercial)

For details about each edition, see http://www.mysql.com/products/workbench/features.html

For more information about the Enterprise edition, visit http://www.mysql.com/enterprise

Chapter 2 Installation

Table of Contents

MySQL Workbench is available for Windows, Linux, and OS X.

Binary distributions of MySQL Workbench are available for the preceding platforms. Source code distributions are also available as a `tar.gz` package, or an RPM package.

MySQL Workbench downloads are available at http://dev.mysql.com/downloads/workbench/. The source code is also available on GitHub.

The following sections explain the installation process for each of these platforms.

2.1 System Requirements

MySQL Workbench is available on a number of operating systems and platforms. For information about those platforms that are officially supported, see http://www.mysql.com/support/supportedplatforms/workbench.html on the MySQL Web site.

General Requirements

General requirements and considerations that apply to all operating systems.

- **MySQL server**: Although it is not required, MySQL Workbench is designed to have either a remote or local MySQL server connection. For additional information about connecting to a MySQL server, see Chapter 5, *MySQL Connections*. For additional information about installing a MySQL server, see Installing and Upgrading MySQL.

  Data modeling does not require a MySQL server connection.

  Some features take advantage of MySQL server features, and as such, they require more recent versions of MySQL Server. For example, the **Performance Dashboard** requires MySQL Server 5.6 or higher.

- **Simultaneous client connections**: Opening a MySQL connection from the MySQL Workbench home page opens a new connection tab in MySQL Workbench for that connection. Each of these tabs requires two MySQL connections to perform basic tasks, such schema discovery and SQL execution. Additionally, performing management related tasks, such as **Server Status**, requires two additional MySQL connections. Essentially, this means that each MySQL connection tab in MySQL Workbench requires four available connections to MySQL. For additional information about "Too many connection" related errors, see Too many connections.

This connection requirement doubles with each connection tab opened in MySQL Workbench, even if the two connection tabs point to the same MySQL server. SQL editor tabs share their connections, so having multiple SQL editor and SQL results tabs does not affect the number of required connections.

Note

On startup, the application checks the OpenGL version and chooses between software and hardware rendering. To determine which rendering method is being used, open the **Help** menu and choose the **System Info** item.

Requirements for Linux

- The requirements for Linux are embedded within their respective packages. Use the platform specific tool (for example, yum or apt) to install the package and their dependencies.
- The "Save password in keychain" functionality requires `gnome-keyring` to store the passwords. Note that on KDE systems, the `gnome-keyring` daemon is not started by default.
- For Linux and OS X, the MySQL server administration features require `sudo` privileges to execute several commands. The sudo user must be capable of executing the following system commands:

```
/usr/bin/sudo
/usr/bin/nohup
/usr/bin/uptime
/usr/bin/which
/usr/bin/stat

/bin/bash
/bin/mkdir
/bin/rm
/bin/rmdir
/bin/dd
/bin/cp
/bin/ls
```

  Additionally, the sudo user must keep the `HOME` environment variable when executing system commands, which means adding the following to `/etc/sudoers`:

```
env_keep +="HOME"
```

  For MySQL Workbench to execute MySQL Enterprise Backup commands, the sudo user must also be able to execute the MySQL Enterprise Backup binary.

Requirements for Windows

- Microsoft .NET 4.0 Framework
- Microsoft Visual C++ 2013 Redistributable Package (MSVC2013)

Note

The 2010 version was used in previous editions of MySQL Workbench 6.

- Windows 7 and above

Note

MySQL Workbench 6.1 supports earlier versions of Windows, including Vista

2.2 Command-line options

In addition to platform-specific command-line options, MySQL Workbench has the following command-line options:

Note

On Microsoft Windows, the command-line options contain one leading dash instead of two. For example, use `-log-level` for Microsoft Windows and `--log-level` for Linux and OS X.

- `--log-level` *`level`*: Controls the verbosity level for logging output from Workbench.

  With increasingly levels of verbosity, the valid values for `level` are: error, warning, info, debug1, debug2, and debug3.

  The location of the generated log files, such as `wb.log`, are as follows:

  Table 2.1 The default location of generated MySQL Workbench log files

Platform	Default location
Linux	`~/.mysql/workbench/log/`
OS X	`~/Library/Application Support/Workbench/log/`
Microsoft Windows	`C:\Users\`*`user_name`*`AppData\Roaming\MySQL\Workbench\log\`

- `--admin` *`instance`*: Open an administration tab to the named MySQL instance.
- `--upgrade-mysql-dbs`: Open the Migration Wizard tab.
- `--migration`: Open the Migration Wizard tab.
- `--log-to-stderr`: Also log to `stderr`.
- `--version`: Show MySQL Workbench version number and exit.
- `--verbose, -v`: Enable diagnostics output.
- `--query` *`[connection|connection_string]`*:
  - Empty: Open a query tab and prompts for a connection.
  - Connection: Open a named connection.
  - Connection_string: Create a connection based on the entered connection string, which should be in the form *`$USER@$HOST:$PORT`*.

- `--model` *`modelfile`*: open the given EER model file.
- `--script` *`script`*: Open the given SQL file in a connection, typically used with the `--query` parameter.
- `--run` *`code`*: Execute the given code using the default language for GRT shell.
- `--run-python` *`script`*: Execute the given code in Python.
- `--run-script` *`file`*: Execute Python code from a file.
- `--open` *`file`*: Open the given file at startup. Deprecated, so instead use specific types such as `--script` or `--model`.
- `--quit-when-done`: Quits MySQL Workbench after `--script` or `--run` finishes.

2.3 MySQL Workbench on Windows

2.3.1 Installing

MySQL Workbench for Windows can be installed using the MySQL Installer that installs and updates all MySQL products on Windows, the standalone .msi installation package, or manually from a Zip file.

Important

Installing MySQL Workbench using an Installer package requires either Administrator or Power User privileges. If you are using the Zip file without an installer, you do not need Administrator or Power User privileges.

Requirements for Windows

- Microsoft .NET 4.0 Framework
- Microsoft Visual C++ 2013 Redistributable Package (MSVC2013)

Note

The 2010 version was used in previous editions of MySQL Workbench 6.

- Windows 7 and above

Note

MySQL Workbench 6.1 supports earlier versions of Windows, including Vista

Installation Using MySQL Installer

The general MySQL Installer download is available at http://dev.mysql.com/downloads/windows/installer/. The MySQL Installer application can install, upgrade, and manage most MySQL products, including MySQL Workbench.

This is the Recommended Approach

Managing all of your MySQL products, including Workbench, with MySQL Installer is the recommended approach. It handles all requirements and prerequisites, configurations, and upgrades.

When executing MySQL Installer, you may choose MySQL Workbench as one of the products to install. It is selected by default, and essentially executes the standalone Installer Package described below.

Installation Using the Installer Package

The standalone download is available at http://dev.mysql.com/downloads/workbench/.

MySQL Workbench can be installed using the Windows Installer (`.msi`) installation package. The MSI package bears the name `mysql-workbench-`*`community-version`*`-winarch.msi`, where *`version`* indicates the MySQL Workbench version number, and *`arch`* the build architecture (either win32 or winx64).

1. To install MySQL Workbench, right-click the MSI file and select the **Install** item from the pop-up menu, or double-click the file.

2. In the **Setup Type** window you may choose a `Complete` or `Custom` installation. To use all features of MySQL Workbench choose the `Complete` option.

3. Unless you choose otherwise, MySQL Workbench is installed in `C:\`*`%PROGRAMFILES%`*`\MySQL\MySQL Workbench 6.3 `*`edition_type`*`\`, where *`%PROGRAMFILES%`* is the default directory for programs for your locale. The `%PROGRAMFILES%` directory is defined as `C:\Program Files\` on most systems.

Installation Using the Zip File

If you have problems running the Installer package, an alternative is to install from a Zip file without an installer. That file is called `mysql-workbench-`*`community-version-arch`*`.zip`, where *`version`* indicates the MySQL Workbench version number, and *`arch`* the build architecture (either win32 or winx64).

To install using the Zip file, download the Zip file to a convenient location and decompress the file using a Zip utility. You can place the resulting directory anywhere on you system. You need not install or configure the application before using it. You may want to create a shortcut on your desktop or the quick launch bar.

2.3.2 Launching

To start MySQL Workbench on Windows, select **Start**, **Programs**, **MySQL**, then select MySQL Workbench. This executes the `MySQLWorkbench.exe` file on your system.

Alternatively, start MySQL Workbench from the command line. To view the available command-line options, issue the command `MySQLWorkbench -help` from the MySQL Workbench installation directory. You will see the following output:

```

MySQLWorkbench.exe [<options>] [<name of a model file or sql script>]

Options:
  -swrendering           Force the diagram canvas to use software rendering instead of OpenGL
  -query [<connection>|<connection string>]
                         Open a query tab and ask for connection if nothing is specified.
                         If named connection is specified it will be opened, else connection
                         will be created based on the given connection string, which should
                         be in form <user>@<host>:<port>
  -admin <instance>      Open a administration tab to the named instance
  -upgrade-mysql-dbs     Open a migration wizard tab
  -model <model file>    Open the given EER model file
  -script <sql file>     Open the given SQL file in an connection, best in conjunction with
                         a query parameter
  -run-script <file>     Execute Python code from a file
  -run <code>            Execute the given Python code
```

```
 -run-python <code>      Execute the given Python code
 -migration              Open the Migration Wizard tab
 -quit-when-done         Quit Workbench when the script is done
 -log-to-stderr          Also log to stderr
 -help, -h               Show command line options and exit
 -log-level=<level>      Valid levels are: error, warning, info, debug1, debug2, debug3
 -verbose, -v            Enable diagnostics output
 -version                Show Workbench version number and exit
 -open <file>            Open the given file at startup (deprecated, use script, model etc.)
```

Use the `-swrendering` option if your video card does not support OpenGL 1.5. The `-version` option can be used to display the MySQL Workbench version number. The other options are self-explanatory.

MySQL Workbench may also be started from MySQL Notifier by choosing **SQL Editor** or **Configure Instance** from the Notifier context menu. For additional information, see MySQL Notifier.

2.3.3 Uninstalling

The method for uninstalling MySQL Workbench depends on how you installed MySQL Workbench.

Removing MySQL Workbench After Installation Using the Installer Package

1. To uninstall MySQL Workbench, open the **Control Panel** and Choose **Add or Remove Programs**. Find the MySQL Workbench entry and choose the **Remove** button. This will remove MySQL Workbench.

Note

If you installed MySQL Workbench using the Installer package, it is not possible to remove MySQL Workbench from the command line. Although you can manually remove some of the components, there is no command-line option for removing MySQL Workbench.

Removing the MySQL Workbench directory manually will not remove all the files belonging to MySQL Workbench.

Removing MySQL Workbench After Installation from the MySQL Installer

Open the MySQL Installer for Windows, click **Remove MySQL Products**, choose MySQL Workbench, and then **Execute**.

What Is Not Removed

Uninstalling MySQL Workbench does not remove your Workbench configuration directory. This directory includes your MySQL connections, configuration settings, cache files, SQL snippets and history, logs, custom modules, and more. These files are stored under your user's `%AppData%` directory.

Note

By default, the Workbench configuration directory is `C:\username\AppData\Roaming\MySQL\Workbench\` where "C:\username\AppData\Roaming\" is the value of your `%AppData%` Windows system variable.

Also, uninstalling Workbench does not remove the `.mysqlworkbench` schema that Workbench creates when sharing SQL snippets across a MySQL connection. For additional information about shared snippets, see Section 8.1.5, “SQL Snippets tab”.

2.4 MySQL Workbench on Linux

2.4.1 Installing

There are binary distributions of MySQL Workbench available for several variants of Linux, including Fedora, Oracle Linux, and Ubuntu.

Installation options include:

- **Official MySQL Yum or APT repository packages**: These binaries are built by the MySQL Release team. For additional information about installing these, see Yum or APT. They contain the newest versions of MySQL Workbench. Typically this package is named `mysql-workbench-community`.
- **Your Linux distributions repository packages**: These binaries are built and maintained by members of the Linux distribution you use, and not by the MySQL team. They are stable but the releases often lag behind. Typically this package is named `mysql-workbench`.
- **Download official MySQL packages**: Downloads are available at http://dev.mysql.com/downloads/workbench.
- **Download the source code and compile yourself**: The source code is available at http://dev.mysql.com/downloads/workbench as a `tar.gz` or RPM package.

Note

32-bit binary builds are not available as of MySQL Workbench 6.2.0. You can use the source code to build your own 32-bit version, as needed.

The procedure for installing on Linux depends on which Linux distribution you are using.

Requirements for Linux

- The requirements for Linux are embedded within their respective packages. Use the platform specific tool (for example, yum or apt) to install the package and their dependencies.
- The "Save password in keychain" functionality requires `gnome-keyring` to store the passwords. Note that on KDE systems, the `gnome-keyring` daemon is not started by default.
- For Linux and OS X, the MySQL server administration features require `sudo` privileges to execute several commands. The sudo user must be capable of executing the following system commands:

```
/usr/bin/sudo
/usr/bin/nohup
/usr/bin/uptime
/usr/bin/which
/usr/bin/stat

/bin/bash
/bin/mkdir
/bin/rm
/bin/rmdir
/bin/dd
/bin/cp
/bin/ls
```

  Additionally, the sudo user must keep the `HOME` environment variable when executing system commands, which means adding the following to `/etc/sudoers`:

```
env_keep +="HOME"
```

For MySQL Workbench to execute MySQL Enterprise Backup commands, the sudo user must also be able to execute the MySQL Enterprise Backup binary.

Installing DEB packages

On Ubuntu, and other systems that use the Debian package scheme, you can either download and install .deb packages or use the APT package manager.

Using the APT Package Manager

Important

Your Linux distribution includes MySQL Workbench builds where "apt-get install mysql-workbench" will install their build of the MySQL Workbench package. To use the official MySQL Workbench builds as provided by the MySQL Release team, you must install the official MySQL APT repository and choose the "mysql-workbench-community" package instead of "mysql-workbench".

- First, install the MySQL APT repository as described in the MySQL APT Repository documentation. For example:

```
shell> sudo dpkg -i mysql-apt-config_0.5.3-1_all.deb
shell> sudo apt-get update
```

- Next, install MySQL Workbench. You might have multiple Workbench packages available, so choose the "mysql-workbench-community" version. For example:

```
shell> sudo apt-get install mysql-workbench-community
```

Manually Installing a Package

You install MySQL Workbench using a command such as:

```
shell> sudo dpkg -i package.deb
```

`package.deb` is the MySQL Workbench package name; for example, `mysql-workbench-community-version1ubu1404-amd64.deb`, where *`version`* is the MySQL Workbench version number.

Note

You may be warned that certain libraries are not available, depending on what you already have installed. Install the required libraries and then install the MySQL Workbench package again.

Installing RPM packages

On Red Hat-based systems, and other systems that use the RPM package format, you can either download and install RPM packages or use the Yum package manager.

Note

Enterprise Linux systems, such as Oracle Linux and Red Hat, require access to the EPEL package repository. For additional information about installing EPEL, see Installing Oracle Enterprise Linux and similar.

Using the Yum Package Manager

Your Linux distribution includes MySQL Workbench builds where "yum install mysql-workbench" will install their build of the MySQL Workbench package. To use the official MySQL Workbench builds as provided by the MySQL Release team, you must install the official MySQL Yum repository and choose the "mysql-workbench-community" package instead of "mysql-workbench".

- First, install the MySQL Yum repository as described in the MySQL Yum Repository documentation. For example:

  ```
  shell> sudo rpm -Uvh mysql-community-release-el7-7.noarch.rpm
  ```

- Next, install MySQL Workbench. You might have multiple Workbench packages available, so choose the "mysql-workbench-community" version. For example:

  ```
  shell> sudo yum install mysql-workbench-community
  ```

Manually Installing a Package

```
shell> sudo rpm -i package.rpm
```

`package.rpm` is the MySQL Workbench package name; for example, `mysql-workbench-community-version-1fc10.x86_64.rpm`, where *`version`* is the MySQL Workbench version number.

Installing Oracle Enterprise Linux and similar

MySQL Workbench requires access to the EPEL repository. EPEL is a repository with additional RPM packages that are not part of the core RHEL/OEL distribution. This includes packages (such as tinyxml) that MySQL Workbench requires.

You need to set up the EPEL repository in yum to resolve the required dependencies. For example, using Oracle Linux 6.8 you would:

```
shell> wget http://download.fedoraproject.org/pub/epel/6/i386/epel-release-6-8.noarch.rpm
shell> rpm -ivh epel-release-6-8.noarch.rpm

shell> yum repolist

Loaded plugins: refresh-packagekit, rhnplugin
repo id                 repo name                                           status
epel                    Extra Packages for Enterprise Linux 6 - x86_64       7,124
```

These instructions also apply to similar Linux distributions such as Red Hat Enterprise Linux, CentOS, and Scientific Linux.

Next, follow the RPM-based installation documentation at Installing RPM packages.

2.4.2 Launching

After MySQL Workbench has been installed, it can be launched by selecting **Applications**, **Programming**, **MySQL Workbench** from the main menu.

MySQL Workbench can also be launched from the command line on Linux. Type the command:

```
shell> /usr/bin/mysql-workbench --help
```

This will display the available command-line options:

```

mysql-workbench [<options>] [<name of a model file or sql script>]

Options:
  --force-sw-render       Force Xlib rendering
  --force-opengl-render   Force OpenGL rendering
  --query [<connection>|<connection string>]
                          Open a query tab and ask for connection if nothing is specified.
                          If named connection is specified it will be opened, else connection
                          will be created based on the given connection string, which should
                          be in form <user>@<host>:<port>
  --admin <instance>      Open a administration tab to the named instance
  --upgrade-mysql-dbs     Open a migration wizard tab
  --model <model file>    Open the given EER model file
  --script <sql file>     Open the given SQL file in an connection, best in conjunction with
                          a query parameter
  --run-script <file>     Execute Python code from a file
  --run <code>            Execute the given Python code
  --run-python <code>     Execute the given Python code
  --migration             Open the Migration Wizard tab
  --quit-when-done        Quit Workbench when the script is done
  --log-to-stderr         Also log to stderr
  --help, -h              Show command line options and exit
  --log-level=<level>     Valid levels are: error, warning, info, debug1, debug2, debug3
  --verbose, -v           Enable diagnostics output
  --version               Show Workbench version number and exit
  --open <file>           Open the given file at startup (deprecated, use script, model etc.)
```

2.4.3 Uninstalling

The procedure for uninstalling MySQL Workbench on Linux depends on the package you are using.

Note

When using apt, the official package name at dev.mysql.com is `mysql-workbench-community`, whereas most Linux distributions use the name `mysql-workbench`. Adjust the following commands accordingly.

Uninstalling DEB packages

To uninstall a Debian package, use the following:

```
shell> sudo apt-get remove mysql-workbench-community
```

Or, alternatively:

```
shell> sudo dpkg -r mysql-workbench-community
```

This command does not remove the configuration files. If you wish to also remove the configuration files, use this command:

```
shell> sudo dpkg --purge mysql-workbench-community
```

Uninstalling RPM packages

Note

When using yum, the official package name at dev.mysql.com is `mysql-workbench-community`, whereas most Linux distributions use the name `mysql-workbench`. Adjust the following commands accordingly.

To uninstall an RPM package, use this command:

```
shell> sudo yum remove mysql-workbench-community
```

Or, alternatively:

```
shell> sudo rpm -e mysql-workbench-community
```

This command does not remove the configuration files.

What Is Not Removed

By default, uninstalling MySQL Workbench does not remove your Workbench configuration directory. This directory includes your MySQL connections, configuration settings, cache files, SQL snippets and history, logs, custom modules, and more. These files are stored under your user's `.mysql/workbench/` directory.

Note

By default, the Workbench configuration directory is `~username/mysql/workbench/` where "~username" is the path to your user's home directory.

Also, uninstalling Workbench does not remove the `.mysqlworkbench` schema that Workbench creates when sharing SQL snippets across a MySQL connection. For additional information about shared snippets, see Section 8.1.5, “SQL Snippets tab”.

2.5 MySQL Workbench on OS X

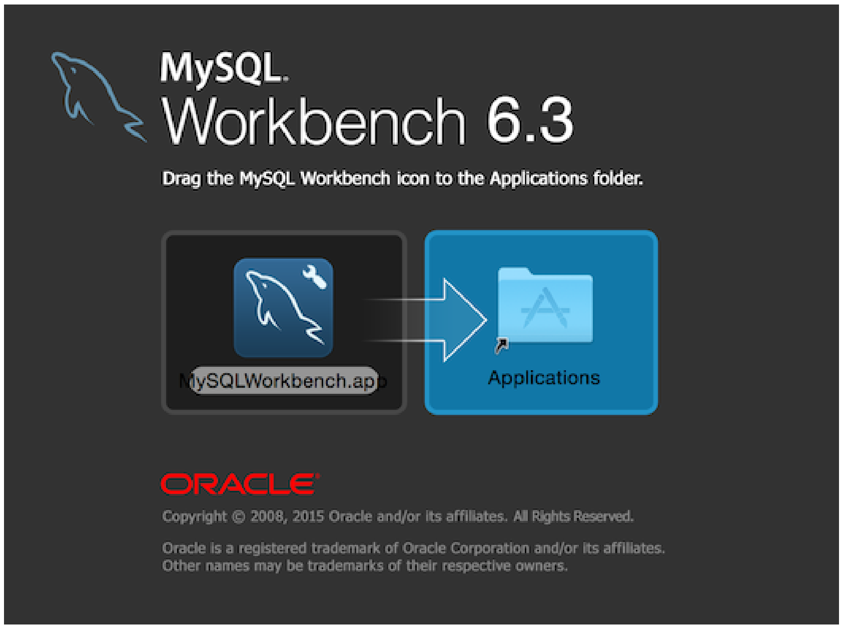
2.5.1 Installing

MySQL Workbench for OS X is distributed as a DMG file. The file is named `mysql-workbench-community-`*`version`*`-osx.dmg`, where *`version`* is the MySQL Workbench version.

Downloads are available at http://dev.mysql.com/downloads/workbench/.

To install MySQL Workbench on OS X, download the file. Double-click the downloaded file. You will be presented with the installation window.

Figure 2.1 MySQL Workbench OS X Installation Window


MySQL
Workbench 6.3
Drag the MySQL Workbench icon to the Applications folder.
MySQLWorkbench.app
Applications
ORACLE
Copyright © 2008, 2015 Oracle and/or its affiliates. All Rights Reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Drag the MySQL Workbench icon onto the Applications icon as instructed. MySQL Workbench is now installed.

You can now launch MySQL Workbench from the Applications folder, or from the command line:

```
shell> /Applications/MySQLWorkbench.app/Contents/MacOS/MySQLWorkbench --help
```

This will display the available command-line options:

```
MySQLWorkbench [<options>] [<name of a model file or sql script>]
Options:
  --query [<connection>|<connection string>]
                          Open a query tab and ask for connection if nothing is specified.
                          If named connection is specified it will be opened,
                          else connection will be created based on the given connection string,
                          which should be in form <user>@<host>:<port>
  --admin <instance>      Open a administration tab to the named instance
  --upgrade-mysql-dbs     Open a migration wizard tab
  --model <model file>    Open the given EER model file
  --script <sql file>     Open the given SQL file in an connection, best in conjunction
                          with a query parameter
  --run-script <file>     Execute Python code from a file
  --run <code>            Execute the given Python code
  --run-python <code>     Execute the given Python code
  --migration             Open the Migration Wizard tab
  --quit-when-done        Quit Workbench when the script is done
  --log-to-stderr         Also log to stderr
  --help, -h              Show command line options and exit
  --log-level=<level>     Valid levels are: error, warning, info, debug1, debug2, debug3
  --verbose, -v           Enable diagnostics output
  --version               Show Workbench version number and exit
  --open <file>           Open the given file at startup (deprecated, use script, model etc.)
```

2.5.2 Launching

To launch MySQL Workbench on OS X, open the Applications folder in the Finder, then double-click MySQL Workbench.

It is also possible to start MySQL Workbench from the command line:

```
shell> open MySQLWorkbench.app [options] [model_file]
```

Specifying options and/or a model file is optional.

2.5.3 Uninstalling

To uninstall MySQL Workbench for OS X, locate MySQL Workbench in the Applications folder, right-click, and select **Move to Trash**.

What Is Not Removed

By default, uninstalling MySQL Workbench does not remove your Workbench configuration directory. This directory includes your MySQL connections, configuration settings, cache files, SQL snippets and history, logs, custom modules, and more. These files are stored under your user's `MySQL/Workbench/` folder.

Note

By default, the Workbench configuration directory is `~username/Library/Application Support/MySQL/Workbench` where "~username" is the path to your user's home directory.

Also, uninstalling Workbench does not remove the `.mysqlworkbench` schema that Workbench creates when sharing SQL snippets across a MySQL connection. For additional information about shared snippets, see Section 8.1.5, “SQL Snippets tab”.

Chapter 3 Configuration

Table of Contents

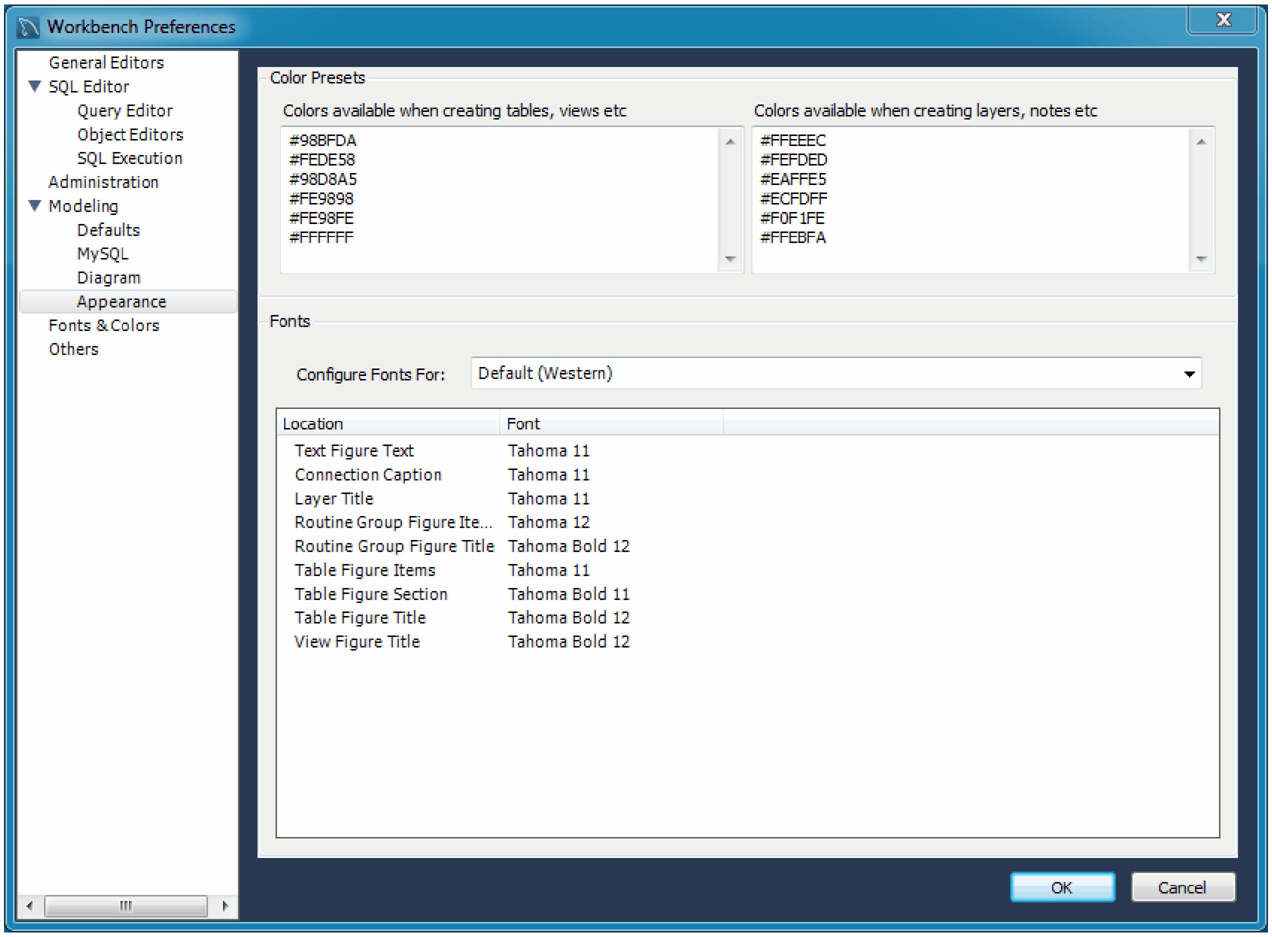
3.1 User Accessibility Options

MySQL Workbench includes methods to improve user accessibility.

Fonts

Under **Preferences**, modeling fonts are adjustable from the **Appearance** section of the **Modeling** menu:

Figure 3.1 Appearance Preferences



Color Presets

Colors available when creating tables, views etc: #98BFDA, #FEDE58, #98D8A5, #FE9898, #FE98FE, #FFFFFF

Colors available when creating layers, notes etc: #FFEEEC, #FEFDED, #EAFFE5, #ECFDFF, #F0F1FE, #FFEBFA

Fonts

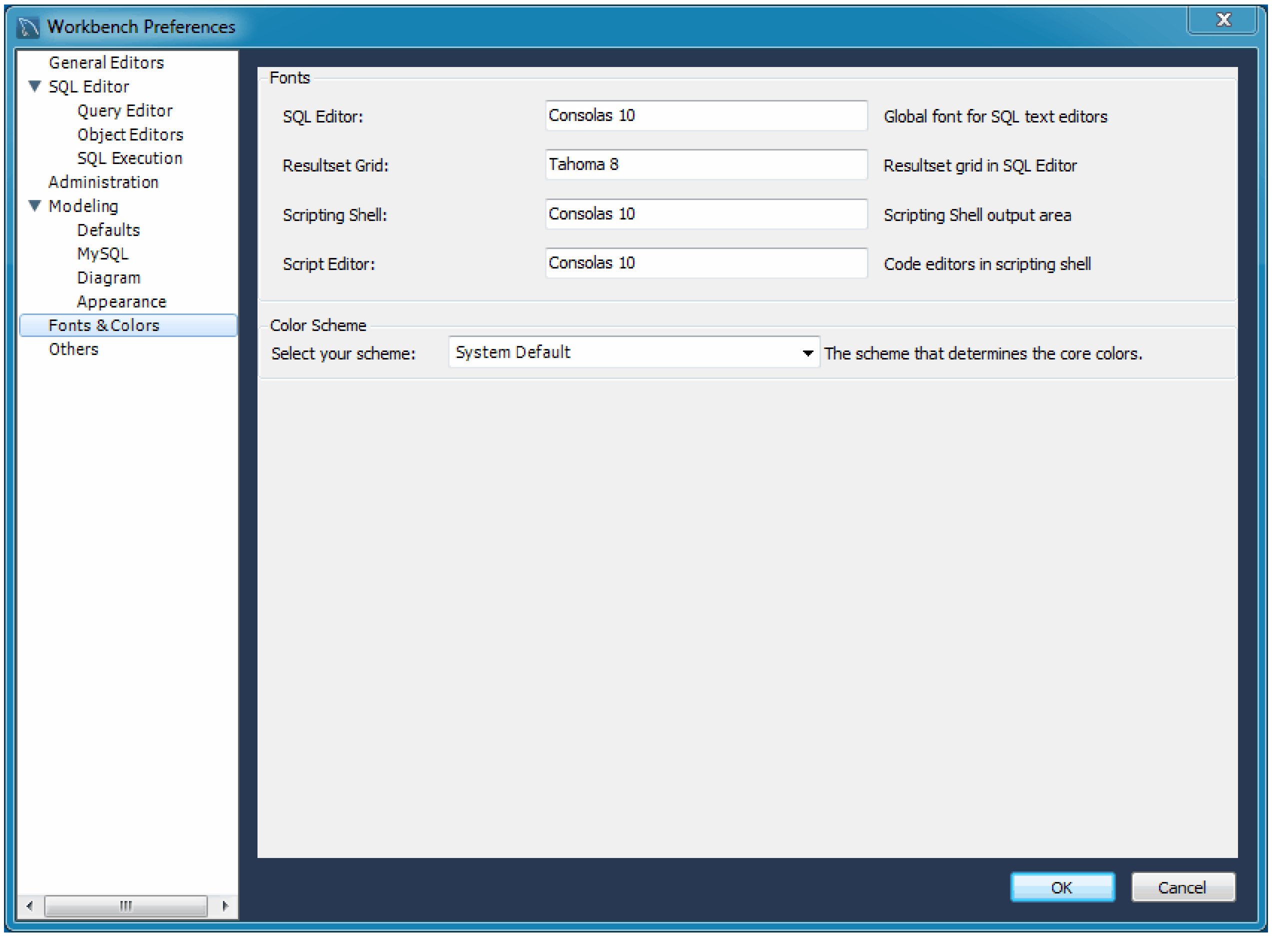
Configure Fonts For: Default (Western)

Location	Font
Text Figure Text	Tahoma 11
Connection Caption	Tahoma 11
Layer Title	Tahoma 11
Routine Group Figure Ite...	Tahoma 12
Routine Group Figure Title	Tahoma Bold 12
Table Figure Items	Tahoma 11
Table Figure Section	Tahoma Bold 11
Table Figure Title	Tahoma Bold 12
View Figure Title	Tahoma Bold 12

Choose the character set under the **Configure Fonts For** setting (typically leave the default setting here) and then adjust the model fonts to fit your needs.

The font types and sizes for other GUI elements are set under the **Fonts & Colors** tab:

Figure 3.2 Fonts & Color Preferences



Fonts

SQL Editor:	Consolas 10	Global font for SQL text editors
Resultset Grid:	Tahoma 8	Resultset grid in SQL Editor
Scripting Shell:	Consolas 10	Scripting Shell output area
Script Editor:	Consolas 10	Code editors in scripting shell

Color Scheme

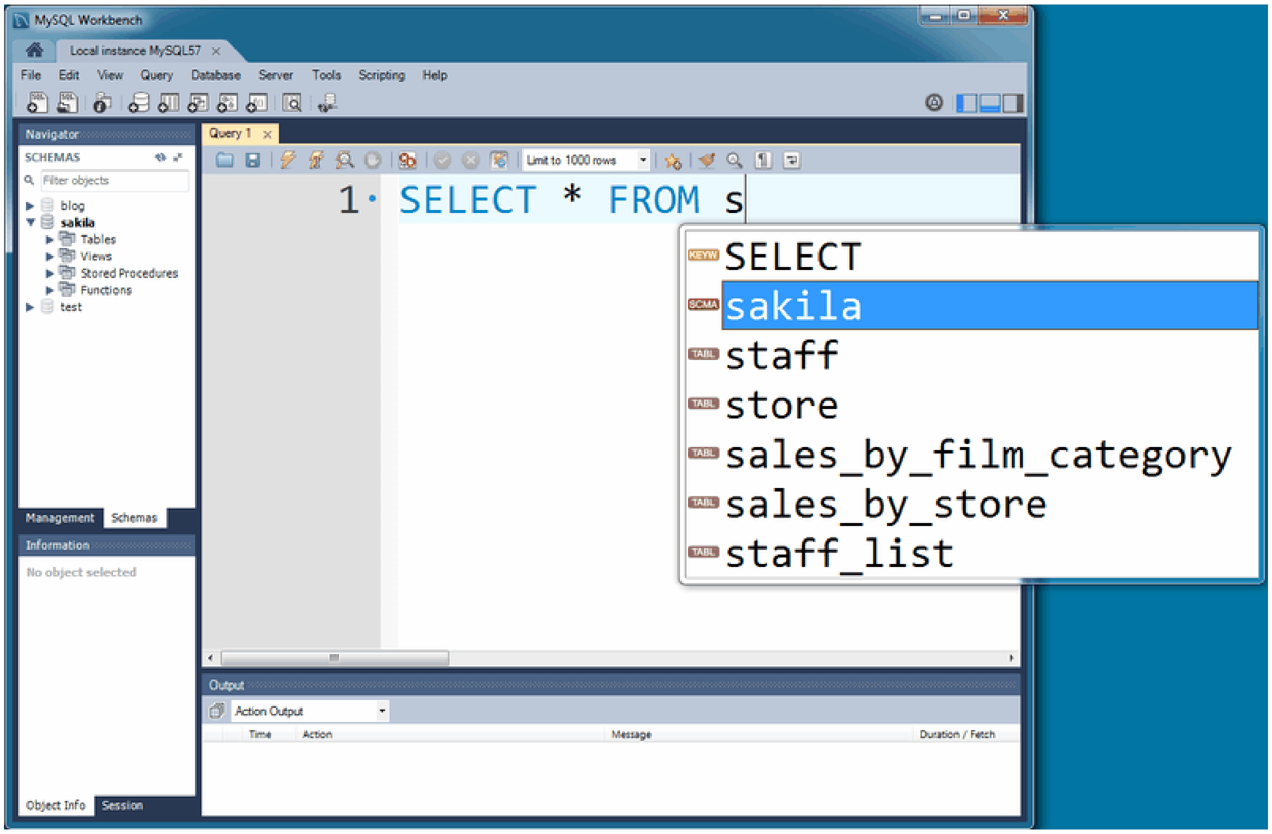
Select your scheme: System Default — The scheme that determines the core colors.

Note

Font changes require a refresh or restart before they take effect.

The following image shows the SQL Editor after changing the **Editor** font size from 10 to 30:

Figure 3.3 SQL Editor with Font size 30



```
SELECT * FROM s
```

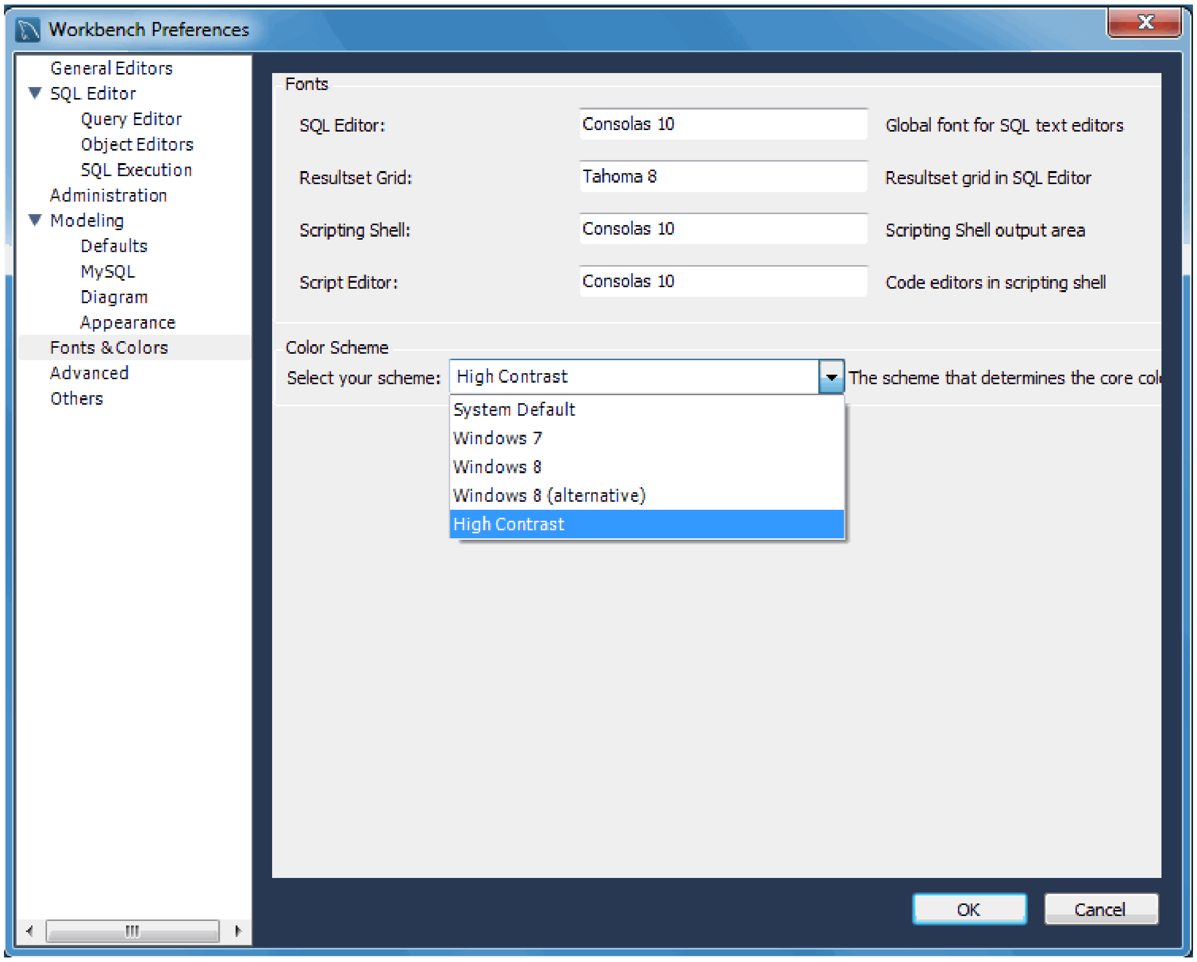
Color Presets

Here you define the colors used in EER diagrams for the tables, views, layers, and notes. You can edit or add additional color choices by entering their ASCII values.

Theming

On Windows, the **Fonts & Colors** preference tab also includes a "Color Scheme" configuration section. From here, you can enable the **High Contrast** color theme. This theme preference affects the MySQL Workbench GUI.

Figure 3.4 High Contrast Preference



Fonts

SQL Editor:	Consolas 10	Global font for SQL text editors
Resultset Grid:	Tahoma 8	Resultset grid in SQL Editor
Scripting Shell:	Consolas 10	Scripting Shell output area
Script Editor:	Consolas 10	Code editors in scripting shell

Color Scheme

Select your scheme: High Contrast The scheme that determines the core col

System Default
Windows 7
Windows 8
Windows 8 (alternative)
High Contrast

Microsoft Active Accessibility (MSAA)

On Windows, MySQL Workbench supports MSAA, which allows use of screen reader applications with MySQL Workbench.

3.2 Workbench Preferences

Use the **Preferences** menu to configure MySQL Workbench to your specific needs. This menu is divided sections, as described below:

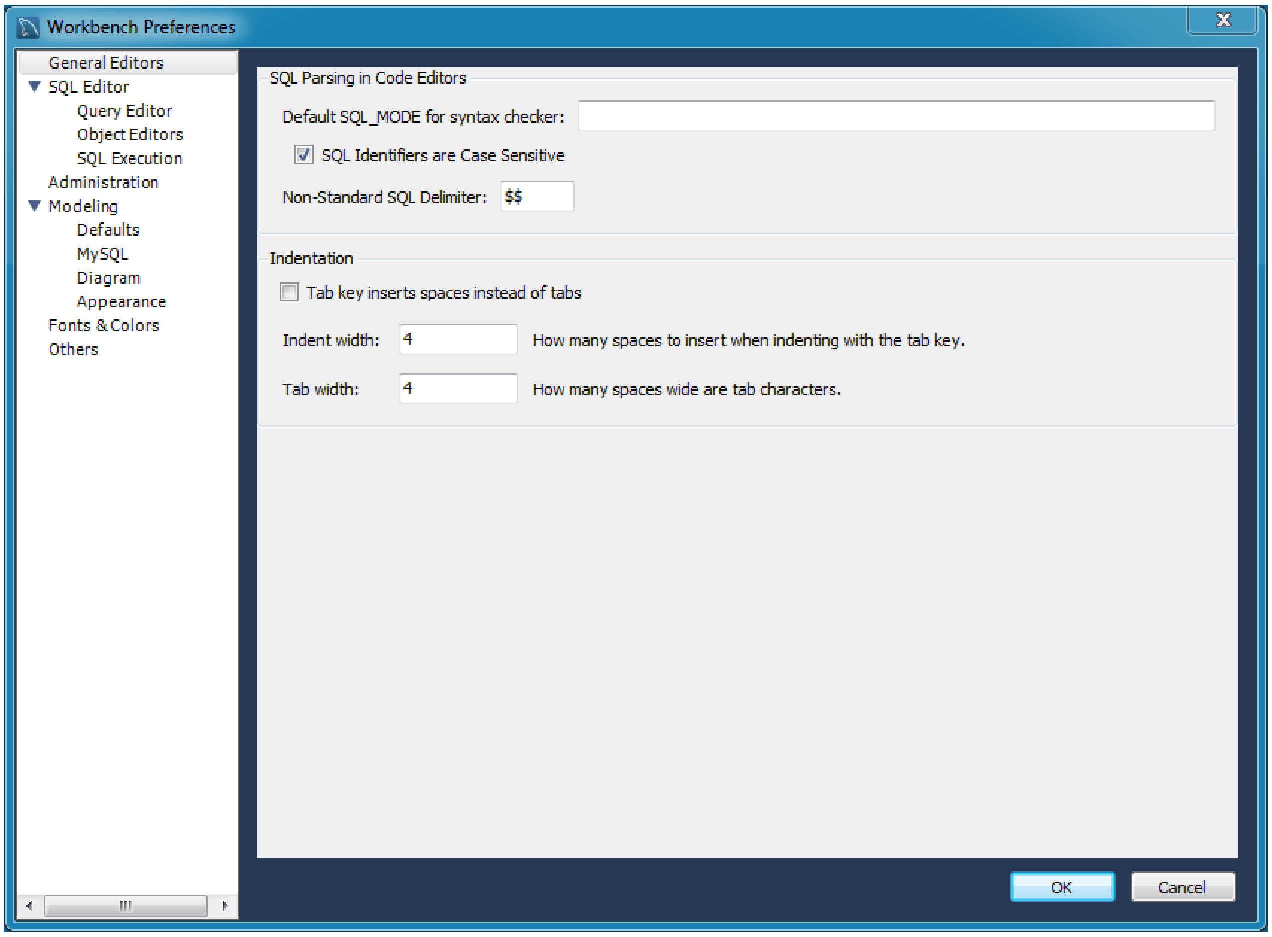
- **General Editors**: General-purpose editor options, such as SQL parsing options.
- **SQL Editor**: SQL editor related preferences that also includes subsections for the **Query Editor**, **Object Editor**, and **SQL Execution**.
- **Administration**: Tools used by the Administrator functionality.
- **Modeling**: Model related preferences that also includes subsections for **Defaults**, **MySQL** (MySQL specific settings), **Diagram** (EER), and **Appearance** (model colors and fonts).
- **Fonts & Colors**: Change fonts for tools such as the SQL editor and results grid.
- **Others**: Miscellaneous options.

A more detailed discussion of these options follows.

3.2.1 General Editors Preferences

The **General Editors** preferences section:

Figure 3.5 Preferences: General Editors



SQL Parsing in Code Editors

Default SQL_MODE for syntax checker:

[✓] SQL Identifiers are Case Sensitive

Non-Standard SQL Delimiter: $$

Indentation

[ ] Tab key inserts spaces instead of tabs

Indent width: 4 How many spaces to insert when indenting with the tab key.

Tab width: 4 How many spaces wide are tab characters.

SQL Parsing in Code Editors

SQL properties that can be set include the `SQL_MODE`, case sensitivity of identifiers, and the SQL delimiter used.

- **Default SQL_MODE for syntax checker**: Optionally configure the SQL_MODE for the SQL editor's SQL syntax checker.

  The document property `SqlMode` defines `SQL_MODE` for all operations affecting SQL parsing at the document scope. The purpose of this option is to preserve the consistency of SQL statements within the document.

  The property has the following functions:

  - Sets the `SQL_MODE` DBMS session variable to the value stored in the `SqlMode` property of the document when performing reverse engineering, forward engineering, or synchronization operations.

- Honors the `SQL_MODE` values defined in `SqlMode` so that SQL parsing is correct.

  Only a subset of all possible `SQL_MODE` values affect the MySQL Workbench SQL parser. These values are: `ANSI_QUOTES`, `HIGH_NOT_PRECEDENCE`, `IGNORE_SPACE`, `NO_BACKSLASH_ESCAPES`, `PIPES_AS_CONCAT`. Other values do not affect the MySQL Workbench SQL parser and are ignored.

  If the value of `SqlMode` is not set, the default value of the `SQL_MODE` session variable defined by the server stays unchanged during operations with the server. However, the MySQL Workbench SQL parser behaves as if `SQL_MODE` is also not set. This may potentially lead to inconsistencies in parsing of SQL statements stored in the document. If you choose to not set the `SqlMode` property, ensure that the default `SQL_MODE` variable defined by the server does not contain any values from the following list: `ANSI_QUOTES`, `HIGH_NOT_PRECEDENCE`, `IGNORE_SPACE`, `NO_BACKSLASH_ESCAPES`, `PIPES_AS_CONCAT`.

  The `SqlMode` property is defined in two locations: globally and at document scope. MySQL Workbench uses the global property to initialize the document property for each new document created. For each document, the property value defined at document scope always has higher priority over the one defined globally.

- [ ] **SQL Identifiers are Case Sensitive**: Whether to treat identifiers separately if their names differ only in letter case. This is enabled by default.

- **Non-Standard SQL Delimiter**: [*$$*]. Define the SQL statement delimiter to be different from the normally used delimiter (such as ";"). Change this if the deliminator you normally use, specifically in stored routines, happens to be the current setting.

Indentation

Note

This preference section was added in MySQL Workbench 6.2.4.

- [ ] Tab key inserts spaces instead of tabs

- Indent width: [*4*] The number of spaces inserted after pressing **Tab** -- this assumes that the *Tab key inserts spaces instead of tabs* option is enabled

- Tab width: [*4*] The width (number of spaces) that tab characters are displayed as in MySQL Workbench

3.2.2 SQL Editor Preferences

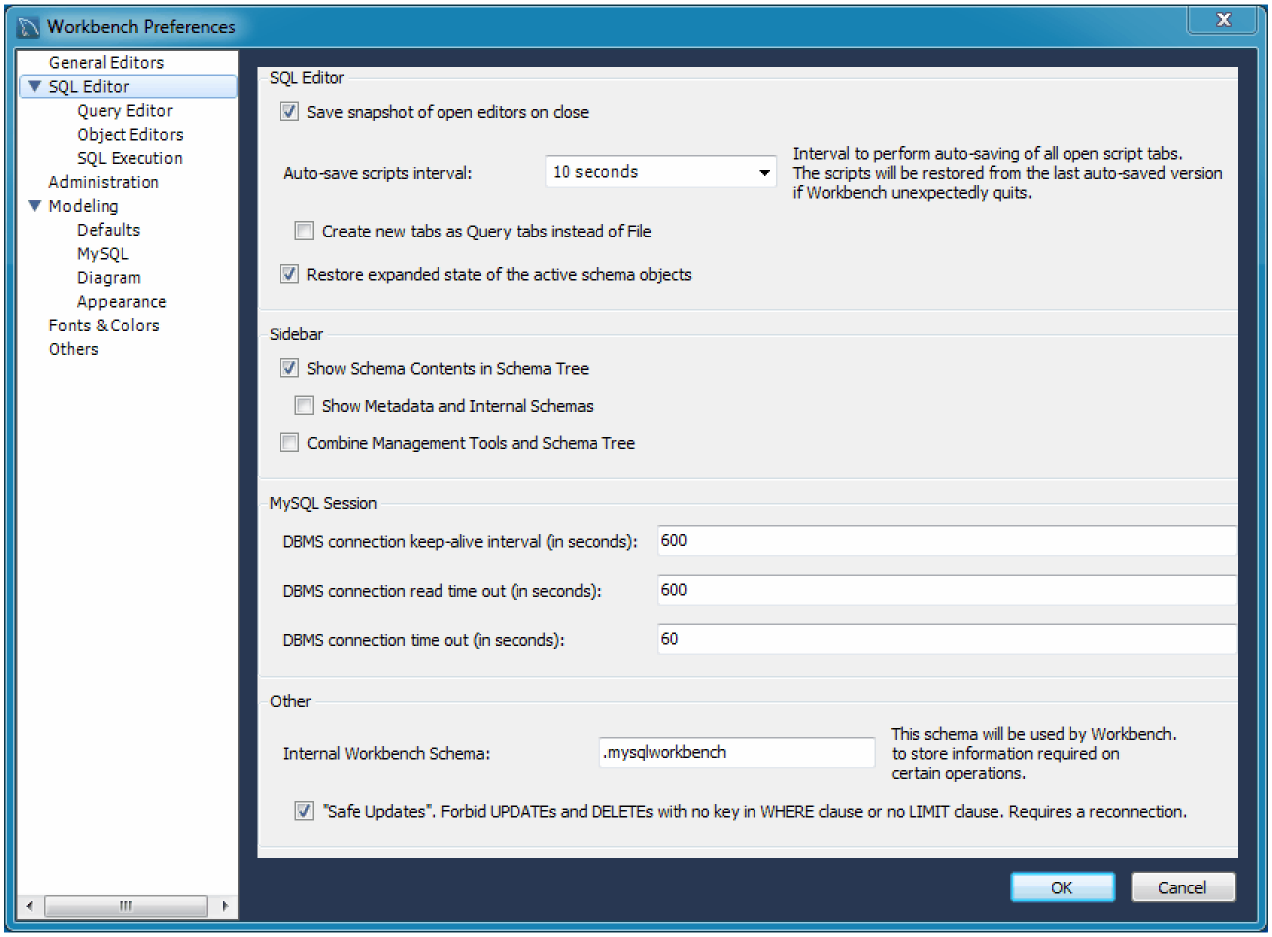
This section provides configuration options that affect the SQL Editor functionality in MySQL Workbench.

The SQL Editor preferences includes three additional sections: Query Editor, Object Editors, and SQL Execution.

Preferences: SQL Editor: Main

SQL Editor

Figure 3.6 Preferences: Main SQL Editor Section



SQL Editor

☑ Save snapshot of open editors on close

Auto-save scripts interval: 10 seconds

Interval to perform auto-saving of all open script tabs. The scripts will be restored from the last auto-saved version if Workbench unexpectedly quits.

☐ Create new tabs as Query tabs instead of File

☑ Restore expanded state of the active schema objects

Sidebar

☑ Show Schema Contents in Schema Tree

☐ Show Metadata and Internal Schemas

☐ Combine Management Tools and Schema Tree

MySQL Session

DBMS connection keep-alive interval (in seconds): 600

DBMS connection read time out (in seconds): 600

DBMS connection time out (in seconds): 60

Other

Internal Workbench Schema: .mysqlworkbench

This schema will be used by Workbench. to store information required on certain operations.

☑ "Safe Updates". Forbid UPDATEs and DELETEs with no key in WHERE clause or no LIMIT clause. Requires a reconnection.

- **Save snapshot of open editors on close**: Enabling will save and reload the SQL Editor tabs after closing/opening MySQL Workbench (including after an unexpected crash).
- **Auto-save scripts interval**: Frequency of the auto-saves.
- **Create new tabs as Query tabs instead of File**: By default, opening a new SQL Editor tab opens as an SQL File tab. Check this option if you prefer the simpler Query tabs that, for example, will not prompt to be saved when closed.
- **Restore expanded state of the active schema objects**: Group nodes that were previously expanded in the active schema when the SQL editor was last closed are re-expanded and loaded. This is enabled by default.

Sidebar

- **Show Schema Contents in Schema Tree**: Enumerating, populating, and drawing large numbers of items can significantly increase loading times. For this reason, this facility can be switched off for models containing large numbers of schemata and tables.
- **Show Data Dictionaries and Internal Schemas**: Whether to show data directories and internal schemas in the schema tree (such as `INFORMATION_SCHEMA`, mysql, and schemas starting with ".").

- **Combine Management Tools and Schema Tree tab**: This affects the Object Browser in the left sidebar, and this option can also be toggled from the sidebar. The management tools and schema tree can be viewable as separate tabs, or as a single long list.

MySQL Session

- **DBMS connection keep-alive interval (in seconds)**: [*`600`*]. Time interval between sending keep-alive messages to the DBMS. Set to 0 to not send keep-alive messages.
- **DBMS connection read time out (in seconds)**: [*`600`*]. The maximum amount of time the query can take to return data from the DBMS. Set 0 to not check the read time out.
- **DBMS connection time out (in seconds)**: [*`60`*]. Maximum time to wait before a connection attempt is aborted.

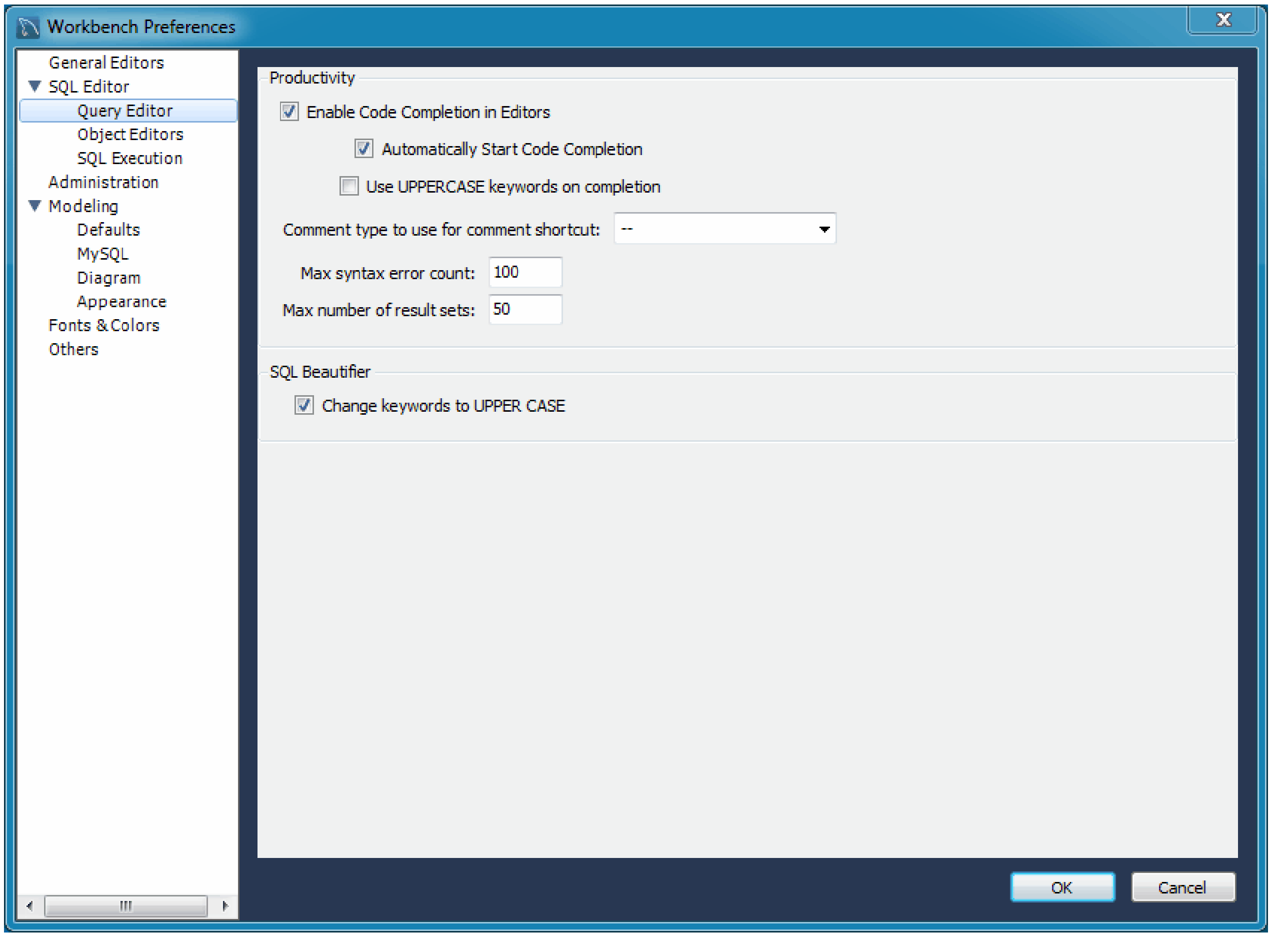
Other

- **Internal Workbench Schema**: [*`.mysqlworkbench`*]. This schema is used by MySQL Workbench to store information required for certain operations, such as saving shared SQL snippets.
- [ ] : **"Safe Updates"**, forbid UPDATE and DELETE queries to execute that lack a corresponding key in a WHERE clause, or lack a LIMIT clause. Setting this option requires a MySQL server reconnection.

  This makes it possible to catch UPDATE and DELETE statements where keys are not used properly and that would probably accidentally change or delete a large number of rows.

Preferences: SQL Editor: Query Editor

Figure 3.7 Preferences: SQL Editor: Query Editor



Productivity

☑ Enable Code Completion in Editors

☑ Automatically Start Code Completion

☐ Use UPPERCASE keywords on completion

Comment type to use for comment shortcut: --

Max syntax error count: 100

Max number of result sets: 50

SQL Beautifier

☑ Change keywords to UPPER CASE

Productivity

- **Enable Code Completion in Editors**: The SQL Editor offers Auto-complete functionality by either pressing the keyboard shortcut (**Modifier** + **Space**), or it will start automatically if the **Automatically Start Code Completion** preference is enabled.
- **Automatically Start Code Completion**: Enabled by default, this will automatically execute the code auto-completion feature while editing SQL in the SQL Editor. If disabled, you will instead use the keyboard shortcut **Modifier** + **Space** to execute the auto-completion routine.
- **Use UPPERCASE keywords on completion**: Normally keywords are shown and inserted as they come from the code editor's configuration file. This setting will always write completed keywords as uppercase.
- **Comment type to use for comment shortcut**: [`--`]. Defaults to "--", with "#" as another option.
- **Max syntax error count**: Large complex scripts may contain errors. Further, a syntax error early on can lead to subsequent syntax errors. For these reasons, it is possible to limit the number of errors displayed using this option. The default is 100 error messages.
- **Max number of result sets**: Maximum number of result sets for SQL queries that can be opened for a single SQL editor. Defaults to 50. Reaching the limit emits a warning.

Note

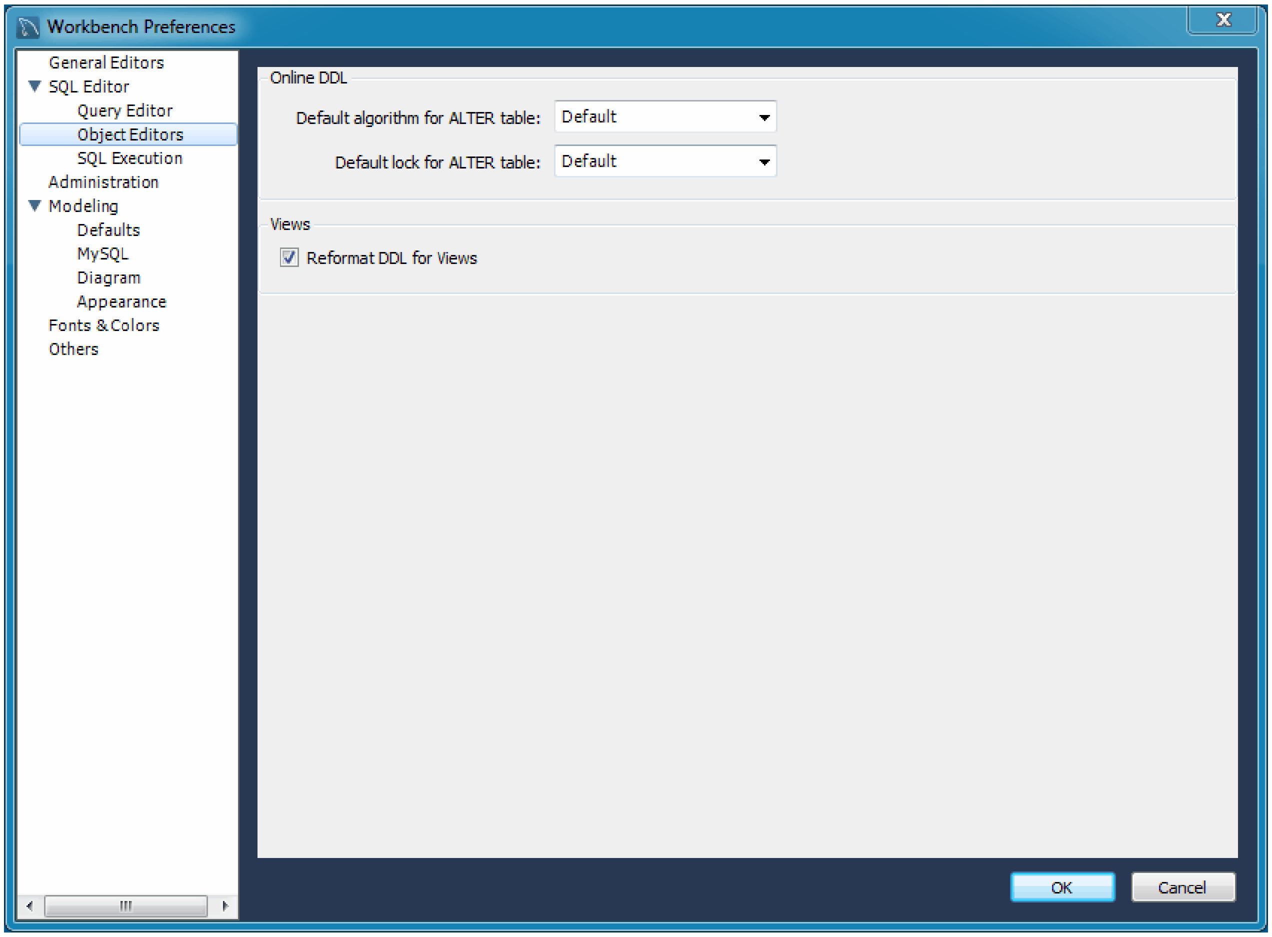
This option was added in MySQL Workbench 6.2.4.

SQL Beautifier

- [ ] **Change keywords to UPPER CASE**: Enabled by default, executing the SQL beautifier will uppercase all SQL keywords.

Preferences: SQL Editor: Object Editors

Figure 3.8 Preferences: SQL Editor: Object Editors



Online DDL

Default algorithm for ALTER table: Default

Default lock for ALTER table: Default

Views

[✓] Reformat DDL for Views

Online DDL

- **Default algorithm for ALTER table**: The default algorithm selected when performing `ALTER TABLE` operations in MySQL Workbench. The setting can also be adjusted for each `ALTER TABLE` operation. Options include "In-Place" (preferred) and "Copy", see the online DDL documentation for more information.

- **Default lock for ALTER table**: The default lock setting for allowing concurrent queries with `ALTER TABLE` in MySQL Workbench. This setting can also be adjusted for each `ALTER TABLE` operation. Options include "None", "Shared", and "Exclusive", see the online DDL documentation for more information.

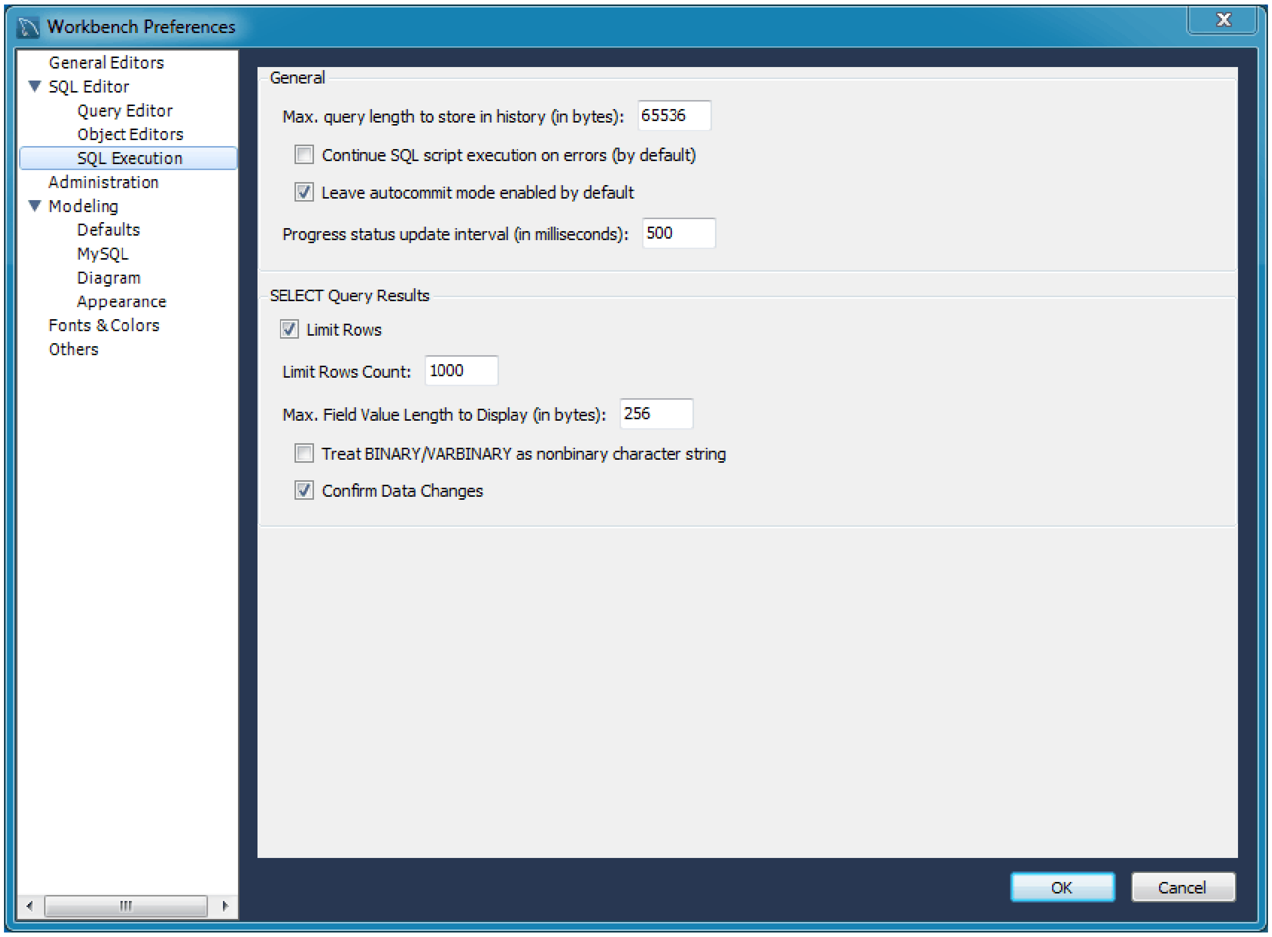
Views

- **Reformat DDL for Views**: Whether to automatically reformat the View DDL that is returned by the MySQL Server.

> **Note**
>
> The MySQL Server does not store the formatting information for View definitions.

Preferences: SQL Editor: SQL Execution

Figure 3.9 Preferences: SQL Editor: SQL Execution



General

Max. query length to store in history (in bytes): 65536

☐ Continue SQL script execution on errors (by default)

☑ Leave autocommit mode enabled by default

Progress status update interval (in milliseconds): 500

SELECT Query Results

☑ Limit Rows

Limit Rows Count: 1000

Max. Field Value Length to Display (in bytes): 256

☐ Treat BINARY/VARBINARY as nonbinary character string

☑ Confirm Data Changes

General

- **Max query length to store in history (in bytes)**: Queries that exceed this size will not be saved in the history when executed. The default is 65536 bytes, and setting to 0 means there is no limit (all queries will be saved).
- **Continue on SQL Script Error**: Should an error occur while executing a script, this option causes execution to continue for the remainder of the script.
- **Leave autocommit mode enabled by default**: Toggles the default autocommit mode for connections. When enabled, each statement will be committed immediately.

Note

All query tabs in the same connection share the same transaction. To have independent transactions, you must open a new connection.

- **Progress status update interval**: When executing long running queries over a slow connection, you may need to increase this value to prevent excess load on the connection. Defaults to 500 milliseconds.

SELECT Query Results

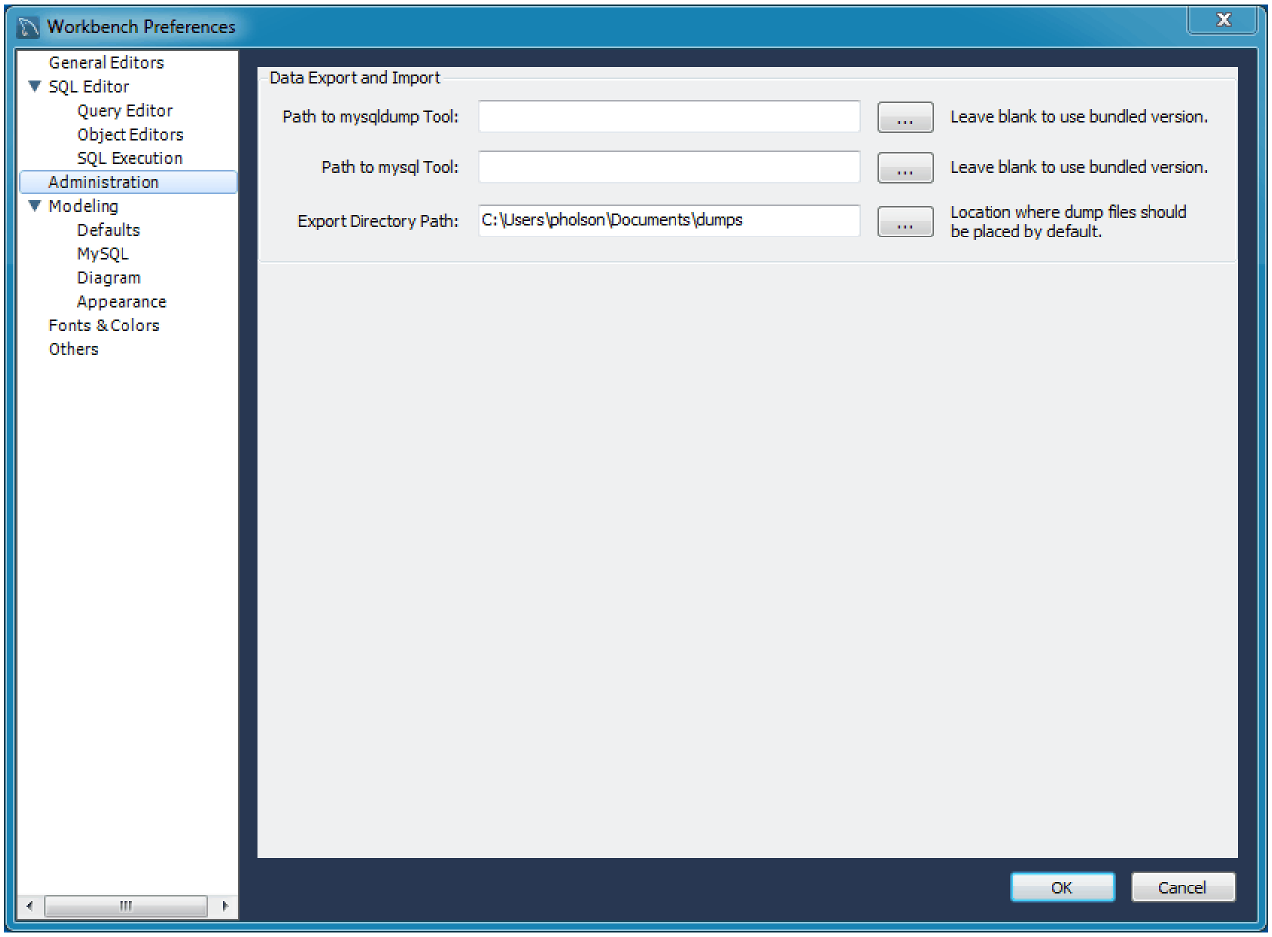
- **Limit Rows**: Queries can sometimes return an excessive number of rows, which can heavily load the connection, and take time to display in MySQL Workbench. To prevent this, you can set a more moderate value here. This limit is defined by the **Limit Rows Count** option.
- **Limit Rows Count**: Specify the maximum number of result rows to return. Defaults to 1000.
- **Max. Field Value Length to Display**: To avoid display problems due to excessive field length, it is possible to set the maximum field length to display (in bytes). Defaults to 256.
- **Treat BINARY/VARBINARY as non-binary character string**: Binary byte string values are not displayed by default in the results grid, but are instead marked as `BLOB` values. These can then be viewed or edited with the `BLOB` editor. Nonbinary character string values are displayed in the results grid, and can be edited in the grid cell or using the `BLOB` editor.

  If this option is turned on, data truncation may result: Binary byte string values may contain null bytes as part of their valid data, whereas for nonbinary character strings, a null byte terminates the string.
- **Confirm Data Changes**: In the SQL Editor, if you edit table data and then click the **Applying changes to data** button, MySQL Workbench launches a wizard to step you through applying your changes. This gives you a chance to review the SQL that will be applied to the live server to make the requested changes. If this option is deselected, the changes will be applied to the server without the wizard being displayed and without giving you a chance to review the changes that will be made.

3.2.3 Administration Preferences

This section provides configuration options that affect the Administration functionality in MySQL Workbench.

Figure 3.10 Preferences: Administration



Data Export and Import

Path to mysqldump Tool: Leave blank to use bundled version.

Path to mysql Tool: Leave blank to use bundled version.

Export Directory Path: C:\Users\pholson\Documents\dumps Location where dump files should be placed by default.

Data Export and Import

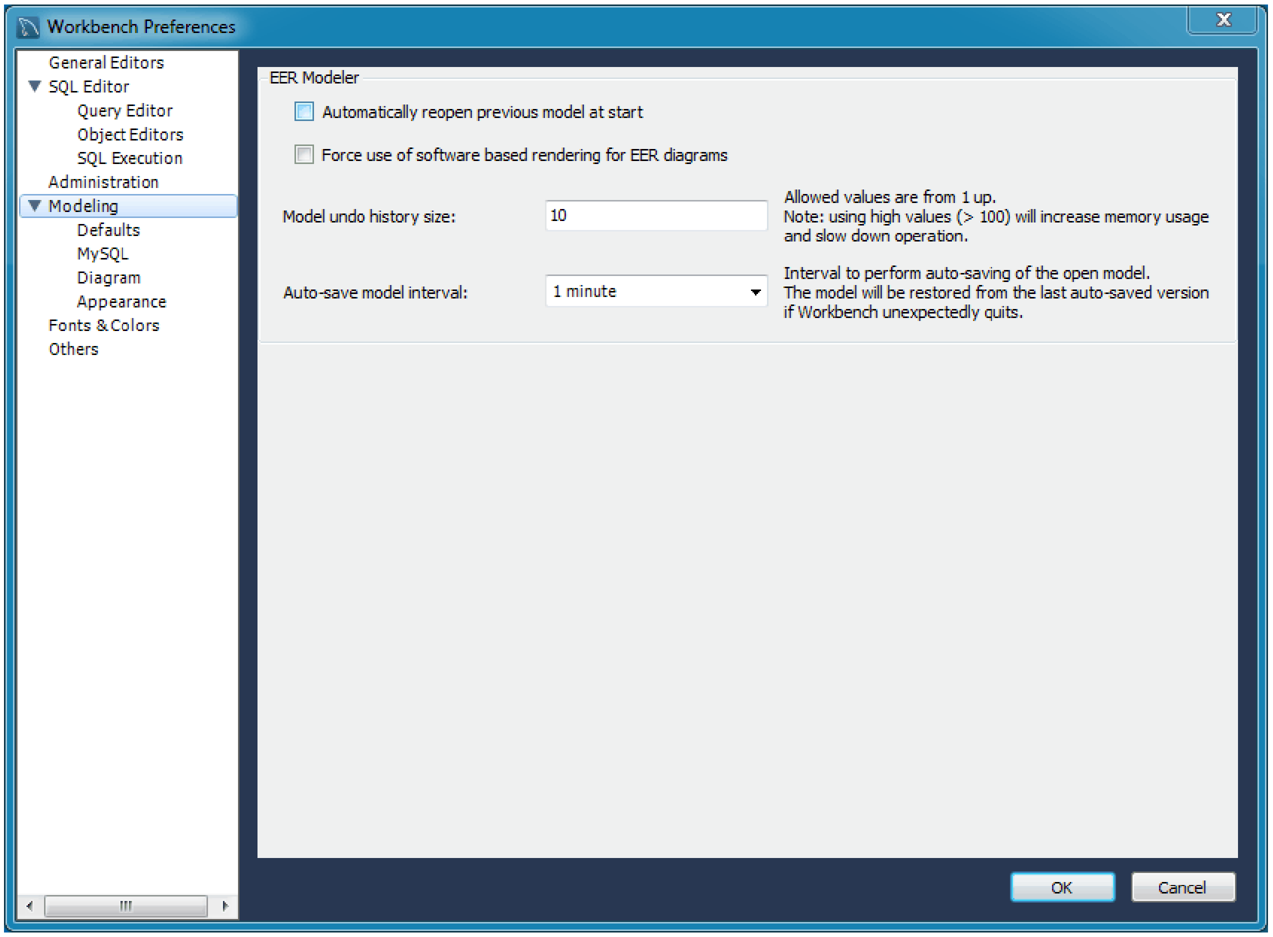
- **Path to mysqldump tool**: Path to your local mysqldump binary. Leave it blank to use the bundled mysqldump binary.
- **Path to mysql tool**: Path to your local mysql client binary. Leave it blank to use the bundled mysql binary.
- **Export Directory Path**: Directory where your exported mysql dumps are located.

3.2.4 Modeling Preferences

This section provides configuration options that affect the Modeling functionality in MySQL Workbench.

Preferences: Modeling: Main

Figure 3.11 Preferences: Modeling



EER Modeler

Automatically reopen previous model at start

Force use of software based rendering for EER diagrams

Model undo history size: 10

Allowed values are from 1 up.
Note: using high values (> 100) will increase memory usage and slow down operation.

Auto-save model interval: 1 minute

Interval to perform auto-saving of the open model.
The model will be restored from the last auto-saved version if Workbench unexpectedly quits.

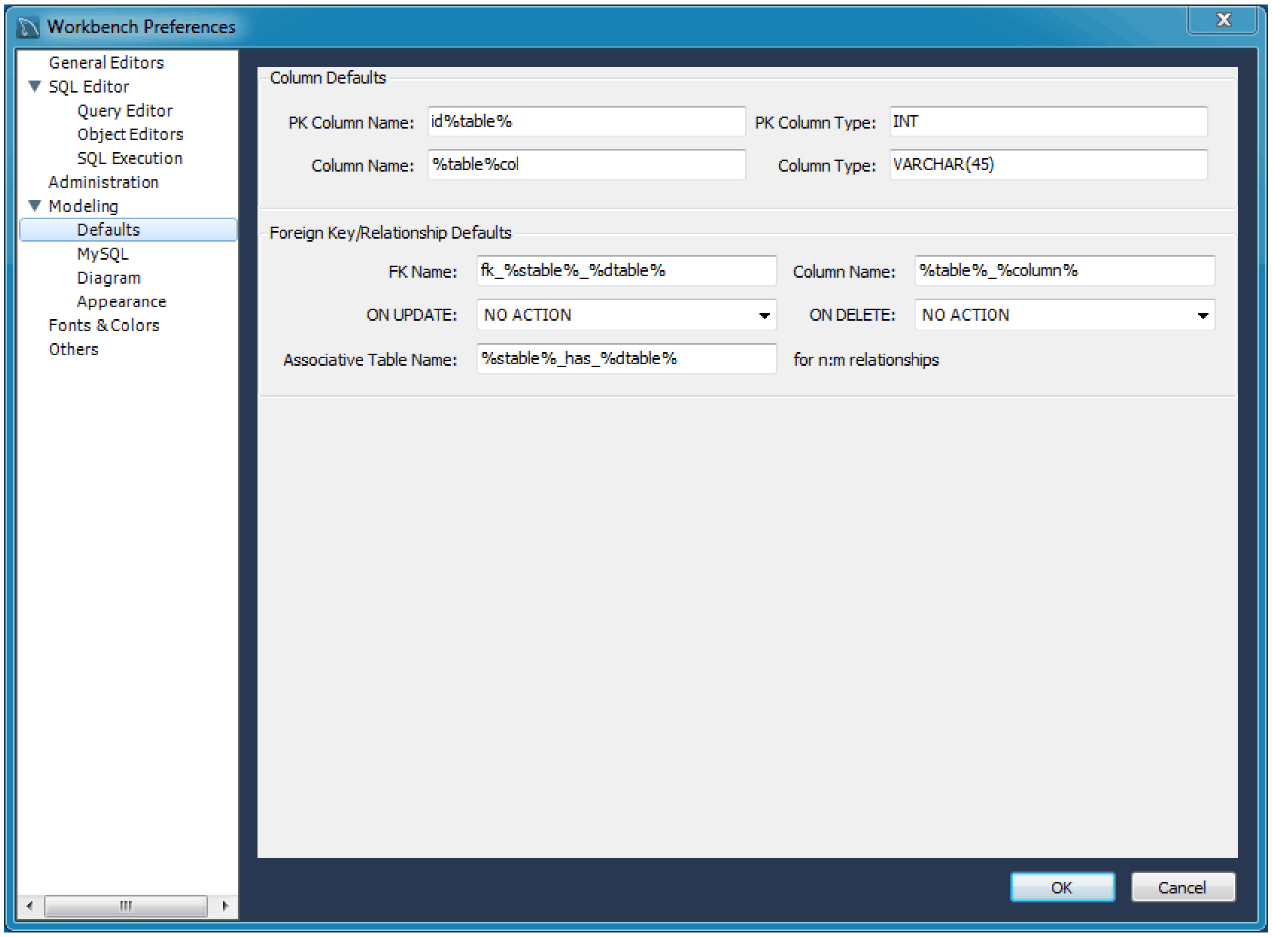
EER Modeler

- **Automatically reopen previous model at start**: Check this if you want the model on which you previously worked to be automatically reopened when you start MySQL Workbench.
- **Force use of software based rendering for EER diagrams**: MySQL Workbench will use OpenGL for rendering when available. However, due to faulty drivers, problems do occasionally occur. These issues can be resolved by selecting the software rendering option here.
- **Model undo history size**: You can limit the size of the undo history here. Set this value to 0 to have an unlimited undo history.
- **Auto-save model interval**: An open model that has not been saved will automatically be saved after this period. On loading a model file, MySQL Workbench will notify the user if the file was not previously saved correctly, due to a crash or power failure. MySQL Workbench can then attempt to recover the last auto-saved version. For automatic recovery to be available for a new file, it will have to have been saved at least once by the user.

Preferences: Modeling: Defaults

Sets default values for modeling object names.

Figure 3.12 Preferences: Modeling: Defaults



Column Defaults

PK Column Name: id%table%
PK Column Type: INT
Column Name: %table%col
Column Type: VARCHAR(45)

Foreign Key/Relationship Defaults

FK Name: fk_%stable%_%dtable%
Column Name: %table%_%column%
ON UPDATE: NO ACTION
ON DELETE: NO ACTION
Associative Table Name: %stable%_has_%dtable% for n:m relationships

The following tables show the object names and their default values.

Column Defaults

Object Name	Default Value
`PK Column Name`	`id%table%`
`PK Column Type`	`INT`
`Column Name`	`%table%col`
`Column Type`	`VARCHAR(45)`

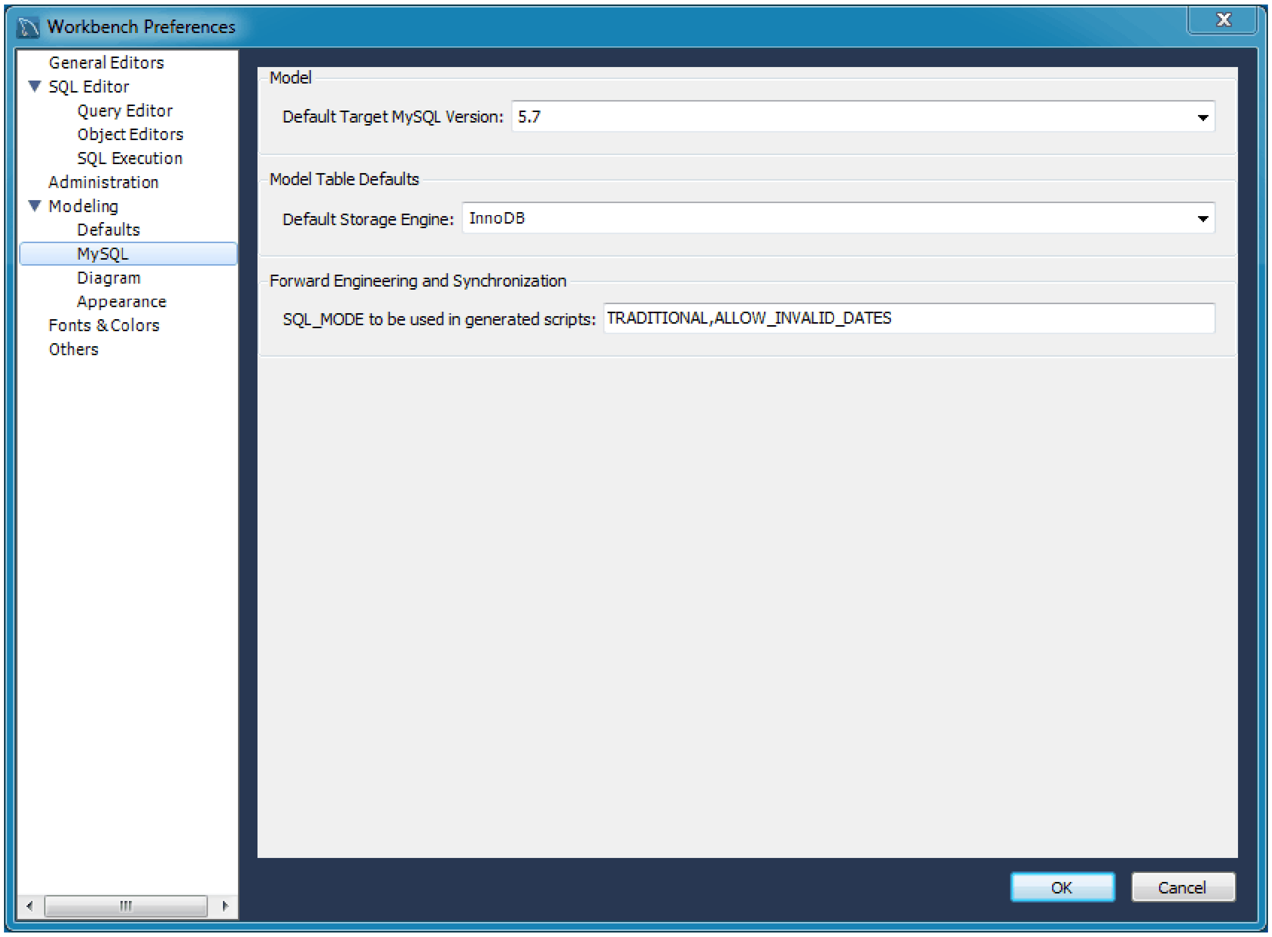
Foreign Key/Relationship Defaults

Object Name	Default Value
`FK Name`	`fk%stable_%dtable%`
`Column Name`	`%table%_%column%`
`ON UPDATE`	`NO ACTION`
`ON DELETE`	`NO ACTION`
`Associative Table Name`	`%stable%_has_%dtable%`

Preferences: Modeling: MySQL

This enables you to set model related options specific to your MySQL version.

Figure 3.13 Preferences: Modeling: MySQL



Model

Default Target MySQL Version: 5.7

Model Table Defaults

Default Storage Engine: InnoDB

Forward Engineering and Synchronization

SQL_MODE to be used in generated scripts: TRADITIONAL,ALLOW_INVALID_DATES

Model

- **Default Target MySQL Version**: A limited subset of validation procedures and table editor options are affected by this MySQL version number.

  Supported MySQL Server 5.6 features include fraction seconds support for `TIME`, `DATETIME`, and `TIMESTAMP`, automatic initialization and updates for `TIMESTAMP` and `DATETIME` (for example, setting them to `CURRENT_TIMESTAMP`), and FULLTEXT index types with the InnoDB engine.

Model Table Defaults

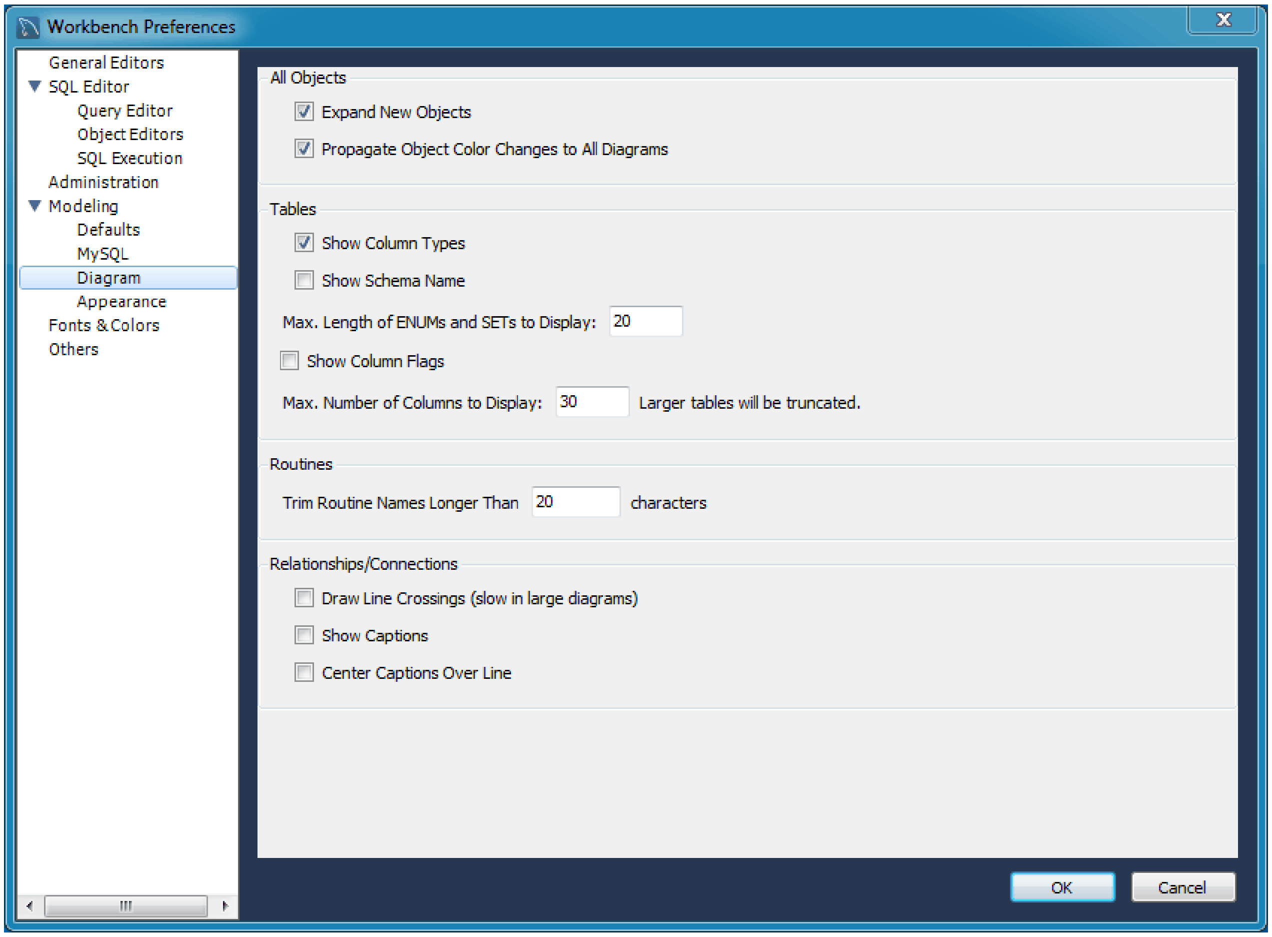
- **Default Storage Engine**: Tables created in MySQL Workbench will be defined using this default storage engine.

Forward Engineering and Synchronization

- **SQL_MODE to be used in generated scripts**: Defaults to "TRADITIONAL,ALLOW_INVALID_DATES", this defines the `SQL_MODE` used by Forward Engineering and Synchronization.

Preferences: Modeling: Diagram

Figure 3.14 Preferences: Modeling: Diagram



All Objects

[✓] Expand New Objects

[✓] Propagate Object Color Changes to All Diagrams

Tables

[✓] Show Column Types

[ ] Show Schema Name

Max. Length of ENUMs and SETs to Display: 20

[ ] Show Column Flags

Max. Number of Columns to Display: 30 Larger tables will be truncated.

Routines

Trim Routine Names Longer Than 20 characters

Relationships/Connections

[ ] Draw Line Crossings (slow in large diagrams)

[ ] Show Captions

[ ] Center Captions Over Line

All Objects

- [ ] **Expand New Objects**: Enabled by default.
- [ ] **Propagate Object Color Changes to All Diagrams**: Enabled by default.

Tables

- [ ] **Show Column Types**: Enabled by default.
- [ ] **Show Schema Name**:
- **Max. Length of ENUMs and SETs to Display**: [*20*]
- [ ] **Show Column Flags**:
- **Max. Number of Columns to Display**: [*30*] Larger tables will be truncated.

Routines

- **Trim Routine Names Longer Than**: [*20*] characters.

Relationships / Connections

- [ ] **Draw Line Crossings (slow in large diagrams)**:

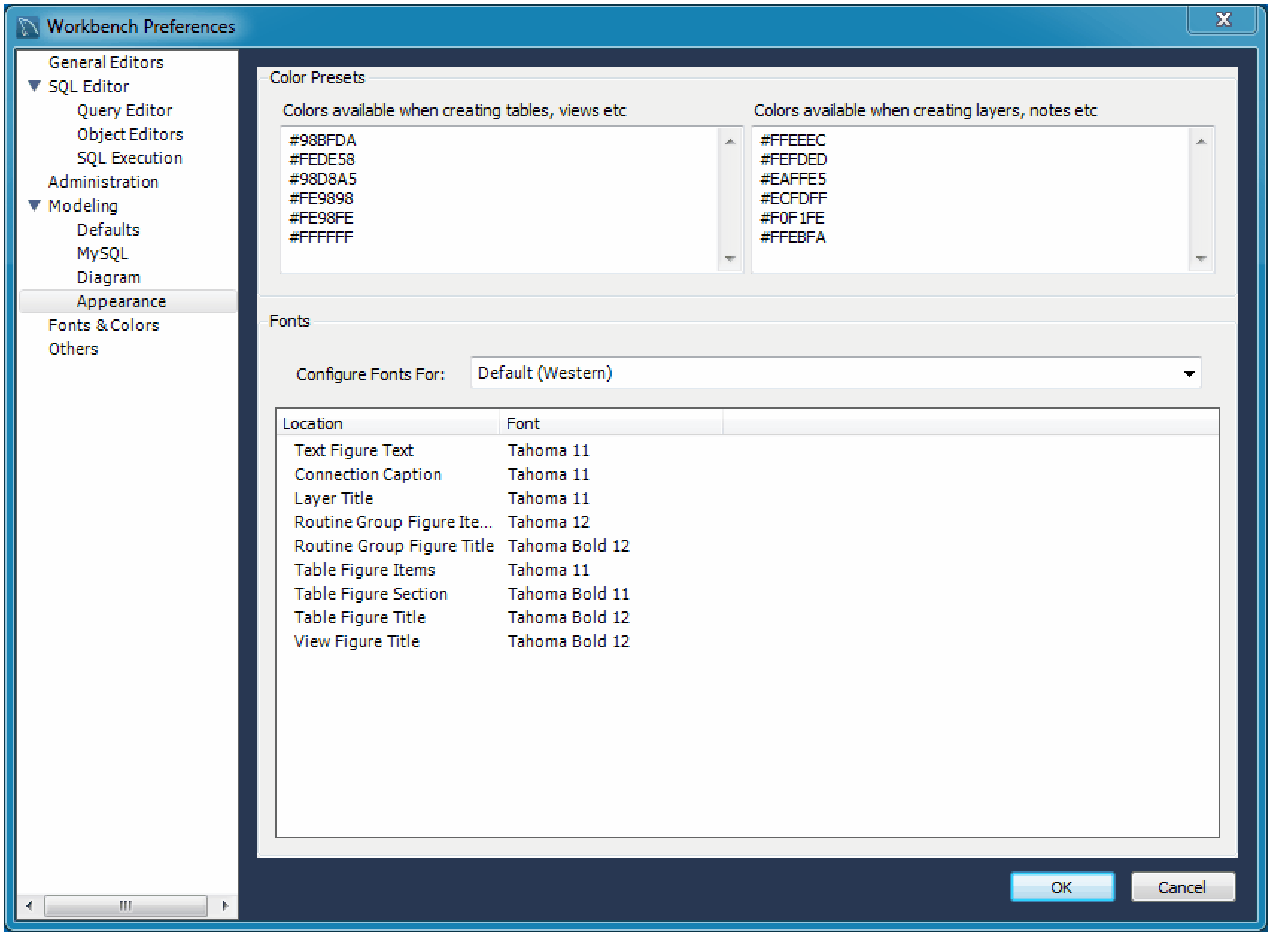
- [ ] **Hide Captions**: Enabled by default.
- [ ] **Center Captions Over Line**:

Preferences: Modeling: Appearance

Use this tab to set the available colors for the objects that appear on an EER diagram canvas. You can also add colors if you wish.

For related information, see Section 3.1, “User Accessibility Options”.

Figure 3.15 Preferences: Modeling: Appearance



Color Presets

Colors available when creating tables, views etc

```
#98BFDA
#FEDE58
#98D8A5
#FE9898
#FE98FE
#FFFFFF
```

Colors available when creating layers, notes etc

```
#FFEEEC
#FEFDED
#EAFFE5
#ECFDFF
#F0F1FE
#FFEBFA
```

Fonts

Configure Fonts For: Default (Western)

Location	Font
Text Figure Text	Tahoma 11
Connection Caption	Tahoma 11
Layer Title	Tahoma 11
Routine Group Figure Ite...	Tahoma 12
Routine Group Figure Title	Tahoma Bold 12
Table Figure Items	Tahoma 11
Table Figure Section	Tahoma Bold 11
Table Figure Title	Tahoma Bold 12
View Figure Title	Tahoma Bold 12

Color Presets

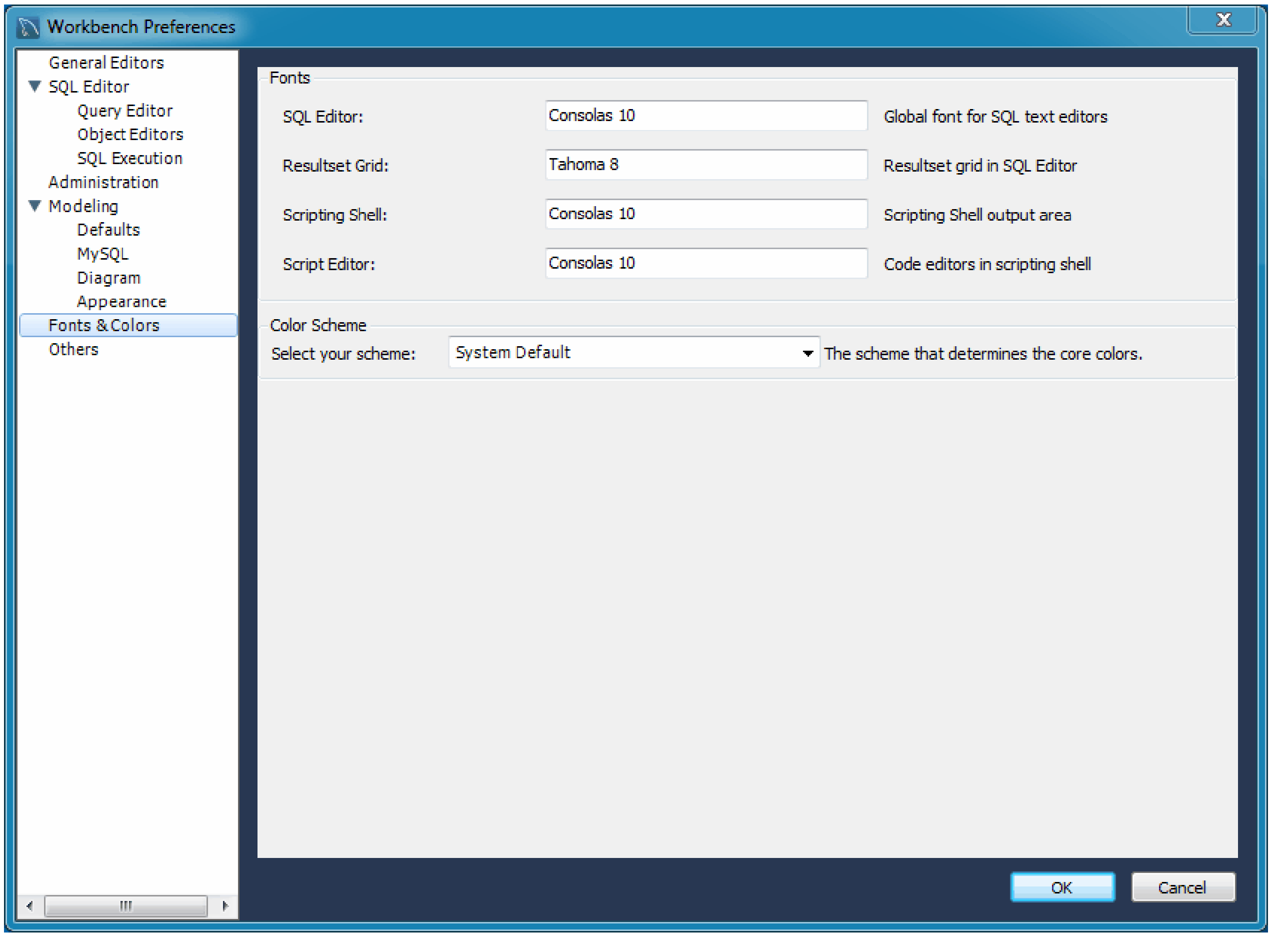
These are the available colors used while modeling, and they are divided into two sections. First, the colors used when creating tables and views. The second section are available colors for items such as layers and notes.

Fonts

These define the fonts and font sizes used while modeling.

3.2.5 Fonts and Colors Preferences

Figure 3.16 Preferences: Fonts and Colors



Fonts

SQL Editor: Consolas 10 Global font for SQL text editors

Resultset Grid: Tahoma 8 Resultset grid in SQL Editor

Scripting Shell: Consolas 10 Scripting Shell output area

Script Editor: Consolas 10 Code editors in scripting shell

Color Scheme

Select your scheme: System Default The scheme that determines the core colors.

Fonts

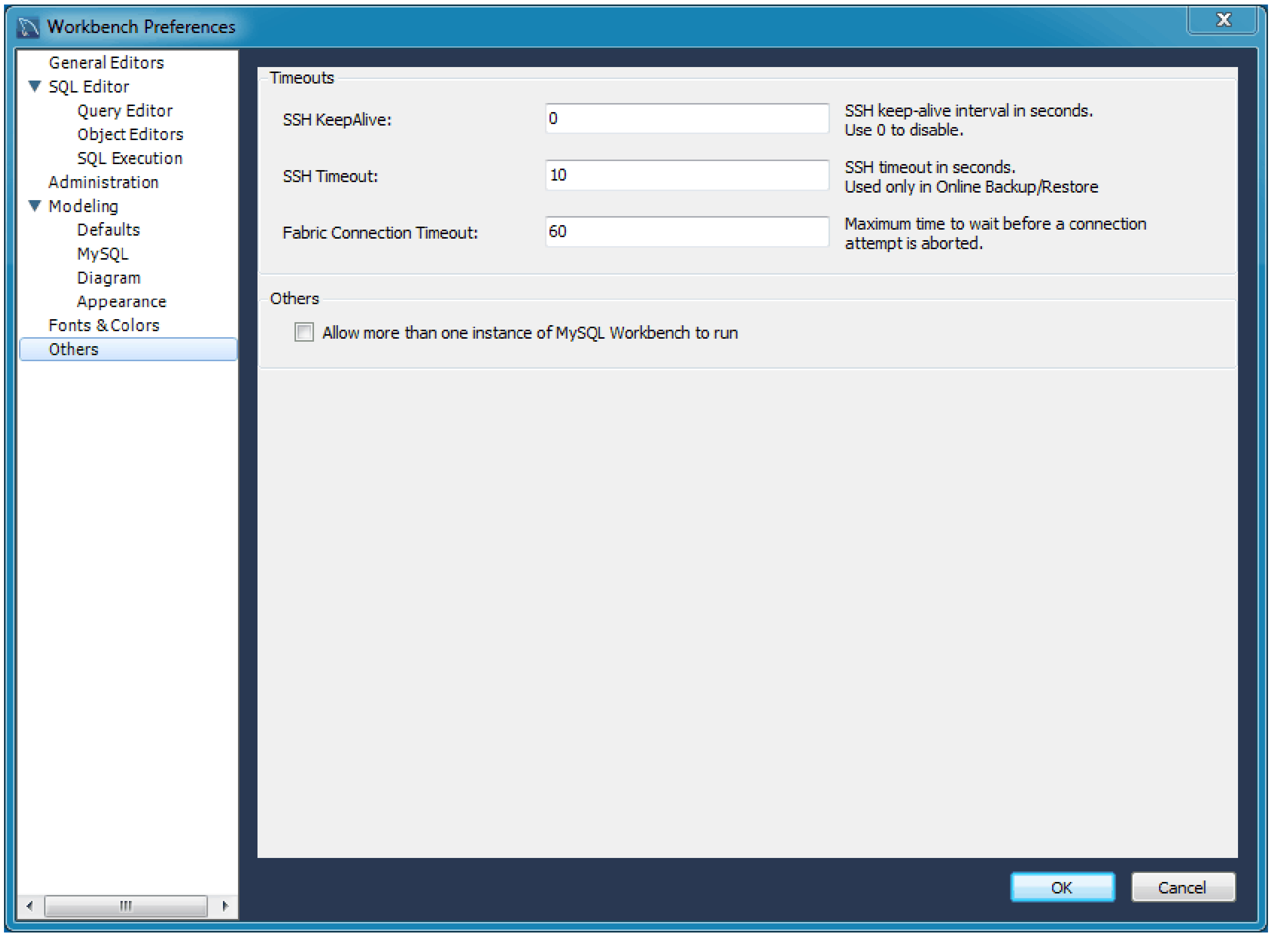
- **SQL Editor**: [`Consolas 10`] -- Global font for SQL text editors.
- **Resultset Grid**: [`Tahoma 8`] -- Resultset grid in SQL editor
- **Scripting Shell**: [`Consolas 10`] -- Scripting Shell output area
- **Script Editor**: [`Consolas 10`] -- Code editors in scripting shell

Color Scheme

On Microsoft Windows, set the scheme that determines the code colors.

3.2.6 Other Preferences

Figure 3.17 Preferences: Others



Timeouts

SSH KeepAlive: 0 — SSH keep-alive interval in seconds. Use 0 to disable.

SSH Timeout: 10 — SSH timeout in seconds. Used only in Online Backup/Restore

Fabric Connection Timeout: 60 — Maximum time to wait before a connection attempt is aborted.

Others

[ ] Allow more than one instance of MySQL Workbench to run

Timeouts

- **SSH KeepAlive**: [*0*] -- This interval (in seconds) without sending any data over the connection, a "keepalive" packet will be sent. This option applies to both SSH tunnel connections and remote management via SSH.
- **SSH Timeout**: [*10*] -- This interval (in seconds) that online backup/restore will timeout when waiting for a result.

  Note

  This option was added in Workbench 6.3.5.

- **Fabric Connection Timeout**: [*60*] -- Maximum time to wait before a connection is aborted.

Others

- [ ] Allow more than one instance of MySQL Workbench to run. By default, only one instance of MySQL Workbench can be running at the same time. .

Note

All MySQL Workbench instances share the same files and settings, so enable at your own risk.

3.3 MySQL Workbench Settings and Log Files

MySQL Workbench saves configuration, cache, and log related files and directories on your system. These files are saved in your user's MySQL Workbench directory as defined by MySQL Workbench, with this base defaulting to:

Table 3.1 Default Local Configuration Base File Path

Operating System	File Path
Windows	%AppData%\MySQL\Workbench\
OS X	~username/Library/Application Support/MySQL/Workbench/
Linux	~username/.mysql/workbench/

A brief description of these directories and files:

Table 3.2 Local Workbench Files and Directory Descriptions

Directory or File	Description
cache/	General behaviors are stored per-connection in *.cache files, and column widths as *.column_widths files
log/	Log files include Workbench startup information, and also per-connection SQL action results performed in Workbench
modules/	Home of installed plugins, for additional information see Section C.3, “Plugins / Tools”
sql_history/	Queries executed in Workbench are stored here, and are available from within MySQL Workbench
snippets/	Saved SQL snippets are stored here, for additional information see Section 8.1.5, “SQL Snippets tab”
audit_cache/	Cache storage by the Audit Log inspector, for additional information see Section 6.6, “MySQL Audit Inspector Interface”
`connections.xml`	Saved MySQL server connection information, as seen on the home screen. For information about backing up and restoring this file, see The Tools Menu
`server_instances.xml`	Stores your MySQL server information, as it relates to your MySQL connections
`wb_options.xml`	Stores your preferences, both configured and default

The cache/ directory

The `cache/` directory contains cache files in the user's MySQL Workbench directory. All cache files are stored as SQLite 3 databases, and they are not meant to be edited outside of MySQL Workbench. The types of cache files are:

- ***.column_widths**:

These are the saved column widths after adjusting columns in the SQL editor's results grid. The fields include column_id, stored as column_name::db_name::table_name, and width, stored as an integer of character length.

- ***.cache**:

  This information (schemas, engines, and other global information) serves as a quick lookup source for the SQL editor's auto completion functionality, and is implicitly updated whenever the schema tree is updated.

All cache/ file names begin with the MySQL connection name. For example, the column width file is named `Local_instance_3306.column_widths` for a MySQL connection named "Local Instance 3306".

Cached files remain after a connection is either renamed or deleted.

The log/ directory

MySQL Workbench start up and SQL actions are logged and stored in the `log/` directory. This directory is in the user's MySQL Workbench directory.

Note

To find these text files, from the main Workbench navigation menu choose **Help** and then **Show Log Files**.

- **wb*.log**:

  Debugging information is generated when MySQL Workbench is started and unexpectedly stopped. Information includes paths used, modules and plugins loaded, system information, and more. The log files are useful when reporting a MySQL Workbench bug.

  The log files rotates when MySQL Workbench is started, in that `wb.log` is renamed to `wb.1.log`, `wb.log` is reset, and the previous `wb.1.log` file is renamed to `wb.2.log`, and so on, all the way up to `wb.9.log`.

- **sql_actions_*.log**:

  A log of all SQL execution results but without the data, for debugging purposes.

  The SQL editor's SQL history does not originate from here, as it is stored in the `sql_history` directory.

The modules/ directory

Custom plugins (modules) are stored in the `modules` directory. For additional information about MySQL Workbench plugins, see Section C.3, “Plugins / Tools”.

The sql_history/ directory

SQL statements executed in the SQL editor are saved in the `sql_history` directory. They are stored as plain text files that are separated one per day (such as `2015-12-15`) and they contain your MySQL Workbench SQL statement history for all MySQL connections. For additional information, see Section 8.1.7, “Output History Panel”.

The snippets/ directory

SQL snippets used by the SQL editor are stored in the `snippets` directory. These files include bundled snippets (such as "SQL DDL Statements") and custom snippets saved under the "My Snippets" tab. For additional information, see Section 8.1.5, “SQL Snippets tab”.

3.4 Tutorial: Add a Custom Link to the Home Page

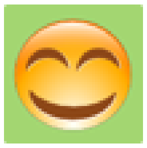
This tutorial introduces the concept of altering the MySQL Workbench home screen by adding your own Shortcut link. Here we will add a shortcut titled "Example" that opens "example.org" as its own browser tab inside MySQL Workbench.

Note

Although adding a link to the Home screen is not a common need, this tutorial is an example that demonstrates the idea of customizing MySQL Workbench.

First, create an icon for your shortcut. This step is optional, and the table below compares the icon used in this tutorial and the default icon if a custom icon is not defined.

Table 3.3 MySQL Workbench Home Screen Icons

Default Icon	Our Example Icon
	

Save your new 52x52 pixel image to a location accessible by MySQL Workbench. By default, source icons are stored here:

Table 3.4 Default Path for Home Screen Icons

Operating System	File Path
Windows	"C:\Program Files (x86)\MySQL\MySQL Workbench CE 6.3.7\images\home\
OS X	/Applications/MySQLWorkbench.app/Contents/Resources/
Linux	/usr/share/mysql-workbench/images/

Next, open `starters_settings.xml` (this file is located under your user's MySQL Workbench directory) and add an entry for your new shortcut ID where the order determines the location on the Home screen. Follow the standard convention by appending your value to "com.mysql.wb.starter.", this tutorial uses "example":

```
<link type="object">com.mysql.wb.starter.example</link>
```

Lastly, open `predefined_starters.xml` and add a new "app.Starter" entry.

Table 3.5 Default Path to predefined_starters.xml

Operating System	File Path
Windows	"C:\Program Files (x86)\MySQL\MySQL Workbench CE 6.3.7\data\
OS X	/Applications/MySQLWorkbench.app/Contents/Resources/data/
Linux	/usr/share/mysql-workbench/data/

This tutorial does not describe this entry in detail, so consider it as a self-explanatory template for now. The important concepts include using the "id" you defined in the previous step, your own URL for the "command", and "smallIcon" as a 52x52 pixel image that is displayed on the MySQL Workbench home screen. Adjust these entries according to your needs, including the path to your icon.

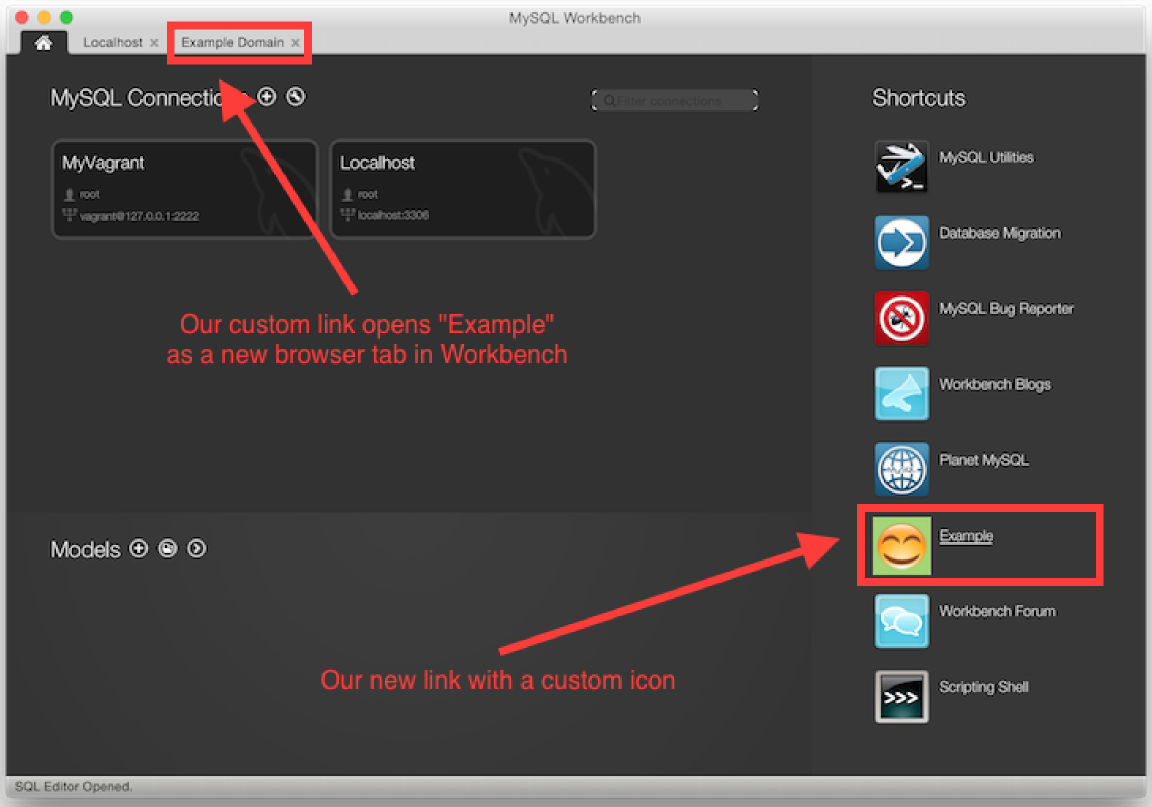
```
<value type="object" struct-name="app.Starter" id="com.mysql.wb.starter.example">
 <value type="string" key="type">Website</value>
 <value type="string" key="title">Example</value>
 <value type="string" key="description">My wonderful example.org</value>
 <value type="string" key="publisher">Example Inc.</value>
 <value type="string" key="authorHome">http://www.example.org/about</value>
 <value type="string" key="smallIcon">/usr/local/share/wb-home-screen-example-icon.png</value>
 <value type="string" key="command">browse:http://example.org</value>
</value>
```

Note

Upgrading MySQL Workbench will overwrite these changes, because `predefined_starters.xml` is stored inside the MySQL Workbench installation directory, and `starters_settings.xml` is reset during installation. Consider saving copies of these changes for future reference.

Restart MySQL Workbench to see the new link on your MySQL Workbench home screen.

Figure 3.18 Home Screen with Custom Link



Additional "app.Starter" options include:

```
Require a specific edition of Workbench:
  Community:
    <value type="string" key="edition">ce</value>
  Commercial:
    <value type="string" key="edition">se</value>

Require a specific version (or higher) of Workbench:
  <value type="string" key="introduction">6.1.0</value>
```

3.5 Common Preferences and Configurations

Commonly used configuration options and preferences include:

- **Rescan for Local MySQL Instances**: Right-click on the Home screen, and this option will scan your system for MySQL instances and add connection tiles to the home screen.
- **Safe Updates**: When enabled (default), Workbench will not execute UPDATE or DELETE statements if a key is not defined in the WHERE clause. In other words, Workbench attempts to prevent big mistakes, such as deleting a large number of (or all) rows. Set from the **SQL Editor** preferences tab.

  For example, "DELETE FROM foo" is considered unsafe, whereas "DELETE FROM foo WHERE id = 1" is safe and will always execute.
- **Default Target MySQL Version**: For modeling, set this **Modeling** MySQL preference to your target MySQL Server version. This affects the generated syntax and database structure in relation to how MySQL changed over time. Having the wrong version may generate invalid syntax for your MySQL server.
- **Combine Management Tools and Schema Tree**: This refers to the left panel in the SQL Editor, where the **Management** and **Schemas** areas are on one or two separate tabs.

  This behavior can also be toggled at runtime by clicking the icon.
- **Save snapshot of open editors on close**: By default, Workbench saves all query tabs and reopens them when you restart Workbench. Use the related **Auto-save scripts interval** setting to modify its behavior. Both are set from the **SQL Editor** preferences tab.

  Related behavior: Right-click on an SQL tab and choose either **Save tab** (to save the tab to a file) or **Close Other Tabs** to close all other SQL editor tabs.
- **Enable Code Completion in Editors**: Code suggestions can be activated either manually, or automatically if the related **Automatically Start Code Completion** setting is also enabled. In addition, enable **Use UPPERCASE keywords on completion** to code suggest upper case SQL keywords, such as "INSERT" instead of "insert".

  Related behavior: The **Context Help** right panel in the SQL editor displays documentation for SQL statements, and is disabled/enabled from the right panel. For example, typing INSERT will load documentation for the INSERT statement in the right panel.

Chapter 4 The Home Screen

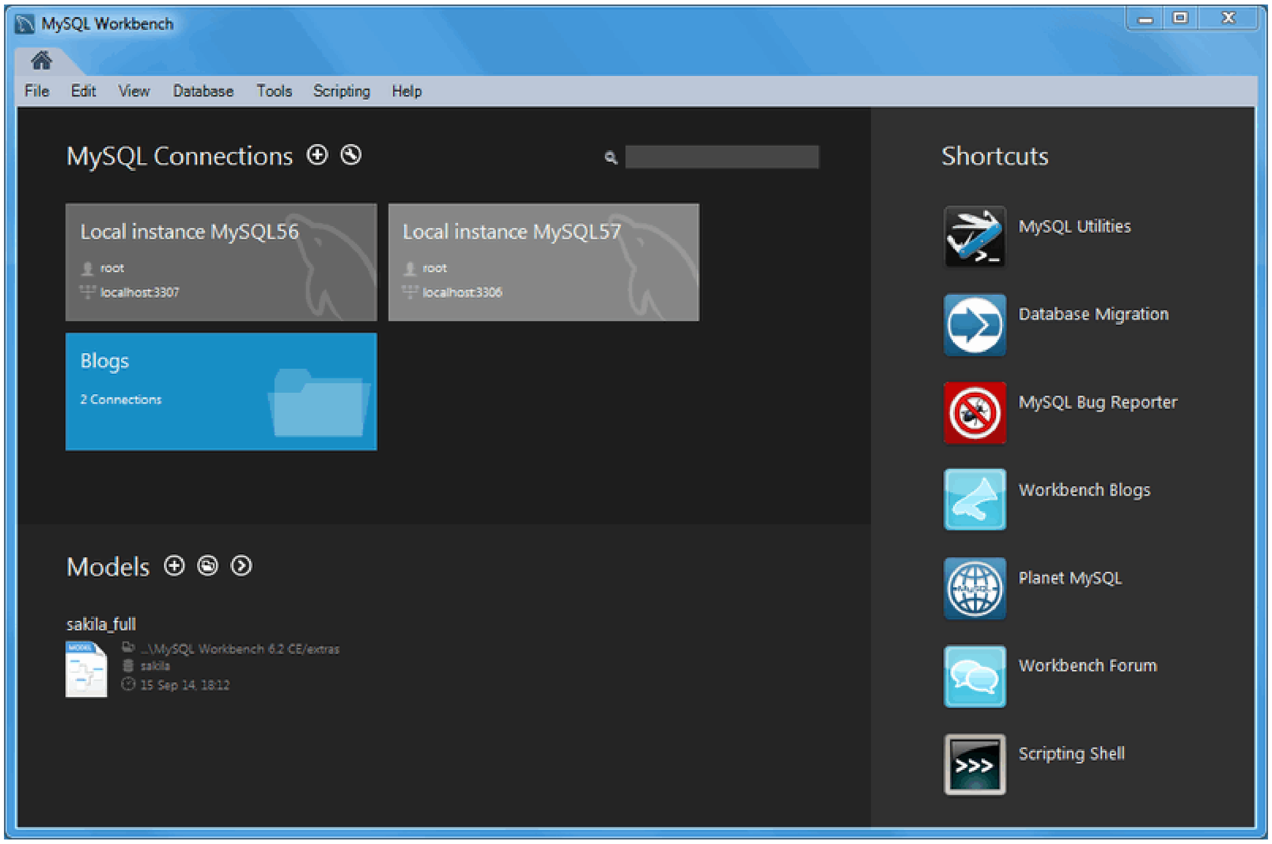
This is the first page you see when opening MySQL Workbench, and it is central to starting MySQL Workbench operations. The three main sections include the **MySQL Connections**, MySQL Workbench **Models**, and external **Shortcuts**.

Note

In MySQL Workbench 5.2 and below, the home screen was different and broken up into sections titled **SQL Development**, **Data Modeling**, and **Server Administration**.

Most MySQL Workbench functionality, such as the SQL editor and MySQL server manager, begin with opening a MySQL connection from the home screen.

Figure 4.1 The Home Window



MySQL Connections

This section lists connections to all of your MySQL servers, and allows you to load, configure, group, and view information about each MySQL connection. For more information, see Chapter 5, *MySQL Connections* and Section 5.2, “Creating A New MySQL Connection (Tutorial)”.

Connection Information

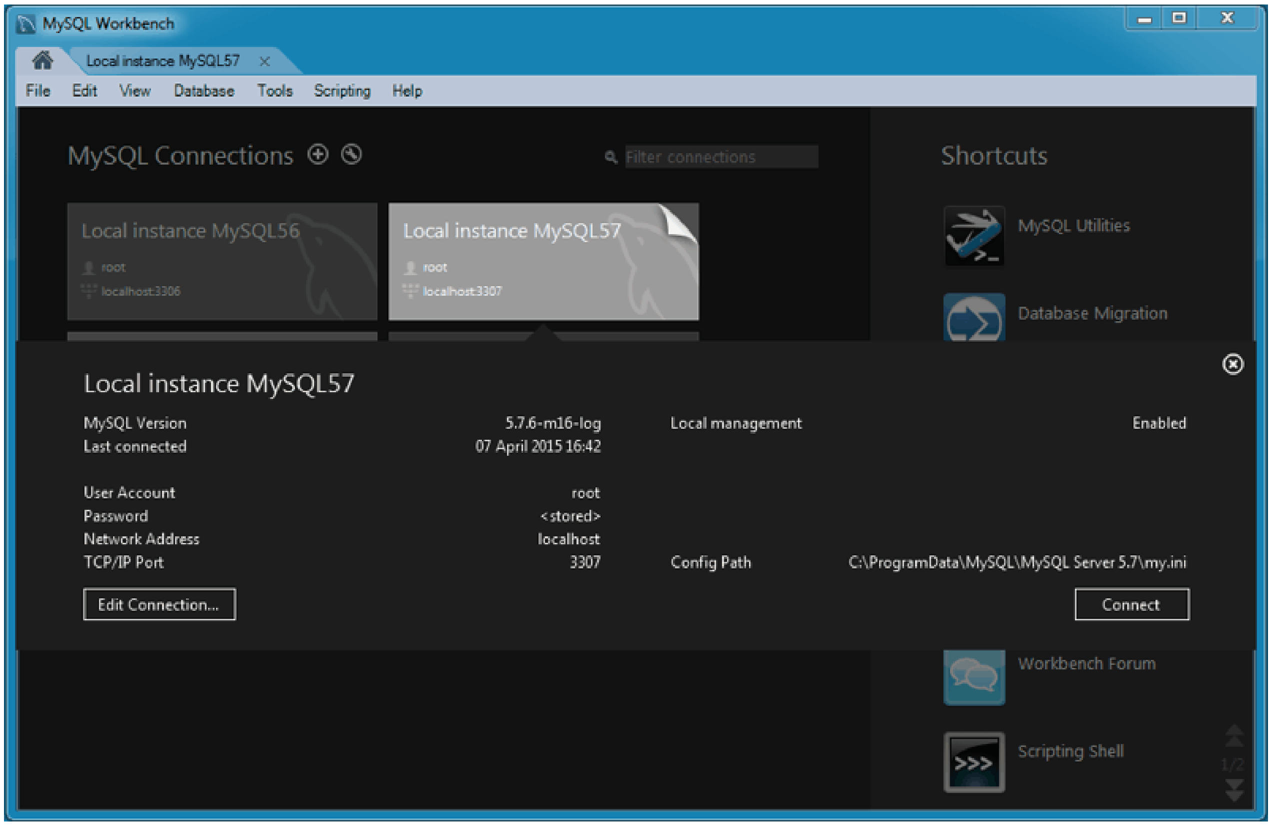
The method to view connection information depends on the operating system.

- On Microsoft Windows and Linux: hover over the right side of a connection title and click the title

- On OS X: hover over a connection title and click the little **(i)** in appears in the bottom right corner

The information will be displayed under the connection tiles, and will appear similar to:

Figure 4.2 Viewing Connection Information



Local instance MySQL57

MySQL Version	5.7.6-m16-log	Local management	Enabled
Last connected	07 April 2015 16:42		
User Account	root		
Password	<stored>		
Network Address	localhost		
TCP/IP Port	3307	Config Path	C:\ProgramData\MySQL\MySQL Server 5.7\my.ini

Connection Groups

You may also create groups of connections. Create a group by either right-clicking a connection and choosing the **Move to group...** context menu option, or you may prefix your connection name with the group name separated by a forward slash (for example, "QA/TestBox") when you create or configure the connection.

Models

The **Models** panel your most recently used models. Each entry lists the date and time that the model was last opened, and its associated database. For further information about modeling, see Chapter 9, *Database Design / Modeling*.

To the right of the **Models** title are three icons. The (+) adds a new model, the (folder) opens an existing model from the disk, and the (>) opens a context menu for additional commands, such as **Create EER Model from Database**.

Workbench Shortcuts

Most of the options you can select on the home screen are straight-forward -- the ones that require explanation are detailed after the screenshot. Here is a quick overview of the simpler options that open your web browser to display information about

- **MySQL Utilities**: If the MySQL Utilities are installed, this opens the `mysqluc` utility. For additional information, see Appendix F, *MySQL Utilities*.
- **Database Migration**: Opens the migration wizard. For additional information, see Chapter 10, *Database Migration Wizard*.
- **MySQL Bug Reporter**: Lets you file a bug report in the MySQL bug system. For additional information, see Appendix D, *How To Report Bugs or Problems*.
- **Workbench Blogs**: Directs you to the Workbench team's blog.
- **Planet MySQL**: Points you to the general MySQL blog aggregator.
- **Workbench Forum**: Sends you to the user forums where you can ask questions and interact with other MySQL Workbench users.
- **Scripting Shell**: Execute Python scripts and develop MySQL Workbench plugins.
- **Oracle eDelivery**: Sends you to your MySQL Workbench downloads (Commercial).
- **Oracle Support**: Sends you to your MOS (My Oracle Support) page (Commercial).

Chapter 5 MySQL Connections

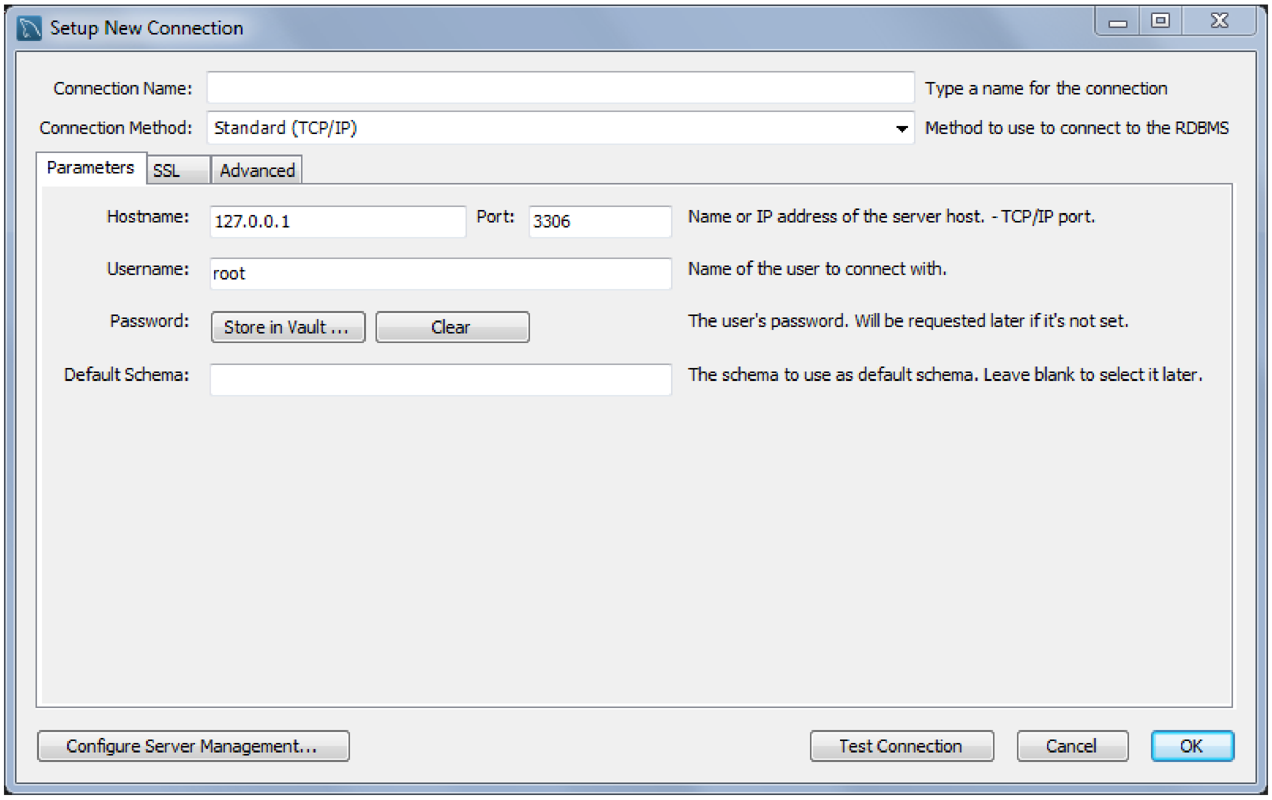
Table of Contents

Manage and create MySQL connections.

5.1 Creating A New MySQL Connection (Simple)

To add a connection, click the `[+]` icon to the right of the **MySQL Connections** title on the Home screen. This opens the **Setup New Connection** form:

Figure 5.1 Setup New Connection Form



Connection Name: Type a name for the connection
Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS
Hostname: 127.0.0.1 Port: 3306 Name or IP address of the server host. - TCP/IP port.
Username: root Name of the user to connect with.
Password: The user's password. Will be requested later if it's not set.
Default Schema: The schema to use as default schema. Leave blank to select it later.

Important

The **Configure Server Management** button (bottom left) opens an optional configuration wizard for setting shell commands on the host. For example, commands to start/stop the MySQL instance, or to edit configuration file. For more information, see Section 5.3.6, “Configure Server Management Wizard”.

Fill out the connection details and optionally click **Configure Server Management** to execute the Server Management wizard. Click **OK** to save the connection.

Important

When opening connections, MySQL Workbench automatically sets the client character set to `utf8`. Manually changing the client character set, such as using `SET NAMES ...`, may cause MySQL Workbench to not correctly display the characters. For additional information about client character sets, see Connection Character Sets and Collations.

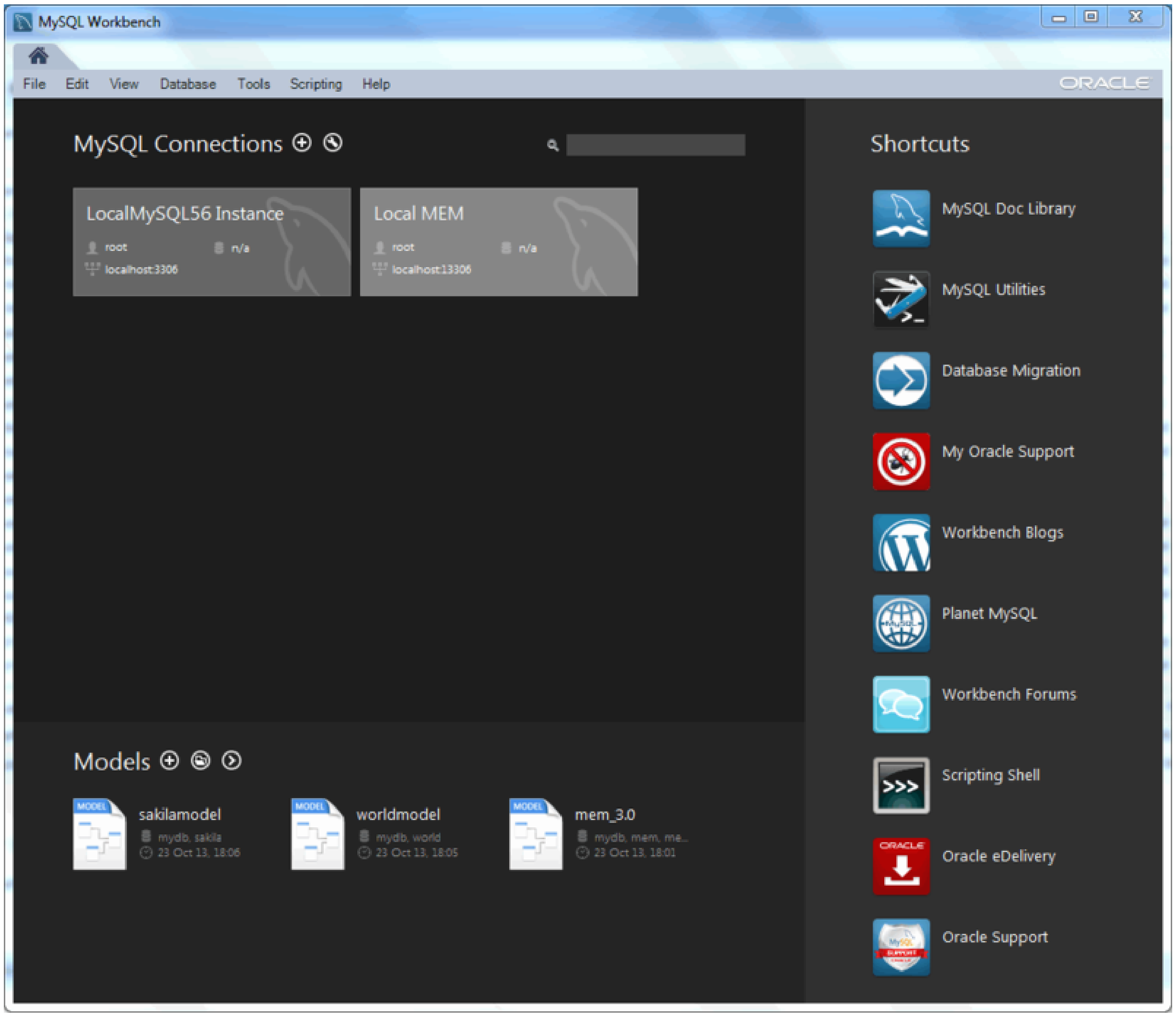
New MySQL connections are added to the Home screen as a tile, and the Section 8.2.1, “Object Browser and Editor Navigator” describes several MySQL Workbench features to monitor and configure each connected MySQL server. A single MySQL Workbench instance can open one or multiple MySQL connections into individual tabs.

For a more detailed overview of this process, see the tutorial titled Section 5.2, “Creating A New MySQL Connection (Tutorial)”.

5.2 Creating A New MySQL Connection (Tutorial)

1. Launch MySQL Workbench. You will be presented with the Home screen.

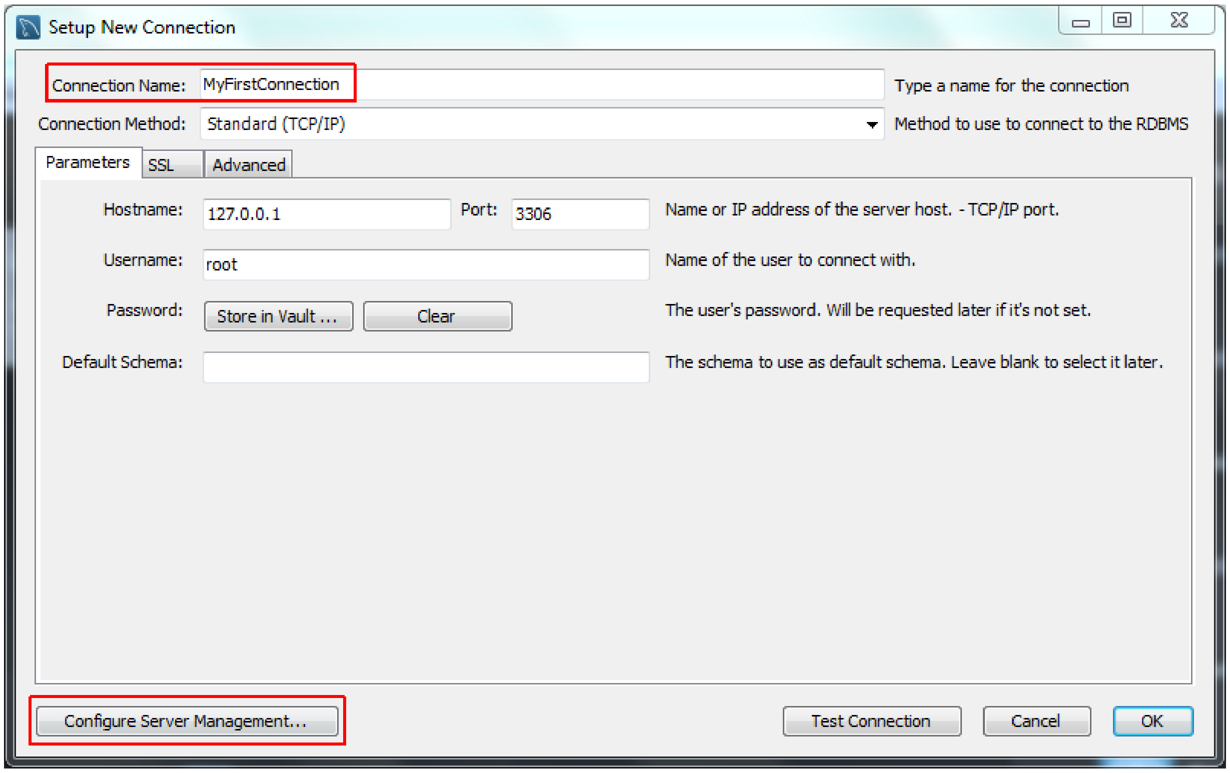
Figure 5.2 Getting Started Tutorial - Home Screen



2. Our example already has two connections created, but let us create a new connection. From the MySQL Workbench Home screen, click the **[+]** icon near the **MySQL Connections** label. This opens the **Setup New Connection** wizard.

3. Define the **Connection Name** value, which we will set to "MyFirstConnection" in this example.

Figure 5.3 Getting Started Tutorial - Setup New Connection: MyFirstConnection


Setup New Connection

Connection Name: MyFirstConnection — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host. - TCP/IP port.

Username: root — Name of the user to connect with.

Password: Store in Vault ... Clear — The user's password. Will be requested later if it's not set.

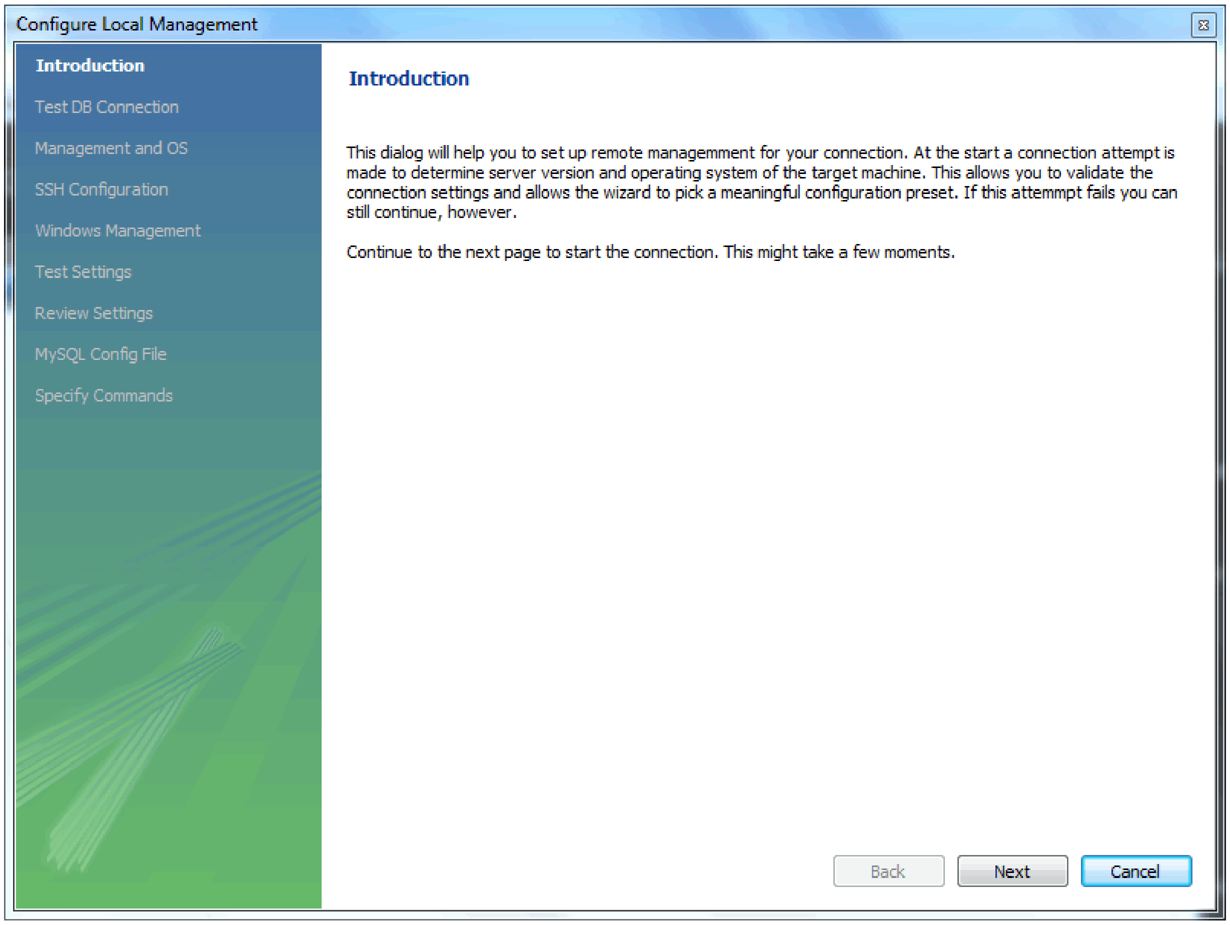
Default Schema: — The schema to use as default schema. Leave blank to select it later.

The default connection values are for a typical local setup, so check them and enter the appropriate values. If you are unsure, click the **Test Connection** button to check the connection parameters. Do not press **OK**.

Next, optionally click **Configure Server Management...**, which opens up the **Configure Local Management** wizard:

4. Read the **Configure Local Management** introduction, and press **Next** to begin defining the new connection parameters.

Figure 5.4 Getting Started Tutorial - Configure Local Management Introduction



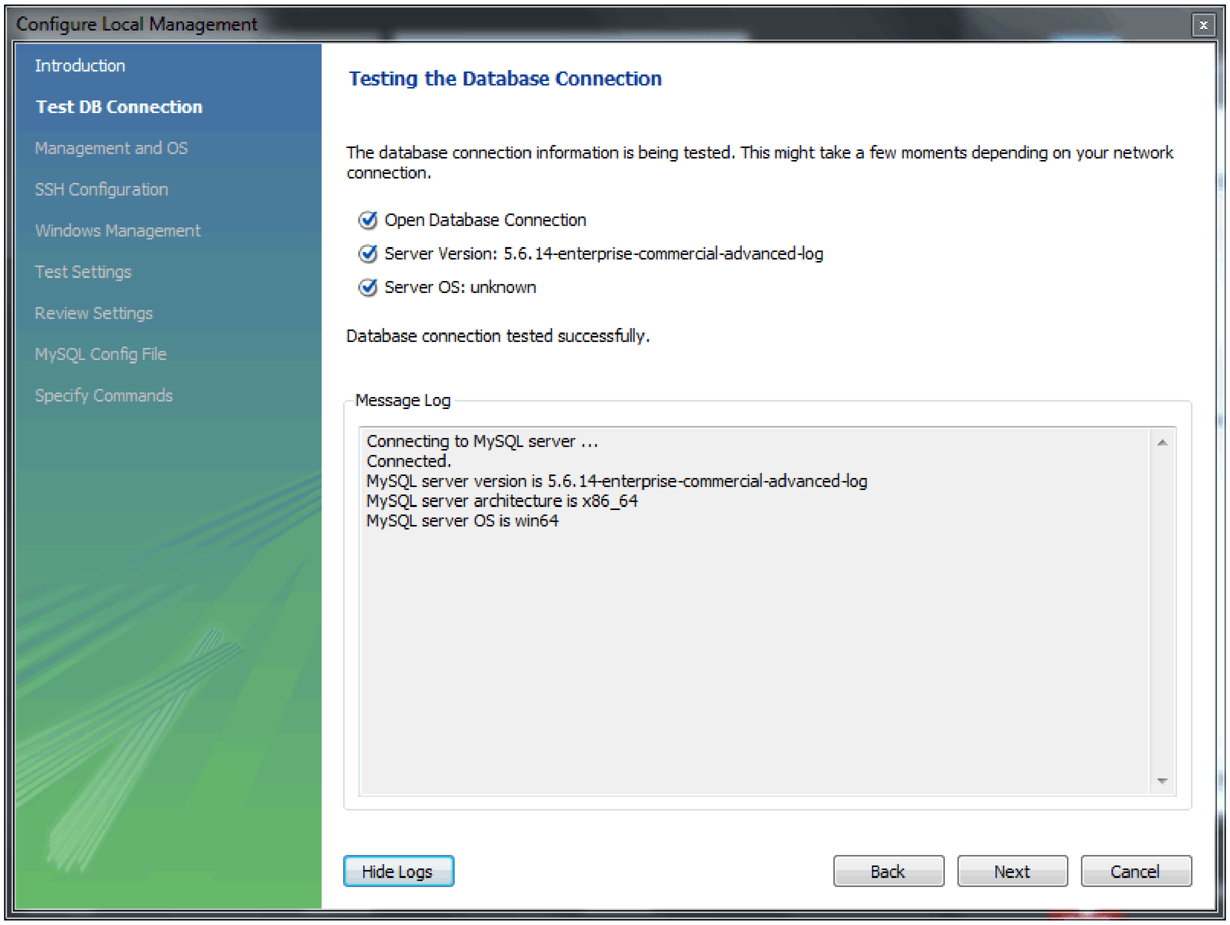
Introduction

This dialog will help you to set up remote managemment for your connection. At the start a connection attempt is made to determine server version and operating system of the target machine. This allows you to validate the connection settings and allows the wizard to pick a meaningful configuration preset. If this attemmpt fails you can still continue, however.

Continue to the next page to start the connection. This might take a few moments.

5. The connection will now be tested. You should see that the connection was successful. If not, click **Back** and check that you have entered the information correctly.

Figure 5.5 Getting Started Tutorial - Test Database Connection



Testing the Database Connection

The database connection information is being tested. This might take a few moments depending on your network connection.

- ✓ Open Database Connection
- ✓ Server Version: 5.6.14-enterprise-commercial-advanced-log
- ✓ Server OS: unknown

Database connection tested successfully.

Message Log

```
Connecting to MySQL server ...
Connected.
MySQL server version is 5.6.14-enterprise-commercial-advanced-log
MySQL server architecture is x86_64
MySQL server OS is win64
```

Toggle the **Show Logs** to view additional details about the tested connection, then click **Next**.

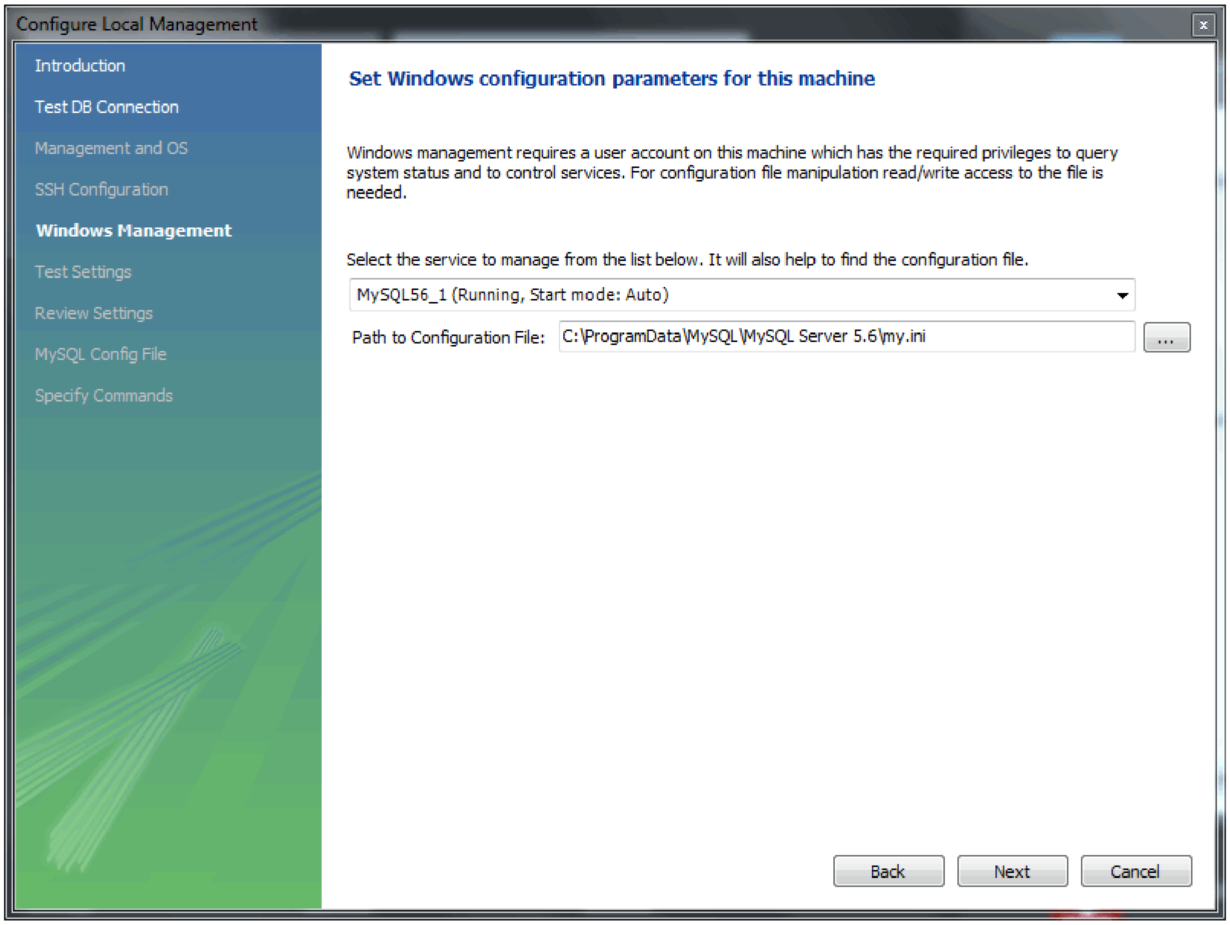
6. Optionally, you may configure a method for remote management if a Remote Host was specified. Setting these options enables MySQL Workbench to determine the location of configuration files, and the correct start and stop commands to use for the connection.

   SSH login based management and Native Windows remote management types are available. The Operating System and MySQL Installation Type are configured for the SSH login variant.

   We are creating a local MySQL connection in this tutorial, so are skipping the **Management and OS** and **SSH Configuration** options, as they are used for configuring a remote MySQL connection.

7. On Microsoft Windows, select the appropriate MySQL service for the MySQL connection.

Figure 5.6 Getting Started Tutorial - Windows Management



Set Windows configuration parameters for this machine

Windows management requires a user account on this machine which has the required privileges to query system status and to control services. For configuration file manipulation read/write access to the file is needed.

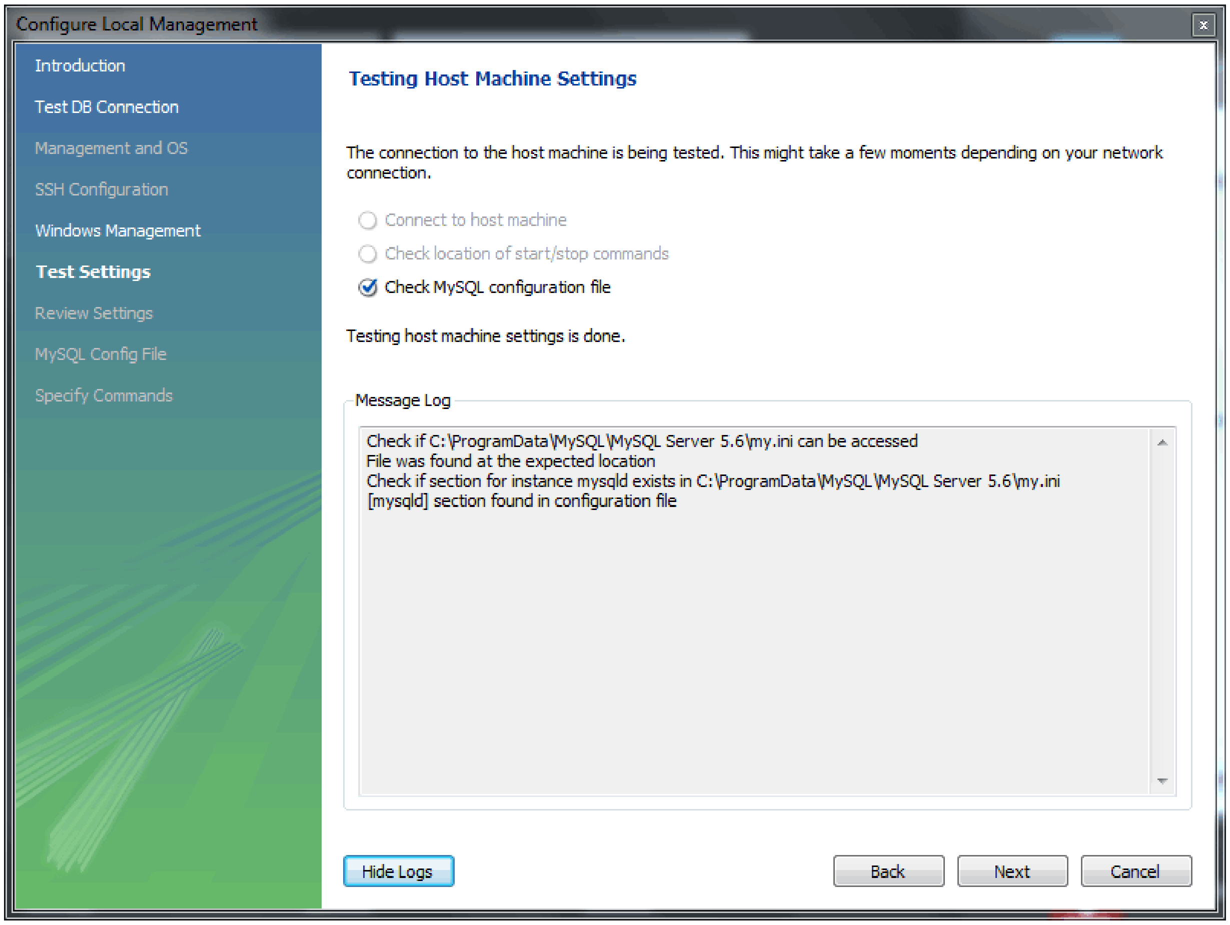
Select the service to manage from the list below. It will also help to find the configuration file.

MySQL56_1 (Running, Start mode: Auto)

Path to Configuration File: C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

8. The wizard will now check its ability to access the start and stop commands, and check access to the MySQL Server configuration file.

Figure 5.7 Getting Started Tutorial - Test Settings



The connection to the host machine is being tested. This might take a few moments depending on your network connection.

Connect to host machine
Check location of start/stop commands
Check MySQL configuration file

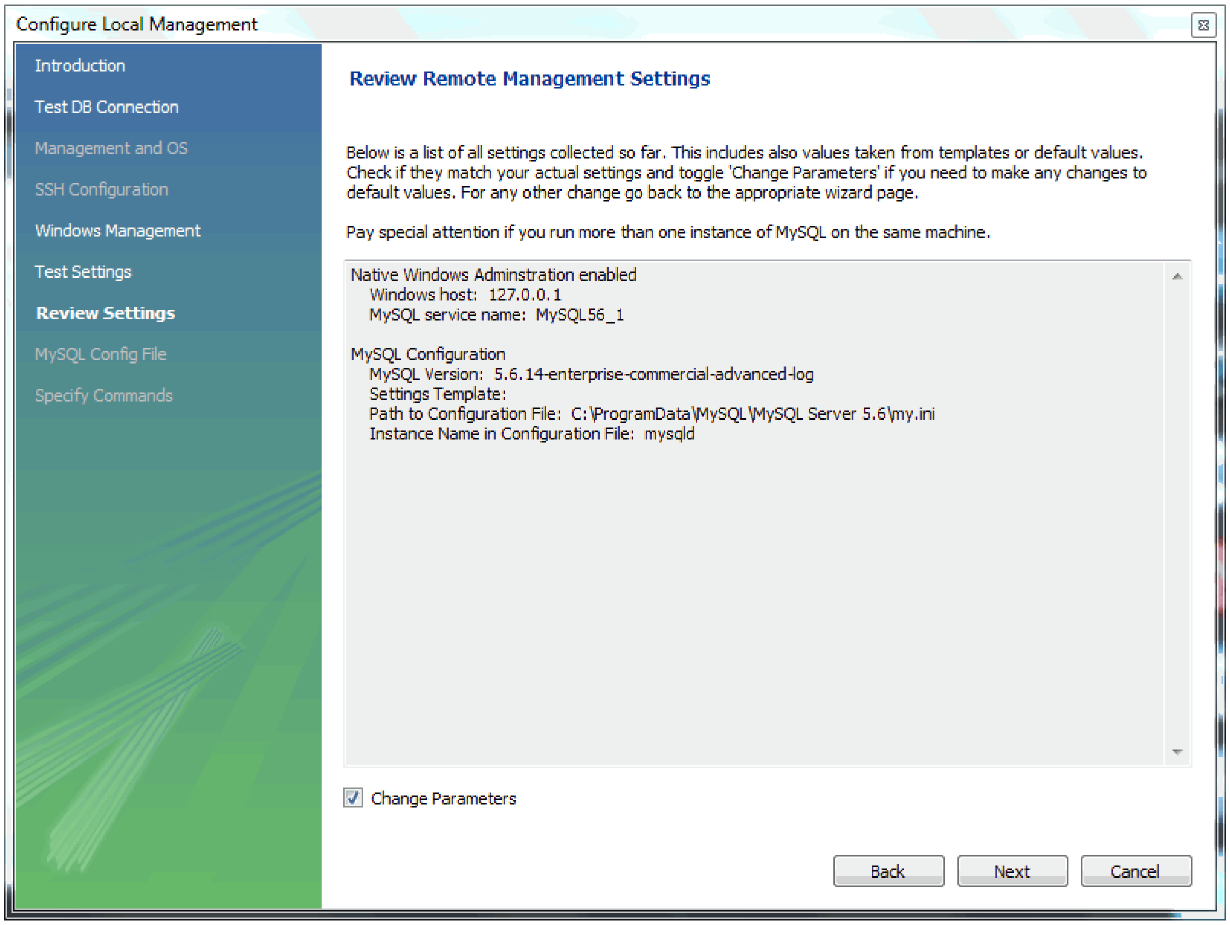
Testing host machine settings is done.

```
Check if C:\ProgramData\MySQL\MySQL Server 5.6\my.ini can be accessed
File was found at the expected location
Check if section for instance mysqld exists in C:\ProgramData\MySQL\MySQL Server 5.6\my.ini
[mysqld] section found in configuration file
```

9. You now have a chance to review the configuration settings. The information displayed varies slightly depending on platform, connection method, and installation type.

   At the **Review Settings** prompt, choose "I'd like to review the settings again" to review the settings. Choosing "Continue" closes the "Configure Server Management" dialog.

Figure 5.8 Getting Started Tutorial - Review Settings



Review Remote Management Settings

Below is a list of all settings collected so far. This includes also values taken from templates or default values. Check if they match your actual settings and toggle 'Change Parameters' if you need to make any changes to default values. For any other change go back to the appropriate wizard page.

Pay special attention if you run more than one instance of MySQL on the same machine.

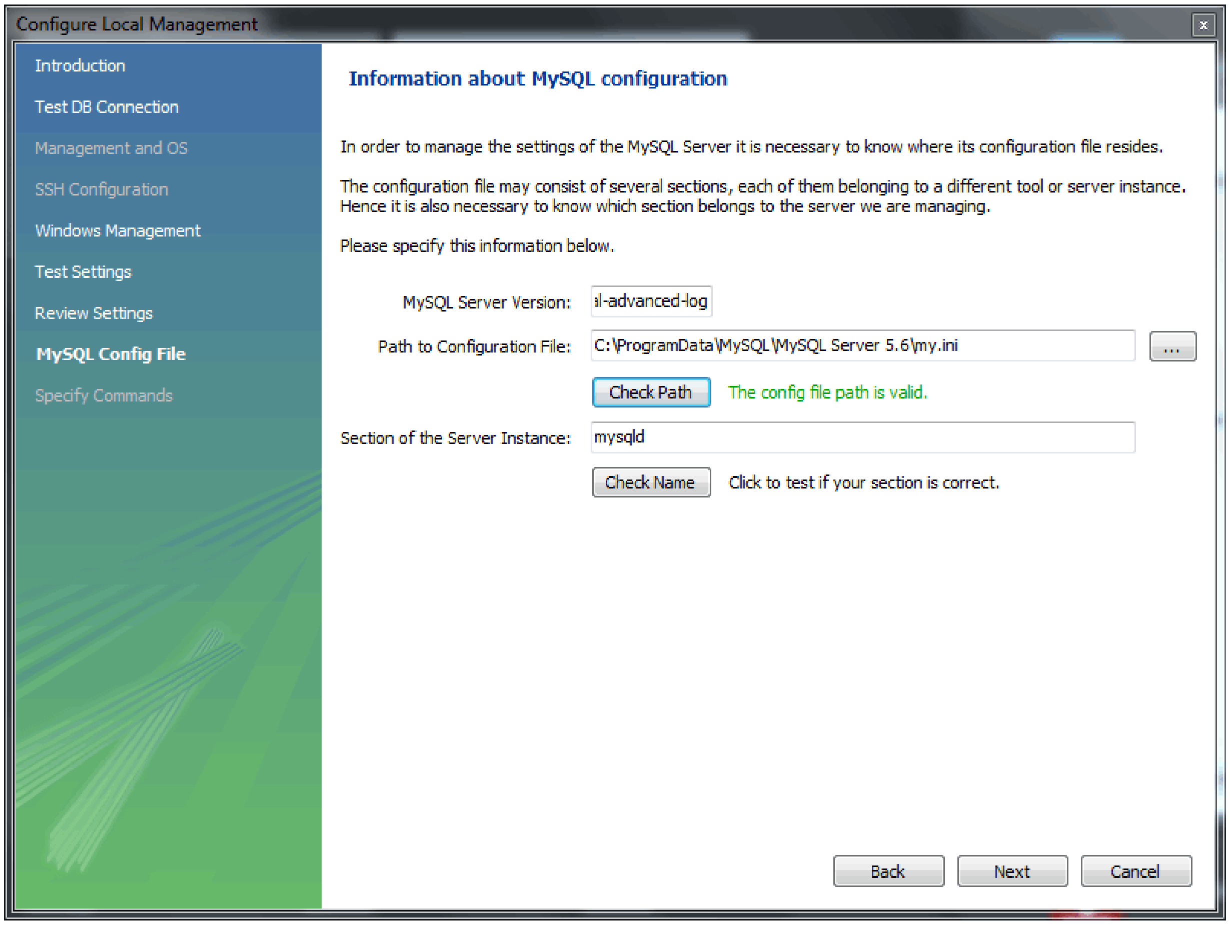
```
Native Windows Adminstration enabled
   Windows host:  127.0.0.1
   MySQL service name:  MySQL56_1

MySQL Configuration
   MySQL Version:  5.6.14-enterprise-commercial-advanced-log
   Settings Template:
   Path to Configuration File:  C:\ProgramData\MySQL\MySQL Server 5.6\my.ini
   Instance Name in Configuration File:  mysqld
```

☑ Change Parameters

Check the **Change Parameters** if you want to check or edit information about the MySQL configuration file. In our example we will check it, and click **Next** to continue.

10. Review the MySQL configuration file information. Click the **Check** buttons to perform the described checks, or optionally change the configuration file path.

Figure 5.9 Getting Started Tutorial - MySQL Config File



Information about MySQL configuration

In order to manage the settings of the MySQL Server it is necessary to know where its configuration file resides.

The configuration file may consist of several sections, each of them belonging to a different tool or server instance. Hence it is also necessary to know which section belongs to the server we are managing.

Please specify this information below.

MySQL Server Version: al-advanced-log

Path to Configuration File: C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

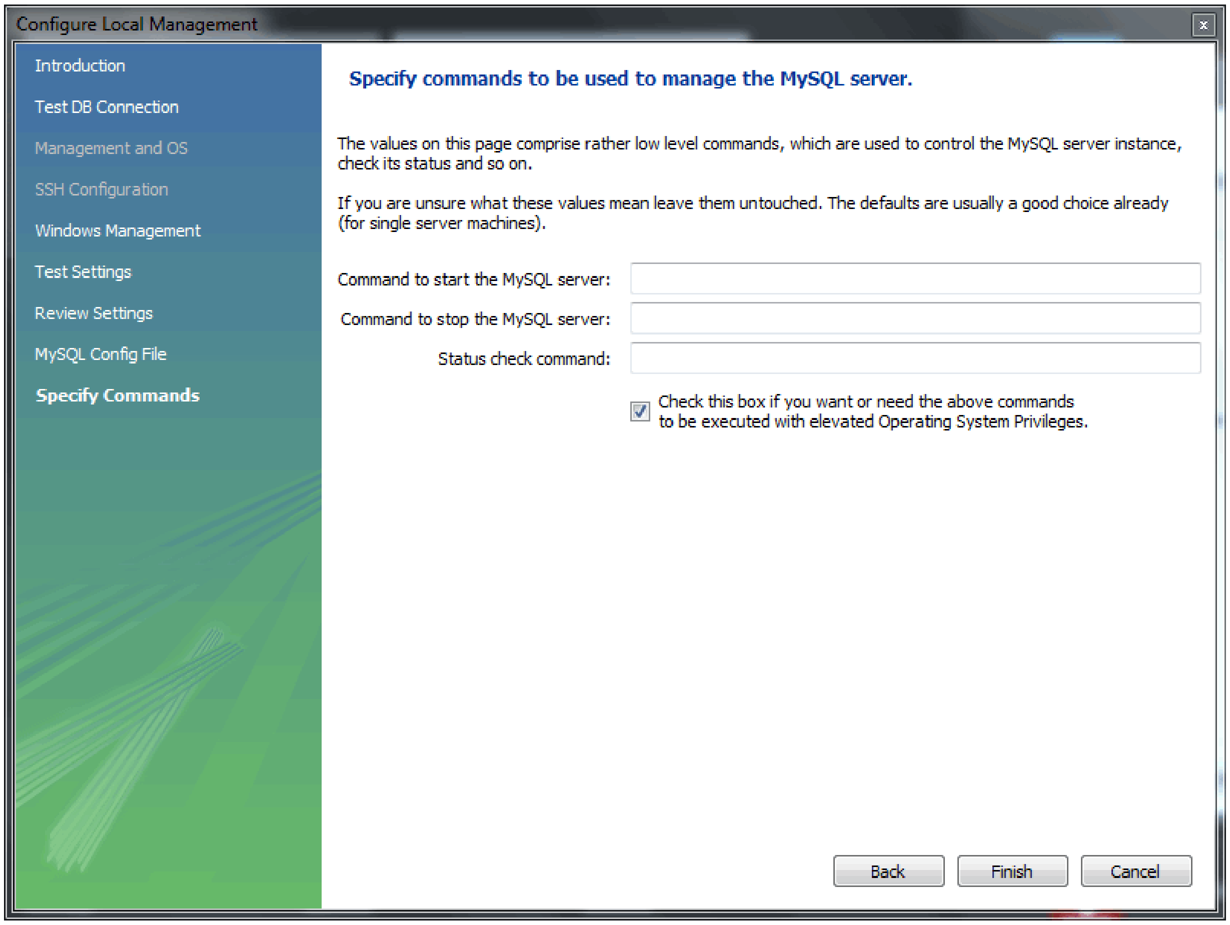
The config file path is valid.

Section of the Server Instance: mysqld

Click to test if your section is correct.

11. Optionally, enter your own commands for starting, stopping, and checking the MySQL connection. Typically the default values are used, which means leaving these optional values blank.

Figure 5.10 Getting Started Tutorial - Specify Commands



Specify commands to be used to manage the MySQL server.

The values on this page comprise rather low level commands, which are used to control the MySQL server instance, check its status and so on.

If you are unsure what these values mean leave them untouched. The defaults are usually a good choice already (for single server machines).

Command to start the MySQL server:

Command to stop the MySQL server:

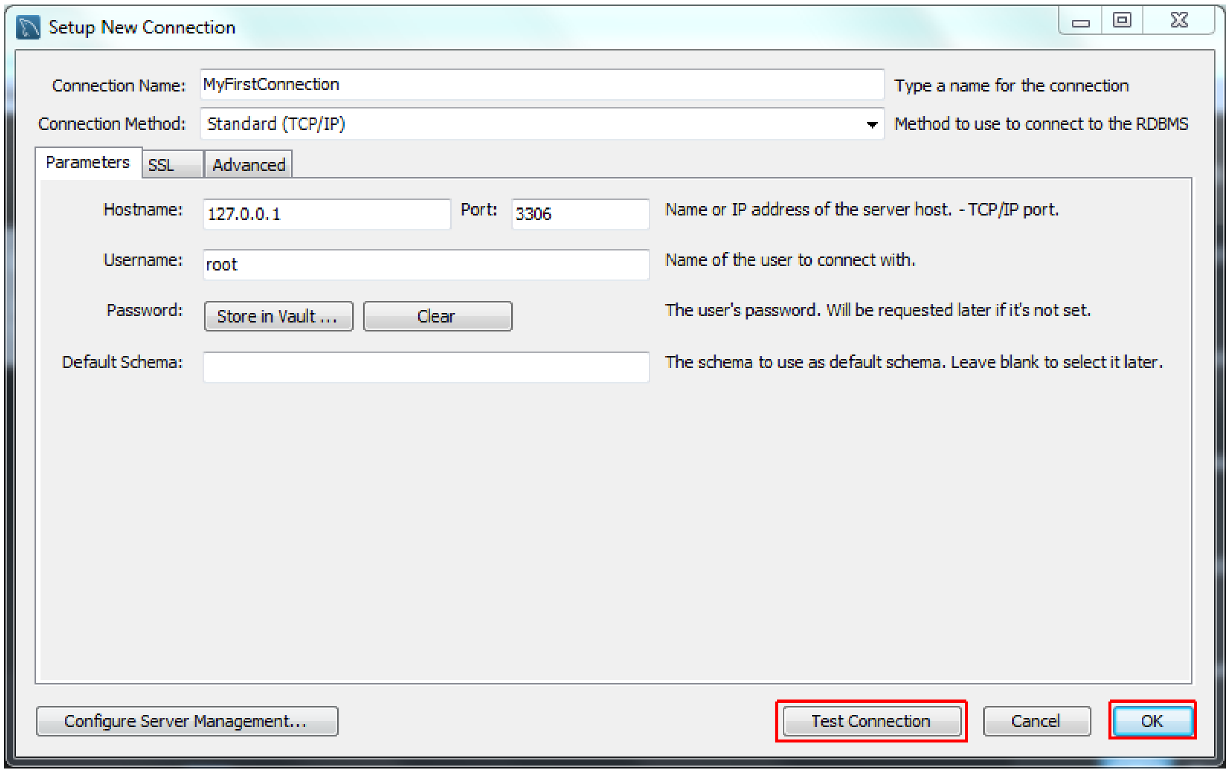
Status check command:

[✓] Check this box if you want or need the above commands to be executed with elevated Operating System Privileges.

Click **Finish** to close the "Configure Server Management" dialog, which reveals the original **Setup New Connection** window.

12. After reviewing the **Setup New Connection** information, press **Test Connection** again to make sure it still functions, and then **OK** to create the new MySQL connection.

Figure 5.11 Getting Started Tutorial - Setup New Connection



Setup New Connection

Connection Name: MyFirstConnection — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host. - TCP/IP port.

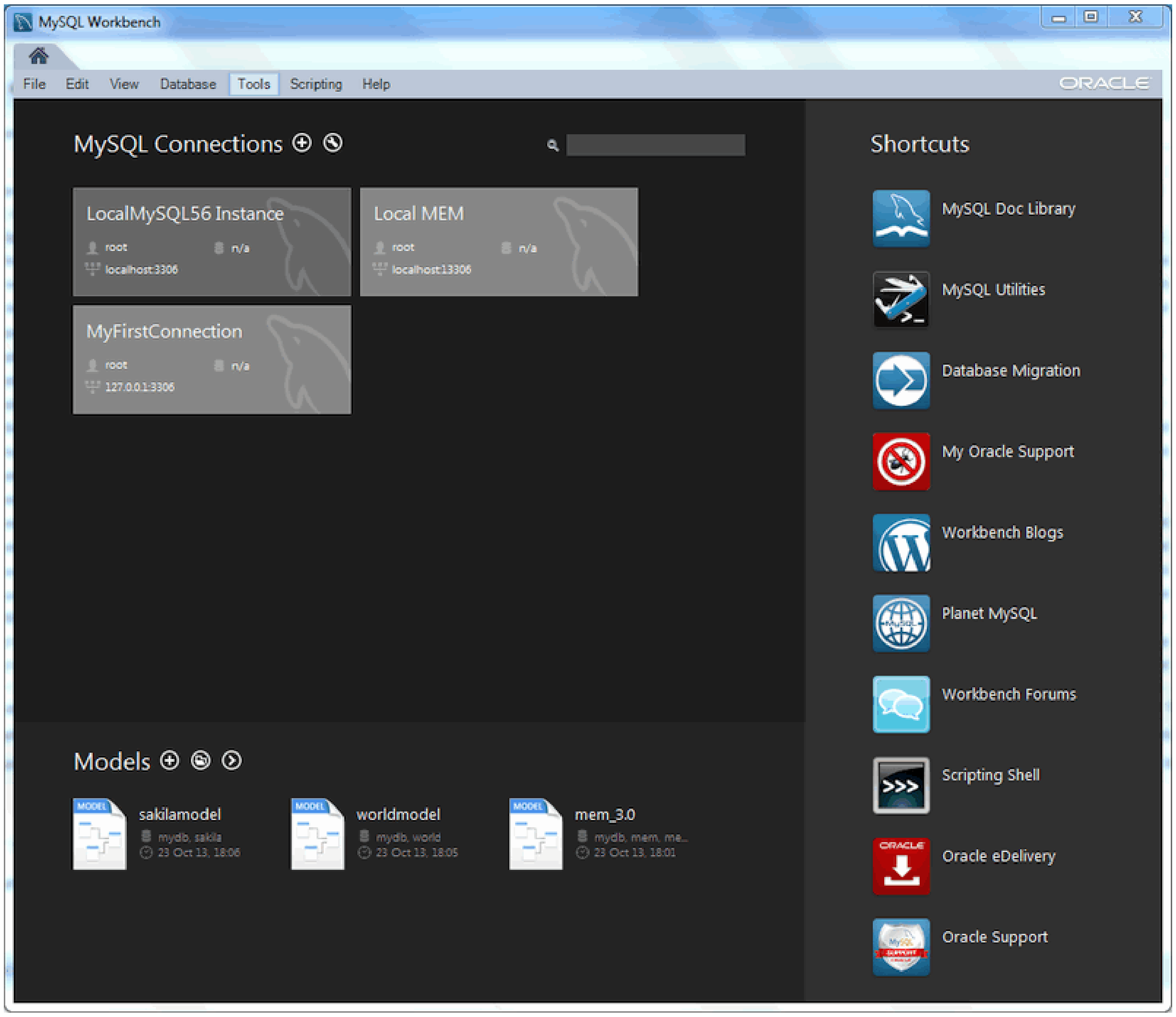
Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

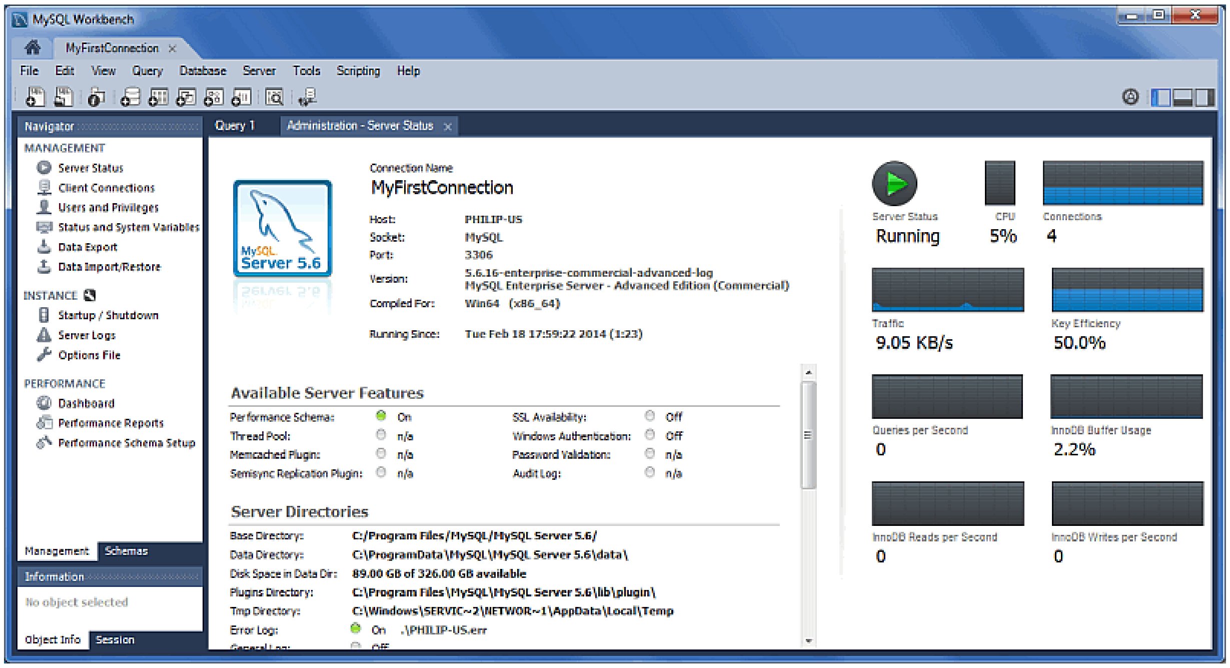
13. Your new **MyFirstConnection** MySQL connection is now listed on the Home screen.

Figure 5.12 Getting Started Tutorial - Home Screen Instance



14. From the Home screen, click the new MySQL connection to open the SQL editor for this connection. The SQL editor is the default page, so now select the **Server Status** from the left **Navigator** panel to display the connected MySQL server's current status.

Figure 5.13 Getting Started Tutorial - Server Status



Connection Name
MyFirstConnection

Host:	PHILIP-US
Socket:	MySQL
Port:	3306
Version:	5.6.16-enterprise-commercial-advanced-log MySQL Enterprise Server - Advanced Edition (Commercial)
Compiled For:	Win64 (x86_64)
Running Since:	Tue Feb 18 17:59:22 2014 (1:23)

Available Server Features

Performance Schema:	● On	SSL Availability:	○ Off
Thread Pool:	○ n/a	Windows Authentication:	○ Off
Memcached Plugin:	○ n/a	Password Validation:	○ n/a
Semisync Replication Plugin:	○ n/a	Audit Log:	○ n/a

Server Directories

Base Directory:	C:/Program Files/MySQL/MySQL Server 5.6/
Data Directory:	C:\ProgramData\MySQL\MySQL Server 5.6\data\
Disk Space in Data Dir:	89.00 GB of 326.00 GB available
Plugins Directory:	C:\Program Files\MySQL\MySQL Server 5.6\lib\plugin\
Tmp Directory:	C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp
Error Log:	● On .\PHILIP-US.err
General Log:	○ Off

Server Status	Running
CPU	5%
Connections	4
Traffic	9.05 KB/s
Key Efficiency	50.0%
Queries per Second	0
InnoDB Buffer Usage	2.2%
InnoDB Reads per Second	0
InnoDB Writes per Second	0

15. Test the other **Navigator** panel options that relate to your new MySQL connection. Check its status, MySQL logs, and measure its performance statistics from the **Dashboard**.

    Notice the **Management** and **Schemas** tabs on the bottom of the **Navigator** panel. The **Schemas** view displays the schemas that are associated with your new MySQL connection. Alternatively, you can merge the Schemas and Management tabs by either clicking the merge icon on the top right of the Navigator panel, or by enabling the **Show Management Tools and Schema Tree in a single tab** SQL Editor preference.

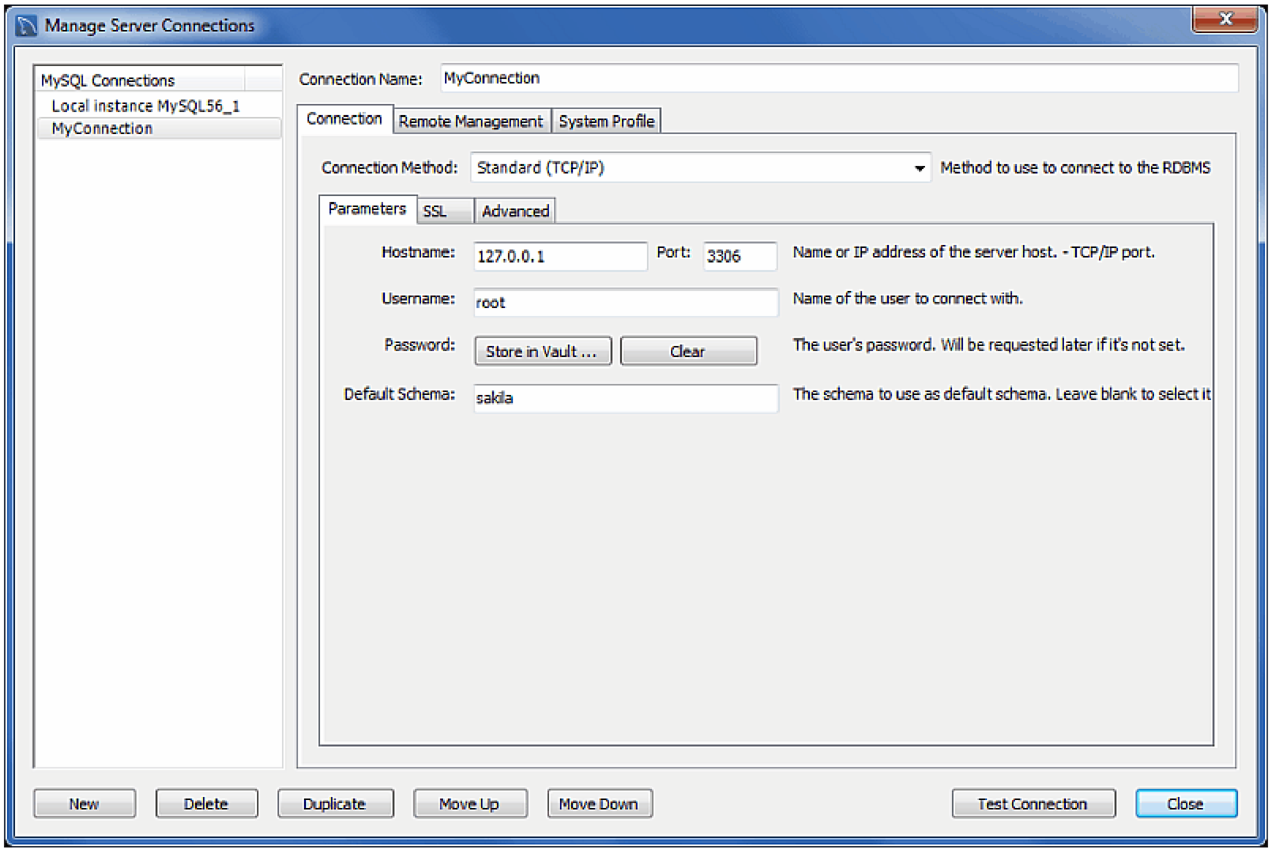
This concludes the "Creating a MySQL connection" tutorial. For additional information about MySQL connections, see Chapter 5, *MySQL Connections*.

5.3 Manage Server Connections

The **Manage Server Connections** dialog is another way to manage MySQL connections. This dialog is invoked by either selecting **Edit Connection** or selecting **Database**, **Manage Connections** from the main menu. It can also be invoked from any of the wizards requiring access to a live database.

After the MySQL connection manager is launched, you are presented with the following dialog, with the **Connection** tab open:

Figure 5.14 Manage Server Connections: Connection Tab



Connection Name: MyConnection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host. - TCP/IP port.

Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: sakila — The schema to use as default schema. Leave blank to select it

- **Connection Name**: The name used to refer to this connection. This connection can then be selected from a list in other wizards requiring a connection.
- **Connection Method**: Method used to connect to the RDBMS.

  After you select a connection method, the fields available in the **Parameters**, **SSL**, and **Advanced** tabs change accordingly. More details about these options and parameters are available below.

Note

The **Test Connection** button will test the selected MySQL connection and report its connection status. It also reports whether or not SSL is enabled.

For testing remote connections, you might also use `ping` to check the hostname, or `telnet` to also check the port. If these fail, then also check the firewall settings on each host, and also that MySQL server is running on the remote host.

Note

Simultaneous client connections: Opening a MySQL connection from the MySQL Workbench home page opens a new connection tab in MySQL Workbench for that connection. Each of these tabs requires two MySQL connections to perform basic tasks, such schema discovery and SQL execution. Additionally, performing management related tasks, such as **Server Status**, requires two additional MySQL connections. Essentially, this means that each MySQL connection tab in MySQL Workbench requires four available connections to MySQL. For additional information about "Too many connection" related errors, see Too many connections.

This connection requirement doubles with each connection tab opened in MySQL Workbench, even if the two connection tabs point to the same MySQL server. SQL editor tabs share their connections, so having multiple SQL editor and SQL results tabs does not affect the number of required connections.

5.3.1 Standard TCP/IP Connection Method

This connection method enables MySQL Workbench to connect to MySQL Server using TCP/IP.

Note

The `--skip-networking` MySQL server configuration option affects the TCP/IP connection method. If disabled, use named pipes or shared memory (on Windows) or Unix socket files (on Unix).

Parameters tab

- **Hostname**: The host name or IP address of the MySQL server.

  Note

  The host name "localhost" might resolve to "127.0.0.1" or "::1" on your host, so note this when checking permissions. For example, if a web application's user only has access to "127.0.0.1" on a host, and a defined connection uses "localhost" that resolves to "::1", this connection may lack the proper permissions to the aforementioned web application. Ping "localhost" on each host to determine where it resolves to.

- **Port**: The TCP/IP port on which the MySQL server is listening (the default is 3306).

- **Username**: User name to use for the connection.

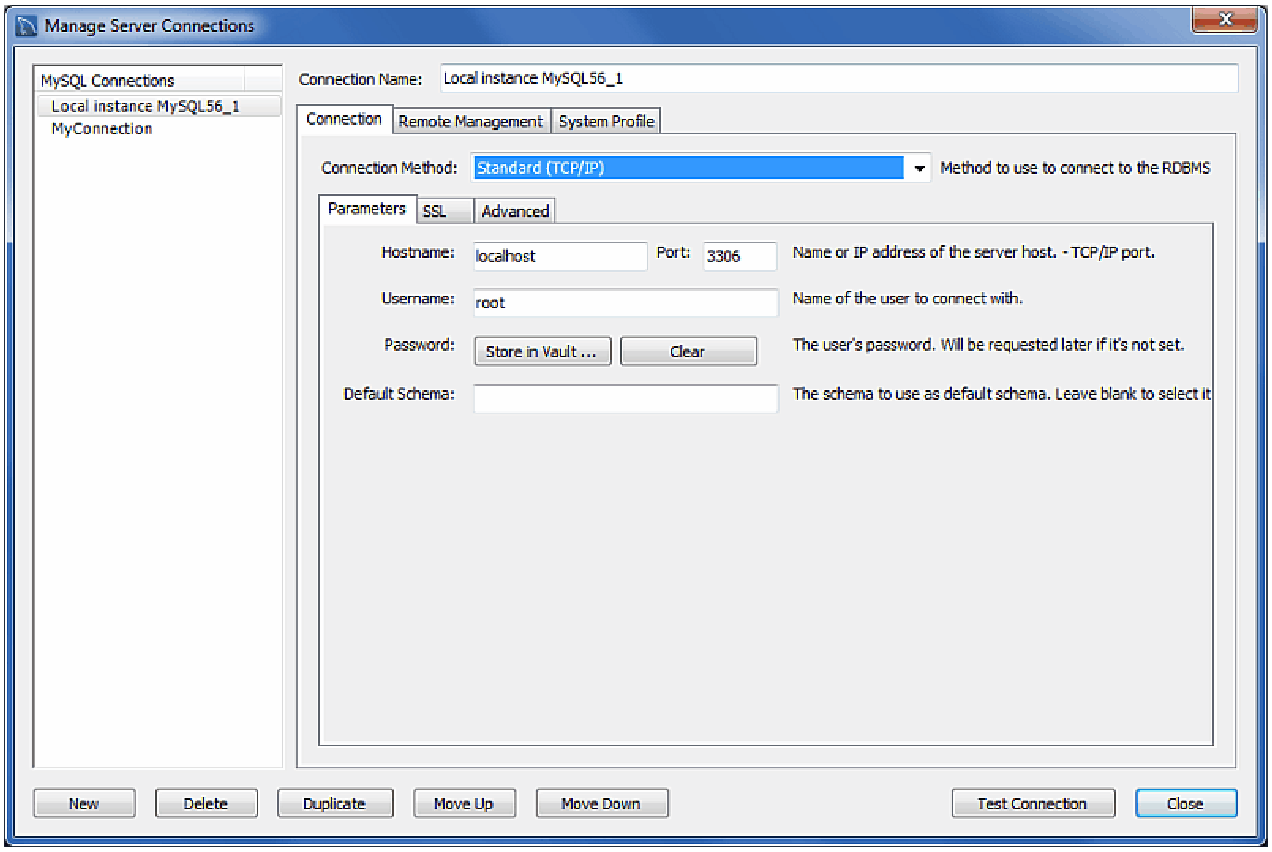
- **Password**: Optional password for the account used. If you enter no password here, you will be prompted to enter the password when MySQL Workbench attempts to establish the connection. MySQL Workbench can store the password in a vault (see Section 5.3.7, “The Password Storage Vault”).

- **Default Schema**: When the connection to the server is established, this is the schema that will be used by default. It becomes the default schema for use in other parts of MySQL Workbench.

Advanced tab

Advanced parameters are less common and include:

Figure 5.15 Standard (TCP/IP) Connection: Advanced Tab


Manage Server Connections
MySQL Connections
Local instance MySQL56_1
MyConnection
Connection Name: Local instance MySQL56_1
Connection
Remote Management
System Profile
Connection Method: Standard (TCP/IP)
Method to use to connect to the RDBMS
Parameters
SSL
Advanced
Hostname: localhost
Port: 3306
Name or IP address of the server host. - TCP/IP port.
Username: root
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's not set.
Default Schema:
The schema to use as default schema. Leave blank to select it
New
Delete
Duplicate
Move Up
Move Down
Test Connection
Close

The **Advanced** tab includes these check boxes:

- **Use compression protocol**: If checked, the communication between the application and the MySQL server will be compressed, which may increase transfer rates. This corresponds to starting a MySQL command-line client with the `--compress` option. This option is unchecked by default.

- **Use ANSI quotes to quote identifiers**: Treat “"” as an identifier quote character (like the “`” quote character) and not as a string quote character. You can still use “`” to quote identifiers with this mode enabled. With this option enabled, you cannot use double quotation marks to quote literal strings, because it is interpreted as an identifier. Note: If this option is checked, it overrides the server setting. This option is unchecked by default.

- **Enable Cleartext Authentication Plugin**: Send the user password in cleartext. Required for some authentication methods. This option is unchecked by default.

- **Use the old authentication protocol**: This option disables Connector/C++'s secure_auth option. This option is unchecked by default. Doing so means you can connect to MySQL Server with MySQL users that utilize the old `mysql_old_password` authentication plugin, which is not recommended. `mysql_old_password` support was removed in MySQL Server 5.7.

Note

This option was removed in Workbench 6.3.6, and did not function in Workbench 6.3.5. For information about upgrading passwords from the old authentication protocol, see Section 5.3.8, “Updating Old Authentication Protocol Passwords”.

> Also, MySQL Server 5.7 does not support the old authentication protocol, because secure_auth can not be disabled. Using the old `mysql_old_password` has not been recommended since MySQL 4.1.

It also includes these options:

SQL_MODE: Override the default `SQL_MODE` used by the server.

Others: Other options for Connector/C++ as option=value pairs, one per line.

SSL tab

SSL parameters include:

- **Use SSL**: This dropdown provides options related to enabling SSL encryption. Choose **No** to disable SSL, **If available** if the client library supports it, or **Require** to require SSL support for the MySQL connection to succeed. This option defaults to **If available**.
- **SSL CA File**: Path to the Certification Authority file for SSL.
- **SSL CERT File**: Path the Certificate file for SSL.
- **SSL Key File**: Path to the Key file for SSL.
- **SSL Cipher**: Optional list of permissible ciphers to use for SSL encryption.
- **SSL Wizard**: Generate SSL certificates for both the MySQL server and MySQL client. Requires access to OpenSSL binaries in the system's PATH. For additional information, see Section 5.3.4, “SSL Wizard (Certificates)”.
- **Files**: Opens a system's file browser that points to the generated SSL files by the SSL Wizard. For additional information, see Section 5.3.4, “SSL Wizard (Certificates)”

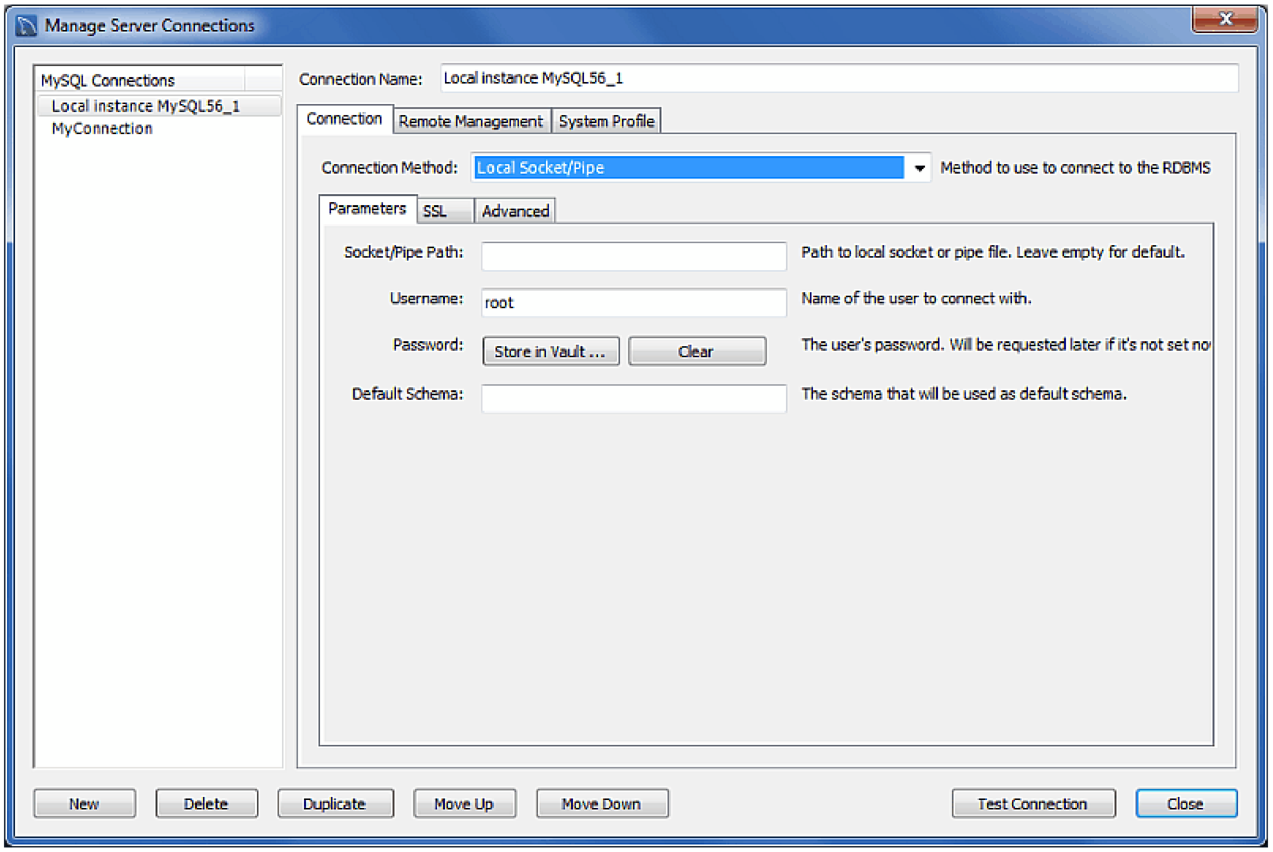
5.3.2 Local Socket/Pipe Connection Method

This connection method enables MySQL Workbench to connect to MySQL Server using a socket file (on Unix) or a named pipe (on Windows).

Parameters

The unique field here is **Socket/Pipe Path**. Enter the name of the socket or pipe here. If the field is left blank, the default socket or pipe name is used. On Unix, the default socket name is `/tmp/mysql.sock`. On Microsoft Windows, the default pipe name is `MySQL`.

Figure 5.16 Manage DB Connections - Socket/Pipe Parameters


Manage Server Connections
MySQL Connections
Local instance MySQL56_1
MyConnection
Connection Name: Local instance MySQL56_1
Connection
Remote Management
System Profile
Connection Method: Local Socket/Pipe
Method to use to connect to the RDBMS
Parameters
SSL
Advanced
Socket/Pipe Path:
Path to local socket or pipe file. Leave empty for default.
Username: root
Name of the user to connect with.
Password:
Store in Vault ...
Clear
The user's password. Will be requested later if it's not set no
Default Schema:
The schema that will be used as default schema.
New
Delete
Duplicate
Move Up
Move Down
Test Connection
Close

Advanced

These are the same options discussed in Section 5.3.1, “Standard TCP/IP Connection Method”, except there is not the **Use compression protocol** option.

SSL

These are the same options discussed in Section 5.3.1, “Standard TCP/IP Connection Method”.

5.3.3 Standard TCP/IP over SSH Connection Method

This connection method enables MySQL Workbench to connect to MySQL Server using TCP/IP over an SSH connection.

Parameters

In addition to a number of parameters that are in common with Standard TCP/IP connections, this connection method features a number of specialized parameters. These are listed here:

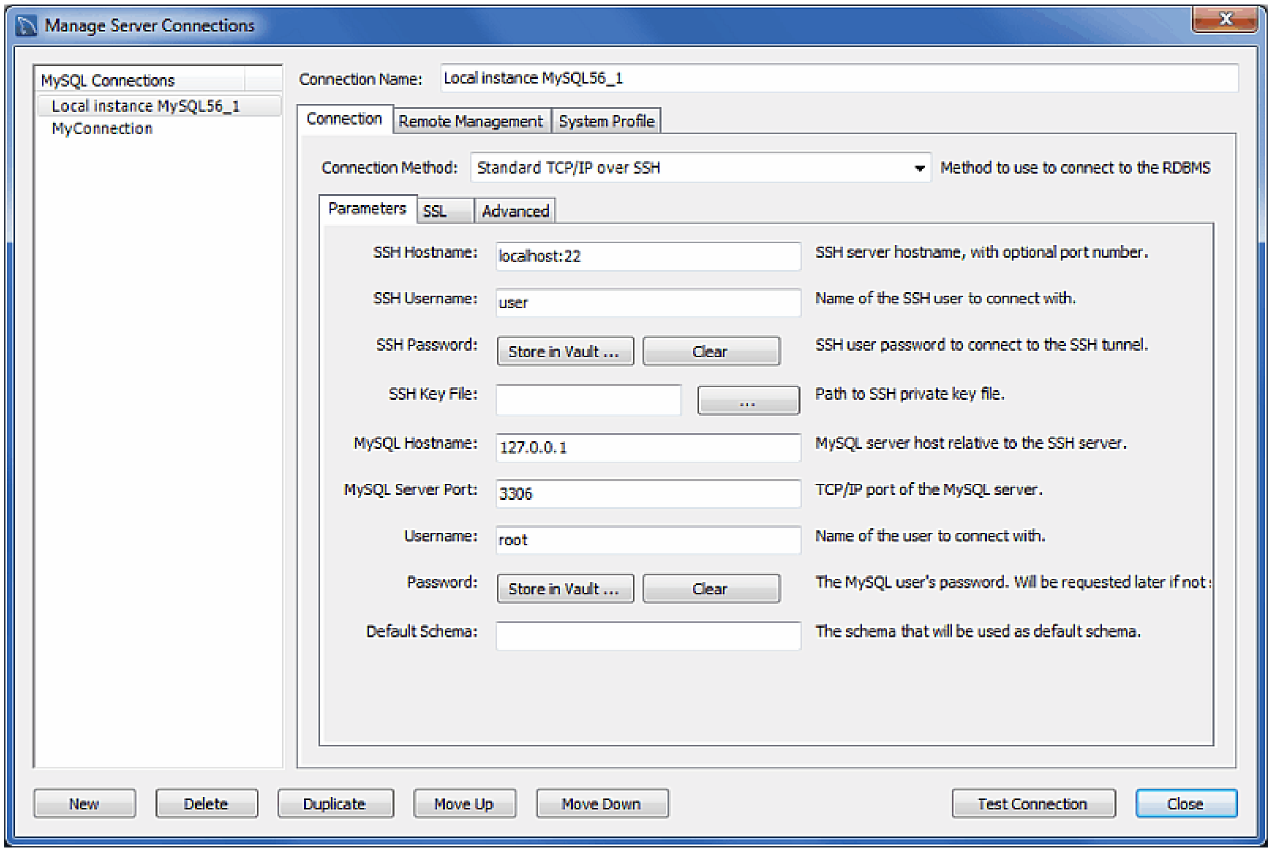
- **SSH Hostname**: This is the name of the SSH server. An optional port number can also be provided. For example, localhost:22.
- **SSH Username**: This is the name of the SSH user name to connect with.
- **SSH Password**: The SSH password. It is recommended that an SSH key file is also used.
- **SSH Key File**: A path to the SSH key file.

If a remote host is missing from the system's list of known hosts, a prompt requires you to confirm the host's fingerprint before storing it. If your stored host fingerprint is different than the host's current fingerprint, then an error is generated and you will be required to handle the discrepancy from outside of MySQL Workbench before creating the connection. Prior to MySQL Workbench 6.1.6, the host SSH fingerprint was not saved by MySQL Workbench.

On Linux and OS X, SSH host fingerprints are stored in `~/.ssh/known_hosts`. On Microsoft Windows, they are stored in a file created by MySQL Workbench under the user's application data folder (`%appdata%`), such as `C:\Users\[username]\AppData\Roaming\MySQL\Workbench\known_hosts`.

The SSH connection options are viewable in the following screenshot:

Figure 5.17 Manage DB Connections - SSH Parameters



Connection Name: Local instance MySQL56_1

Connection Method: Standard TCP/IP over SSH — Method to use to connect to the RDBMS

SSH Hostname:	localhost:22	SSH server hostname, with optional port number.
SSH Username:	user	Name of the SSH user to connect with.
SSH Password:		SSH user password to connect to the SSH tunnel.
SSH Key File:		Path to SSH private key file.
MySQL Hostname:	127.0.0.1	MySQL server host relative to the SSH server.
MySQL Server Port:	3306	TCP/IP port of the MySQL server.
Username:	root	Name of the user to connect with.
Password:		The MySQL user's password. Will be requested later if not
Default Schema:		The schema that will be used as default schema.

Advanced

The options here are the same as for the Standard TCP/IP connection. See Section 5.3.1, "Standard TCP/IP Connection Method".

SSL

The options here are the same as for the Standard TCP/IP connection. See Section 5.3.1, "Standard TCP/IP Connection Method".

5.3.4 SSL Wizard (Certificates)

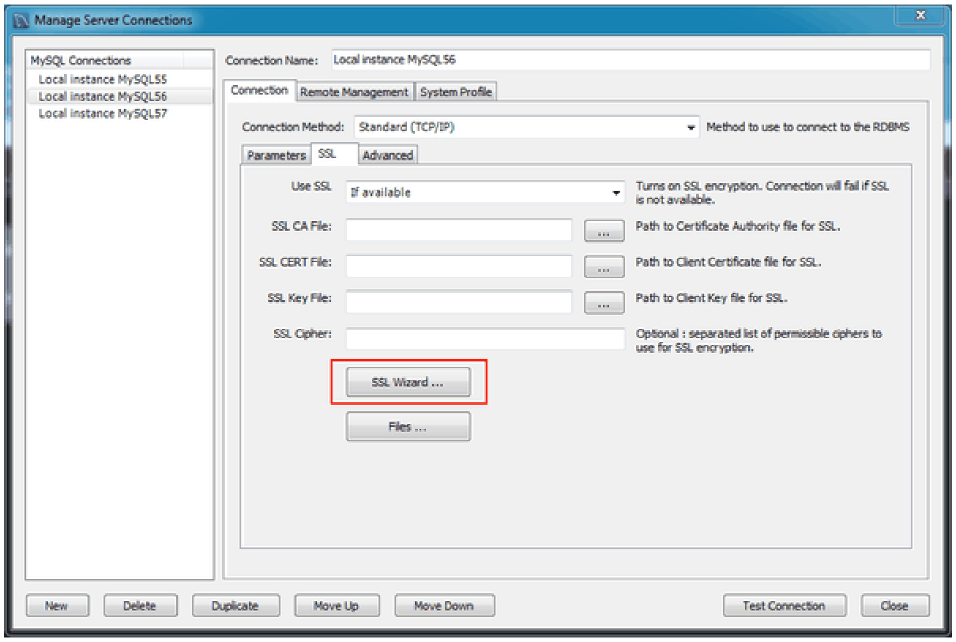
This wizard helps create SSL certificates for both MySQL clients and MySQL servers. Connections in MySQL Workbench are updated with the certificates by the wizard. This wizard requires OpenSSL to create the certificates. An example MySQL configuration file (`my.cnf` / `my.ini`) is also generated that utilizes the generated certificates.

Note

The OpenSSL binary should be in the system's PATH.

Start the SSL wizard from the **SSL** tab of a MySQL connection. Locate this tab in the MySQL connection editor. Click **SSL Wizard** to execute the wizard:

Figure 5.18 SSL Wizard: Start



Connection Name: Local instance MySQL56

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Use SSL: If available — Turns on SSL encryption. Connection will fail if SSL is not available.

SSL CA File: — Path to Certificate Authority file for SSL.

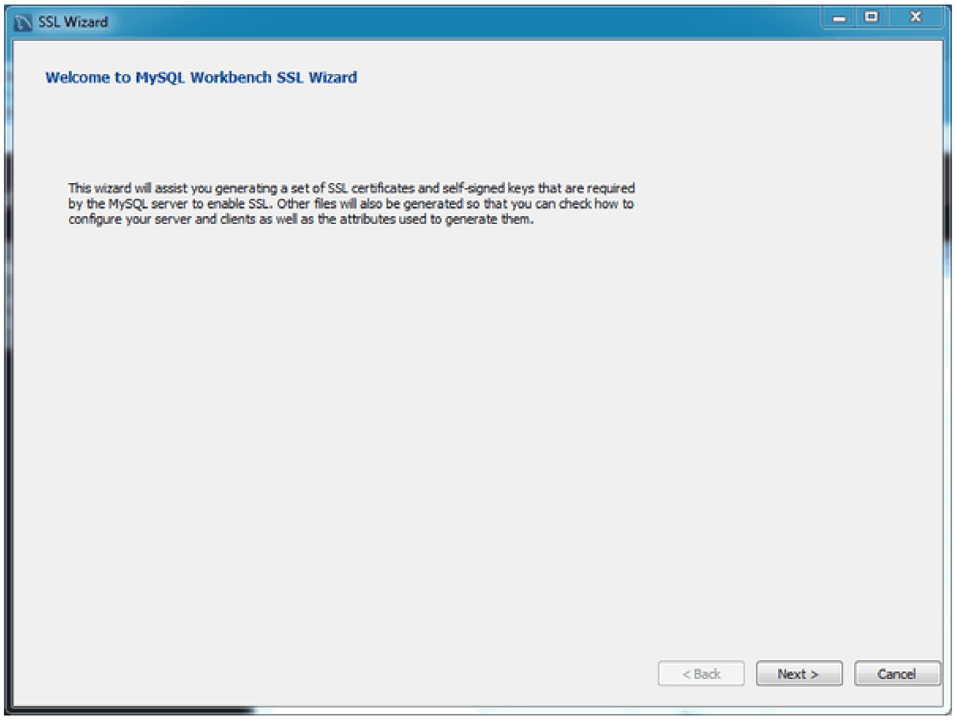
SSL CERT File: — Path to Client Certificate file for SSL.

SSL Key File: — Path to Client Key file for SSL.

SSL Cipher: — Optional : separated list of permissible ciphers to use for SSL encryption.

Read the informative text on the welcome screen:

Figure 5.19 SSL Wizard: Welcome



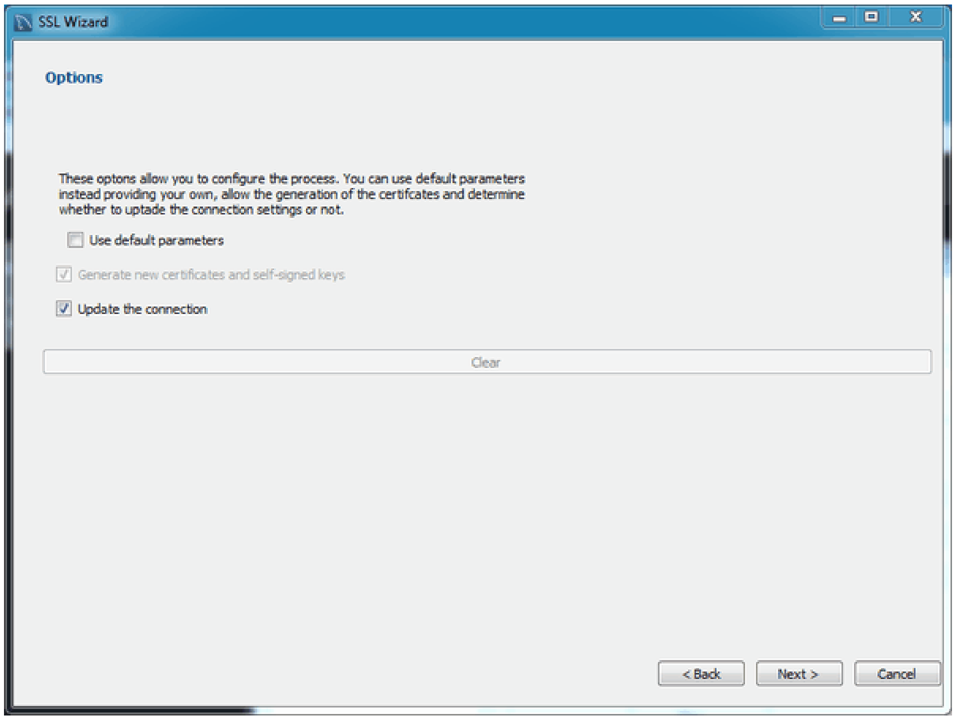
Welcome to MySQL Workbench SSL Wizard

This wizard will assist you generating a set of SSL certificates and self-signed keys that are required by the MySQL server to enable SSL. Other files will also be generated so that you can check how to configure your server and clients as well as the attributes used to generate them.

Check the options that apply:

- [ ] *Use default parameters*: Check this to skip entering the optional attributes, such as Country, State, Organization, and so on. By default, these fields are empty.
- [ ] *Generate new certificates and self-signed keys*: Check this to generate new files, otherwise the existing files are used. You might disable this if you already generated SSL certificates but forgot where the files are located, or how to configure them.
- [ ] *Update the connection*: Updates the defined MySQL connection (in Workbench) with the generated certificate information.

Figure 5.20 SSL Wizard: Options



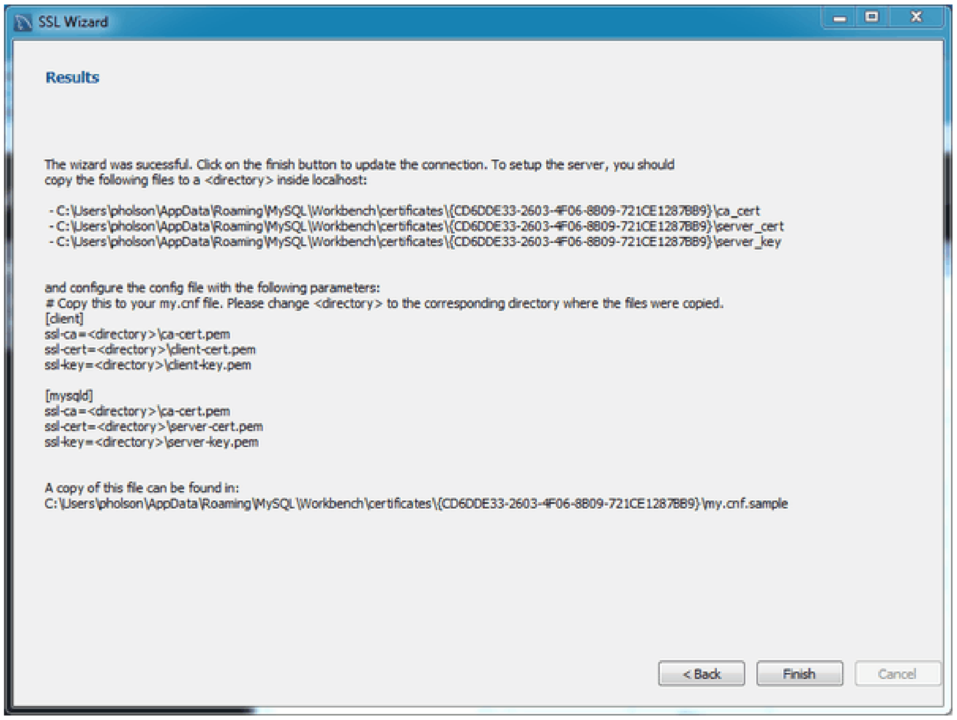
These optons allow you to configure the process. You can use default parameters instead providing your own, allow the generation of the certifcates and determine whether to uptade the connection settings or not.

- [ ] Use default parameters
- [✓] Generate new certificates and self-signed keys
- [✓] Update the connection

The results page describes the generated files, and provides requirements that you must perform to complete the operation. For example, you must manually edit your MySQL configuration file (`my.ini` or `my.cnf`) and define the SSL options.

Consider leaving this screen open, and close it after you copied the files and altered your MySQL configuration file to enable SSL connections. The wizard does not perform these actions for you.

Figure 5.21 SSL Wizard: Results



The wizard was sucessful. Click on the finish button to update the connection. To setup the server, you should copy the following files to a <directory> inside localhost:

```
 - C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\ca_cert
 - C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\server_cert
 - C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\server_key
```

and configure the config file with the following parameters:

```
# Copy this to your my.cnf file. Please change <directory> to the corresponding directory where the files were copied.
[client]
ssl-ca=<directory>\ca-cert.pem
ssl-cert=<directory>\client-cert.pem
ssl-key=<directory>\client-key.pem

[mysqld]
ssl-ca=<directory>\ca-cert.pem
ssl-cert=<directory>\server-cert.pem
ssl-key=<directory>\server-key.pem
```

A copy of this file can be found in:
C:\Users\pholson\AppData\Roaming\MySQL\Workbench\certificates\{CD6DDE33-2603-4F06-8809-721CE1287B89}\my.cnf.sample

Here an example process of using the generated SSL files to set up an SSL connection. Adjust your paths as they will be different.

1. Create a directory to store the certificate files. In our simple example, we have MySQL Workbench installed on the same host as the MySQL Server, and we created "`C:\certs`" on this system.

2. Copy and paste the results to a new (temporary) file, but change <directory> to the path (`C:\certs`) we created. For example:

   ```
   [client]
   ssl-ca=C:\certs\ca-cert.pem
   ssl-cert=C:\certs\client-cert.pem
   ssl-key=C:\certs\client-key.pem

   [mysqld]
   ssl-ca=C:\certs\ca-cert.pem
   ssl-cert=C:\certs\\server-cert.pem
   ssl-key=C:\certs\\server-key.pem
   ```

   Warning

   MySQL Server interprets "\s" as a space, so we added an extra backslash to escape it. That is why you see "\\server-key.pem" in the above example, because MySQL Server would interpret "\server-key.pem" as " erver-key.pem".

3. Open the MySQL Server configuration file. In this example, its location is "`C:\ProgramData\MySQL\MySQL Server 5.7\my.ini`".

Note

The location of your configuration file depends on how MySQL Server was installed. The connection editor defines and displays its location, as does the **Options File** page in MySQL Workbench.

4. Add the client certificate information under the `[client]` section:

   ```
   [client]
   ssl-ca=C:\certs\ca-cert.pem
   ssl-cert=C:\certs\client-cert.pem
   ssl-key=C:\certs\client-key.pem
   ```

   Add the server certificate information under the `[mysqld]` section:

   ```
   [mysqld]
   ssl-ca=C:\certs\ca-cert.pem
   ssl-cert=C:\certs\\server-cert.pem
   ssl-key=C:\certs\\server-key.pem
   ```

5. Update the paths to the SSL client certificates in your MySQL connection, under the SSL tab. There are three paths to update.

6. Restart the MySQL Server. In the log, you should see something like "Warning CA certificate `C:\certs\ca-cert.pem` is self signed."

7. In MySQL Workbench's MySQL connection editor, clicking **Test Connection** should confirm your SSL connection.

Additionally, consider setting **Use SSL** to "Required". Or, if you are experiencing problems, set it to "If available" while debugging the problem.

5.3.5 System Profile

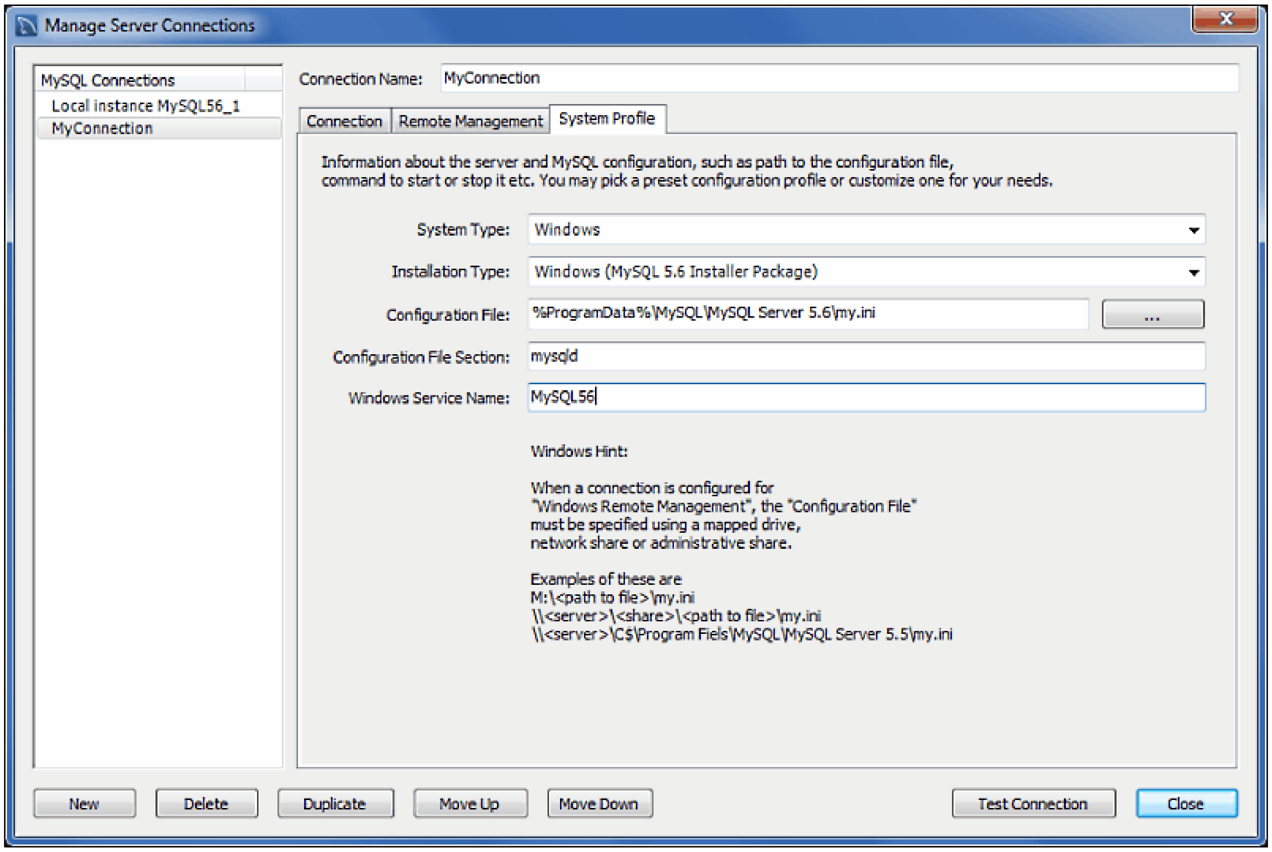
The **System Profile** tab enables you to specify host-specific information. This is achieved primarily through selecting a **System Type**, along with its corresponding **Installation Type**. These profile settings contain standard information that is used in managing the host's MySQL instance.

Here are some of the available installation types:

- FreeBSD, MySQL package or Custom
- Linux, including distributions such as Fedora, Oracle, RHEL, SLES, Ubuntu, Generic, and Custom
- OS X, MySQL package or Custom
- OpenSolaris, MySQL package or Custom
- Windows, with different installation methods, MySQL versions, and build architectures

Choose the appropriate **System Type** and **Installation Type** to set default parameters that includes commands used to start and stop MySQL, commands to check the server status, the location of the `my.ini` or `my.cnf` configuration file, and on Windows, the Windows Service Name. These default values are customizable.

Figure 5.22 Manage Server Connections: System Profile Tab



Connection Name: MyConnection

Information about the server and MySQL configuration, such as path to the configuration file, command to start or stop it etc. You may pick a preset configuration profile or customize one for your needs.

System Type: Windows

Installation Type: Windows (MySQL 5.6 Installer Package)

Configuration File: %ProgramData%\MySQL\MySQL Server 5.6\my.ini

Configuration File Section: mysqld

Windows Service Name: MySQL56

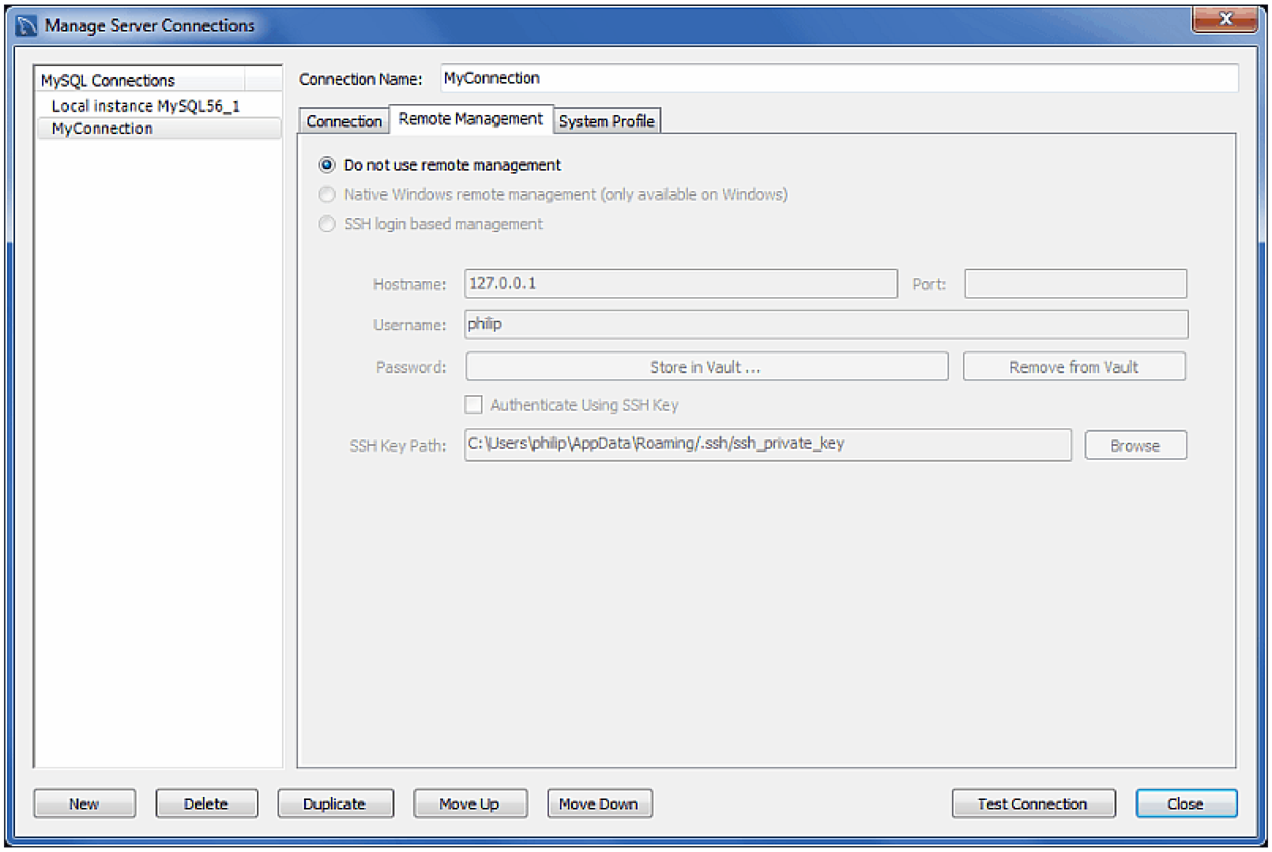
Windows Hint:

When a connection is configured for "Windows Remote Management", the "Configuration File" must be specified using a mapped drive, network share or administrative share.

Examples of these are
M:\<path to file>\my.ini
\\<server>\<share>\<path to file>\my.ini
\\<server>\C$\Program Fiels\MySQL\MySQL Server 5.5\my.ini

The **Remote Management** tab is available when connecting to MySQL remotely.

Figure 5.23 Manage Server Connections: Remote Management Tab



Connection Name: MyConnection

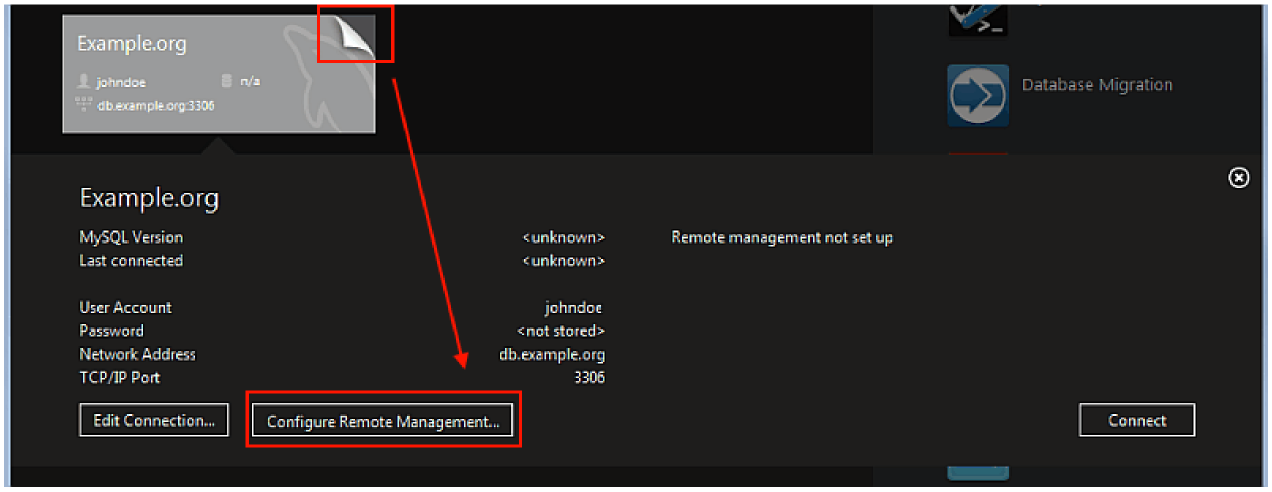
(●) Do not use remote management
( ) Native Windows remote management (only available on Windows)
( ) SSH login based management

Hostname: 127.0.0.1 Port:
Username: philip
Password:
[ ] Authenticate Using SSH Key
SSH Key Path: C:\Users\philip\AppData\Roaming/.ssh/ssh_private_key

5.3.6 Configure Server Management Wizard

Clicking the [+] icon from the Home page launches the **Setup New Connection** wizard. The wizard provides a MySQL connection form to create a new MySQL connection, and includes a **Configure Server Management** option as a step-by-step approach to creating a new MySQL server connection. This can also be executed later (on remote connections) when from the home page by clicking the top right corner of a MySQL connection tile:

Figure 5.24 Configure Remote Management



Example.org

MySQL Version <unknown>
Last connected <unknown>

User Account johndoe
Password <not stored>
Network Address db.example.org
TCP/IP Port 3306

Remote management not set up

Executing this wizard is required to perform tasks requiring shell access to the host. For example, starting/stopping the MySQL instance and editing the configuration file.

For a tutorial that demonstrates the steps outlined below, see the tutorial titled Section 5.2, “Creating A New MySQL Connection (Tutorial)”.

The steps presented in the wizard are as follows:

1. **Test DB Connection**

   On this page, MySQL Workbench tests your database connection and displays the results. If an error occurs, click **Show Logs** to view the related logs.

2. **Management and OS**

   Used to specify a remote management type and target operating system, which is available when the Host Machine is defined as a remote host.

   The SSH login based management option includes configuration entries for the Operating System and MySQL Installation Type.

3. **SSH Configuration**

   If you specified a Remote Host on the Specify Host Machine page, you will be presented with the Host SSH Connection page, that enables you to use SSH for the connection to the server instance. This facility enables you to create a secure connection to remotely administer and configure the server instance. You must enter the host name and user name of the account that will be used to log in to the server for administration and configuration activities. If you do not enter the optional SSH Key for use with the server, then you will be prompted for the password when the connection is established by MySQL Workbench.

   Note

   This connection is to enable remote administration and configuration of the MySQL Server itself. It is not the same as the connection used to connect to a server for general database manipulation.

   Note

   You must use an SSH connection type when managing a remote server if you wish to start or stop the server or edit its configuration file. Other administrative functions do not require an SSH connection.

4. **Windows Management**

   If a Windows server is used, then setting the Windows configuration parameters is mandatory. Windows management requires a user account with the required privileges to query the system status, and to control services. And read/write access to the configuration file is needed to allow editing of the file.

5. **Test Settings**

   The wizard now attempts a connection to your server and reports the results. If an error occurs, click **Show Logs** to view the related logs.

   MySQL Workbench must know where the MySQL Server configuration file is located to be able to display configuration information. The wizard is able to determine the most likely location of the configuration file, based on the selection made on the Operating System page of the wizard. However, it is possible to test that this information is correct by clicking the **Check path** and **Check section** buttons. The wizard then reports whether the configuration file and server configuration section can in fact be accessed. It is also possible to manually enter the location of the configuration file, and the section pertaining to MySQL Server data; these manually entered values should be tested using the buttons provided. Click the **Next** button to continue.

6. **Review Settings**

   The modified settings may be reviewed, which also includes the default values. Check the Change Parameters checkbox if the MySQL Config File section will be edited, and then click **Next** to continue.

7. **MySQL Config File**

   Allows configuration of the MySQL server version. It also allows the editing and validation of the configuration file path, and validation of the server instance section. Click **Next** to continue.

8. **Specify Commands**

   Optionally set the commands required to start, stop, and check the status of the running MySQL server instance. Commands can be customized, if required, but the defaults are suitable in most cases. The defaults depend on the selected options on the **Operating System** page of the wizard. Click **Next** to continue.

9. **Complete Setup**

   Name the MySQL server instance on the final step. This name is used throughout MySQL Workbench as a reference to this MySQL connection. After setting a suitable name, click **Finish** to save the instance.

5.3.7 The Password Storage Vault

The vault provides a convenient secure storage for passwords used to access MySQL servers. By using the vault, you need not enter credentials every time MySQL Workbench attempts to connect to a server.

Note

The hostname is used for storing password information. For example, a local connection might use "localhost", "127.0.0.1", or "::1", but these are stored separately in the password storage vault, even if they all resolve to the same place.

The vault is implemented differently on each platform:

- **Windows**: The vault is an encrypted file in the MySQL Workbench `data` directory. This is where `connections.xml` and related files are located. The file is encrypted using a Windows API which performs the encryption based on the current user, so only the current user can decrypt it. As a result it is not possible to decrypt the file on any other computer. It is possible to delete the file, in which case all stored passwords are lost, but MySQL Workbench will otherwise perform as expected. You then must re-enter passwords as required.

- **OS X**: The vault is implemented using the OS X Secure Keychain. The keychain contents is also viewable from the native `Keychain Access.app` utility.

- **Linux**: The vault works by storing passwords using the `gnome-keyring` daemon, which must be running for password persistence to work. The daemon is automatically started in GNOME desktops, but normally is not in KDE and others. The `gnome-keyring` daemon stores passwords for MySQL Workbench on non-GNOME platforms, but it must be started manually.

5.3.8 Updating Old Authentication Protocol Passwords

MySQL 4.1 extended password hashes from 16 to 41 bytes. However, upgrading MySQL does not automatically update the old password passwords, so existing passwords continue to be stored in the deprecated format. This is because MySQL does not store passwords as plain text, so regenerating password hashes requires user intervention.

The associated `secure_auth` option was enabled by default as of MySQL 5.6. It is always enabled as of MySQL 5.7, meaning it can *not* be disabled. A future MySQL release will remove this option. With this option enabled, a user with a password defined in the old format will not be able to login to MySQL.

With all that said, the deprecated password format does not function with MySQL 5.7. All passwords using the old format must be updated. This section documents how to upgrade these passwords using MySQL Workbench. For information about migrating away from the old password format using the MySQL command line instead of Workbench, see Migrating Away from Pre-4.1 Password Hashing and the mysql_old_password Plugin.

Note

The method that MySQL stores a password is defined by an authentication plugin. The old method uses the `mysql_old_password` authentication plugin, and the current default method uses `mysql_native_password`. As of MySQL 5.6, a `sha256_password` option is also available although it requires an SSL or encrypted connection. When Workbench upgrades passwords, it upgrades `mysql_old_password` to `mysql_native_password`. For additional information about authentication plugins, see Pluggable Authentication.

Options Depend on your secure_auth Option

Upgrading a password does have constraints. Here are two scenarios:

- If the `secure_auth` MySQL Server option is disabled, then you can log in using the user with the old password format and update the user's own MySQL password. However, this is not an option as of MySQL Workbench 6.3.5 because compatibility with the old password format was removed. For this reason, a user's ability to upgrade their own password format must be done using the MySQL command line as described in Migrating Away from Pre-4.1 Password Hashing and the mysql_old_password Plugin.

  Note

  If using the MySQL command line is not an option, then you could use an older version of MySQL Workbench (version 6.3.4 and earlier), which allows you to enable a Use the old authentication protocol option under the **Advanced** connections tab. Older versions of Workbench are available at https://downloads.mysql.com/archives/workbench/.

  As stated earlier, `secure_auth` is enabled by default as of MySQL 5.6, and always enabled as of MySQL 5.7.

- If `secure_auth` is enabled, you can not log in if your user's password is stored in the old format. Attempts will fail and emit an error similar to "*ERROR 2049 (HY000): Connection using old (pre-4.1.1) authentication protocol refused (client option 'secure_auth' enabled)*". To upgrade the password, you can either disable `secure_auth` (not recommended) then update as described above, or log in as a different and privileged user, such as root, to change the password for a different user.

Using Workbench to Upgrade Your Password

Keeping the above in mind, there are two methods to update passwords using Workbench.

Open the Users and Privileges tab from Workbench's Management navigator. Select the user you want to update from the **User Accounts** section. If using the old password format, you will see text beginning with "This account is using the pre-mysql-4.1.1 password hashing type." in the lower right corner of the screen, and also a large **Upgrade** button on the right. From here, you can:

- Option for all MySQL versions:

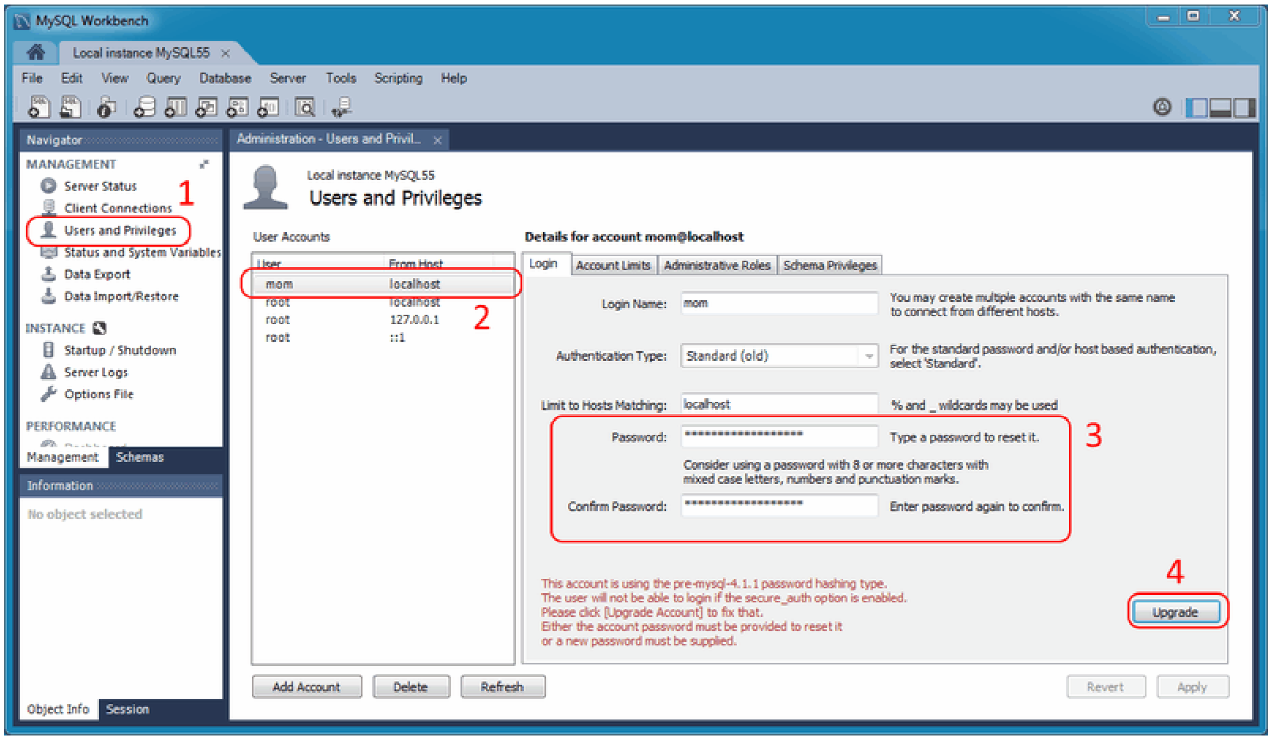
  Manually enter a new password, or the current password, and click **Upgrade**. This upgrades the password to the newer password format, and the MySQL user can now log in using the new password that you defined.

- Option for MySQL 5.6 and later:

  Rather than editing the password field, leave it alone and immediately click **Upgrade**. From here, you can generate a random password and tag it as expired by clicking **Reset To Expired**. Use this temporary random password to login the user, and MySQL will prompt for a new password when the user first logs in.

The following screen shots demonstrate the steps used in both methods:

Figure 5.25 Upgrade Old Password: Setting a New Password



User Accounts

User	From Host
mom	localhost
root	localhost
root	127.0.0.1
root	::1

Details for account mom@localhost

Login Name: mom — You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: Standard (old) — For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: localhost — % and _ wildcards may be used

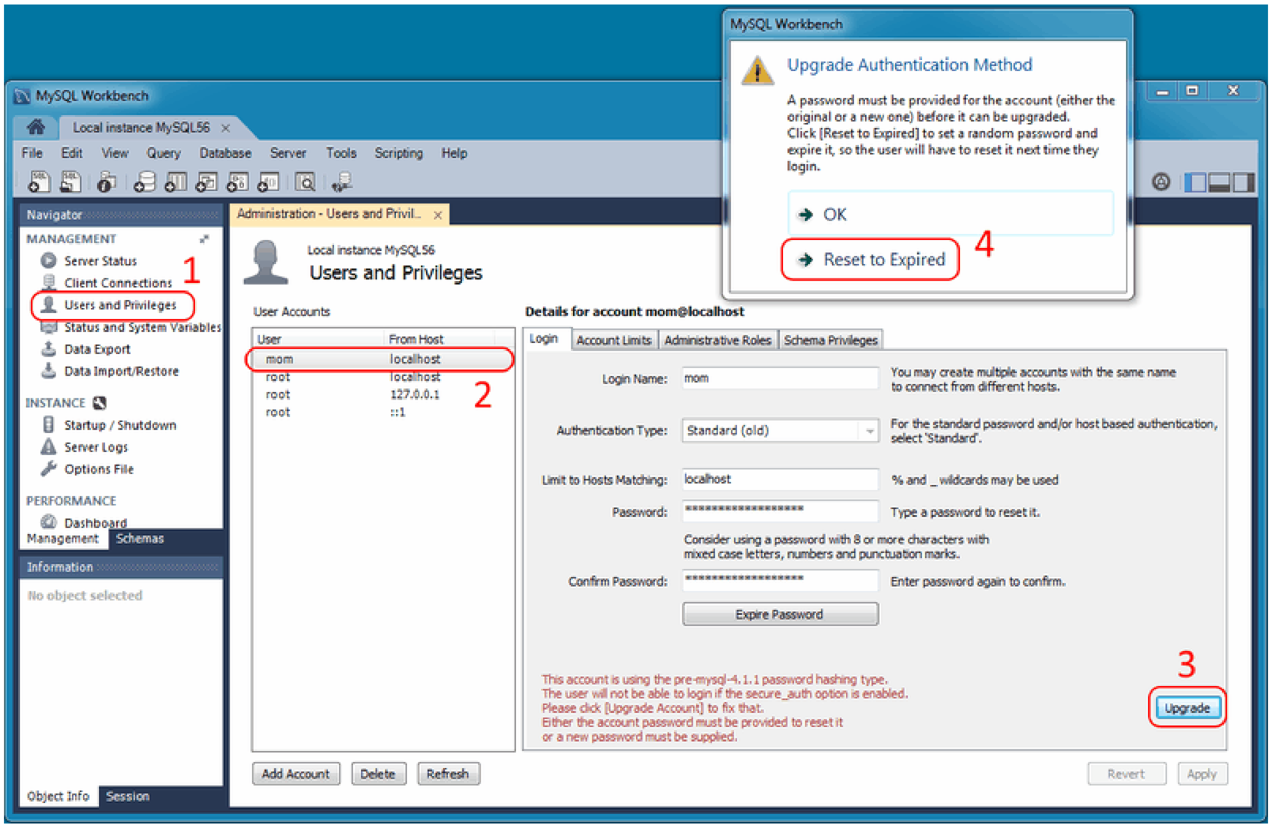
Password: ****************** — Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: ****************** — Enter password again to confirm.

This account is using the pre-mysql-4.1.1 password hashing type.
The user will not be able to login if the secure_auth option is enabled.
Please click [Upgrade Account] to fix that.
Either the account password must be provided to reset it
or a new password must be supplied.

Figure 5.26 Upgrade Old Password: Reset to Random Expired Password



MySQL Workbench

Upgrade Authentication Method

A password must be provided for the account (either the original or a new one) before it can be upgraded.
Click [Reset to Expired] to set a random password and expire it, so the user will have to reset it next time they login.

Local instance MySQL56
Users and Privileges

User Accounts

User	From Host
mom	localhost
root	localhost
root	127.0.0.1
root	::1

Details for account mom@localhost

Login Name: mom — You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: Standard (old) — For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: localhost — % and _ wildcards may be used

Password: ****************** — Type a password to reset it.

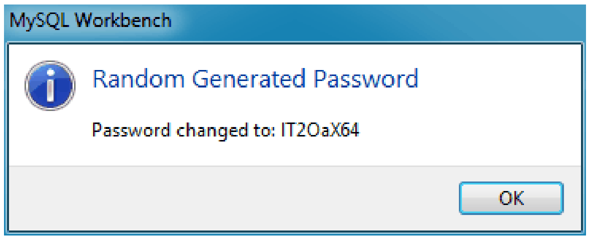
Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: ****************** — Enter password again to confirm.

This account is using the pre-mysql-4.1.1 password hashing type.
The user will not be able to login if the secure_auth option is enabled.
Please click [Upgrade Account] to fix that.
Either the account password must be provided to reset it
or a new password must be supplied.

When resetting to a random password, you must save the password and give it to the user. You will find the random password in the new popup window, that is similar to:

Figure 5.27 Upgrade Old Password: Random Password Popup



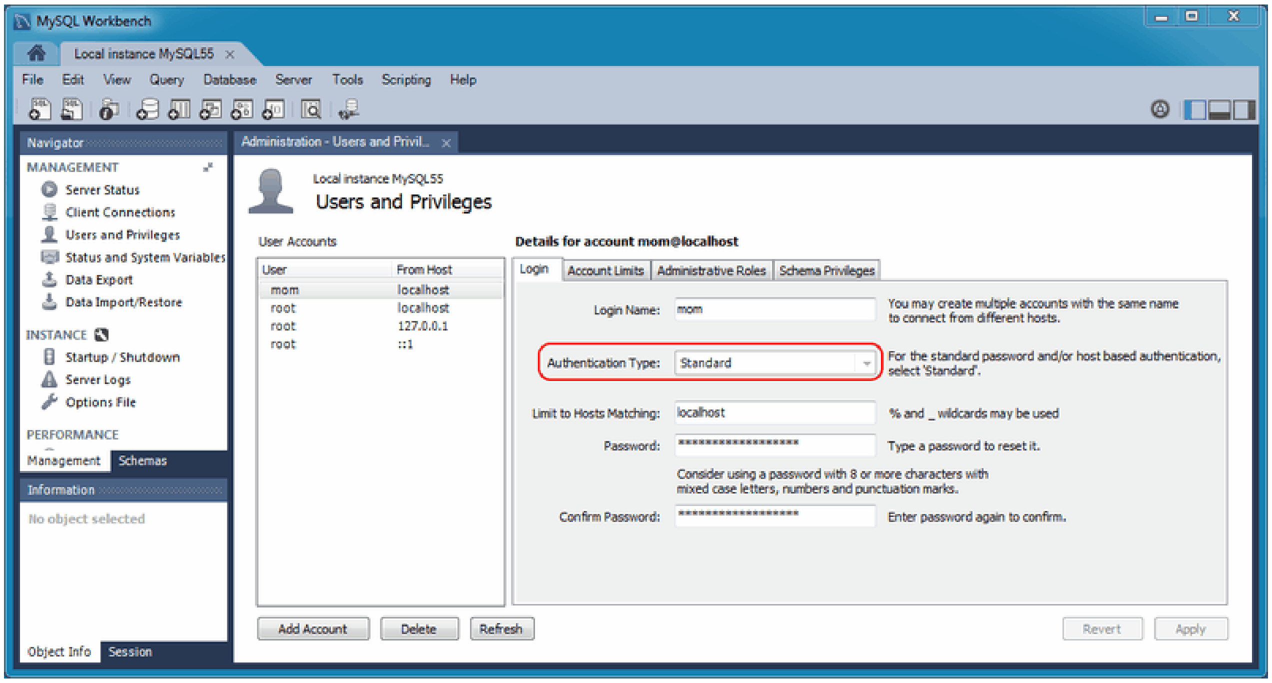
MySQL Workbench

Random Generated Password

Password changed to: IT2OaX64

After completing the upgrade, notice the new **Authentication Type** for the connection. In our examples, it changed from **Standard (old)** to **Standard**. In other words, the authentication type changed from `mysql_old_password` to `mysql_native_password`.

Figure 5.28 Upgraded Password: Standard (old) to Standard



User	From Host
mom	localhost
root	localhost
root	127.0.0.1
root	::1

Login Name: mom — You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: Standard — For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: localhost — % and _ wildcards may be used

Password: ****************** — Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: ****************** — Enter password again to confirm.

5.4 MySQL Fabric Integration

Browse, view status, and connect to any MySQL instance in a Fabric Cluster.

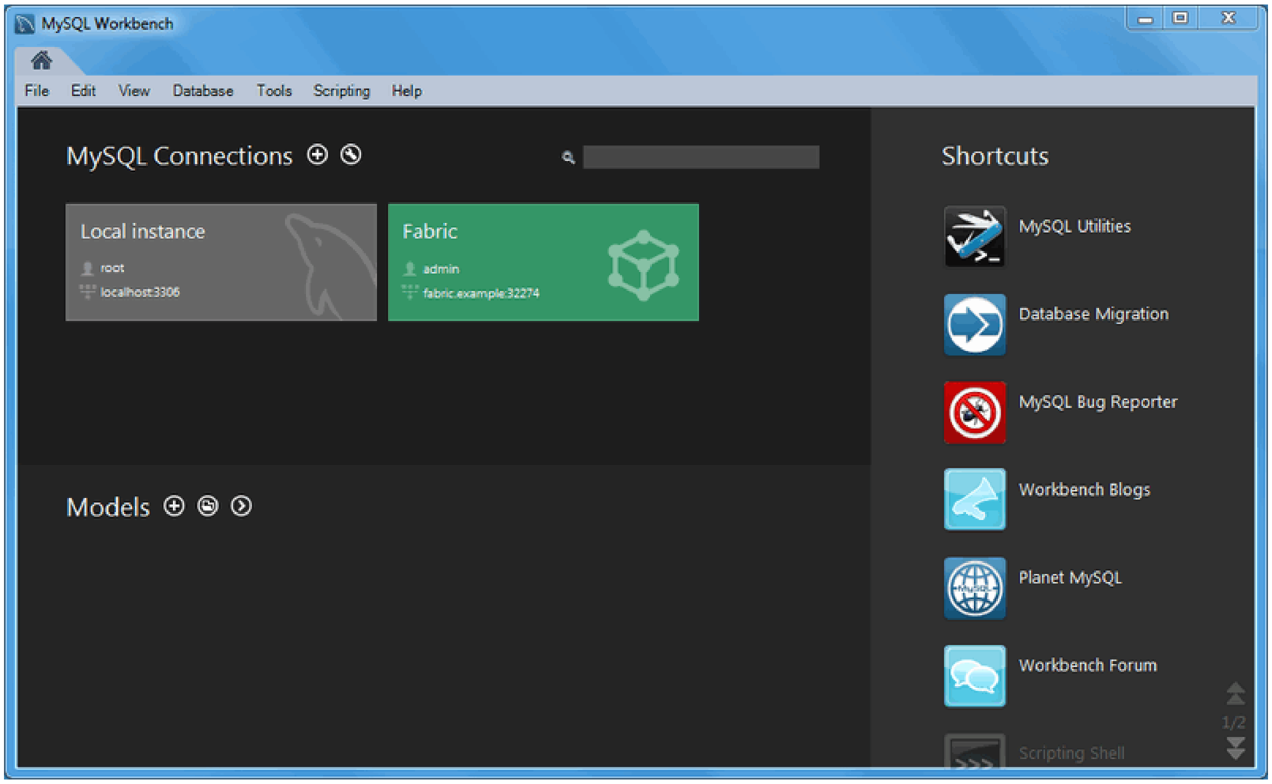
Note

This requires Connector/Python and MySQL Fabric 1.5 and above installed, including the Python module. This feature also requires MySQL Workbench 6.3+.

Fabric 1.5 support was added in MySQL Workbench 6.3, and due to incompatible protocol changes, Fabric 1.4 support was dropped. Also, MySQL Workbench 6.2 added Fabric 1.4 support.

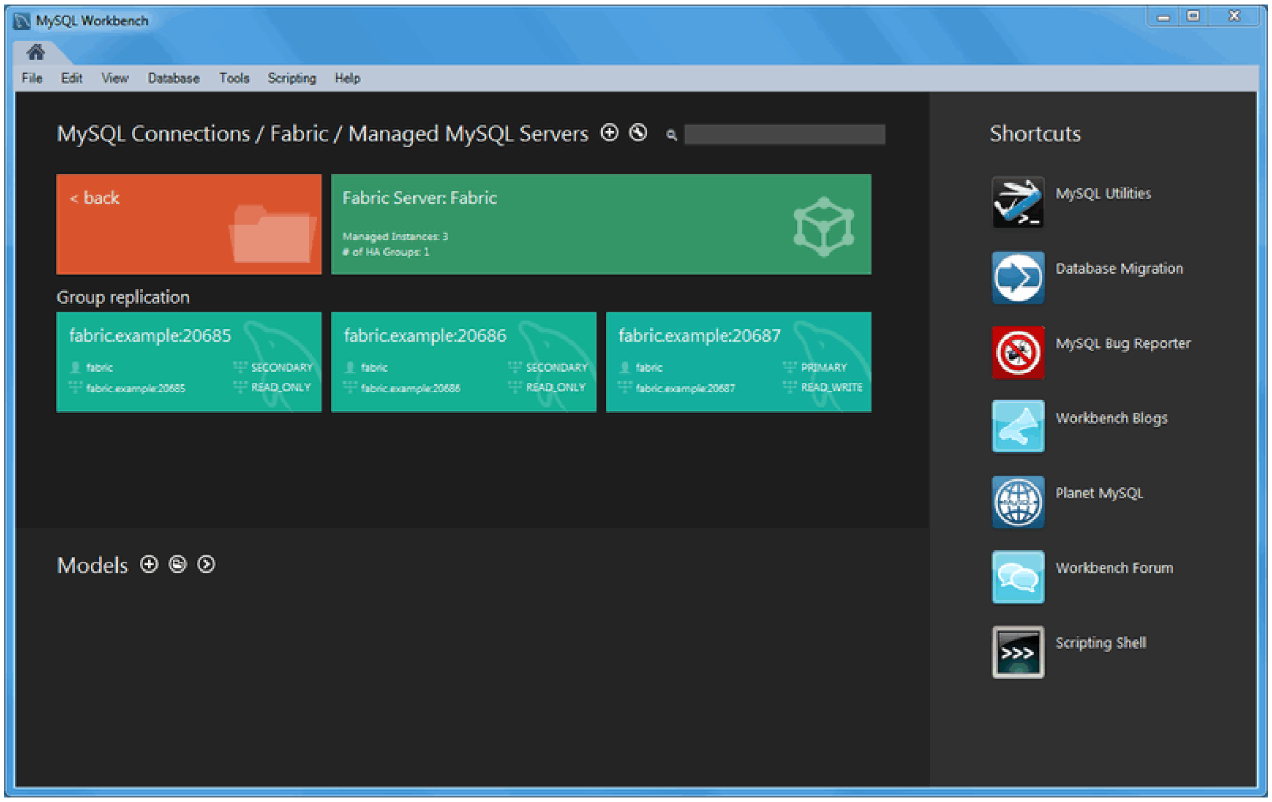
To set up a managed Fabric connection, create a new MySQL connection with the new **MySQL Fabric Management Node** connection method. The connection tiles have a different look:

Figure 5.29 Fabric Connection Group Tile



Clicking the new fabric group tile shows the managed connections:

Figure 5.30 Fabric Connection Group Tiles



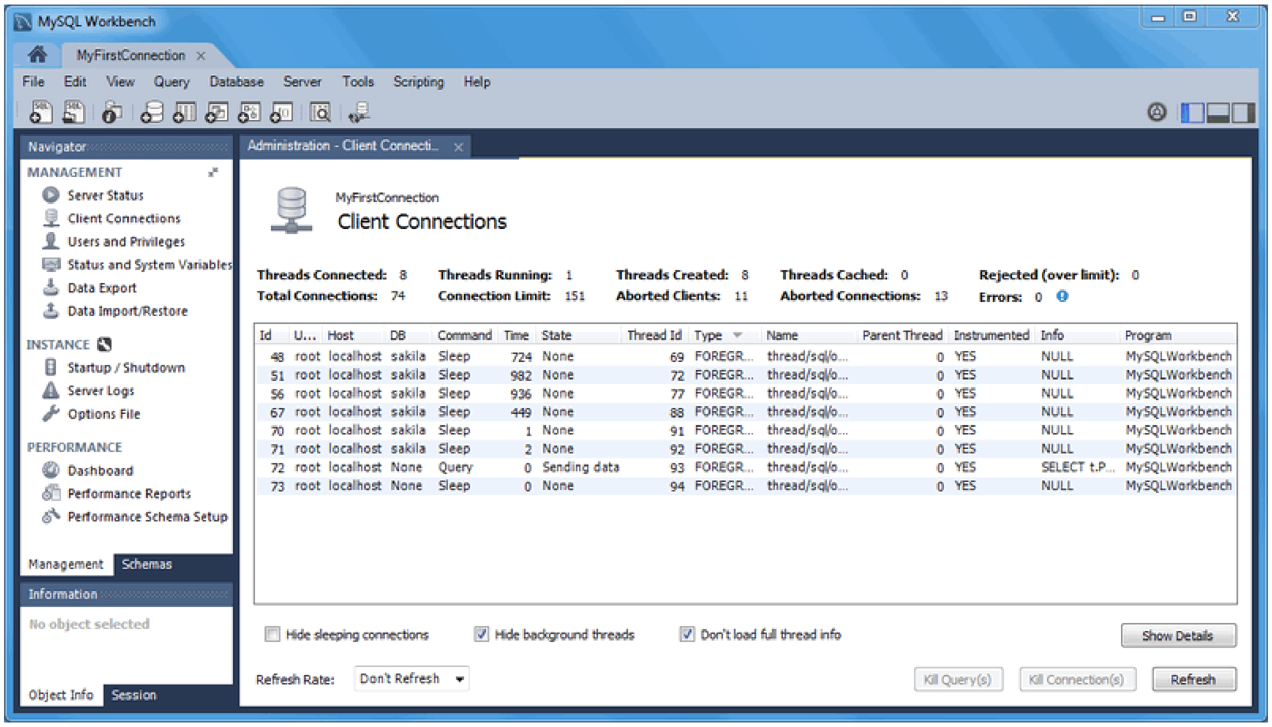
5.5 Client Connections

The client connection browser lists the active and sleeping MySQL client connections, and adds the ability to kill statements and connections, and view additional connection details and attributes.

Note

The connection details viewer requires MySQL 5.6 or greater. Only basic connection information is available for previous versions of MySQL, such as the connection hosts, database, and state.

Figure 5.31 Client Connection Overview



MyFirstConnection
Client Connections

Threads Connected: 8 Threads Running: 1 Threads Created: 8 Threads Cached: 0 Rejected (over limit): 0
Total Connections: 74 Connection Limit: 151 Aborted Clients: 11 Aborted Connections: 13 Errors: 0

Id	U...	Host	DB	Command	Time	State	Thread Id	Type	Name	Parent Thread	Instrumented	Info	Program
48	root	localhost	sakila	Sleep	724	None	69	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench
51	root	localhost	sakila	Sleep	982	None	72	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench
56	root	localhost	sakila	Sleep	936	None	77	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench
67	root	localhost	sakila	Sleep	449	None	88	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench
70	root	localhost	sakila	Sleep	1	None	91	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench
71	root	localhost	sakila	Sleep	2	None	92	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench
72	root	localhost	None	Query	0	Sending data	93	FOREGR...	thread/sql/o...	0	YES	SELECT t.P...	MySQLWorkbench
73	root	localhost	None	Sleep	0	None	94	FOREGR...	thread/sql/o...	0	YES	NULL	MySQLWorkbench

☐ Hide sleeping connections ☑ Hide background threads ☑ Don't load full thread info

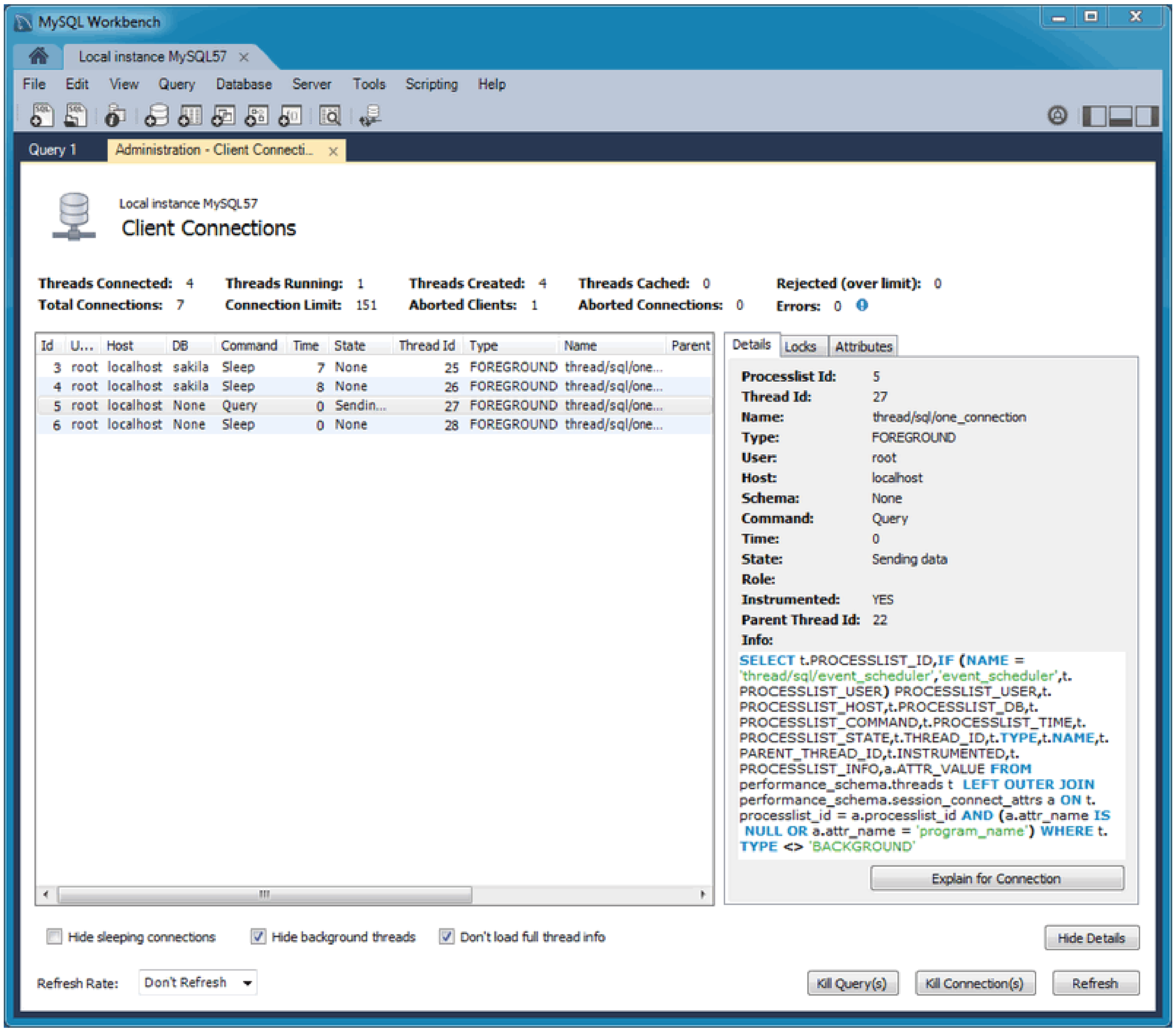
Refresh Rate: Don't Refresh

Client Connections and Metadata locks

The **Client Connections** management window includes a **Show Details** for connections to MySQL 5.6 and above. These details are separated into three tabs:

- **Details**: connection details such as Process ID, Type, User, Host, Instrumented, and additional information.

Figure 5.32 Client Connections Details



Local instance MySQL57

Client Connections

Threads Connected: 4	Threads Running: 1	Threads Created: 4	Threads Cached: 0	Rejected (over limit): 0
Total Connections: 7	Connection Limit: 151	Aborted Clients: 1	Aborted Connections: 0	Errors: 0

Id	U...	Host	DB	Command	Time	State	Thread Id	Type	Name	Parent
3	root	localhost	sakila	Sleep	7	None	25	FOREGROUND	thread/sql/one...	
4	root	localhost	sakila	Sleep	8	None	26	FOREGROUND	thread/sql/one...	
5	root	localhost	None	Query	0	Sendin...	27	FOREGROUND	thread/sql/one...	
6	root	localhost	None	Sleep	0	None	28	FOREGROUND	thread/sql/one...	

Processlist Id:	5
Thread Id:	27
Name:	thread/sql/one_connection
Type:	FOREGROUND
User:	root
Host:	localhost
Schema:	None
Command:	Query
Time:	0
State:	Sending data
Role:	
Instrumented:	YES
Parent Thread Id:	22
Info:	

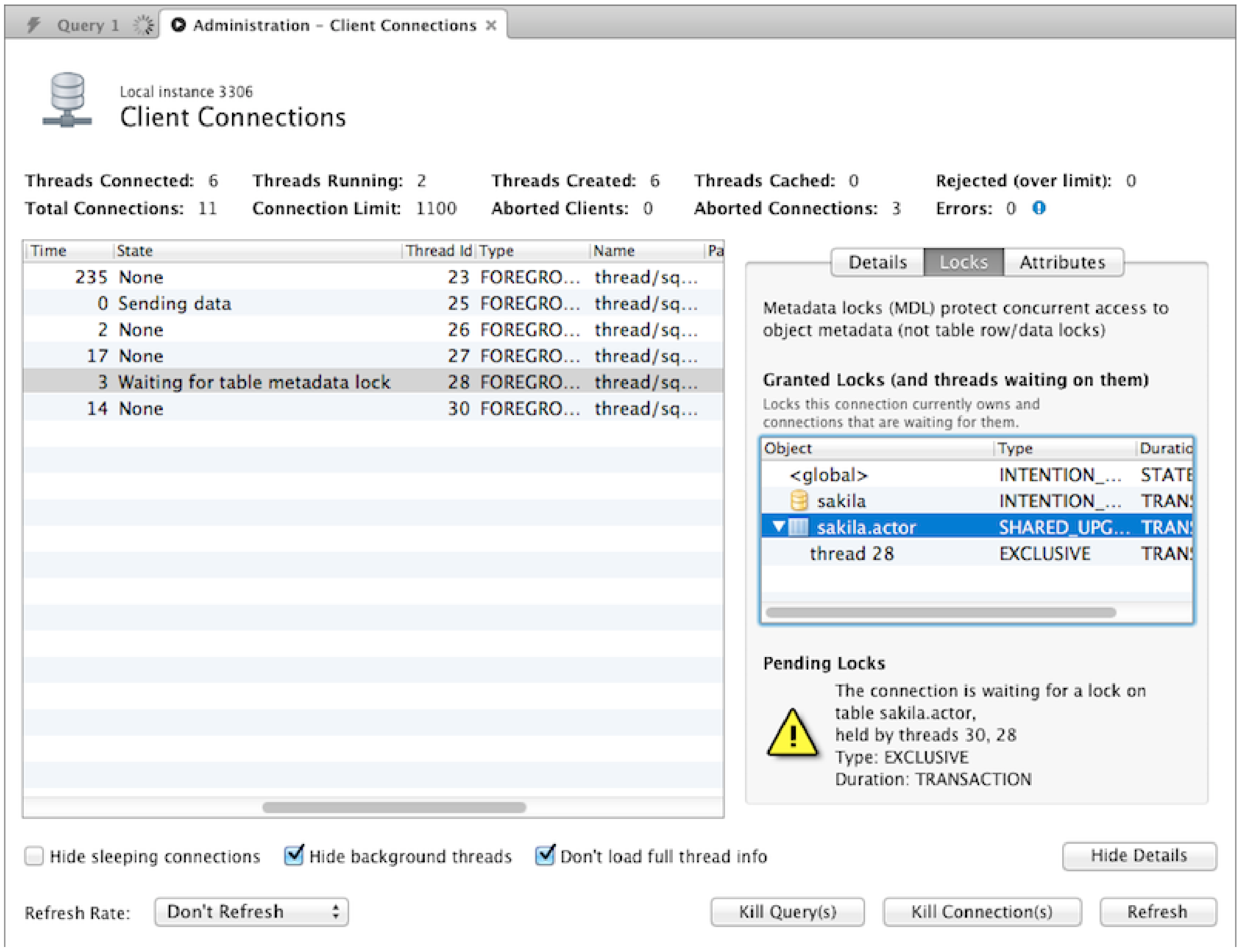
```
SELECT t.PROCESSLIST_ID,IF (NAME =
'thread/sql/event_scheduler','event_scheduler',t.
PROCESSLIST_USER) PROCESSLIST_USER,t.
PROCESSLIST_HOST,t.PROCESSLIST_DB,t.
PROCESSLIST_COMMAND,t.PROCESSLIST_TIME,t.
PROCESSLIST_STATE,t.THREAD_ID,t.TYPE,t.NAME,t.
PARENT_THREAD_ID,t.INSTRUMENTED,t.
PROCESSLIST_INFO,a.ATTR_VALUE FROM
performance_schema.threads t  LEFT OUTER JOIN
performance_schema.session_connect_attrs a ON t.
processlist_id = a.processlist_id AND (a.attr_name IS
 NULL OR a.attr_name = 'program_name') WHERE t.
TYPE <> 'BACKGROUND'
```

☐ Hide sleeping connections ☑ Hide background threads ☑ Don't load full thread info

Refresh Rate: Don't Refresh

- **Locks**: MySQL uses metadata locking to manage access to objects such as tables and triggers. Sometimes a query might be blocked while being manipulated by another connection from another user. The **Locks** feature utilizes these MySQL metadata locks (MDL) to show the locked connections that are blocked or being waiting on, and shows information about the locks, what they are waiting for, and what they hold.

Figure 5.33 Metadata Locks Browser



Threads Connected: 6 Threads Running: 2 Threads Created: 6 Threads Cached: 0 Rejected (over limit): 0
Total Connections: 11 Connection Limit: 1100 Aborted Clients: 0 Aborted Connections: 3 Errors: 0

Time	State	Thread Id	Type	Name	Pa
235	None	23	FOREGRO...	thread/sq...	
0	Sending data	25	FOREGRO...	thread/sq...	
2	None	26	FOREGRO...	thread/sq...	
17	None	27	FOREGRO...	thread/sq...	
3	Waiting for table metadata lock	28	FOREGRO...	thread/sq...	
14	None	30	FOREGRO...	thread/sq...	

Metadata locks (MDL) protect concurrent access to object metadata (not table row/data locks)

Granted Locks (and threads waiting on them)
Locks this connection currently owns and connections that are waiting for them.

Object	Type	Duratio
<global>	INTENTION_...	STATE
sakila	INTENTION_...	TRAN
sakila.actor	SHARED_UPG...	TRAN
thread 28	EXCLUSIVE	TRAN

Pending Locks
The connection is waiting for a lock on table sakila.actor,
held by threads 30, 28
Type: EXCLUSIVE
Duration: TRANSACTION

Hide sleeping connections [✓ Hide background threads] [✓ Don't load full thread info]

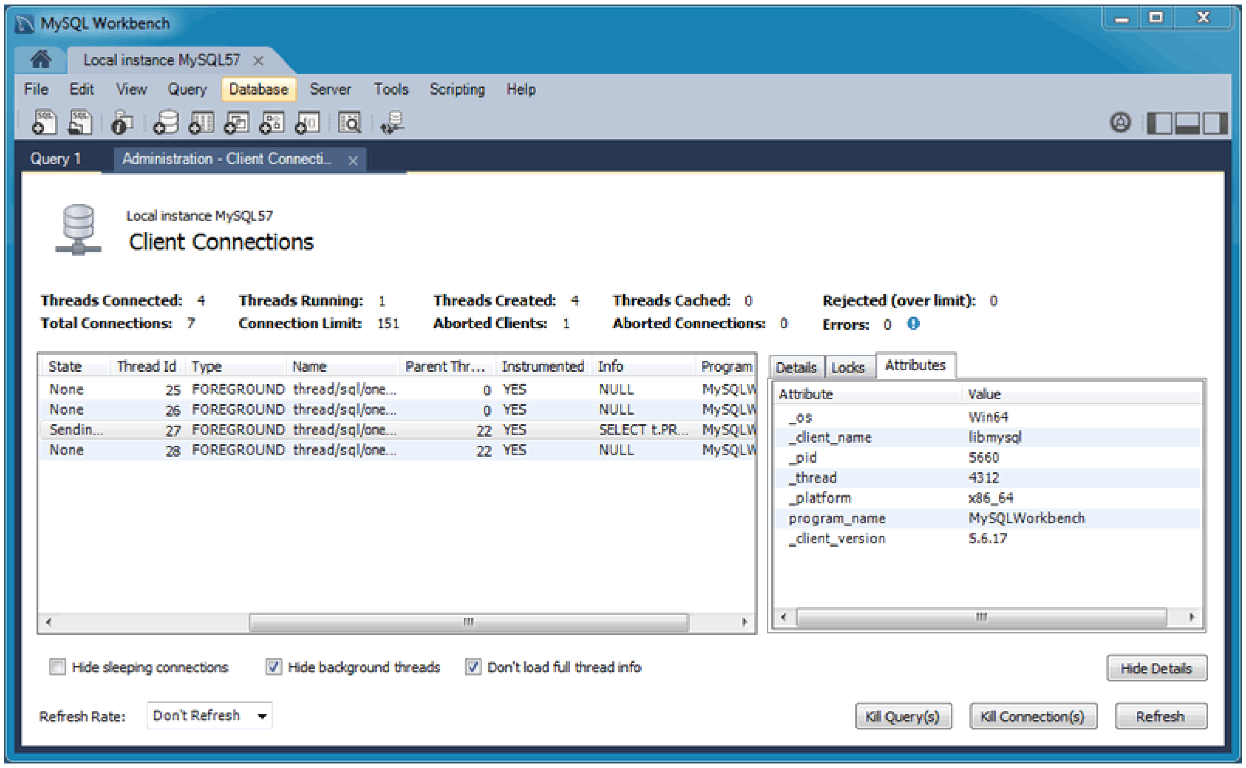
Refresh Rate: Don't Refresh

Note

The metadata lock information is provided in the performance schema as of MySQL server 5.7.3.

- **Attributes**: these are connection attributes such as OS, Client Name, Client Version, and Platform.

Figure 5.34 Client Connection Attributes


MySQL Workbench
Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Query 1
Administration - Client Connecti...
Local instance MySQL57
Client Connections

Threads Connected: 4 Threads Running: 1 Threads Created: 4 Threads Cached: 0 Rejected (over limit): 0
Total Connections: 7 Connection Limit: 151 Aborted Clients: 1 Aborted Connections: 0 Errors: 0

State	Thread Id	Type	Name	Parent Thr...	Instrumented	Info	Program
None	25	FOREGROUND	thread/sql/one...	0	YES	NULL	MySQLW
None	26	FOREGROUND	thread/sql/one...	0	YES	NULL	MySQLW
Sendin...	27	FOREGROUND	thread/sql/one...	22	YES	SELECT t.PR...	MySQLW
None	28	FOREGROUND	thread/sql/one...	22	YES	NULL	MySQLW

Attribute	Value
_os	Win64
_client_name	libmysql
_pid	5660
_thread	4312
_platform	x86_64
program_name	MySQLWorkbench
_client_version	5.6.17

☐ Hide sleeping connections ☑ Hide background threads ☑ Don't load full thread info

Refresh Rate: Don't Refresh

Chapter 6 Administrative Tasks

Table of Contents

MySQL Workbench provides a visual GUI to administer your MySQL environment. The available visual tools help configure your MySQL servers, administer users, perform backup and recovery, inspect audit data, and view database health.

6.1 Server Management

Manage your MySQL instances with a comprehensive view of your MySQL server connections. The visual tree based navigation provides detailed information about server and status variables, including the number of threads, bytes sent and received by clients, buffer allocations size, and more.

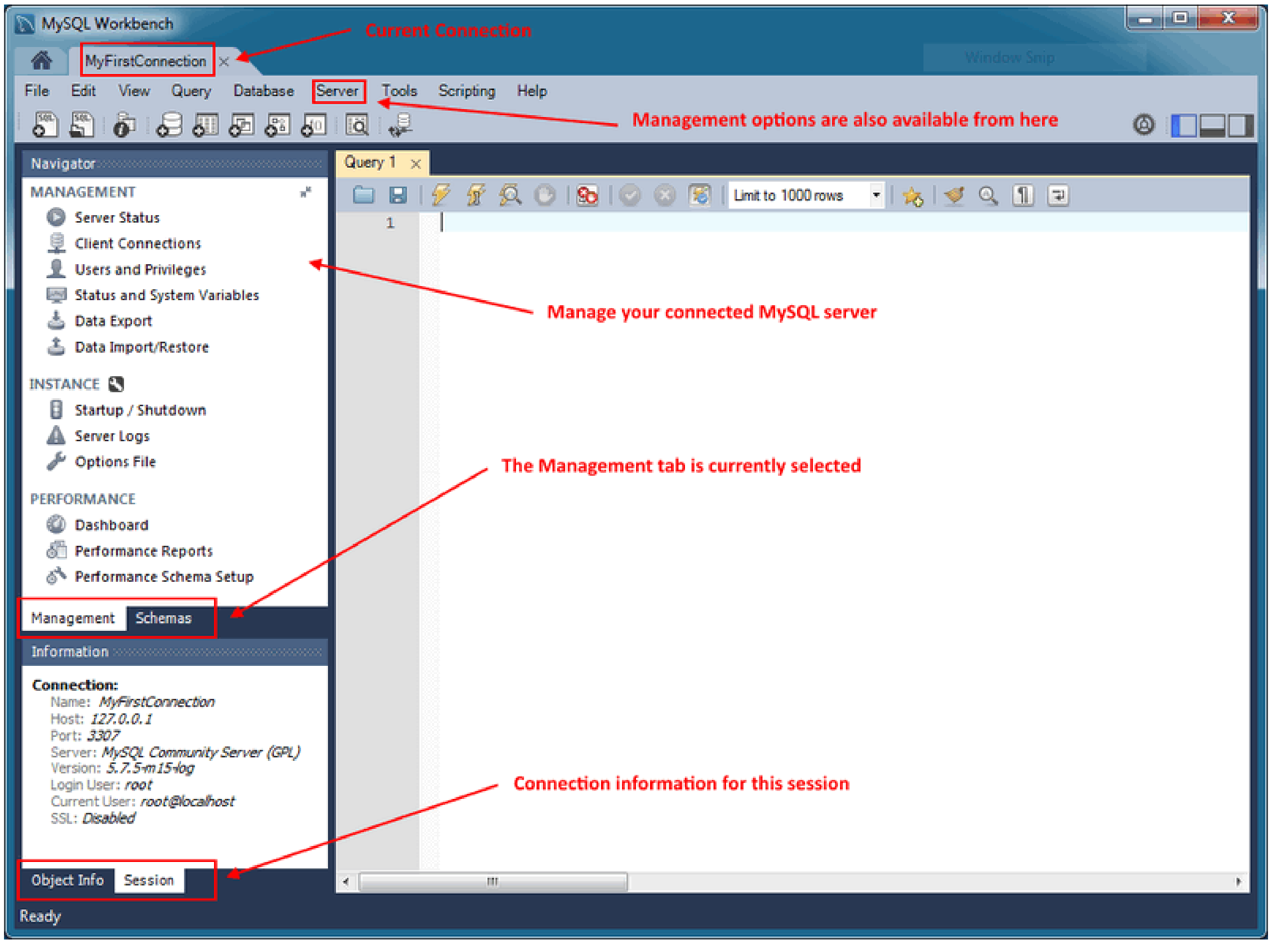
6.1.1 MySQL Connection Navigator

The Navigator panel has a **Management** tab with functionality to monitor and configure your selected MySQL connection.

Note

The Navigator panel also has a **Schemas** tab for managing databases using your MySQL Connection. For information about the Schemas tab, see Section 8.2.1, “Object Browser and Editor Navigator”.

Figure 6.1 SQL Editor - Navigator Management Tab



Connection:
Name: *MyFirstConnection*
Host: *127.0.0.1*
Port: *3307*
Server: *MySQL Community Server (GPL)*
Version: *5.7.5-m15-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Disabled*

The Navigator Management tab is separated into the **MANAGEMENT**, **INSTANCE**, and **PERFORMANCE** sections, and the Commercial edition of MySQL Workbench also includes the **MYSQL ENTERPRISE** section.

6.1.2 Server Logs

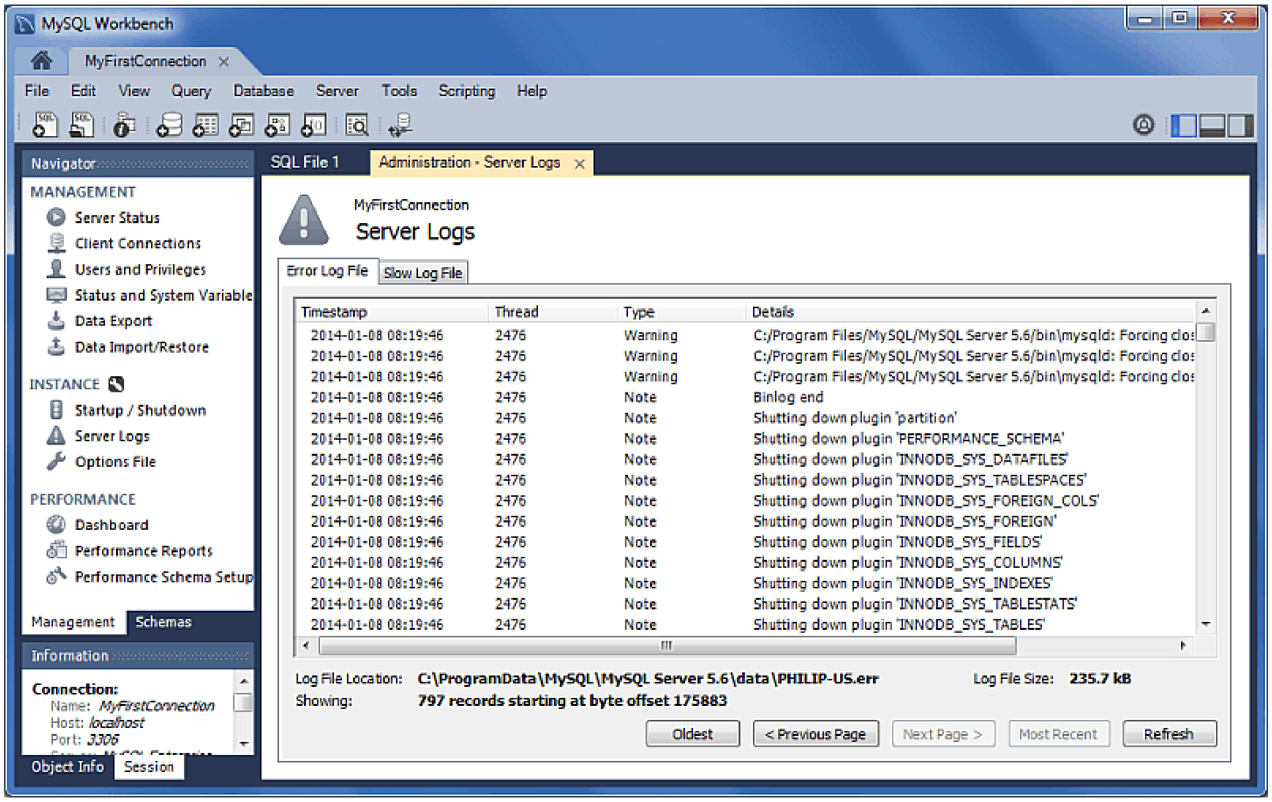
The Server Logs page displays your MySQL Server logs, and includes tabs for the general error logs, and also the slow logs.

Access this window from either the Navigator panel, or by selecting **Server**, **Server Logs** from the main menu.

Error Log File

General MySQL errors, for more information see The Error Log

Figure 6.2 Navigator Management: Instance: Server Logs: Error Log



MyFirstConnection
Server Logs

Timestamp	Thread	Type	Details
2014-01-08 08:19:46	2476	Warning	C:/Program Files/MySQL/MySQL Server 5.6/bin\mysqld: Forcing clo
2014-01-08 08:19:46	2476	Warning	C:/Program Files/MySQL/MySQL Server 5.6/bin\mysqld: Forcing clo
2014-01-08 08:19:46	2476	Warning	C:/Program Files/MySQL/MySQL Server 5.6/bin\mysqld: Forcing clo
2014-01-08 08:19:46	2476	Note	Binlog end
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'partition'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'PERFORMANCE_SCHEMA'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_DATAFILES'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_TABLESPACES'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_FOREIGN_COLS'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_FOREIGN'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_FIELDS'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_COLUMNS'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_INDEXES'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_TABLESTATS'
2014-01-08 08:19:46	2476	Note	Shutting down plugin 'INNODB_SYS_TABLES'

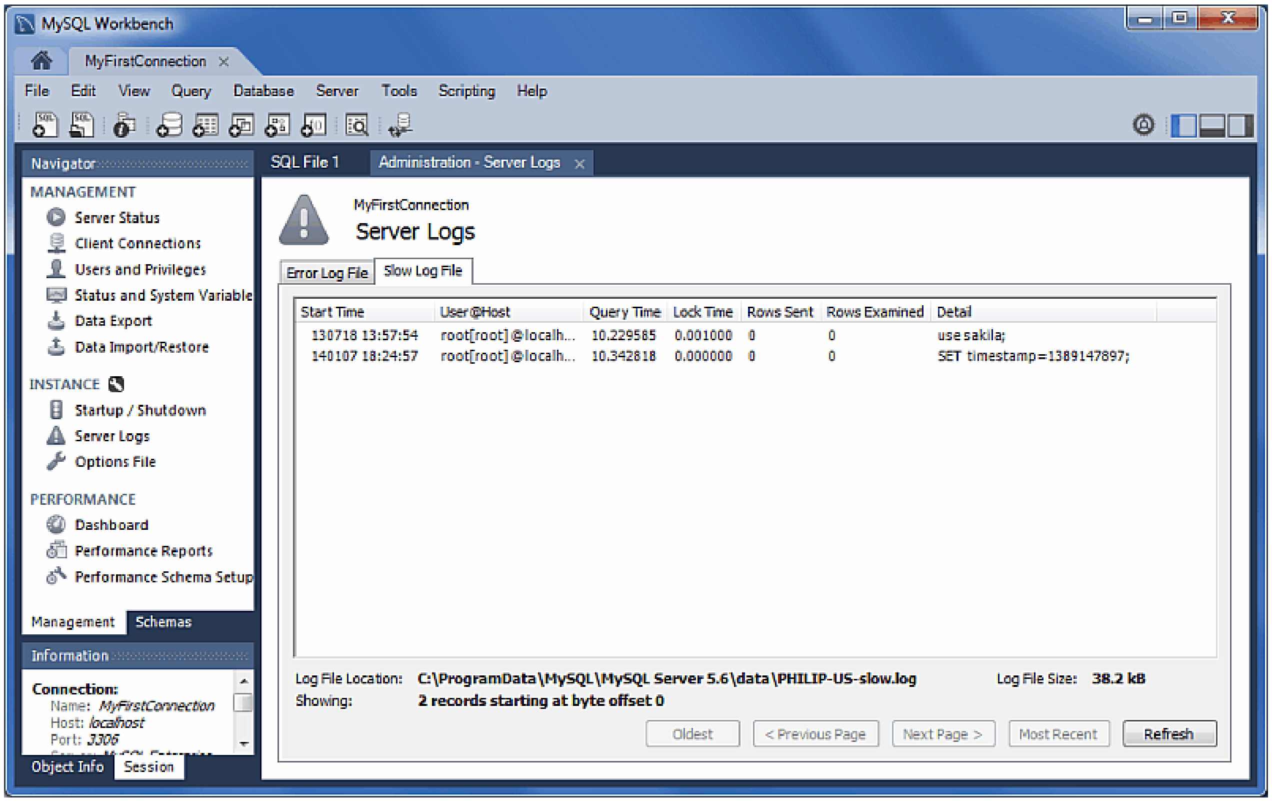
Log File Location: **C:\ProgramData\MySQL\MySQL Server 5.6\data\PHILIP-US.err** Log File Size: **235.7 kB**
Showing: **797 records starting at byte offset 175883**

Connection:
Name: *MyFirstConnection*
Host: *localhost*
Port: *3306*

Slow Log File

Slow queries (when available), for more information see The Slow Query Log

Figure 6.3 Navigator Management: Instance: Server Logs: Slow Log



Start Time	User@Host	Query Time	Lock Time	Rows Sent	Rows Examined	Detail
130718 13:57:54	root[root] @localh...	10.229585	0.001000	0	0	use sakila;
140107 18:24:57	root[root] @localh...	10.342818	0.000000	0	0	SET timestamp=1389147897;

Log File Location: **C:\ProgramData\MySQL\MySQL Server 5.6\data\PHILIP-US-slow.log** Log File Size: **38.2 kB**
Showing: **2 records starting at byte offset 0**

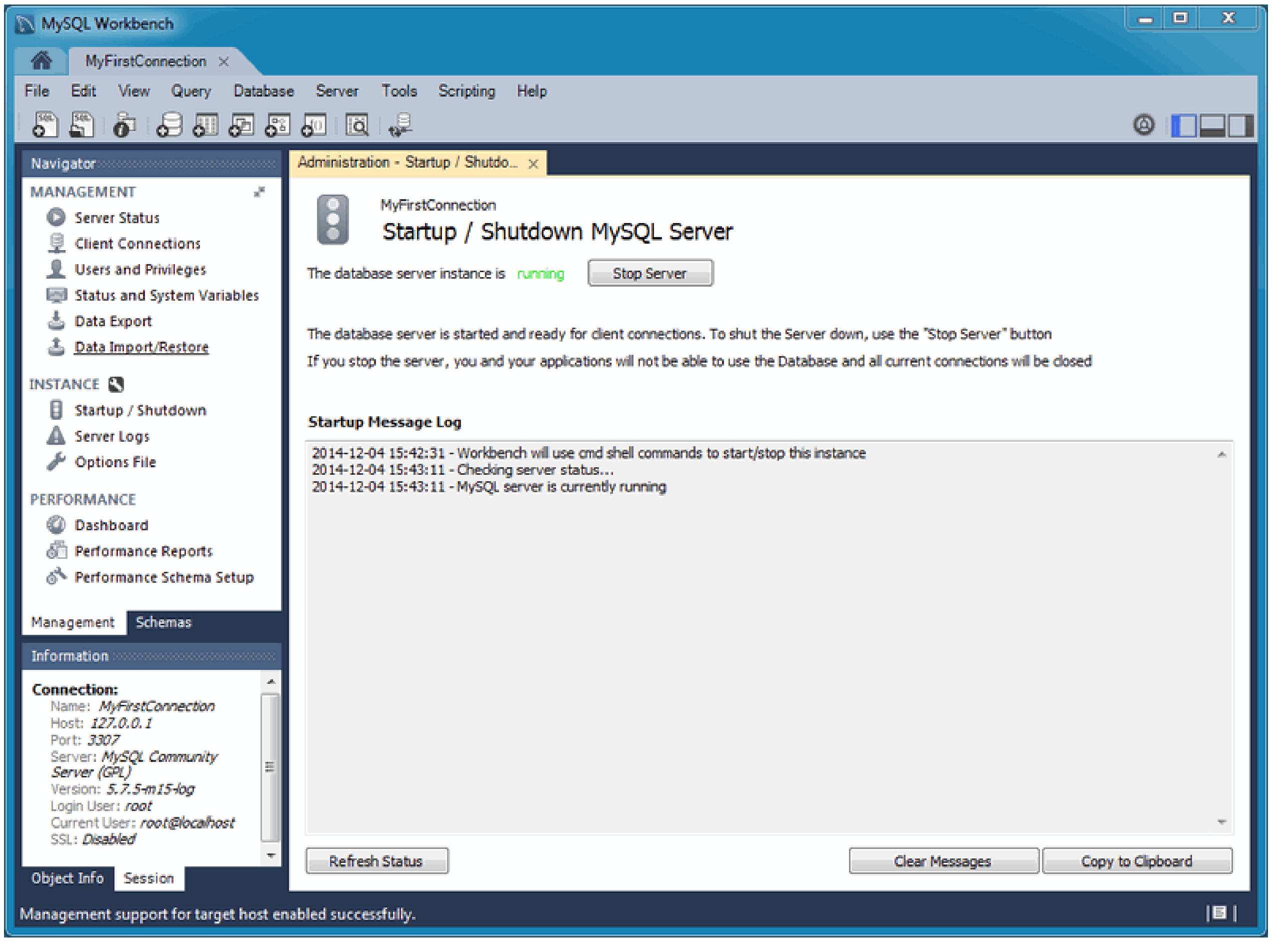
6.1.3 Service Control

The **Startup / Shutdown** functionality includes:

- Viewing the **Startup Message Log**
- Start up and shut down the MySQL instance
- View the current status of the MySQL instance

Access this window from either the Navigator panel, or by selecting **Server**, **Startup/Shutdown** from the main menu.

Figure 6.4 Navigator Management: Instance: Startup / Shutdown



MyFirstConnection

Startup / Shutdown MySQL Server

The database server instance is running

The database server is started and ready for client connections. To shut the Server down, use the "Stop Server" button

If you stop the server, you and your applications will not be able to use the Database and all current connections will be closed

Startup Message Log

```
2014-12-04 15:42:31 - Workbench will use cmd shell commands to start/stop this instance
2014-12-04 15:43:11 - Checking server status...
2014-12-04 15:43:11 - MySQL server is currently running
```

Connection:
Name: MyFirstConnection
Host: 127.0.0.1
Port: 3307
Server: MySQL Community Server (GPL)
Version: 5.7.5-m15-log
Login User: root
Current User: root@localhost
SSL: Disabled

Management support for target host enabled successfully.

6.1.4 Configuration (options file)

The **Options File** editor is used to view and edit the MySQL configuration file (`my.ini` on Windows, or `my.cnf` on Linux / OS X) by selecting check boxes and other GUI controls, and then making edits. MySQL Workbench divides the options file into its own groupings as a set of tabs (such as General, Logging, InnoDB, and more). Make an edit and click **Apply** to commit the changes.

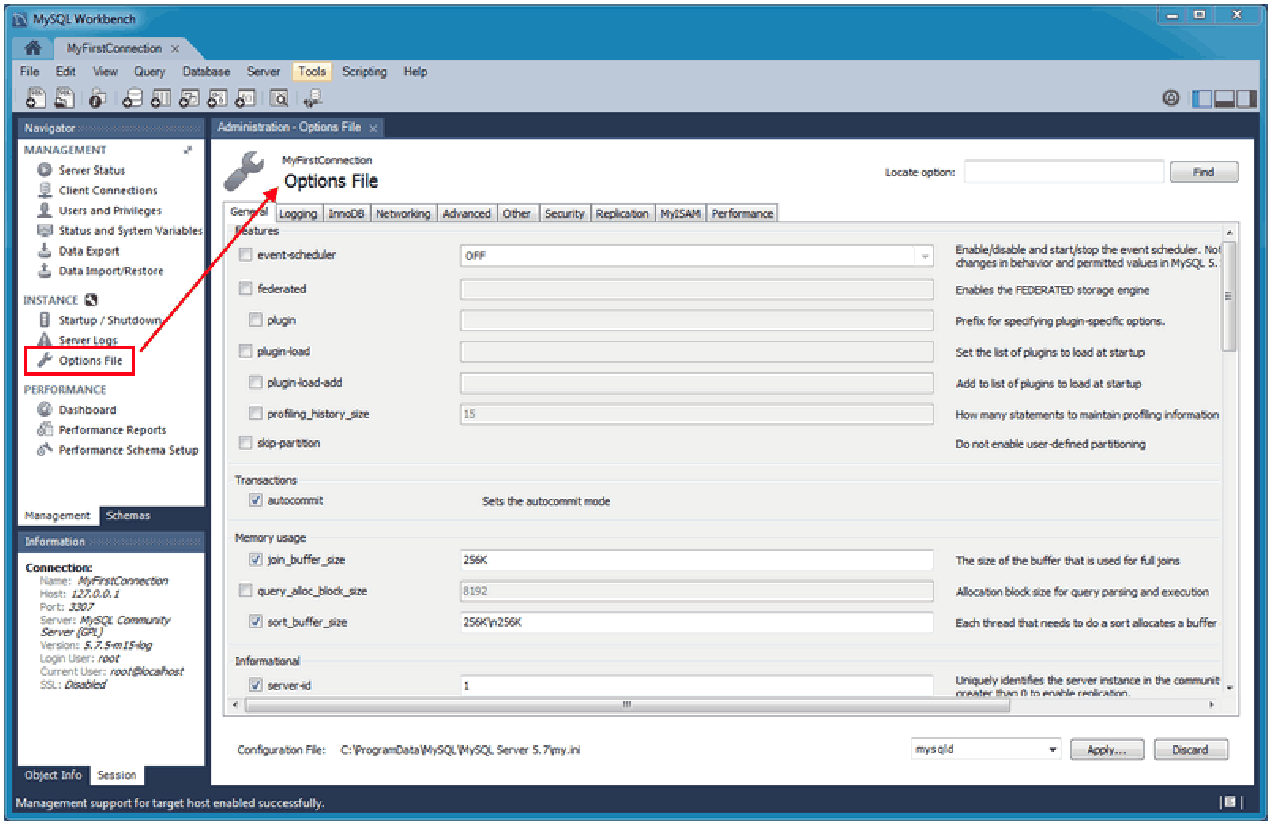
Access this window from either the Navigator panel, or by selecting **Server**, **Options File** from the main menu.

The options file editor includes the following components:

- Option file groupings, as divided into convenient tabs by MySQL Workbench
- A **Locate option** search field to search your MySQL options configuration file
- **Configuration File** path, so you know the configuration file you are editing
- An options file group selector, to select the option [group] to edit. Because the same option can be defined under multiple groupings, it is important to choose the correct group when making edits. `[mysqld]` (the MySQL server) is the default and most common group. For additional information about groups, see Using Option Files.

A screenshot with the General tab selected:

Figure 6.5 Navigator Management: Instance: Options File: General



Connection:
Name: MyFirstConnection
Host: 127.0.0.1
Port: 3307
Server: MySQL Community Server (GPL)
Version: 5.7.5-m15-log
Login User: root
Current User: root@localhost
SSL: Disabled

☐ event-scheduler	OFF	Enable/disable and start/stop the event scheduler. Not changes in behavior and permitted values in MySQL 5.
☐ federated		Enables the FEDERATED storage engine
☐ plugin		Prefix for specifying plugin-specific options.
☐ plugin-load		Set the list of plugins to load at startup
☐ plugin-load-add		Add to list of plugins to load at startup
☐ profiling_history_size	15	How many statements to maintain profiling information
☐ skip-partition		Do not enable user-defined partitioning
☑ autocommit		Sets the autocommit mode
☑ join_buffer_size	256K	The size of the buffer that is used for full joins
☐ query_alloc_block_size	8192	Allocation block size for query parsing and execution
☑ sort_buffer_size	256K\n256K	Each thread that needs to do a sort allocates a buffer
☑ server-id	1	Uniquely identifies the server instance in the communit greater than 0 to enable replication.

Configuration File: C:\ProgramData\MySQL\MySQL Server 5.7\my.ini   mysqld

Management support for target host enabled successfully.

6.2 Users and Privileges

A listing of all users and privileges that relate to the MySQL connection. You may also manage (add) user accounts, adjust privileges, and expire passwords.

Access this window from either the Navigator panel, or by selecting **Server**, **Users and Privileges** from the main menu.

The **Users and Privileges** page has several sections:

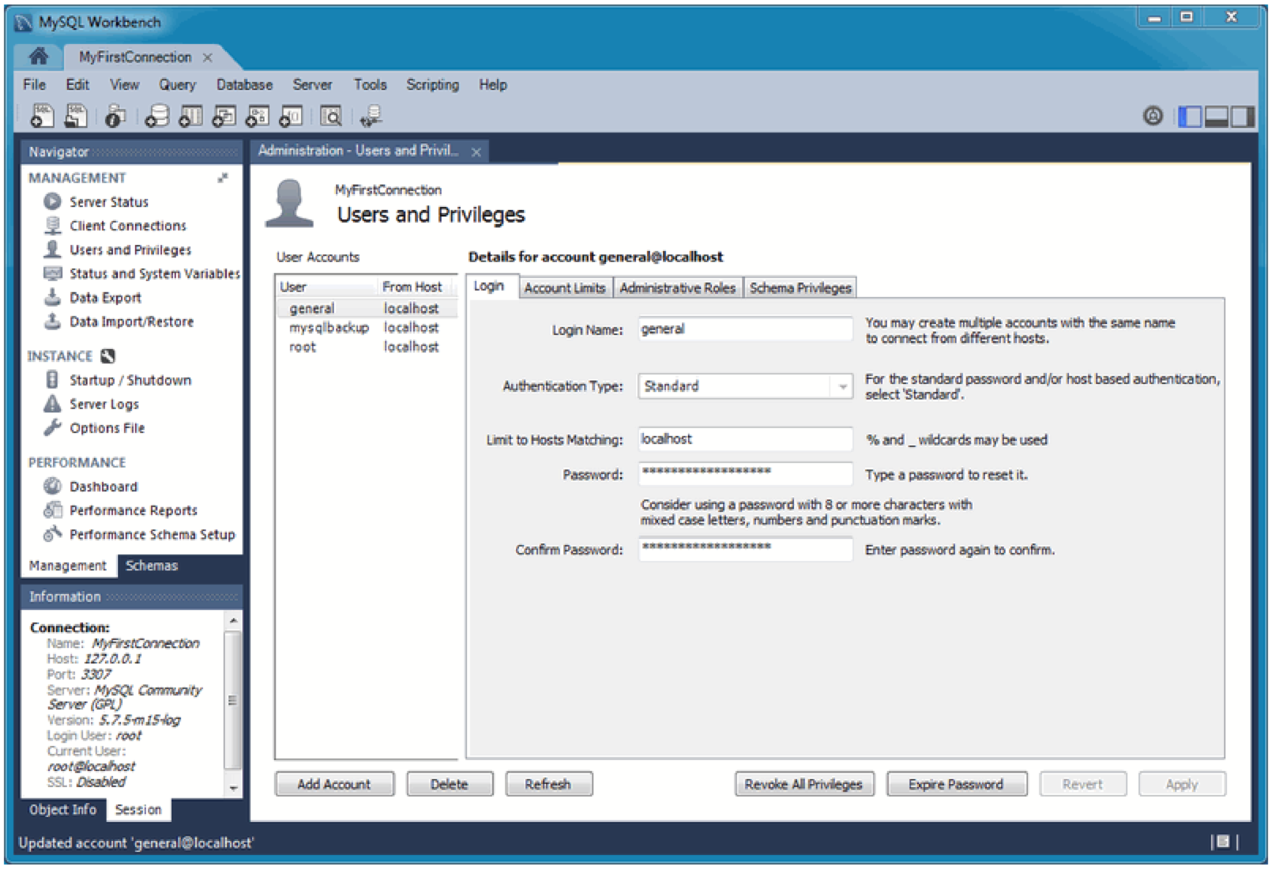
User Accounts

Lists each user account that is associated to the active MySQL connection.

Login

Login information related to the selected user account.

Figure 6.6 Navigator Management: User And Privileges: Login



MyFirstConnection

Users and Privileges

User Accounts

User	From Host
general	localhost
mysqlbackup	localhost
root	localhost

Details for account general@localhost

Login Name: general — You may create multiple accounts with the same name to connect from different hosts.

Authentication Type: Standard — For the standard password and/or host based authentication, select 'Standard'.

Limit to Hosts Matching: localhost — % and _ wildcards may be used

Password: ****************** — Type a password to reset it.

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: ****************** — Enter password again to confirm.

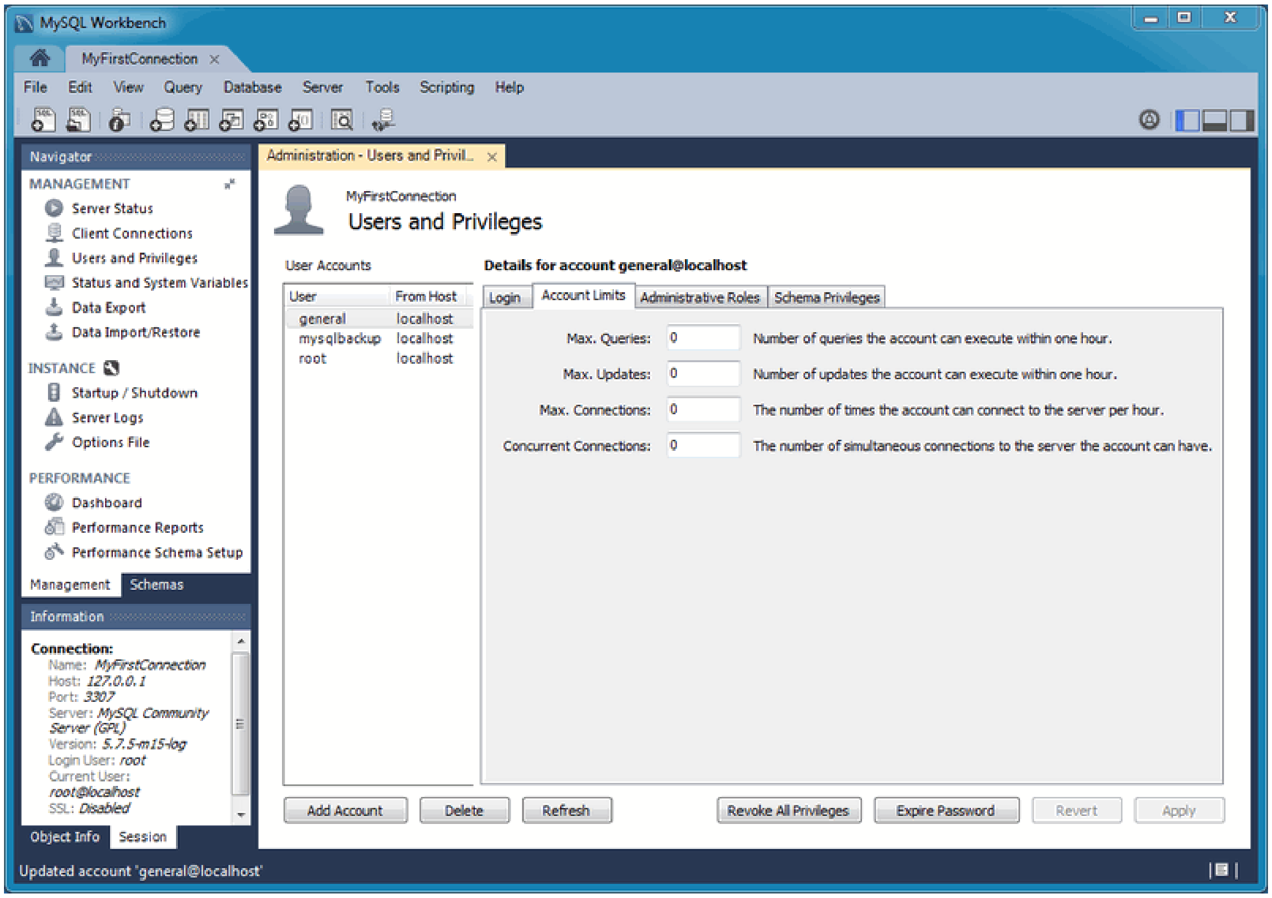
Connection:
Name: *MyFirstConnection*
Host: *127.0.0.1*
Port: *3307*
Server: *MySQL Community Server (GPL)*
Version: *5.7.5-m15-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Disabled*

Updated account 'general@localhost'

Account Limits

Define limits for the user account, such as the maximum number of queries, updates, connections, and concurrent connections that an account can execute in one hour.

Figure 6.7 Navigator Management: User And Privileges: Account Limits



User	From Host
general	localhost
mysqlbackup	localhost
root	localhost

Max. Queries: 0 Number of queries the account can execute within one hour.

Max. Updates: 0 Number of updates the account can execute within one hour.

Max. Connections: 0 The number of times the account can connect to the server per hour.

Concurrent Connections: 0 The number of simultaneous connections to the server the account can have.

Connection:
Name: *MyFirstConnection*
Host: *127.0.0.1*
Port: *3307*
Server: *MySQL Community Server (GPL)*
Version: *5.7.5-m15-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Disabled*

Updated account 'general@localhost'

Administrative Roles

To aid in assigning privileges to MySQL Server users, MySQL Workbench introduces the concept of Administrative Roles. Roles are a quick way of granting a set of privileges to a user, based on the work the user must carry out on the server. It is also possible to assign multiple roles to a user. To assign roles, click the User Account you wish to modify, then click the **Administrative Roles** tab. Then click the check boxes according to the roles you wish to allocate to the user. After you select a role to a user, you will see the accumulated privileges in the **Global Privileges Assigned to User** panel. For example, if you select the role `BackupAdmin`, the privileges granted include `EVENT`, `LOCK TABLES`, `SELECT`, `SHOW DATABASES`. If you also select the role of `ReplicationAdmin`, the list of privileges expands to include `REPLICATION CLIENT`, `REPLICATION SLAVE` and `SUPER`.

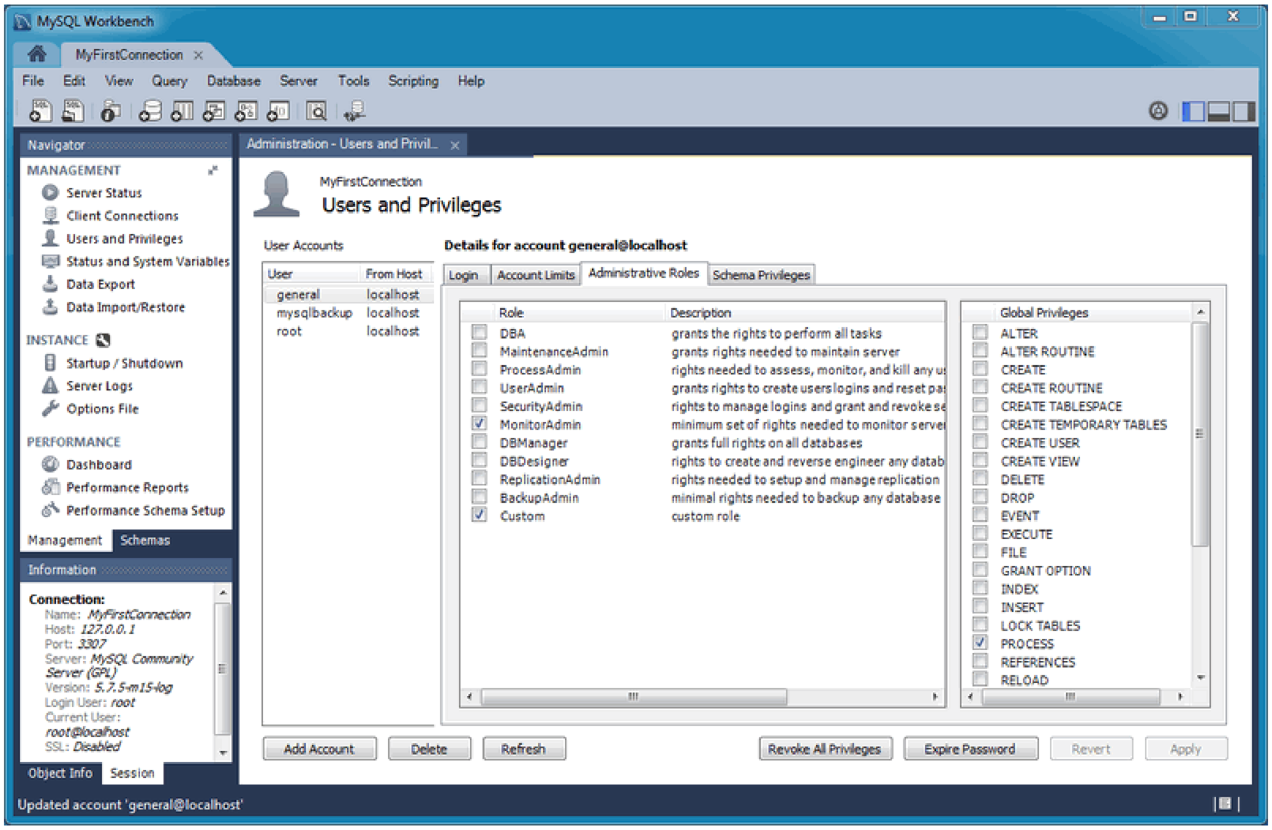
These roles are available:

- **DBA**: Grants all privileges
- **MaintenanceAdmin**: Grants privileges to maintain the server
- **ProcessAdmin**: Grants privileges to monitor and kill user processes
- **UserAdmin**: Grants privileges to create users and reset passwords
- **SecurityAdmin**: Grants privileges to manage logins and grant and revoke server privileges
- **MonitorAdmin**: Grants privileges to monitor the server

- **DBManager**: Grants privileges to manage databases
- **DBDesigner**: Grants privileges to create and reverse engineer any database schema
- **ReplicationAdmin**: Grants privileges to set up and manage replication
- **BackupAdmin**: Grants privileges required to back up databases
- **Custom**: Lists other (custom) privileges that are assigned to the user account

The *Password Validation Plugin* (available as of MySQL Server 5.6.6) is supported in Workbench. For information about what these settings mean, see The Password Validation Plugin.

Figure 6.8 Navigator Management: User And Privileges: Administrative Roles



User Accounts

User	From Host
general	localhost
mysqlbackup	localhost
root	localhost

Details for account general@localhost

	Role	Description
☐	DBA	grants the rights to perform all tasks
☐	MaintenanceAdmin	grants rights needed to maintain server
☐	ProcessAdmin	rights needed to assess, monitor, and kill any us
☐	UserAdmin	grants rights to create users logins and reset pas
☐	SecurityAdmin	rights to manage logins and grant and revoke se
✓	MonitorAdmin	minimum set of rights needed to monitor server
☐	DBManager	grants full rights on all databases
☐	DBDesigner	rights to create and reverse engineer any datab
☐	ReplicationAdmin	rights needed to setup and manage replication
☐	BackupAdmin	minimal rights needed to backup any database
✓	Custom	custom role

	Global Privileges
☐	ALTER
☐	ALTER ROUTINE
☐	CREATE
☐	CREATE ROUTINE
☐	CREATE TABLESPACE
☐	CREATE TEMPORARY TABLES
☐	CREATE USER
☐	CREATE VIEW
☐	DELETE
☐	DROP
☐	EVENT
☐	EXECUTE
☐	FILE
☐	GRANT OPTION
☐	INDEX
☐	INSERT
☐	LOCK TABLES
✓	PROCESS
☐	REFERENCES
☐	RELOAD

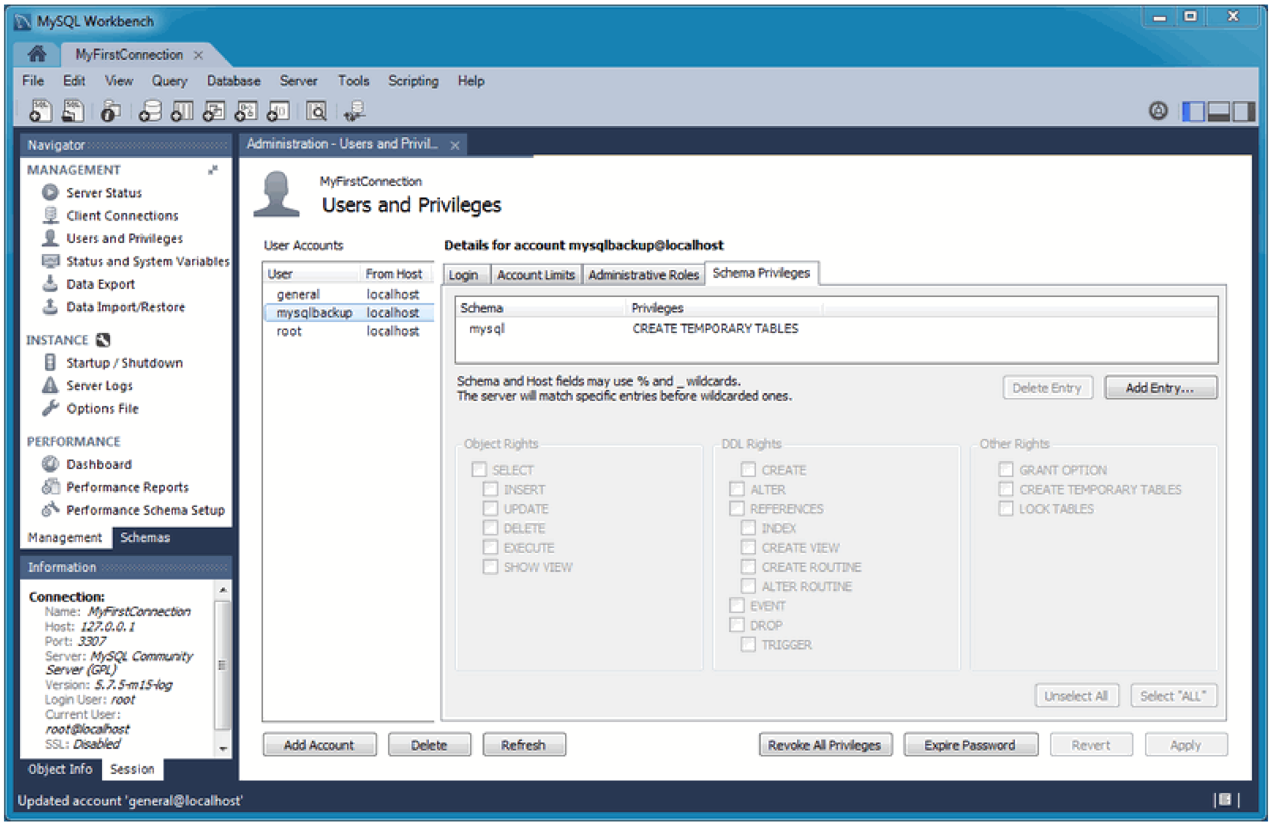
Connection:
Name: *MyFirstConnection*
Host: *127.0.0.1*
Port: *3307*
Server: *MySQL Community Server (GPL)*
Version: *5.7.5-m15-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Disabled*

Updated account 'general@localhost'

Schema Privileges

Additional schema privileges that the account can use. For example, the standard `mysqlbackup` user has "CREATE TEMPORARY TABLES" on the `mysql` schema.

Figure 6.9 Navigator Management: User And Privileges: Schema Privileges



Connection:
Name: *MyFirstConnection*
Host: *127.0.0.1*
Port: *3307*
Server: *MySQL Community Server (GPL)*
Version: *5.7.5-m15-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Disabled*

User	From Host
general	localhost
mysqlbackup	localhost
root	localhost

Details for account mysqlbackup@localhost

Schema	Privileges
mysql	CREATE TEMPORARY TABLES

Schema and Host fields may use % and _ wildcards.
The server will match specific entries before wildcarded ones.

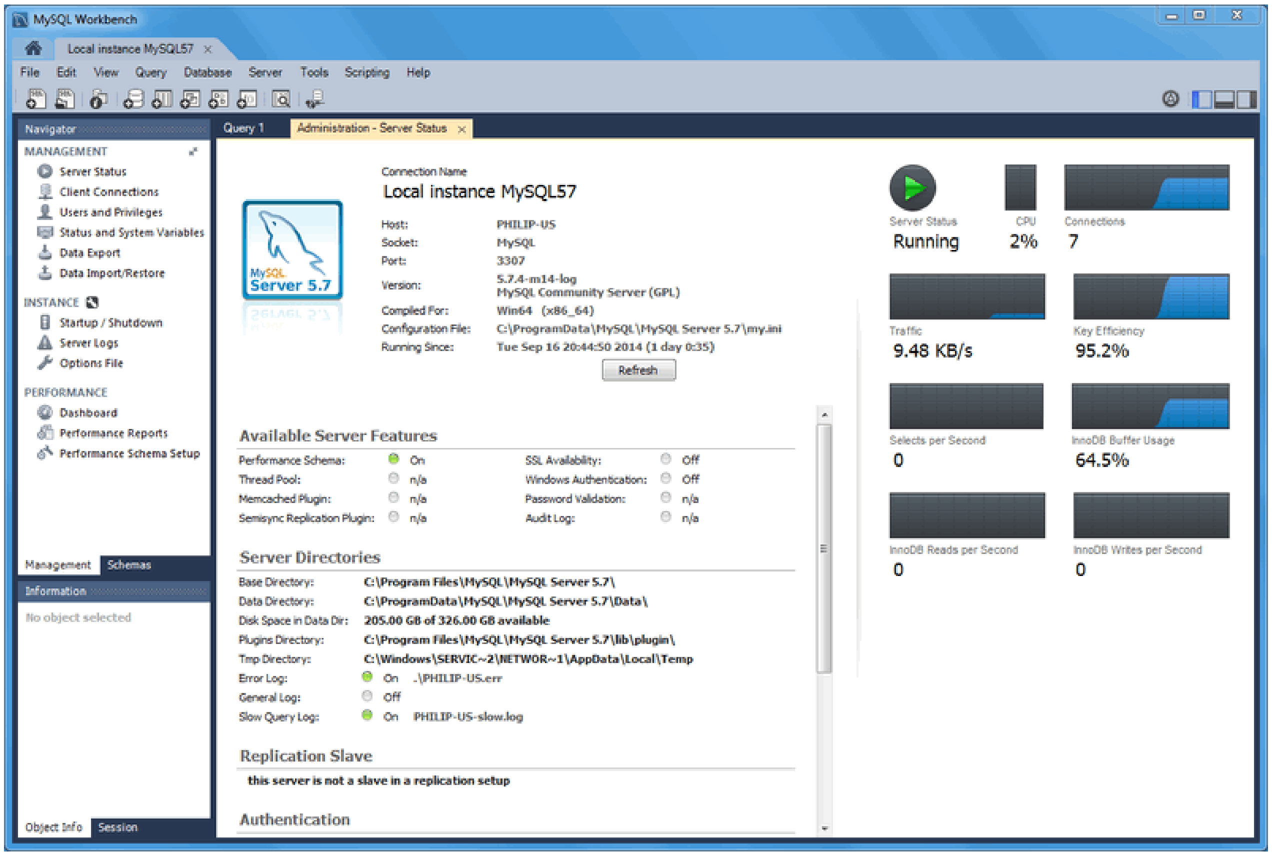
Updated account 'general@localhost'

6.3 Server Status

Get an immediate view into the basic health indicators and counters for your MySQL environment. This includes viewing the server's running state (stopped/running), available features, primary server directories, replication state, and security settings for authentication and SSL. Reports also include information and graphs to track memory usage, connections, hit rates, and more.

Access this window from either the Navigator panel, or by selecting **Server**, **Server Status** from the main menu.

Figure 6.10 Navigator Management: Server Status



Connection Name
Local instance MySQL57

Host:	PHILIP-US
Socket:	MySQL
Port:	3307
Version:	5.7.4-m14-log MySQL Community Server (GPL)
Compiled For:	Win64 (x86_64)
Configuration File:	C:\ProgramData\MySQL\MySQL Server 5.7\my.ini
Running Since:	Tue Sep 16 20:44:50 2014 (1 day 0:35)

Available Server Features

Performance Schema:	On	SSL Availability:	Off
Thread Pool:	n/a	Windows Authentication:	Off
Memcached Plugin:	n/a	Password Validation:	n/a
Semisync Replication Plugin:	n/a	Audit Log:	n/a

Server Directories

Base Directory:	C:\Program Files\MySQL\MySQL Server 5.7\
Data Directory:	C:\ProgramData\MySQL\MySQL Server 5.7\Data\
Disk Space in Data Dir:	205.00 GB of 326.00 GB available
Plugins Directory:	C:\Program Files\MySQL\MySQL Server 5.7\lib\plugin\
Tmp Directory:	C:\Windows\SERVIC~2\NETWOR~1\AppData\Local\Temp
Error Log:	On .\PHILIP-US.err
General Log:	Off
Slow Query Log:	On PHILIP-US-slow.log

Replication Slave

this server is not a slave in a replication setup

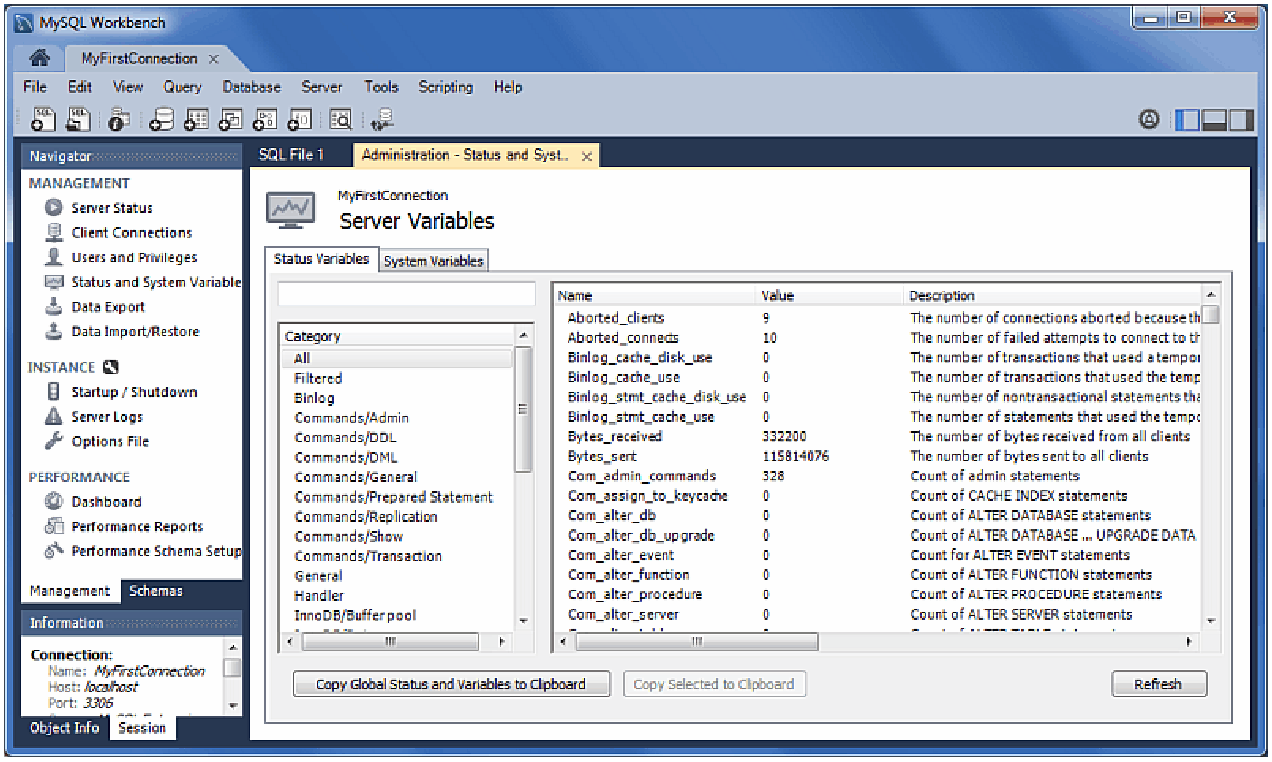
Authentication

6.4 Status and System Variables

Status and System Variables: Lists all server variables for the MySQL connection. You may also copy all or selected variables to your clipboard.

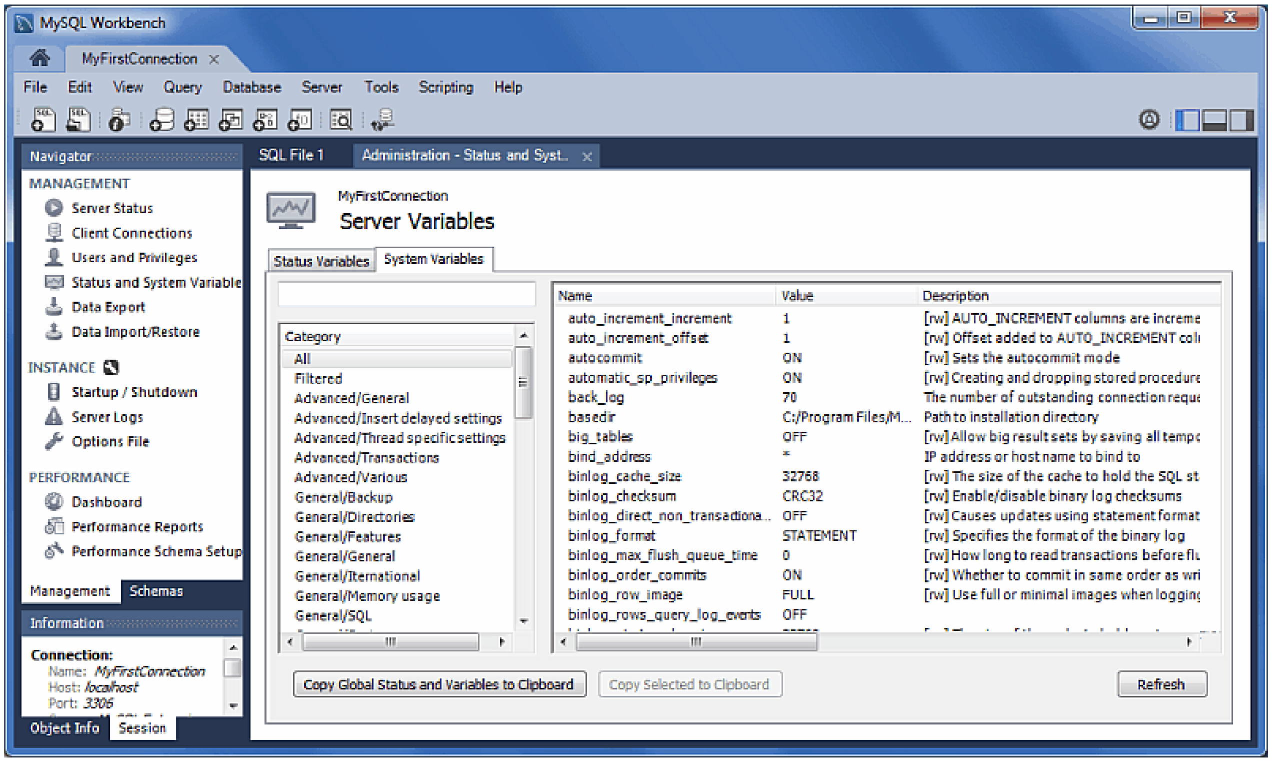
Access this window from either the Navigator panel, or by selecting **Server**, **Status and System Variables** from the main menu.

Figure 6.11 Navigator Management: Status Variables



Name	Value	Description
Aborted_clients	9	The number of connections aborted because th
Aborted_connects	10	The number of failed attempts to connect to th
Binlog_cache_disk_use	0	The number of transactions that used a tempor
Binlog_cache_use	0	The number of transactions that used the temp
Binlog_stmt_cache_disk_use	0	The number of nontransactional statements tha
Binlog_stmt_cache_use	0	The number of statements that used the tempo
Bytes_received	332200	The number of bytes received from all clients
Bytes_sent	115814076	The number of bytes sent to all clients
Com_admin_commands	328	Count of admin statements
Com_assign_to_keycache	0	Count of CACHE INDEX statements
Com_alter_db	0	Count of ALTER DATABASE statements
Com_alter_db_upgrade	0	Count of ALTER DATABASE ... UPGRADE DATA
Com_alter_event	0	Count for ALTER EVENT statements
Com_alter_function	0	Count of ALTER FUNCTION statements
Com_alter_procedure	0	Count of ALTER PROCEDURE statements
Com_alter_server	0	Count of ALTER SERVER statements

Figure 6.12 Navigator Management: Status Variables



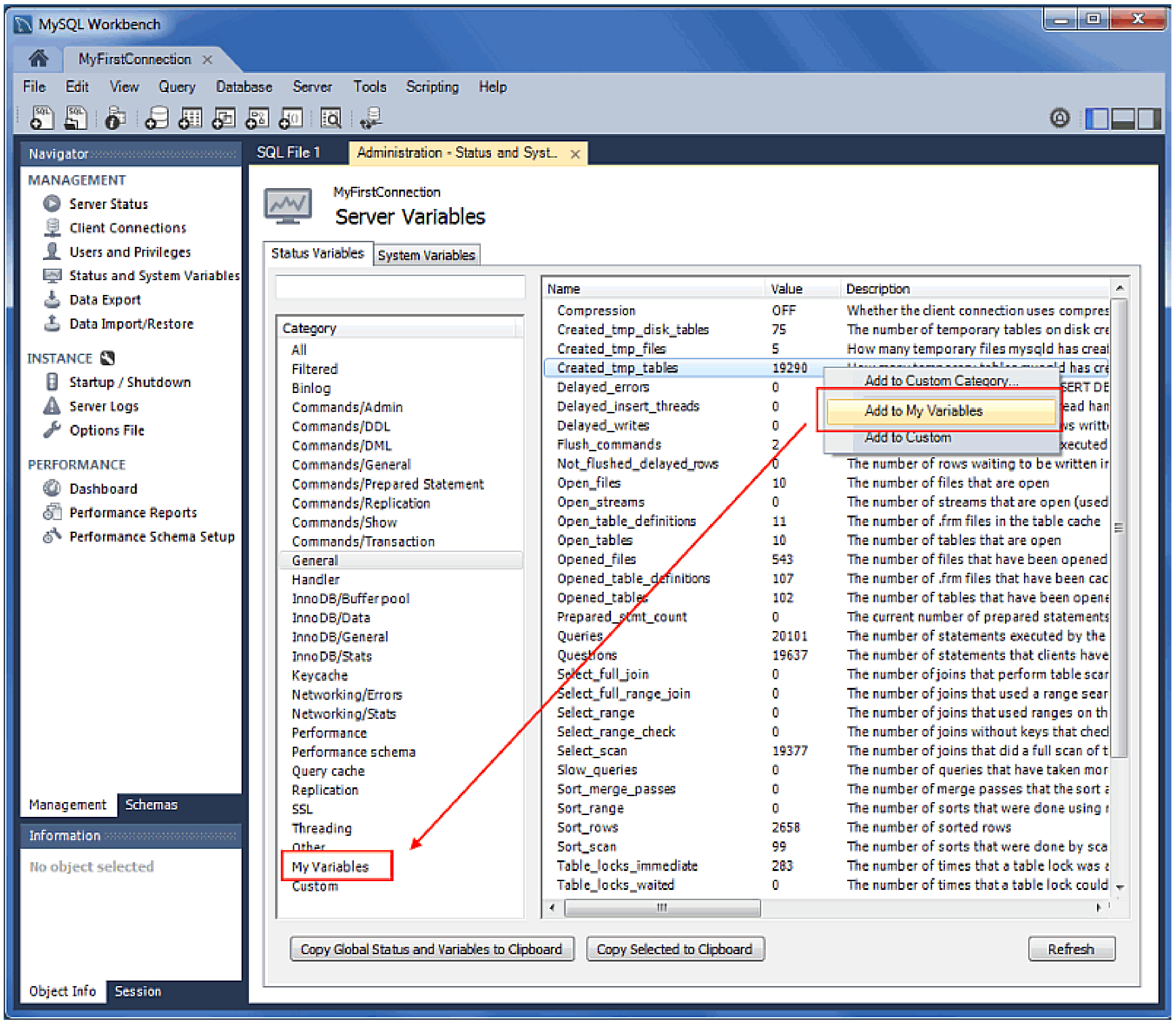
Name	Value	Description
auto_increment_increment	1	[rw] AUTO_INCREMENT columns are increme
auto_increment_offset	1	[rw] Offset added to AUTO_INCREMENT colu
autocommit	ON	[rw] Sets the autocommit mode
automatic_sp_privileges	ON	[rw] Creating and dropping stored procedure
back_log	70	The number of outstanding connection reque
basedir	C:/Program Files/M...	Path to installation directory
big_tables	OFF	[rw] Allow big result sets by saving all tempo
bind_address	*	IP address or host name to bind to
binlog_cache_size	32768	[rw] The size of the cache to hold the SQL st
binlog_checksum	CRC32	[rw] Enable/disable binary log checksums
binlog_direct_non_transactiona...	OFF	[rw] Causes updates using statement format
binlog_format	STATEMENT	[rw] Specifies the format of the binary log
binlog_max_flush_queue_time	0	[rw] How long to read transactions before flu
binlog_order_commits	ON	[rw] Whether to commit in same order as wri
binlog_row_image	FULL	[rw] Use full or minimal images when logging
binlog_rows_query_log_events	OFF	

Custom Variable Grouping

The status and system variables are each categorized by groups (such as InnoDB or Logging), and you may also create your own custom groups. Right-click on a variable, choose the custom group (or create a new one), and add the variable to the aforementioned group. Your custom groups are listed along with the pre-existing groups.

In the example below, we created a custom group titled "My Variables" and are in the process of adding the `Created_tmp_variables` variable to it.

Figure 6.13 Navigator Management: Adding A Variable To A Custom Group



Name	Value	Description
Compression	OFF	Whether the client connection uses compres
Created_tmp_disk_tables	75	The number of temporary tables on disk cre
Created_tmp_files	5	How many temporary files mysqld has creat
Created_tmp_tables	19290	d has cre
Delayed_errors	0	SERT DE
Delayed_insert_threads	0	ead han
Delayed_writes	0	ws writt
Flush_commands	2	xecuted
Not_flushed_delayed_rows	0	The number of rows waiting to be written ir
Open_files	10	The number of files that are open
Open_streams	0	The number of streams that are open (used
Open_table_definitions	11	The number of .frm files in the table cache
Open_tables	10	The number of tables that are open
Opened_files	543	The number of files that have been opened
Opened_table_definitions	107	The number of .frm files that have been cac
Opened_tables	102	The number of tables that have been opene
Prepared_stmt_count	0	The current number of prepared statements
Queries	20101	The number of statements executed by the
Questions	19637	The number of statements that clients have
Select_full_join	0	The number of joins that perform table scar
Select_full_range_join	0	The number of joins that used a range sear
Select_range	0	The number of joins that used ranges on th
Select_range_check	0	The number of joins without keys that checl
Select_scan	19377	The number of joins that did a full scan of t
Slow_queries	0	The number of queries that have taken mor
Sort_merge_passes	0	The number of merge passes that the sort a
Sort_range	0	The number of sorts that were done using r
Sort_rows	2658	The number of sorted rows
Sort_scan	99	The number of sorts that were done by sca
Table_locks_immediate	283	The number of times that a table lock was a
Table_locks_waited	0	The number of times that a table lock could

6.5 Data Export and Import

There are three ways to export and import data in MySQL Workbench, each serving a different purpose.

Table 6.1 Methods to Export or Import data in MySQL Workbench

GUI Location	Data Set	Export Types	Import Types	Additional Details
Object Browser context menu	Tables	JSON, CSV	JSON, CSV	Simple table operations, includes moderate control over the output type (this method was added in version 6.3.0)
Result Grid menu under the SQL editor	Result set (after performing an SQL query)	CSV, HTML, JSON, SQL, XML, Excel XML, TXT	CSV	Simple data operations, includes little control
Management Navigator	Databases and/or Tables	SQL	SQL	Detailed database and table operations, standard backup/ restore behavior using the `mysqldump` command and meta data, includes control over how data is handled, and includes meta data
Management Navigator	Databases and/or Tables	SQL	SQL	Detailed database and table operations, includes control over how data is handled, can be scheduled and incremental, includes meta data, uses MySQL Enterprise Backup (commercial)

6.5.1 Table Data Export and Import Wizard

This wizard supports import and export operations using CSV and JSON files, and includes several configuration options (separators, column selection, encoding selection, and more). The wizard can be performed against local or remotely connected MySQL servers, and the import action includes table, column, and type mapping.

Note

This wizard only exports/imports tables using the JSON or CSV format. For an overview of the data export and import options in MySQL Workbench, see Section 6.5, “Data Export and Import”.

The wizard is accessible from the object browser's context menu by right-clicking on a table and choose either **Table Data Export Wizard** or **Table Data Import Wizard**.

Figure 6.14 Table Data Wizards: Open

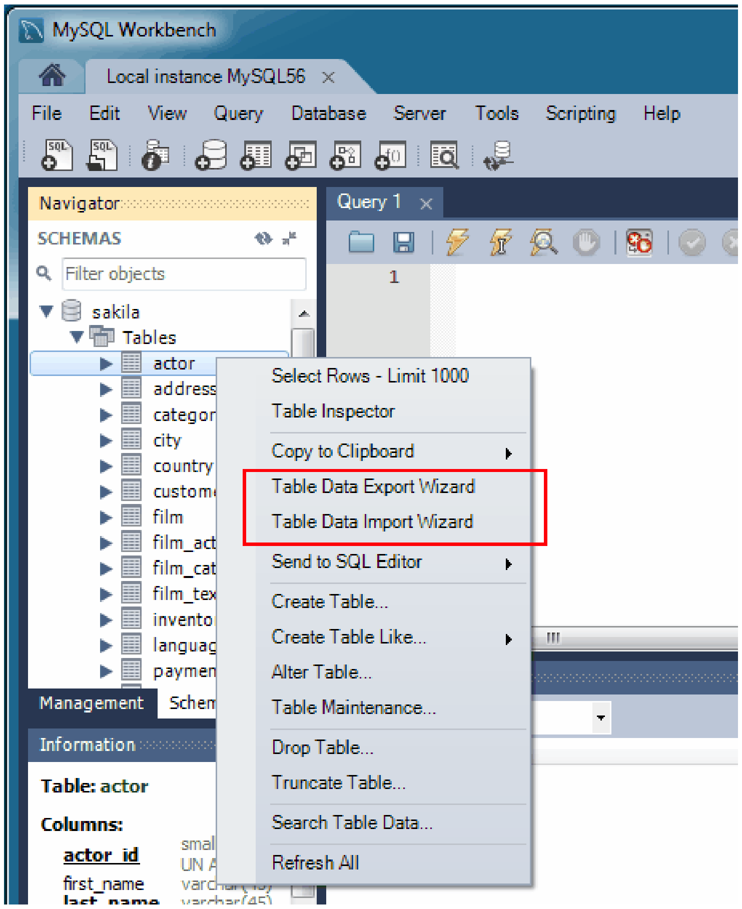


Table Data Export Wizard

Export table data to either a JSON or CSV file. The following example exports the *sakila.actor* table to a CSV file.

Figure 6.15 Table Data Export: Source

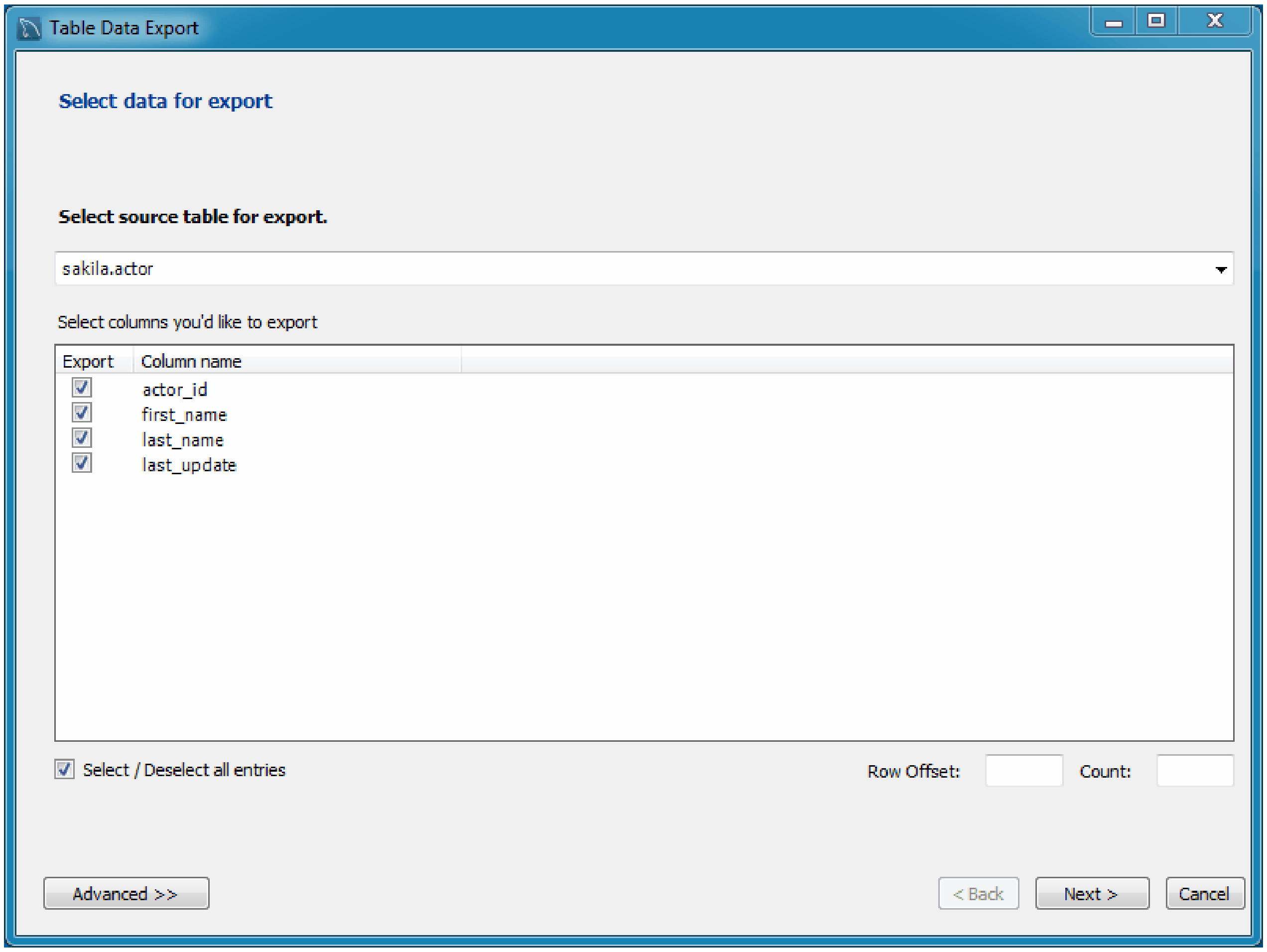


Table Data Export

Select data for export

Select source table for export.

sakila.actor

Select columns you'd like to export

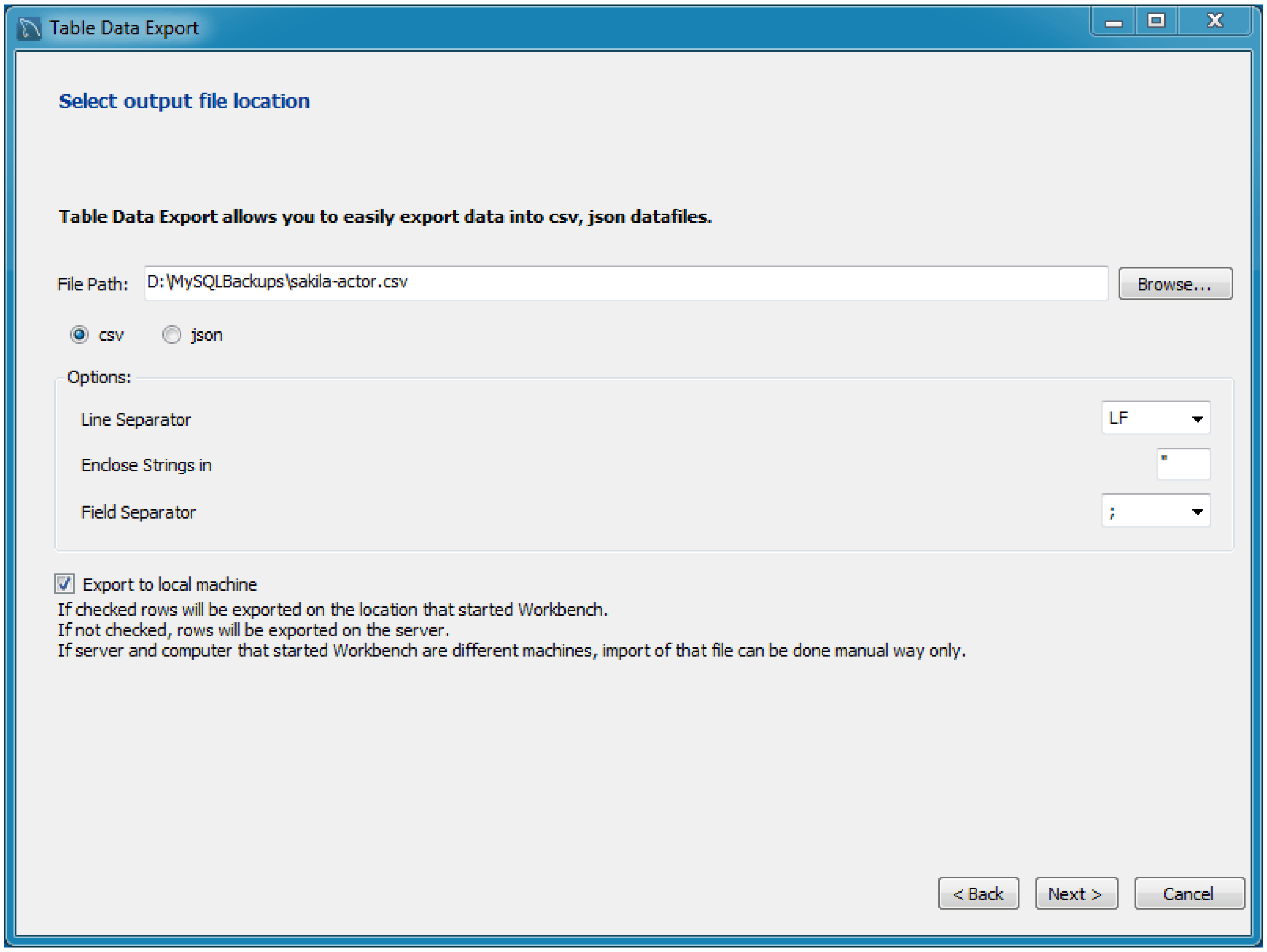
Export	Column name
✓	actor_id
✓	first_name
✓	last_name
✓	last_update

✓ Select / Deselect all entries

Row Offset:

Count:

Figure 6.16 Table Data Export: CSV Configuration



Select output file location

Table Data Export allows you to easily export data into csv, json datafiles.

File Path: D:\MySQLBackups\sakila-actor.csv

[✓ csv] json

Options:

Line Separator: LF

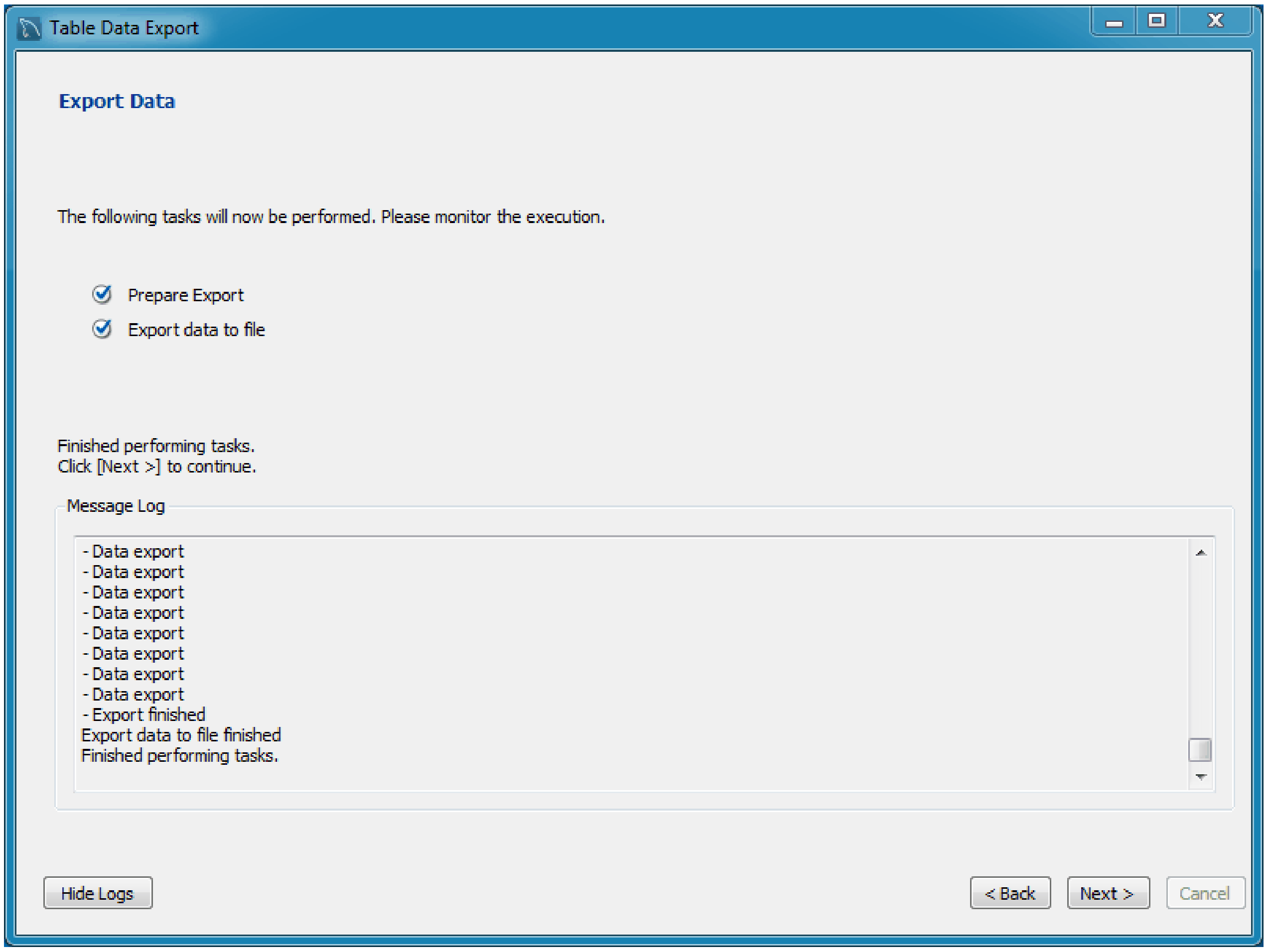
Enclose Strings in: "

Field Separator: ;

[✓] Export to local machine

If checked rows will be exported on the location that started Workbench.
If not checked, rows will be exported on the server.
If server and computer that started Workbench are different machines, import of that file can be done manual way only.

Figure 6.17 Table Data Export: Results



The following tasks will now be performed. Please monitor the execution.

- ✓ Prepare Export
- ✓ Export data to file

Finished performing tasks.
Click [Next >] to continue.

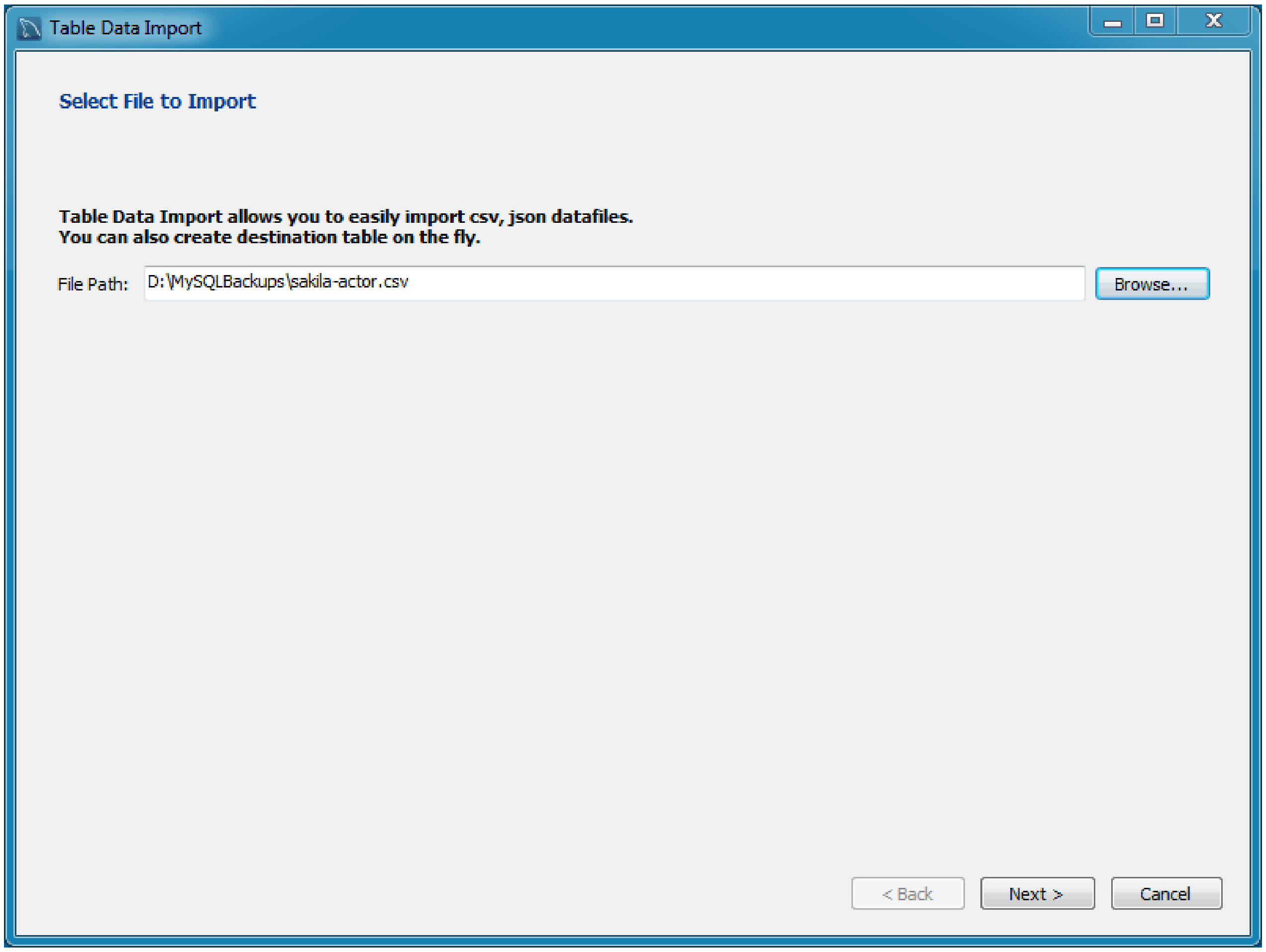
Message Log

```
- Data export
- Data export
- Data export
- Data export
- Data export
- Data export
- Data export
- Data export
- Export finished
Export data to file finished
Finished performing tasks.
```

Table Data Import Wizard

Import table data from either a JSON or CSV file. The following example imports the *sakila.actor* table from a CSV file.

Figure 6.18 Table Data Import: CSV Source

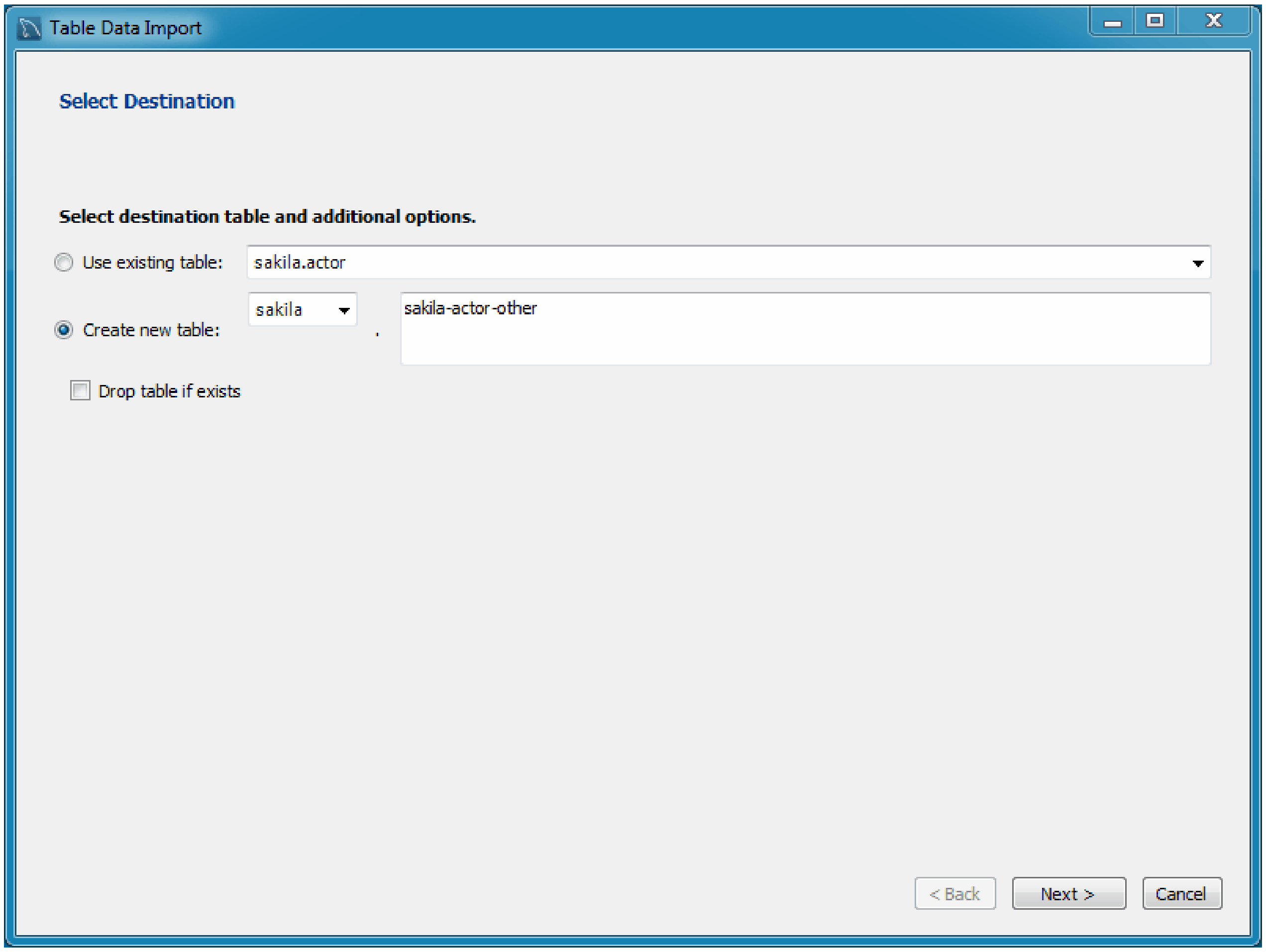


Select File to Import

Table Data Import allows you to easily import csv, json datafiles.
You can also create destination table on the fly.

File Path: D:\MySQLBackups\sakila-actor.csv

Figure 6.19 Table Data Import: Destination Table



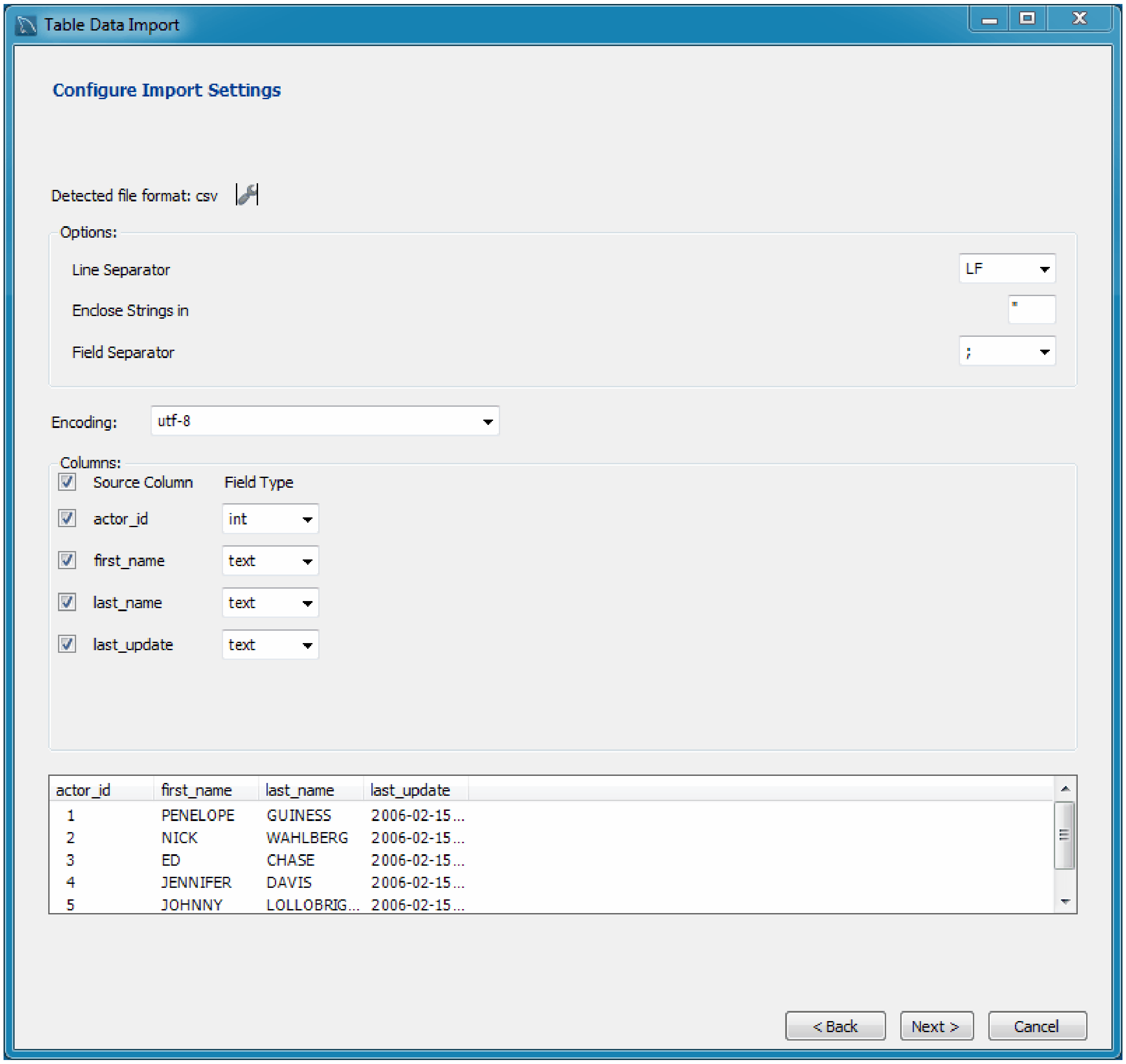
Select destination table and additional options.

Use existing table: sakila.actor

[✓ Create new table:] sakila . sakila-actor-other

☐ Drop table if exists

Figure 6.20 Table Data Import: CSV Configuration



Configure Import Settings

Detected file format: csv

Options:

Line Separator	LF
Enclose Strings in	"
Field Separator	;

Encoding: utf-8

Columns:

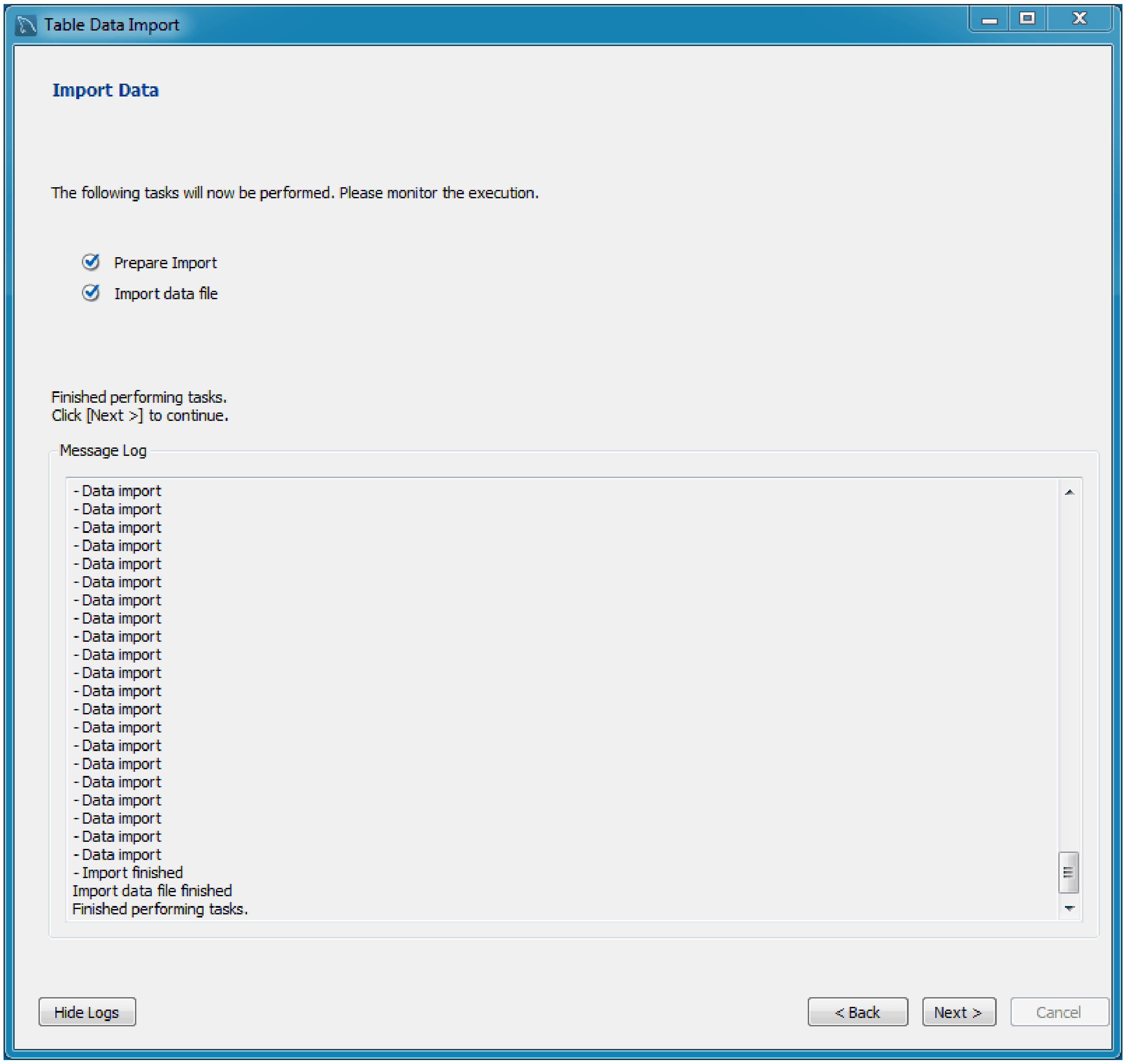
✓ Source Column	Field Type
✓ actor_id	int
✓ first_name	text
✓ last_name	text
✓ last_update	text

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15...
2	NICK	WAHLBERG	2006-02-15...
3	ED	CHASE	2006-02-15...
4	JENNIFER	DAVIS	2006-02-15...
5	JOHNNY	LOLLOBRIG...	2006-02-15...

Note

The **Encoding** field should correspond with your CSV file.

Figure 6.21 Table Data Import: Results



The following tasks will now be performed. Please monitor the execution.

✓ Prepare Import
✓ Import data file

Finished performing tasks.
Click [Next >] to continue.

Message Log

```
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Data import
- Import finished
Import data file finished
Finished performing tasks.
```

6.5.2 SQL Data Export and Import Wizard

Use this wizard to either export or import SQL generated from MySQL Workbench or with the `mysqldump` command.

Access these wizards from either the Navigator panel, or by selecting **Server** from the main menu, and then either **Data Import** or **Data Export**.

Note

This wizard only exports/imports the MySQL SQL format. For an overview of the data export and import options in MySQL Workbench, see Section 6.5, “Data Export and Import”.

Data Export

This tab allows you to export your MySQL data. Select each schema you want to export, optionally choose specific schema objects/tables from each schema, and generate the export. Configuration options include exporting to a project folder or self-contained SQL file, optionally dump stored routines and events, or skip table data.

Note

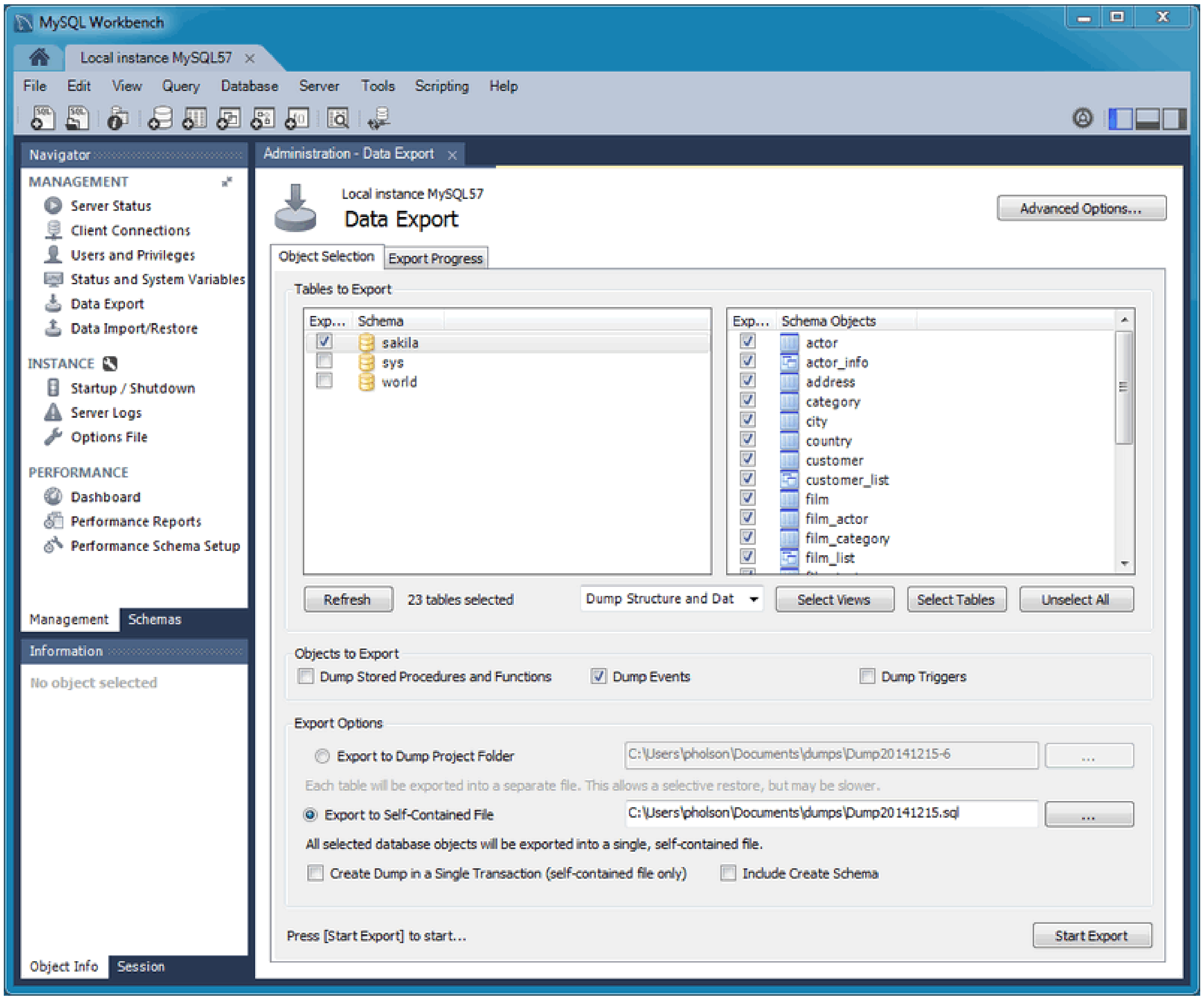
Alternatively, use Export a Result Set to export a specific result set in the SQL editor to another format such as CSV, JSON, HTML, and XML.

Select the Database objects to export, and configure the related options.

Note

Click **Refresh** to load the current objects.

Figure 6.22 Navigator Management: Data Export: Object Selection



Tables to Export

Exp...	Schema
✓	sakila
	sys
	world

Exp...	Schema Objects
✓	actor
✓	actor_info
✓	address
✓	category
✓	city
✓	country
✓	customer
✓	customer_list
✓	film
✓	film_actor
✓	film_category
✓	film_list

23 tables selected

Dump Structure and Dat

Objects to Export

Dump Stored Procedures and Functions [ ] Dump Events [✓] Dump Triggers [ ]

Export Options

Export to Dump Project Folder: C:\Users\pholson\Documents\dumps\Dump20141215-6

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

[✓ Export to Self-Contained File]: C:\Users\pholson\Documents\dumps\Dump20141215.sql

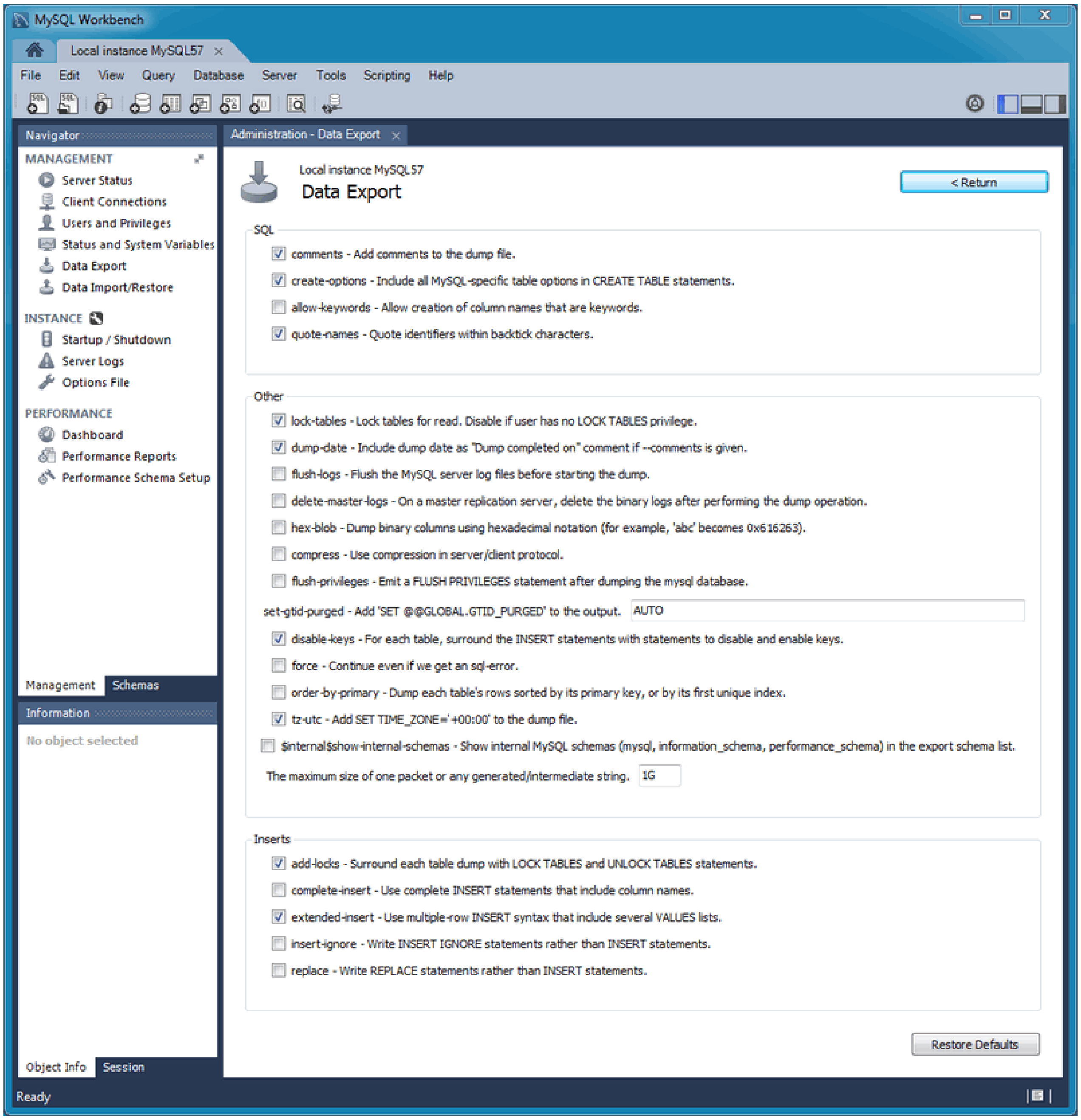
All selected database objects will be exported into a single, self-contained file.

Create Dump in a Single Transaction (self-contained file only) [ ] Include Create Schema [ ]

Press [Start Export] to start...

Optionally open the **Advanced Options** tab that allows you to refine the export operation. For example, add table locks, use replace instead of insert statements, quote identifiers with backtick characters, and more.

Figure 6.23 Navigator Management: Data Export: Advanced Options



SQL

- [✓] comments - Add comments to the dump file.
- [✓] create-options - Include all MySQL-specific table options in CREATE TABLE statements.
- [ ] allow-keywords - Allow creation of column names that are keywords.
- [✓] quote-names - Quote identifiers within backtick characters.

Other

- [✓] lock-tables - Lock tables for read. Disable if user has no LOCK TABLES privilege.
- [✓] dump-date - Include dump date as "Dump completed on" comment if --comments is given.
- [ ] flush-logs - Flush the MySQL server log files before starting the dump.
- [ ] delete-master-logs - On a master replication server, delete the binary logs after performing the dump operation.
- [ ] hex-blob - Dump binary columns using hexadecimal notation (for example, 'abc' becomes 0x616263).
- [ ] compress - Use compression in server/client protocol.
- [ ] flush-privileges - Emit a FLUSH PRIVILEGES statement after dumping the mysql database.

set-gtid-purged - Add 'SET @@GLOBAL.GTID_PURGED' to the output. AUTO

- [✓] disable-keys - For each table, surround the INSERT statements with statements to disable and enable keys.
- [ ] force - Continue even if we get an sql-error.
- [ ] order-by-primary - Dump each table's rows sorted by its primary key, or by its first unique index.
- [✓] tz-utc - Add SET TIME_ZONE='+00:00' to the dump file.
- [ ] $internal$show-internal-schemas - Show internal MySQL schemas (mysql, information_schema, performance_schema) in the export schema list.

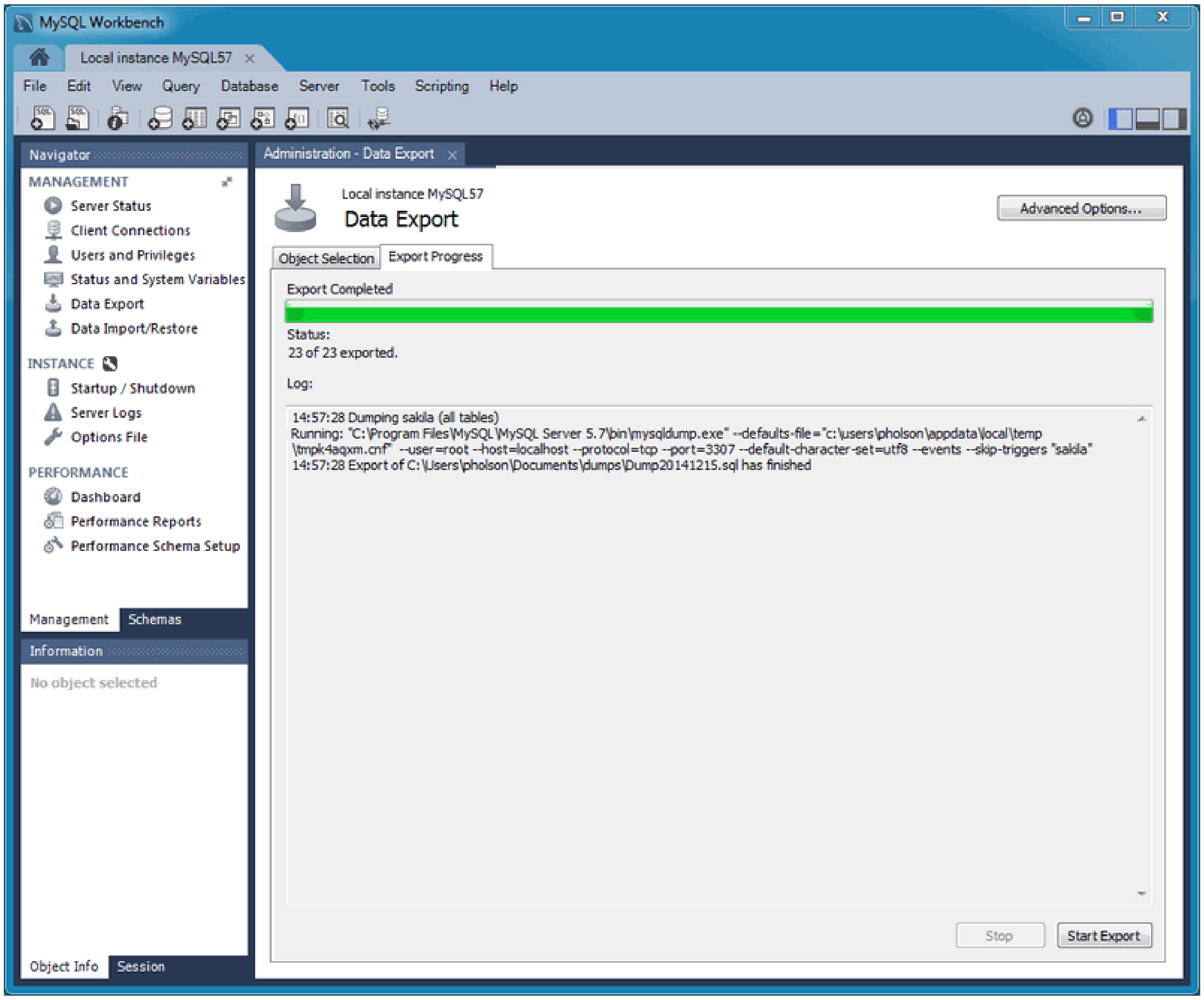
The maximum size of one packet or any generated/intermediate string. 1G

Inserts

- [✓] add-locks - Surround each table dump with LOCK TABLES and UNLOCK TABLES statements.
- [ ] complete-insert - Use complete INSERT statements that include column names.
- [✓] extended-insert - Use multiple-row INSERT syntax that include several VALUES lists.
- [ ] insert-ignore - Write INSERT IGNORE statements rather than INSERT statements.
- [ ] replace - Write REPLACE statements rather than INSERT statements.

Click **Start Export** to begin the export process:

Figure 6.24 Navigator Management: Data Export: Export Progress



Export Completed

Status:
23 of 23 exported.

Log:

```
14:57:28 Dumping sakila (all tables)
Running: "C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqldump.exe" --defaults-file="c:\users\pholson\appdata\local\temp
\tmpk4aqxm.cnf"  --user=root --host=localhost --protocol=tcp --port=3307 --default-character-set=utf8 --events --skip-triggers "sakila"
14:57:28 Export of C:\Users\pholson\Documents\dumps\Dump20141215.sql has finished
```

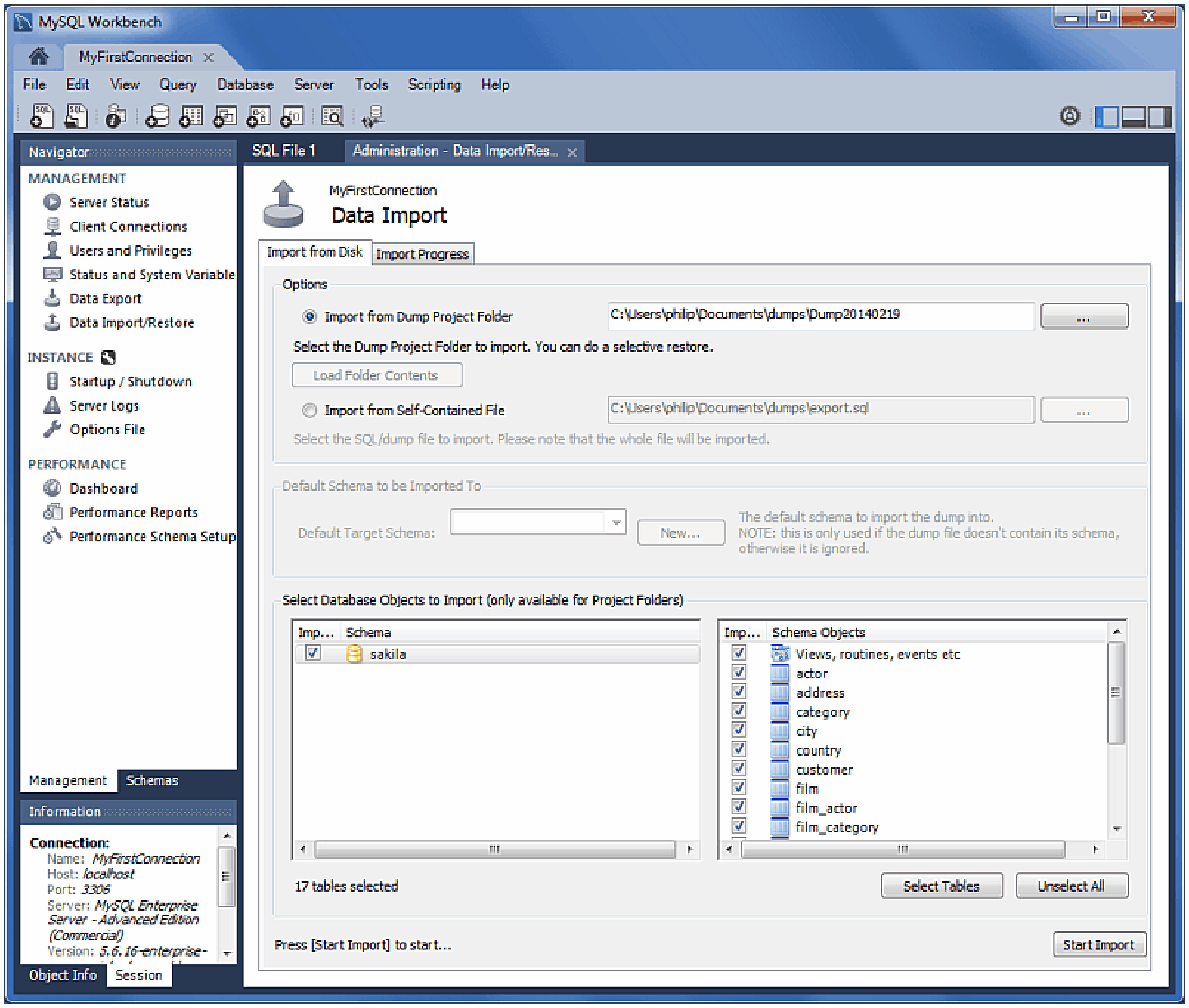
This functionality uses the `mysqldump` command.

Data Import/Restore

Restore exported data from the **Data Export** operation, or from other exported data from the `mysqldump` command.

Choose the project folder or self-contained SQL file, choose the schema that the data will be imported to, or choose **New** to define a new schema.

Figure 6.25 Navigator Management: Data Import: Import From Disk



MyFirstConnection

Data Import

Options

[✓ Import from Dump Project Folder] C:\Users\philip\Documents\dumps\Dump20140219

Select the Dump Project Folder to import. You can do a selective restore.

Import from Self-Contained File C:\Users\philip\Documents\dumps\export.sql

Select the SQL/dump file to import. Please note that the whole file will be imported.

Default Schema to be Imported To

Default Target Schema:

The default schema to import the dump into.
NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.

Select Database Objects to Import (only available for Project Folders)

Imp...	Schema
✓	sakila

Imp...	Schema Objects
✓	Views, routines, events etc
✓	actor
✓	address
✓	category
✓	city
✓	country
✓	customer
✓	film
✓	film_actor
✓	film_category

17 tables selected

Press [Start Import] to start...

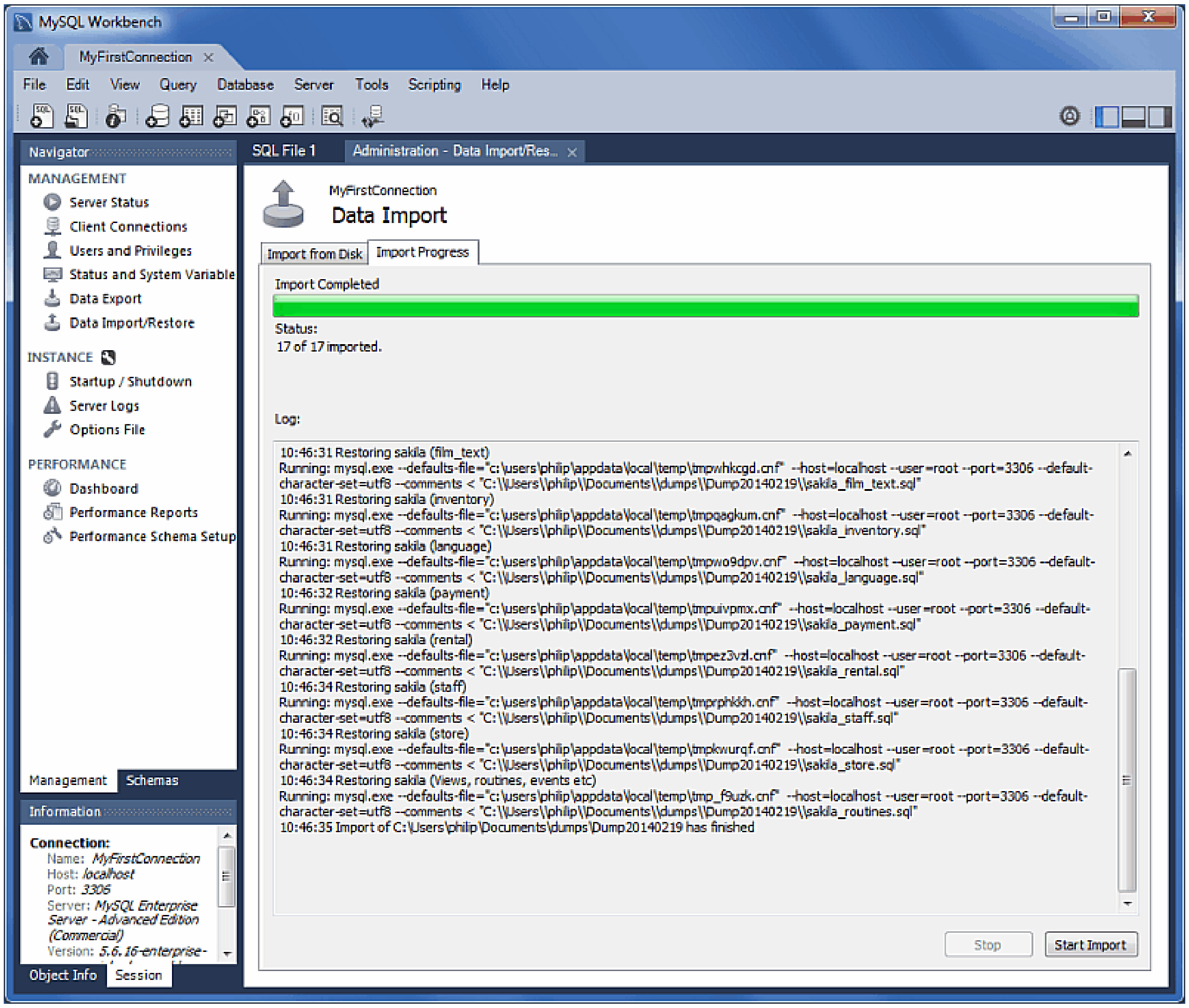
Connection:
Name: *MyFirstConnection*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.16-enterprise-*

Note

You may only select specific data objects (tables) to import if the data export operation used project folders instead of a self-contained SQL file.

Click **Start Import** to begin the import process:

Figure 6.26 Navigator Management: Data Import: Import Progress



Connection:
Name: *MyFirstConnection*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.16-enterprise-*

MyFirstConnection

Import Completed

Status:
17 of 17 imported.

Log:

```
10:46:31 Restoring sakila (film_text)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmpwhkcgd.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_film_text.sql"
10:46:31 Restoring sakila (inventory)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmpqagkum.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_inventory.sql"
10:46:31 Restoring sakila (language)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmpwo9dpv.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_language.sql"
10:46:32 Restoring sakila (payment)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmpuivpmx.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_payment.sql"
10:46:32 Restoring sakila (rental)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmpez3vzl.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_rental.sql"
10:46:34 Restoring sakila (staff)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmprphkkh.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_staff.sql"
10:46:34 Restoring sakila (store)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmpkwurqf.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_store.sql"
10:46:34 Restoring sakila (Views, routines, events etc)
Running: mysql.exe --defaults-file="c:\users\philip\appdata\local\temp\tmp_f9uzk.cnf"  --host=localhost --user=root --port=3306 --default-
character-set=utf8 --comments < "C:\\Users\\philip\\Documents\\dumps\\Dump20140219\\sakila_routines.sql"
10:46:35 Import of C:\Users\philip\Documents\dumps\Dump20140219 has finished
```

6.5.3 Result Data Export and Import

Export or Import data directly from or into the result set.

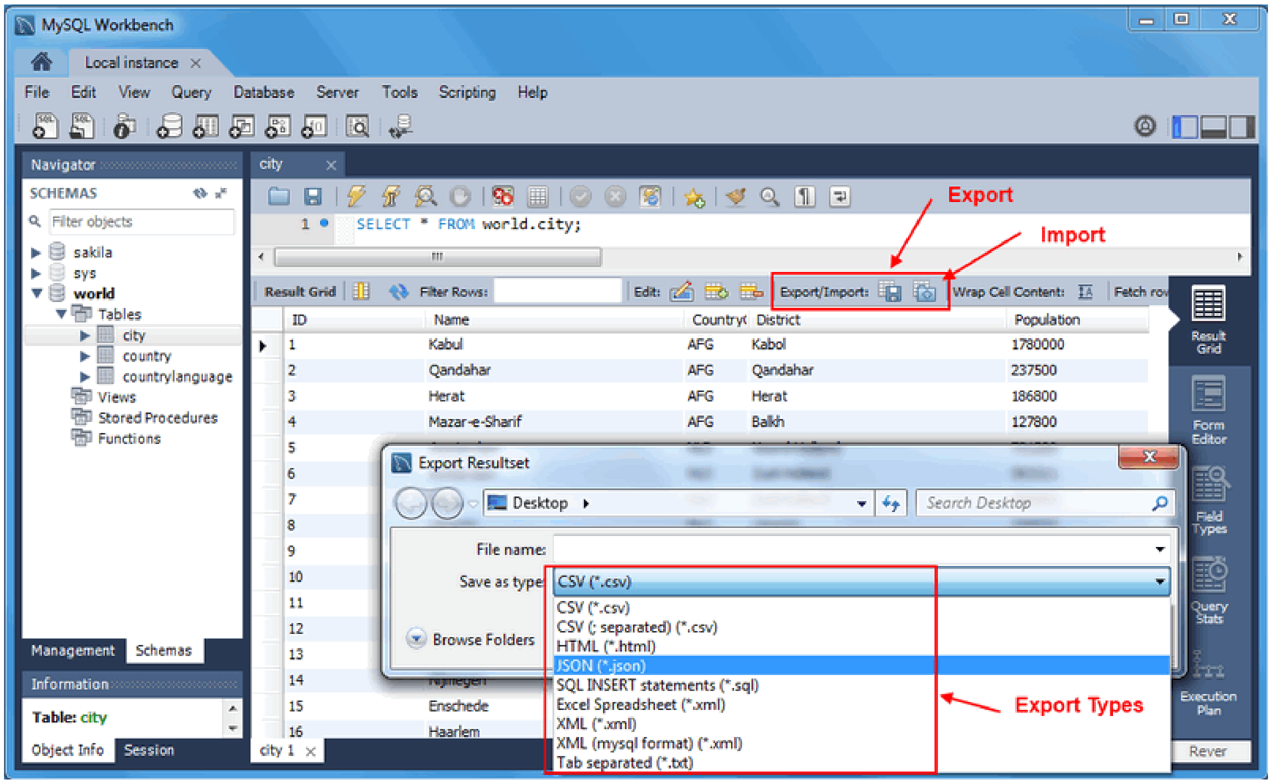
Export a Result Set

A result set in the visual SQL editor can be exported to common file formats including CSV, JSON, HTML, and XML.

Note

This wizard only exports/imports data from a result set. For an overview of the data export and import options in MySQL Workbench, see Section 6.5, "Data Export and Import".

Figure 6.27 Exporting a Result Set



```
SELECT * FROM world.city;
```

Import into a Result Set

Records from a CSV file can be imported into the result set of the visual SQL editor.

Note

The result set must have a unique row identifier (such as a Primary Key or NOT NULL unique index) as otherwise values can not be imported because the result set will be read-only.

Note

Alternatively, use Section 6.5, “Data Export and Import” to export larger sets of data, such as entire tables and databases.

6.6 MySQL Audit Inspector Interface

MySQL Workbench offers a GUI interface to the Audit Inspector.

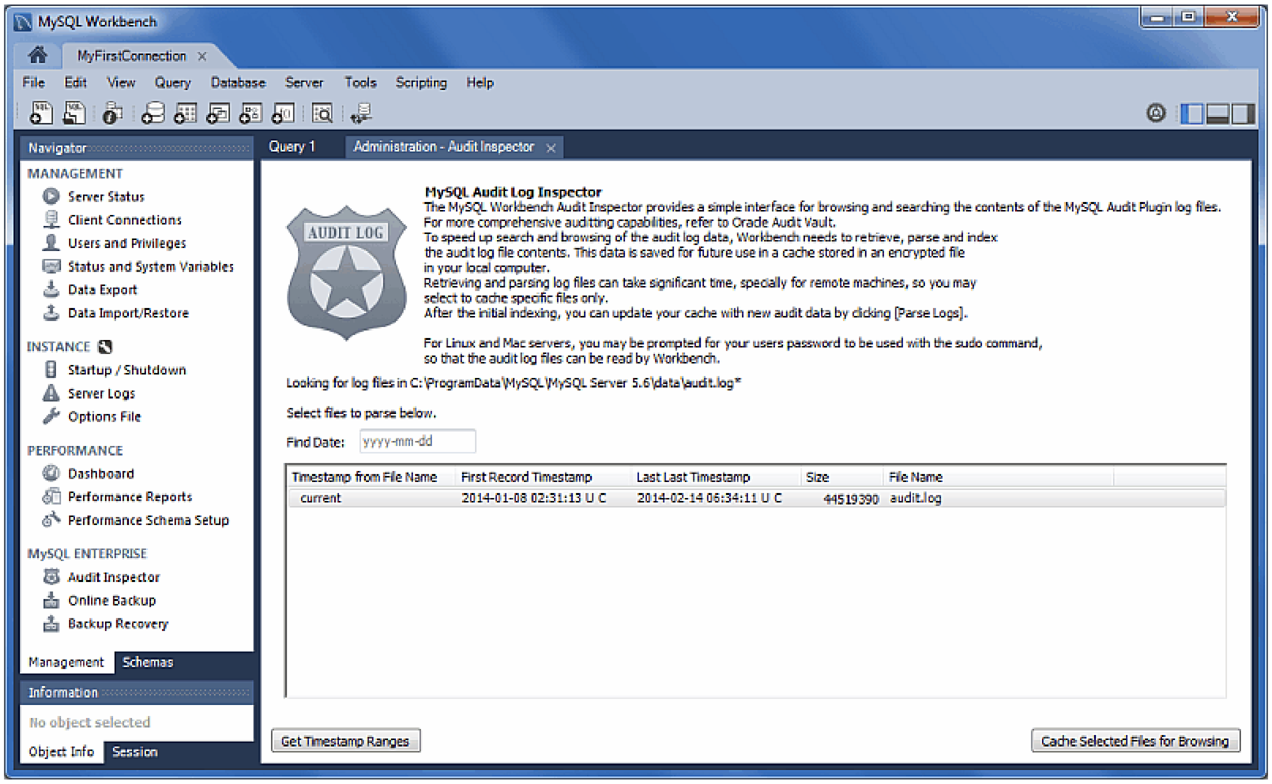
Initially, when you first load the **Audit Inspector**, you must use MySQL Workbench to cache the audit log for performance reasons. MySQL Workbench will then parse, index, and retrieve values from the encrypted cached file on your local computer.

Note

MySQL Workbench will prompt for **sudo** access if the MySQL Workbench user is unable to read the audit log file.

At this stage, you also set a password for the encrypted file that will be used when viewing this file. The initial screen looks similar to:

Figure 6.28 Workbench: Audit Inspector: Initializing



MySQL Audit Log Inspector
The MySQL Workbench Audit Inspector provides a simple interface for browsing and searching the contents of the MySQL Audit Plugin log files.
For more comprehensive auditting capabilities, refer to Oracle Audit Vault.
To speed up search and browsing of the audit log data, Workbench needs to retrieve, parse and index
the audit log file contents. This data is saved for future use in a cache stored in an encrypted file
in your local computer.
Retrieving and parsing log files can take significant time, specially for remote machines, so you may
select to cache specific files only.
After the initial indexing, you can update your cache with new audit data by clicking [Parse Logs].

For Linux and Mac servers, you may be prompted for your users password to be used with the sudo command,
so that the audit log files can be read by Workbench.

Looking for log files in C:\ProgramData\MySQL\MySQL Server 5.6\data\audit.log*

Select files to parse below.

Find Date: yyyy-mm-dd

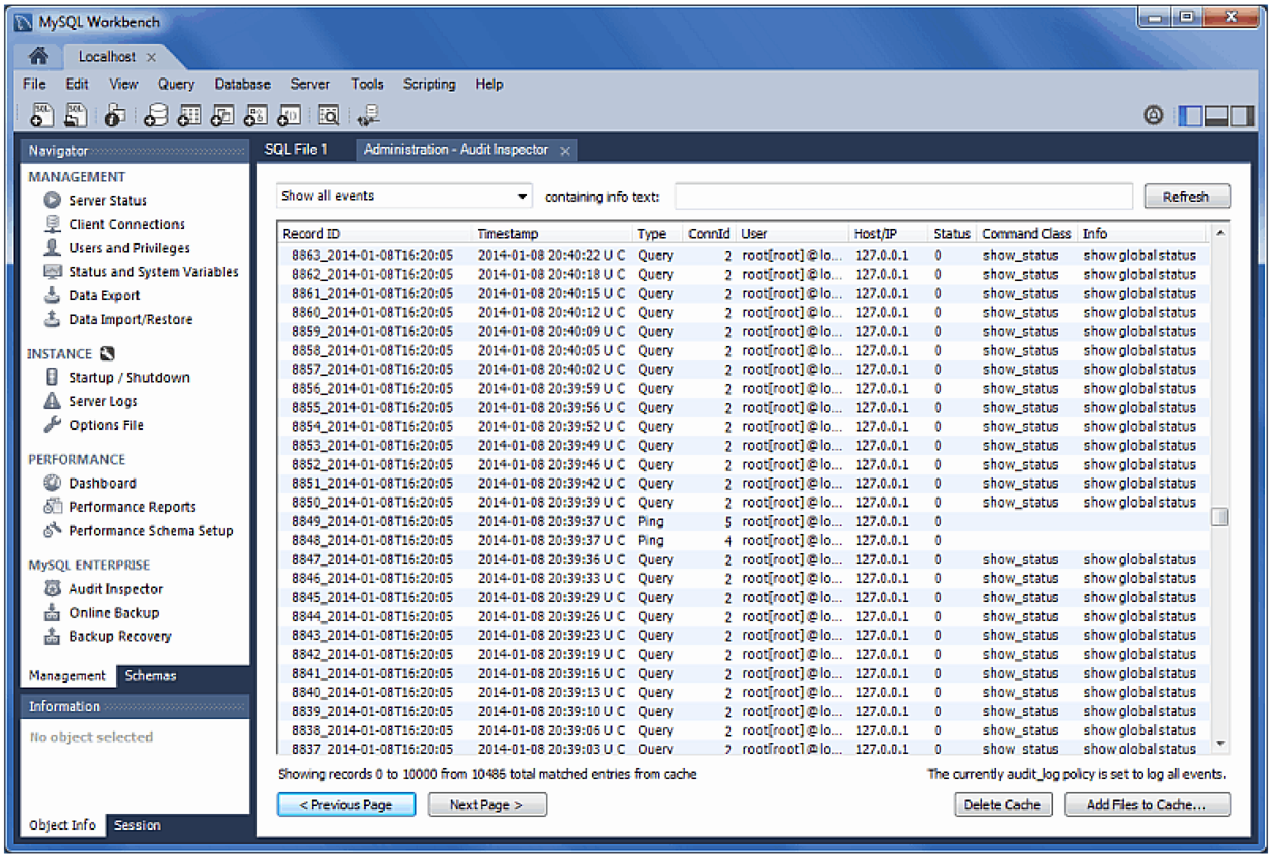
Timestamp from File Name	First Record Timestamp	Last Last Timestamp	Size	File Name
current	2014-01-08 02:31:13 U C	2014-02-14 06:34:11 U C	44519390	audit.log

Note

Generating the cache file can take a long time. If you press **Abort** during the caching process, MySQL Workbench will save the results that were cached at the point you pressed **Abort**.

After caching an audit log, the **Audit Inspector** page will display the results:

Figure 6.29 Workbench: Audit Inspector



Record ID	Timestamp	Type	ConnId	User	Host/IP	Status	Command Class	Info
8863_2014-01-08T16:20:05	2014-01-08 20:40:22 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8862_2014-01-08T16:20:05	2014-01-08 20:40:18 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8861_2014-01-08T16:20:05	2014-01-08 20:40:15 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8860_2014-01-08T16:20:05	2014-01-08 20:40:12 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8859_2014-01-08T16:20:05	2014-01-08 20:40:09 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8858_2014-01-08T16:20:05	2014-01-08 20:40:05 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8857_2014-01-08T16:20:05	2014-01-08 20:40:02 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8856_2014-01-08T16:20:05	2014-01-08 20:39:59 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8855_2014-01-08T16:20:05	2014-01-08 20:39:56 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8854_2014-01-08T16:20:05	2014-01-08 20:39:52 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8853_2014-01-08T16:20:05	2014-01-08 20:39:49 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8852_2014-01-08T16:20:05	2014-01-08 20:39:46 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8851_2014-01-08T16:20:05	2014-01-08 20:39:42 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8850_2014-01-08T16:20:05	2014-01-08 20:39:39 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8849_2014-01-08T16:20:05	2014-01-08 20:39:37 U C	Ping	5	root[root] @lo...	127.0.0.1	0		
8848_2014-01-08T16:20:05	2014-01-08 20:39:37 U C	Ping	4	root[root] @lo...	127.0.0.1	0		
8847_2014-01-08T16:20:05	2014-01-08 20:39:36 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8846_2014-01-08T16:20:05	2014-01-08 20:39:33 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8845_2014-01-08T16:20:05	2014-01-08 20:39:29 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8844_2014-01-08T16:20:05	2014-01-08 20:39:26 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8843_2014-01-08T16:20:05	2014-01-08 20:39:23 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8842_2014-01-08T16:20:05	2014-01-08 20:39:19 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8841_2014-01-08T16:20:05	2014-01-08 20:39:16 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8840_2014-01-08T16:20:05	2014-01-08 20:39:13 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8839_2014-01-08T16:20:05	2014-01-08 20:39:10 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8838_2014-01-08T16:20:05	2014-01-08 20:39:06 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status
8837_2014-01-08T16:20:05	2014-01-08 20:39:03 U C	Query	2	root[root] @lo...	127.0.0.1	0	show_status	show global status

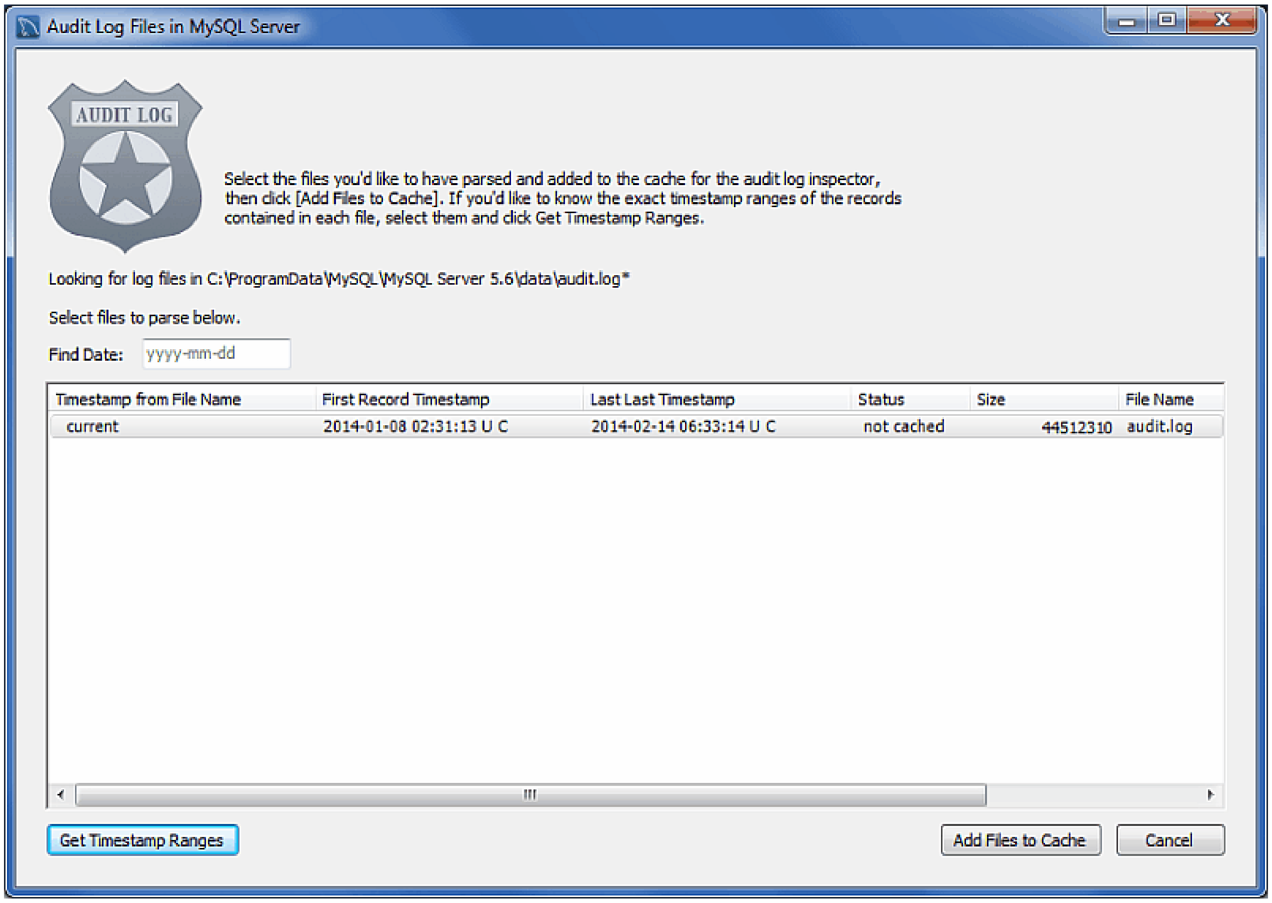
Showing records 0 to 10000 from 10486 total matched entries from cache

The currently audit_log policy is set to log all events.

The search field offers criteria for narrowing the displayed events, including **Show events of type Fetch** and **Show events of type Query**, and defaults to **Show all events**. Custom filters are also available.

You can **Add Files to Cache** from the main Audit Inspector page:

Figure 6.30 Workbench: Audit Inspector: Add Files to Cache



Select the files you'd like to have parsed and added to the cache for the audit log inspector, then click [Add Files to Cache]. If you'd like to know the exact timestamp ranges of the records contained in each file, select them and click Get Timestamp Ranges.

Looking for log files in C:\ProgramData\MySQL\MySQL Server 5.6\data\audit.log*

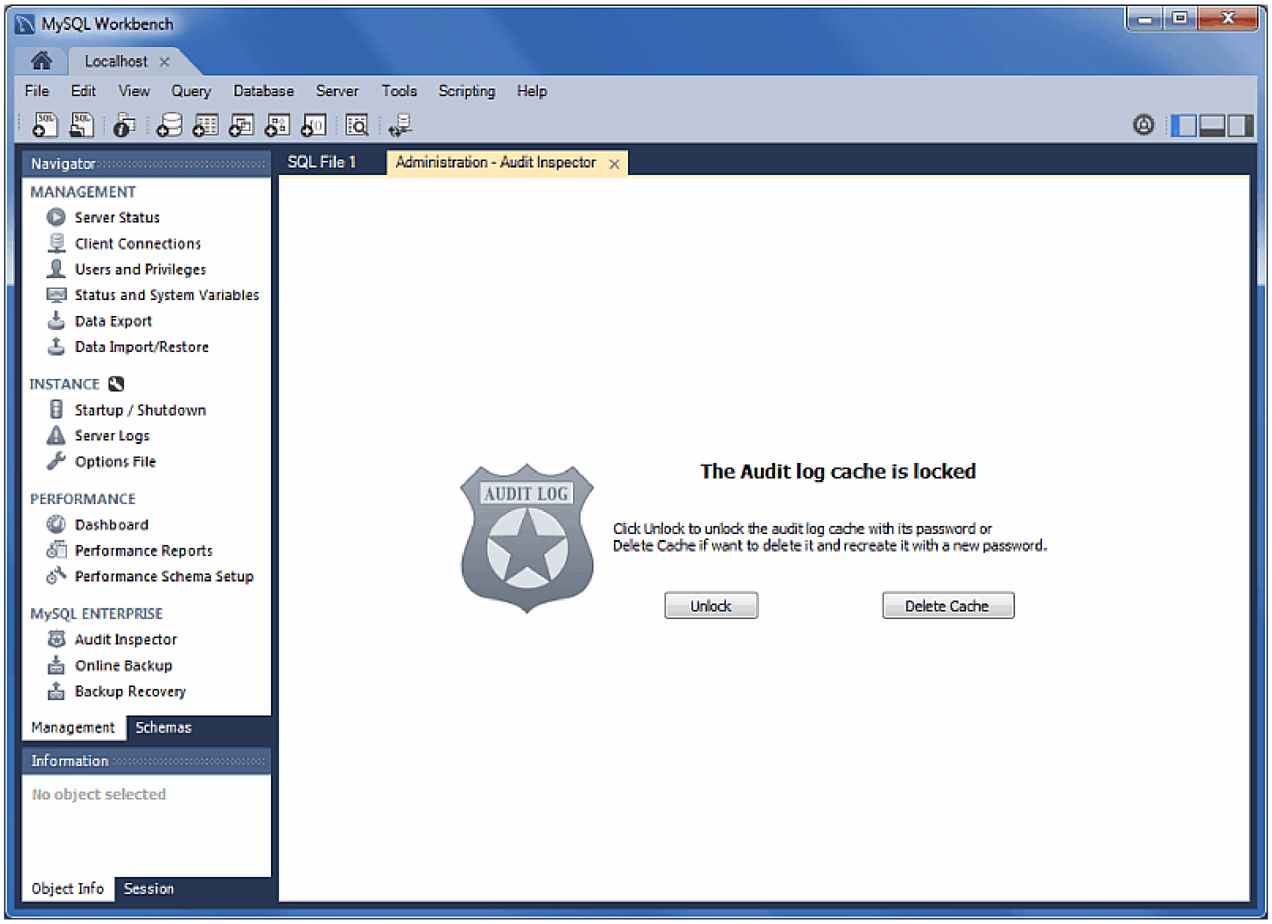
Select files to parse below.

Find Date: yyyy-mm-dd

Timestamp from File Name	First Record Timestamp	Last Last Timestamp	Status	Size	File Name
current	2014-01-08 02:31:13 U C	2014-02-14 06:33:14 U C	not cached	44512310	audit.log

Future uses of the **Audit Inspector** will require the password that you set during the initial step. The login page:

Figure 6.31 Workbench: Audit Inspector: Unlock



The Audit log cache is locked

Click Unlock to unlock the audit log cache with its password or
Delete Cache if want to delete it and recreate it with a new password.

6.7 MySQL Enterprise Backup Interface

MySQL Workbench include a MySQL Enterprise Backup GUI interface that is listed in the Management Navigator tab for a MySQL connection. There are two MySQL Enterprise Backup related sections in the Navigator:

- **Online Backup**: Sets a backup profile that defines `what` should be backed up, `where` the backup should be stored, and `when` (the frequency) MySQL should be backed up.
- **Restore**: Restores the MySQL server to a specific point in time, typically by restoring a backup that was created by the **Online Backup** feature in MySQL Workbench.

For information comparing the different methods to import/export data using MySQL Workbench, see Section 6.5, “Data Export and Import”.

The MySQL Enterprise Backup configuration is located on the MySQL server, and not locally. This information includes the MySQL Enterprise Backup configuration backup profiles, job scheduling, backup operations, and data. This also means that the backup operations are executed with (or without) MySQL Workbench running.

6.7.1 General Requirements

MySQL Enterprise Backup is a MySQL Enterprise Feature that is separate from MySQL Workbench. For more information about its functionality, see the MySQL Enterprise Backup documentation at http://

dev.mysql.com/doc/mysql-enterprise-backup/en. MySQL Workbench provides an interface to MySQL Enterprise Backup, as described in the following documentation. In addition to having MySQL Enterprise Backup installed on your host, the following general requirements also apply:

- A recent version of MySQL Enterprise Backup. The MySQL Enterprise Backup support policy is to support the current GA version of MySQL Enterprise Backup, and the major version before that. This dictates the minimum MySQL Enterprise Backup version required by MySQL Workbench, which is the major version before the current GA release.
- Managing both local and remote MySQL instances is available on Linux and OS X, and managing local MySQL instances is available on Microsoft Windows. Remote management is configured using SSH Remote Management.
- A MySQL connection with a root user.
- The MySQL server configuration file path must be set and correct for the MySQL connection.
- The user running MySQL Workbench must be a sudo user (Linux / OS X) that is able to execute the MySQL Enterprise Backup binary.
- Additionally, the sudo user must keep the `HOME` environment variable when executing system commands, which means adding the following to `/etc/sudoers`:

  ```
  env_keep +="HOME"
  ```

And the **Prerequisites** set in the **Settings** tab are:

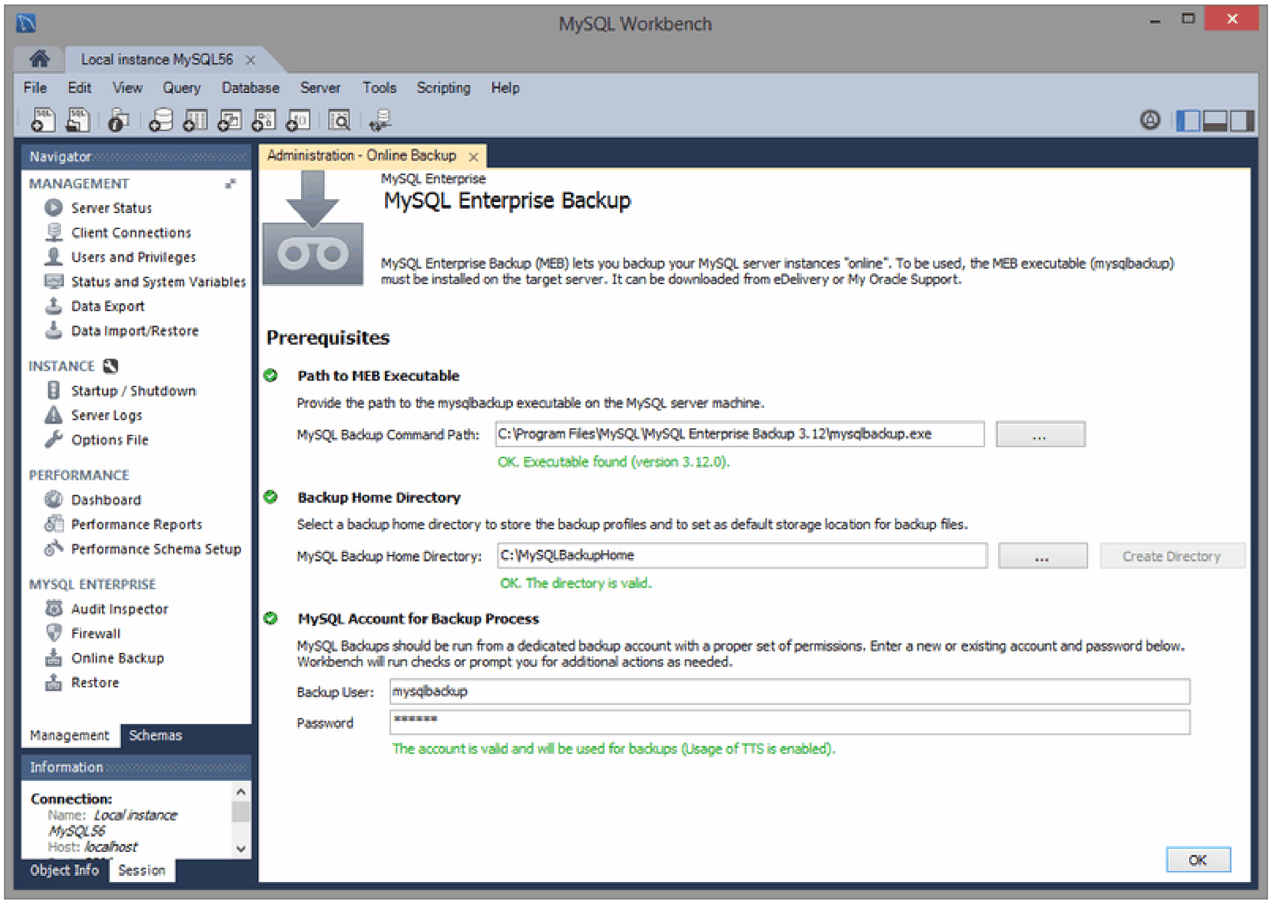
- A path to the MySQL Enterprise Backup executable. MySQL Enterprise Backup is available via eDelivery or My Oracle Support (MOS). MySQL Workbench attempts to locate the MySQL Enterprise Backup executable, so check the path and adjust it accordingly.
- The path to the **Backup Home Directory**, where backup profiles and data is stored. This can be created from within Workbench from the **Settings** tab.
- The MySQL account for the **Backup Process**. The available actions depends on the current state of this set up, with options including:
  - **Create MEB Account**: Available if a backup user does not already exist.
  - **Change Password**: Available if a backup user does exist.
  - **Fix Grants for MEB...**: Available if the user's privileges are invalid, which alters the user by adding the `RELOAD`, `SUPER`, and `REPLICATION CLIENT ON *.*` privileges.

Uninstalling Workbench notes

- Uninstalling Workbench does not remove the associated MySQL Enterprise Backup backup tasks. To stop the scheduled backups, edit the related "Task Scheduler" entries on Windows, or remove the associated cron jobs on Linux and OS X.
- Uninstalling Workbench does not remove the MySQL Enterprise Backup master related configuration file, the configuration files generated for each defined profile, nor the MySQL backups.

The **Settings** tab:

Figure 6.32 Workbench: MySQL Enterprise Backup Settings



Connection:
Name: *Local instance MySQL56*
Host: *localhost*

MySQL Enterprise Backup

MySQL Enterprise Backup (MEB) lets you backup your MySQL server instances "online". To be used, the MEB executable (mysqlbackup) must be installed on the target server. It can be downloaded from eDelivery or My Oracle Support.

Prerequisites

Path to MEB Executable

Provide the path to the mysqlbackup executable on the MySQL server machine.

MySQL Backup Command Path: C:\Program Files\MySQL\MySQL Enterprise Backup 3.12\mysqlbackup.exe

OK. Executable found (version 3.12.0).

Backup Home Directory

Select a backup home directory to store the backup profiles and to set as default storage location for backup files.

MySQL Backup Home Directory: C:\MySQLBackupHome

OK. The directory is valid.

MySQL Account for Backup Process

MySQL Backups should be run from a dedicated backup account with a proper set of permissions. Enter a new or existing account and password below. Workbench will run checks or prompt you for additional actions as needed.

Backup User: mysqlbackup

Password ******

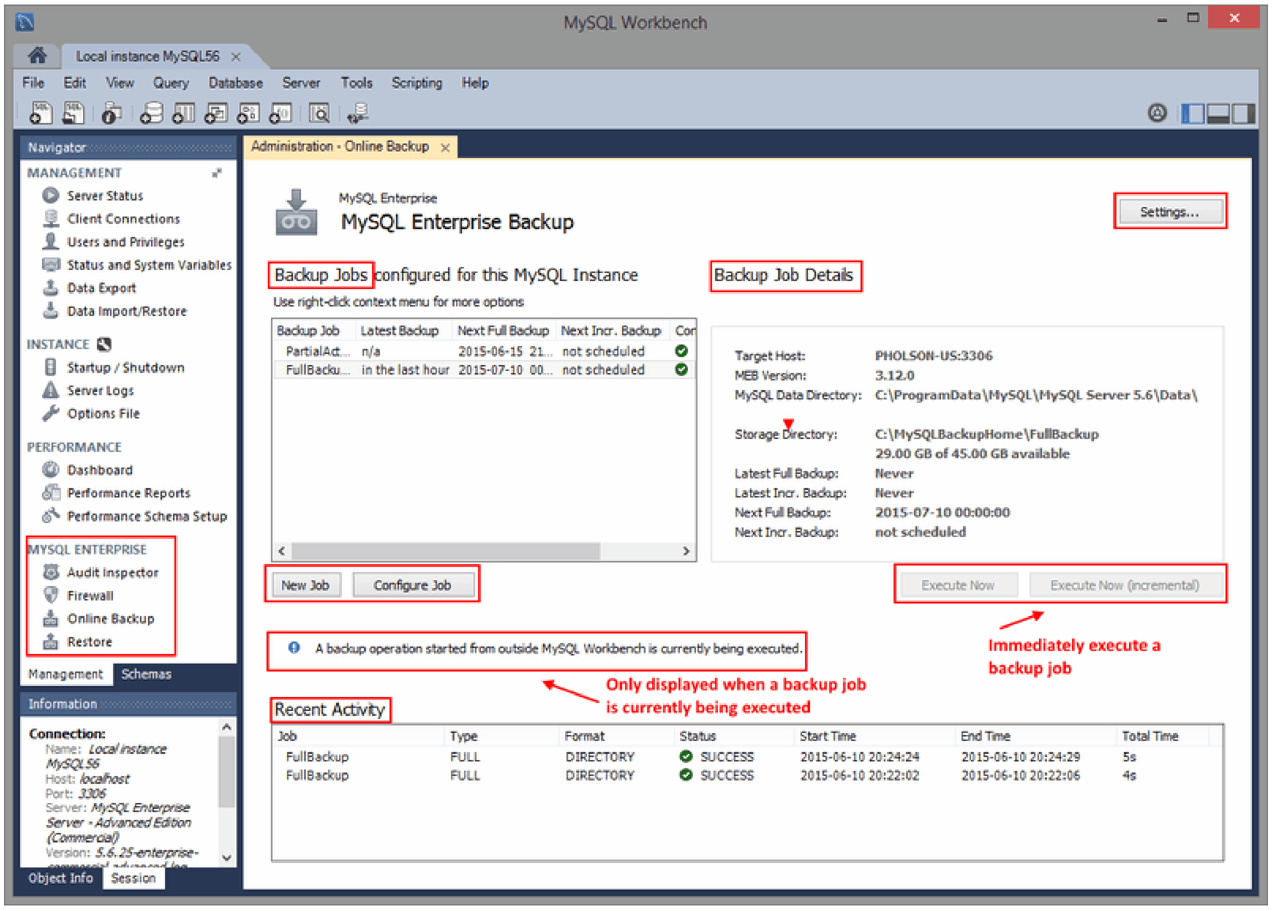
The account is valid and will be used for backups (Usage of TTS is enabled).

If any of the requirements are not met, then an error will be generated when attempting to use MySQL Enterprise Backup features.

6.7.2 Online Backup

Sets a backup profile that defines `what` should be backed up, `where` the backup should be stored, and `when` (the frequency) it should be backed up. The main page:

Figure 6.33 Workbench: MySQL Enterprise Backup



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*

MySQL Enterprise
MySQL Enterprise Backup

Backup Jobs configured for this MySQL Instance
Use right-click context menu for more options

Backup Job	Latest Backup	Next Full Backup	Next Incr. Backup	Cor
PartialAct...	n/a	2015-06-15 21...	not scheduled	✓
FullBacku...	in the last hour	2015-07-10 00...	not scheduled	✓

Backup Job Details

Target Host:	PHOLSON-US:3306
MEB Version:	3.12.0
MySQL Data Directory:	C:\ProgramData\MySQL\MySQL Server 5.6\Data\
Storage Directory:	C:\MySQLBackupHome\FullBackup 29.00 GB of 45.00 GB available
Latest Full Backup:	Never
Latest Incr. Backup:	Never
Next Full Backup:	2015-07-10 00:00:00
Next Incr. Backup:	not scheduled

A backup operation started from outside MySQL Workbench is currently being executed.

Recent Activity

Job	Type	Format	Status	Start Time	End Time	Total Time
FullBackup	FULL	DIRECTORY	✓ SUCCESS	2015-06-10 20:24:24	2015-06-10 20:24:29	5s
FullBackup	FULL	DIRECTORY	✓ SUCCESS	2015-06-10 20:22:02	2015-06-10 20:22:06	4s

The **Online Backup** page is separated into three sections:

- **Backup Jobs**: Used for managing backup jobs for the MySQL server. A backup job (profile) is a configuration file used to store information about what is backed up, where the backup is stored, and optionally when backups will be performed.

  Right-clicking on a Backup Job is an alternative way to access the available actions, such as `Configure Job`, `Delete Job`, and `Execute Backup`. Right-clicking also offers two additional options:

  - **Execute Backup to Image File**: Saves the backup to a single file, and prompts for the file name.

  - **Copy Backup Command to Clipboard**: Generates a command for executing the backup, and copies it to your clipboard. You might execute this command in the shell or terminal, which looks similar to: `/bin/mysqlbackup --defaults-file="/var/lib/meb/foo.cnf" --show-progress=stdout backup --with-timestamp`.

- **Backup Job Details**: Displays information about the state of a specific (selected) backup job. It includes information from the **Settings** page, and information specific to the selected backup.

- **Recent Activity**: Historical information about the backup operations performed on the server. View the backup log by right-clicking an entry and choosing `View Backup Log`

A progress dialog is generated for the backup operation.

Backup Jobs

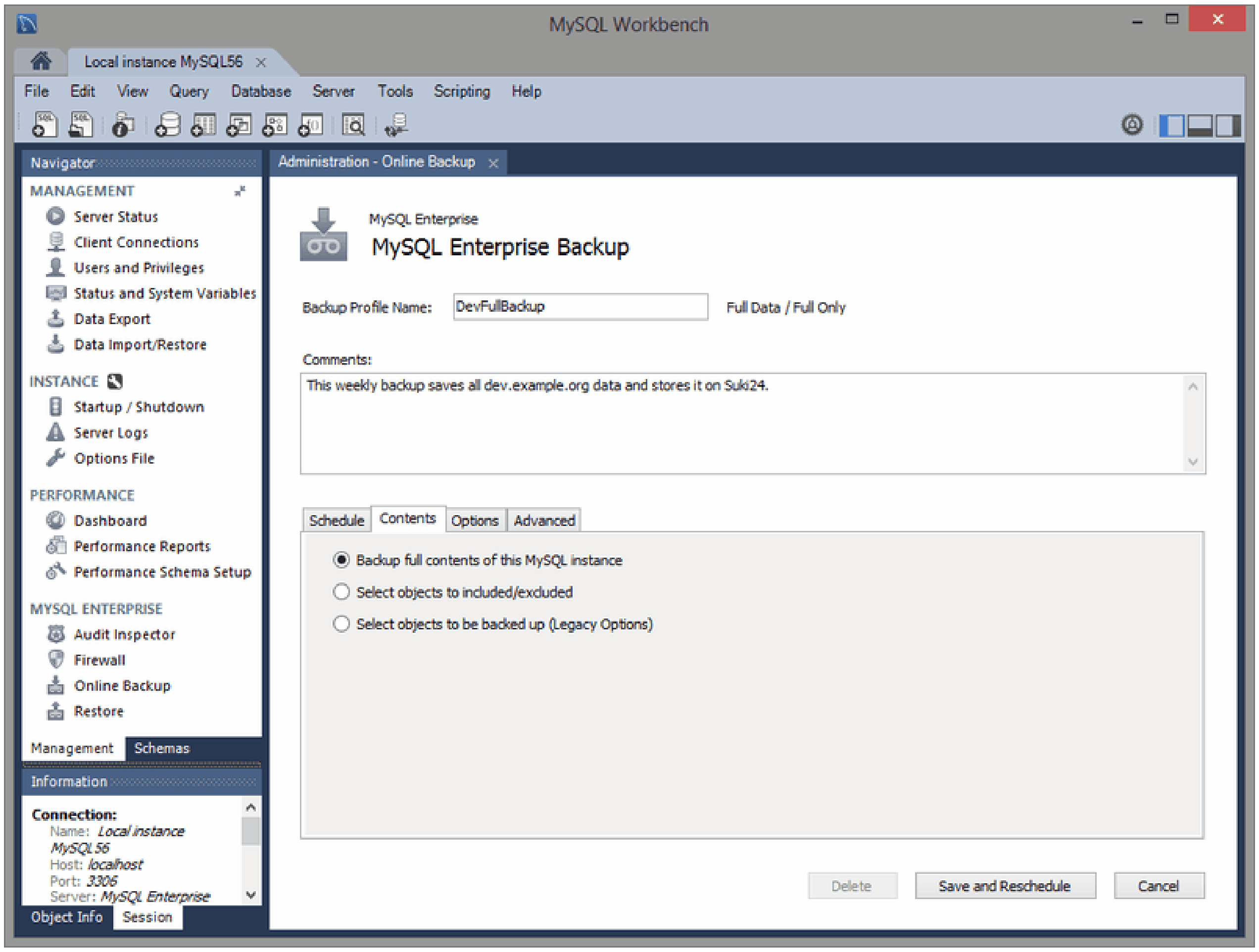
The following information applies to the **New Job** operation, and **Configure Job** is used to modify existing jobs.

The **Backup Profile Name** and its associated **Comments** field are used to identify the backup job's profile, and this name is listed on the main page.

The **New Job** scheduling page separates the configuration information into four tabs. The **Contents** tab defines the schemas and tables to back up, and whether the job is a full or partial backup.

- `Full backup`: All schemas and tables are backed up.
- `Partial`: Select the schemas and tables (objects) that you want to back up. Choose **Select objects to included/excluded** to open the table inclusion (and exclusion) options. For additional information about the include, exclude, and **Transportable Tablespace** options, see the MySQL Enterprise Backup documentation titled Partial Backup and Restore Options.

Figure 6.34 Workbench: MySQL Enterprise Backup Configuration: Contents for Full Backups



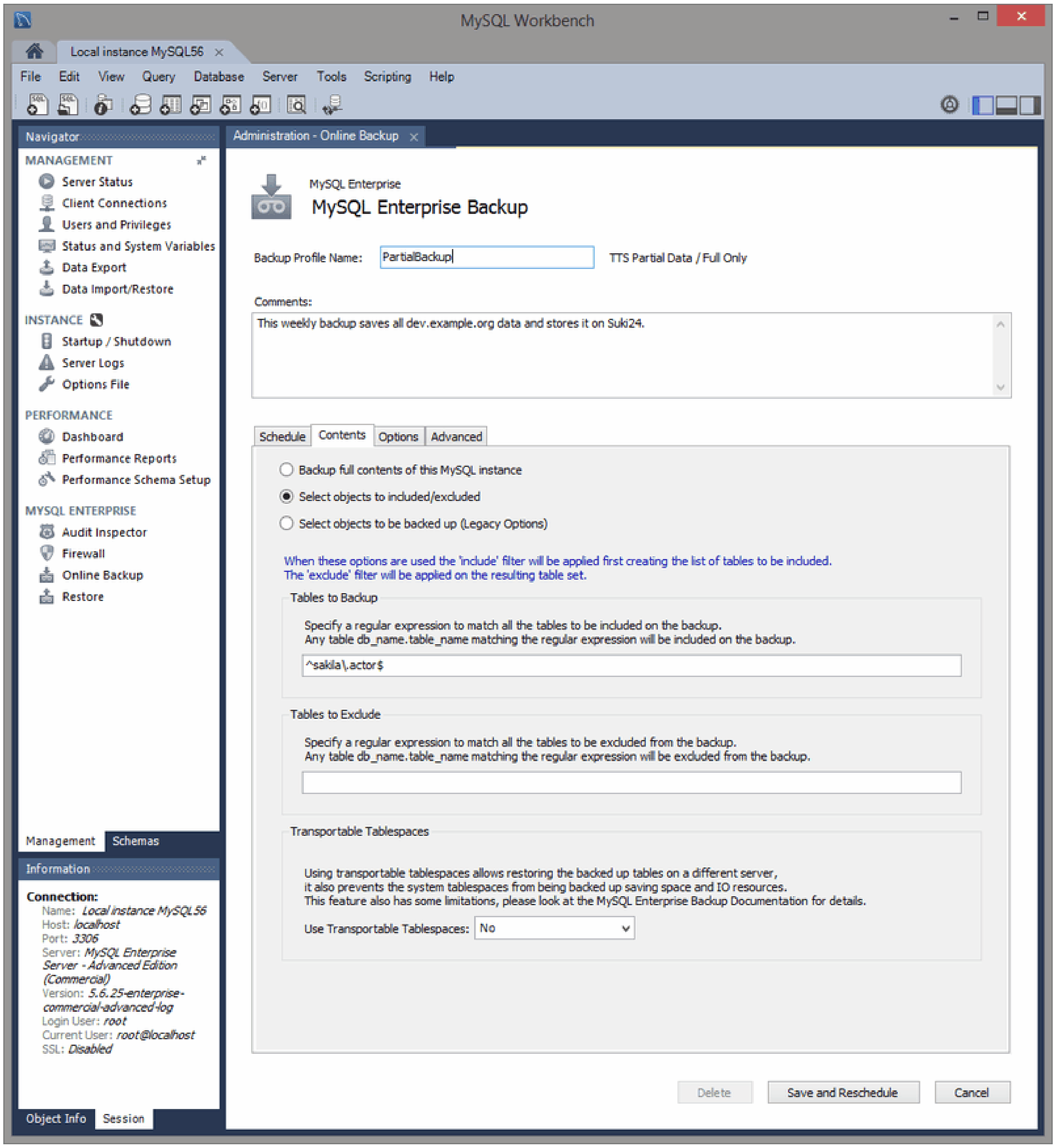
Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise*

Backup Profile Name: DevFullBackup
Full Data / Full Only

Comments:
This weekly backup saves all dev.example.org data and stores it on Suki24.

[✓ Backup full contents of this MySQL instance]
Select objects to included/excluded
Select objects to be backed up (Legacy Options)

Figure 6.35 Workbench: MySQL Enterprise Backup Configuration: Contents for Partial Backups



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Disabled*

Backup Profile Name: PartialBackup TTS Partial Data / Full Only

Comments:
This weekly backup saves all dev.example.org data and stores it on Suki24.

( ) Backup full contents of this MySQL instance
(●) Select objects to included/excluded
( ) Select objects to be backed up (Legacy Options)

When these options are used the 'include' filter will be applied first creating the list of tables to be included.
The 'exclude' filter will be applied on the resulting table set.

Tables to Backup

Specify a regular expression to match all the tables to be included on the backup.
Any table db_name.table_name matching the regular expression will be included on the backup.

^sakila\.actor$

Tables to Exclude

Specify a regular expression to match all the tables to be excluded from the backup.
Any table db_name.table_name matching the regular expression will be excluded from the backup.

Transportable Tablespaces

Using transportable tablespaces allows restoring the backed up tables on a different server,
it also prevents the system tablespaces from being backed up saving space and IO resources.
This feature also has some limitations, please look at the MySQL Enterprise Backup Documentation for details.

Use Transportable Tablespaces: No

The **Options** tab includes settings to modify the default behavior of the backup process.

- **Backup Storage Directory**: By default, the **Backup Storage Directory** is stored under a sub-folder using the name of the **Backup Profile Name** in the `MySQL Backup Home Directory` setting.

  A new sub-folder is created for each backup, named with its timestamp. An example subdirectory is "2016-02-22_17-49-18" where 17:49:18 is the time.

Incremental backups are also stored in the **Backup Storage Directory** directory, but in their own `inc/` sub-folder. Each incremental backup also creates its own timestamped sub-folder within `inc/`.

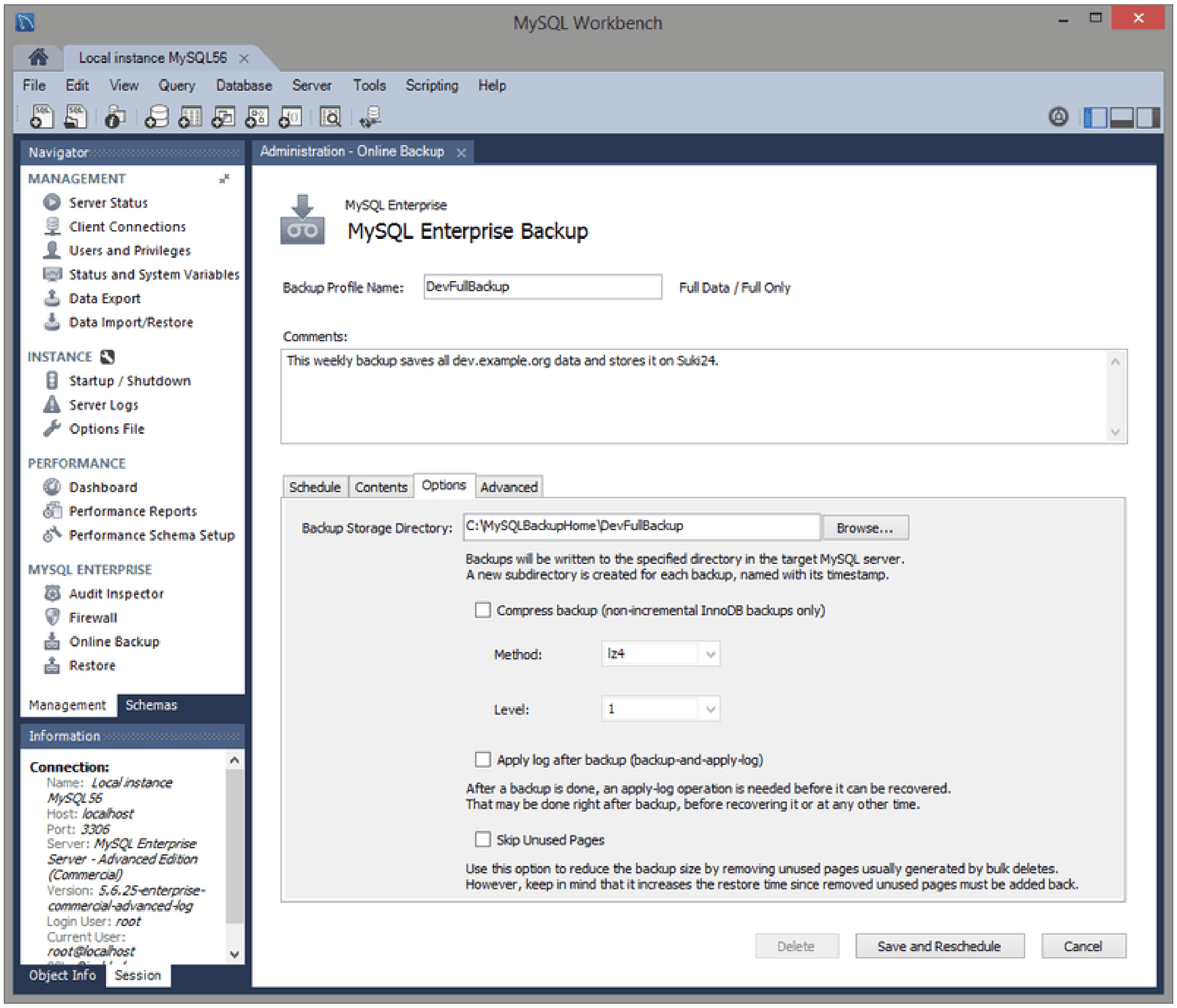
- **Compress Backup**: Optionally compress non-incremental InnoDB backups.
- **Apply Log after backup**: After a backup is completed, an `apply-log` operation is needed before it can be completed. This can be done after a backup, before recovery, or at any other time. Disabled by default.
- **Skip Unused Pages**: Use this option to reduce the backup size by removing unused pages that are typically generated by bulk deletes. Disabled by default.

Note

Enabling this increases the restoration time because the removed unused pages must be added back during the recovery process.

Additional options include compression and `apply-log`, and the option to **Skip Unused Pages**.

Figure 6.36 Workbench: MySQL Enterprise Backup Configuration: Options



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*
Current User: *root@localhost*

Backup Profile Name: DevFullBackup   Full Data / Full Only

Comments:
This weekly backup saves all dev.example.org data and stores it on Suki24.

Backup Storage Directory: C:\MySQLBackupHome\DevFullBackup

Backups will be written to the specified directory in the target MySQL server.
A new subdirectory is created for each backup, named with its timestamp.

☐ Compress backup (non-incremental InnoDB backups only)

Method: lz4

Level: 1

☐ Apply log after backup (backup-and-apply-log)

After a backup is done, an apply-log operation is needed before it can be recovered.
That may be done right after backup, before recovering it or at any other time.

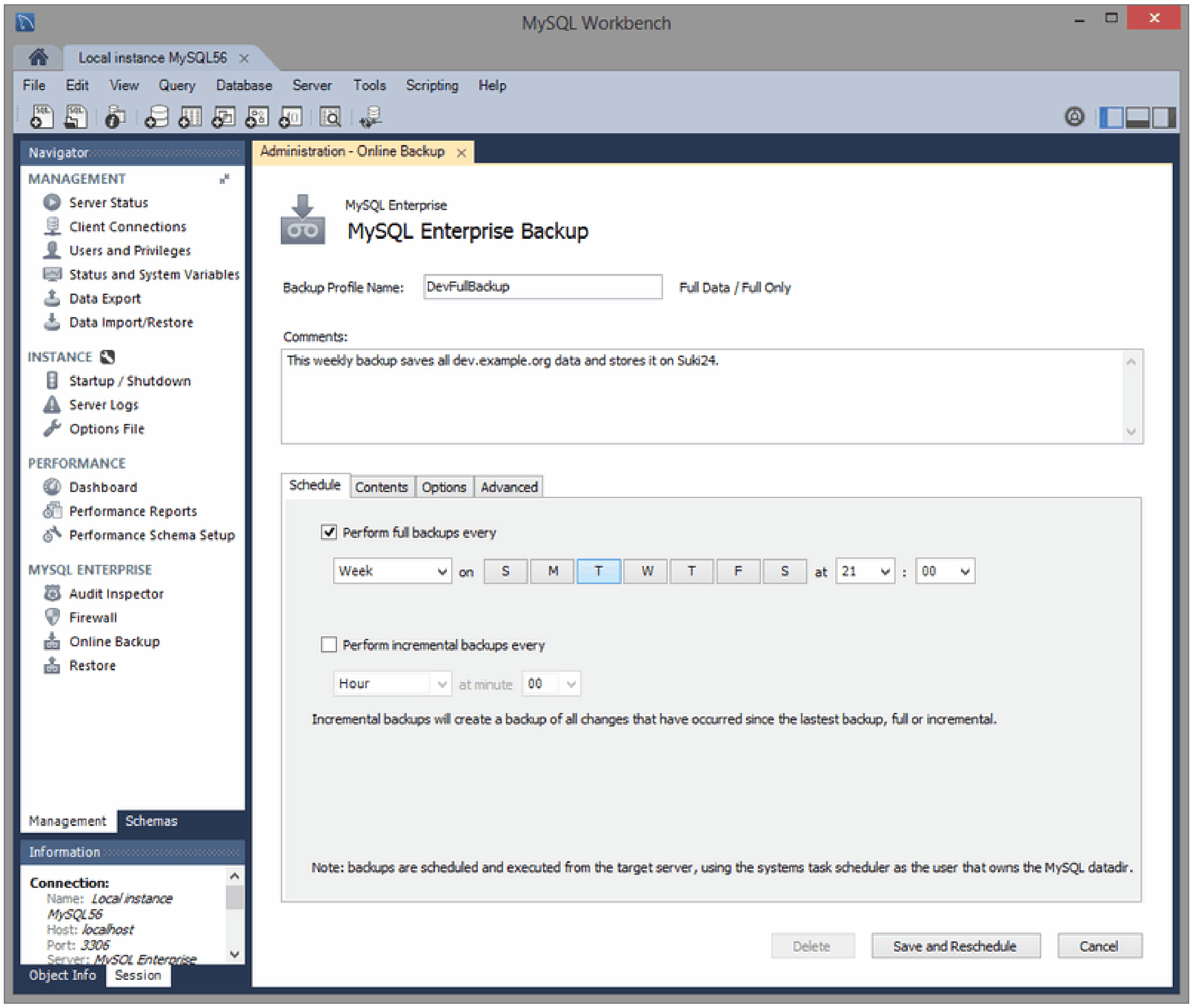
☐ Skip Unused Pages

Use this option to reduce the backup size by removing unused pages usually generated by bulk deletes.
However, keep in mind that it increases the restore time since removed unused pages must be added back.

The **Schedule** tab optionally sets a backup schedule for both full and incremental backups. The schedule uses the Windows Tasks Scheduler on Microsoft Windows, and a cron job on Linux and OS X. It is scheduled using the operating system user that is scheduling the backup, which is typically the MySQL user.

A full backup is slower than the incremental backup that merges with a full backup. A common scenario is to set a full backup as weekly, and an incremental backup as daily. For additional information about backup performance, see Optimizing Backup Performance.

Figure 6.37 Workbench: MySQL Enterprise Backup Configuration: Schedule



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise*

Backup Profile Name: DevFullBackup   Full Data / Full Only

Comments:
This weekly backup saves all dev.example.org data and stores it on Suki24.

[✓] Perform full backups every
Week on S M [✓ T] W T F S at 21 : 00

[ ] Perform incremental backups every
Hour at minute 00

Incremental backups will create a backup of all changes that have occurred since the lastest backup, full or incremental.

Note: backups are scheduled and executed from the target server, using the systems task scheduler as the user that owns the MySQL datadir.

The **Advanced** tab allows you to pass in additional MySQL Enterprise Backup options.

Note

These additional options are not validated.

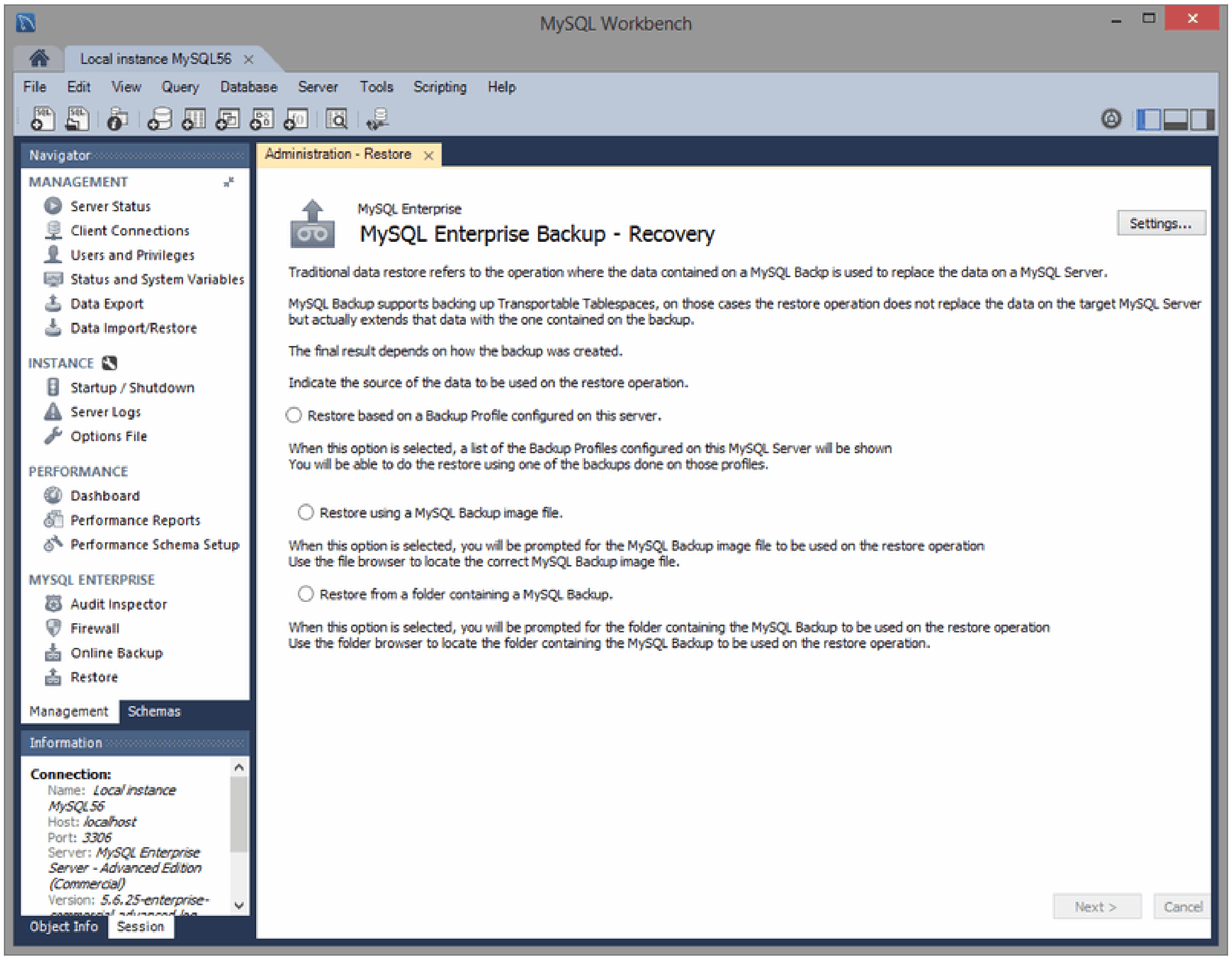
To recover backups, see Section 6.7.3, “Backup Recovery”.

6.7.3 Backup Recovery

The Backup Recovery wizard is used to recover MySQL Enterprise Backup backups. For more information about creating MySQL Enterprise Backup backups using MySQL Workbench, see Section 6.7.2, “Online Backup”

The Backup Recovery wizard allows you to restore backups from folders, image files, and backup profiles created by Section 6.7.2, “Online Backup”. Click **Restore** from the Navigator, then choose the appropriate option:

Figure 6.38 Workbench: Backup Recovery: Start



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*

MySQL Enterprise
MySQL Enterprise Backup - Recovery

Traditional data restore refers to the operation where the data contained on a MySQL Backp is used to replace the data on a MySQL Server.

MySQL Backup supports backing up Transportable Tablespaces, on those cases the restore operation does not replace the data on the target MySQL Server but actually extends that data with the one contained on the backup.

The final result depends on how the backup was created.

Indicate the source of the data to be used on the restore operation.

○ Restore based on a Backup Profile configured on this server.

When this option is selected, a list of the Backup Profiles configured on this MySQL Server will be shown
You will be able to do the restore using one of the backups done on those profiles.

○ Restore using a MySQL Backup image file.

When this option is selected, you will be prompted for the MySQL Backup image file to be used on the restore operation
Use the file browser to locate the correct MySQL Backup image file.

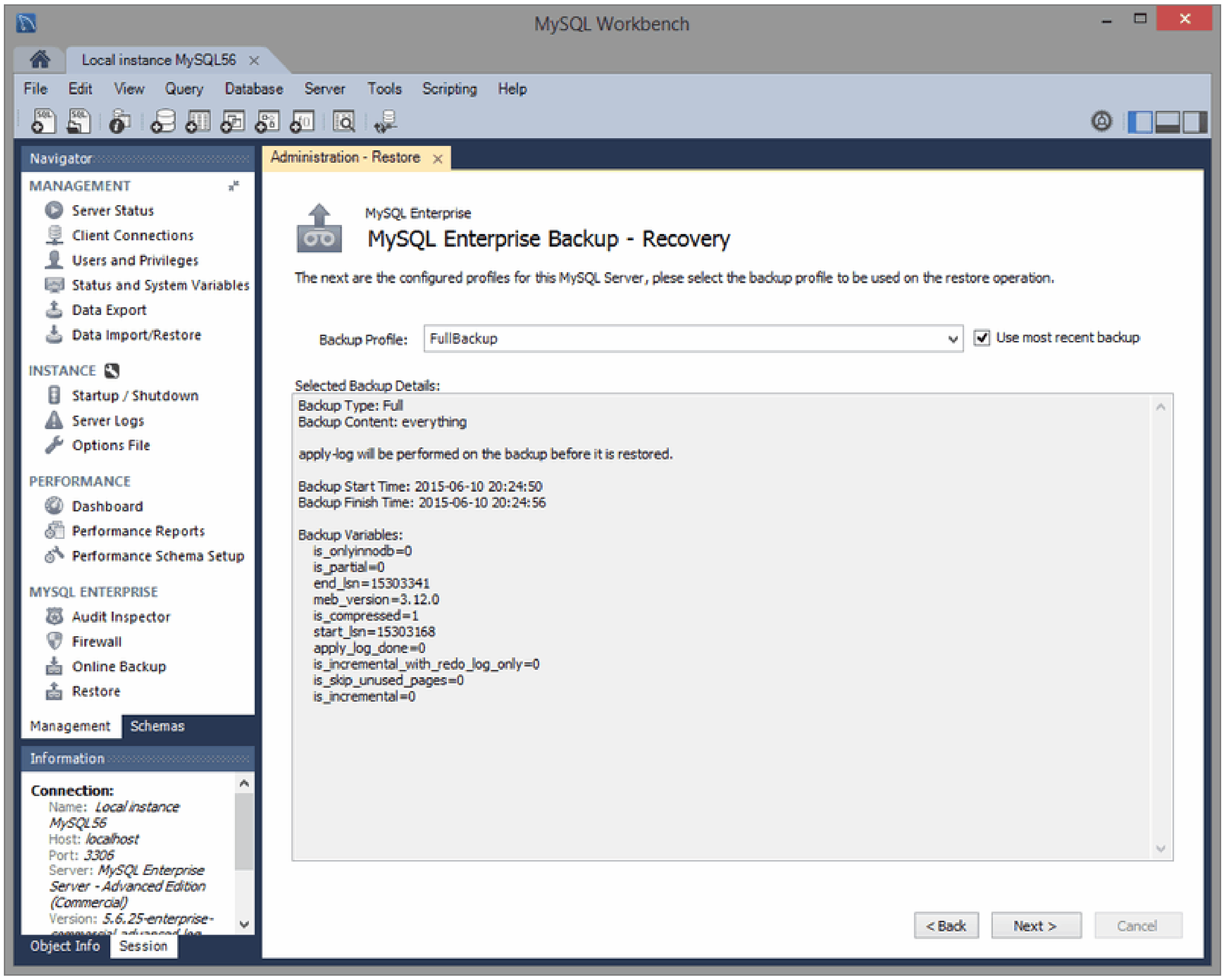
○ Restore from a folder containing a MySQL Backup.

When this option is selected, you will be prompted for the folder containing the MySQL Backup to be used on the restore operation
Use the folder browser to locate the folder containing the MySQL Backup to be used on the restore operation.

- **Backup Profile**: Choose from the available MySQL Enterprise Backup profiles on your system.
- **Backup Image File**: Opens the system's file browser; choose a backup image file to restore.
- **Backup Folder**: Opens the system's file browser; choose a backup folder to restore.

In our example, we will restore a full backup profile that was created by MySQL Workbench Online Backup. After choosing the "FullBackup" profile that we created earlier, use the next page to review its upcoming restoration:

Figure 6.39 Workbench: Backup Recovery: Profile



MySQL Enterprise Backup - Recovery

The next are the configured profiles for this MySQL Server, plese select the backup profile to be used on the restore operation.

Backup Profile: FullBackup ☑ Use most recent backup

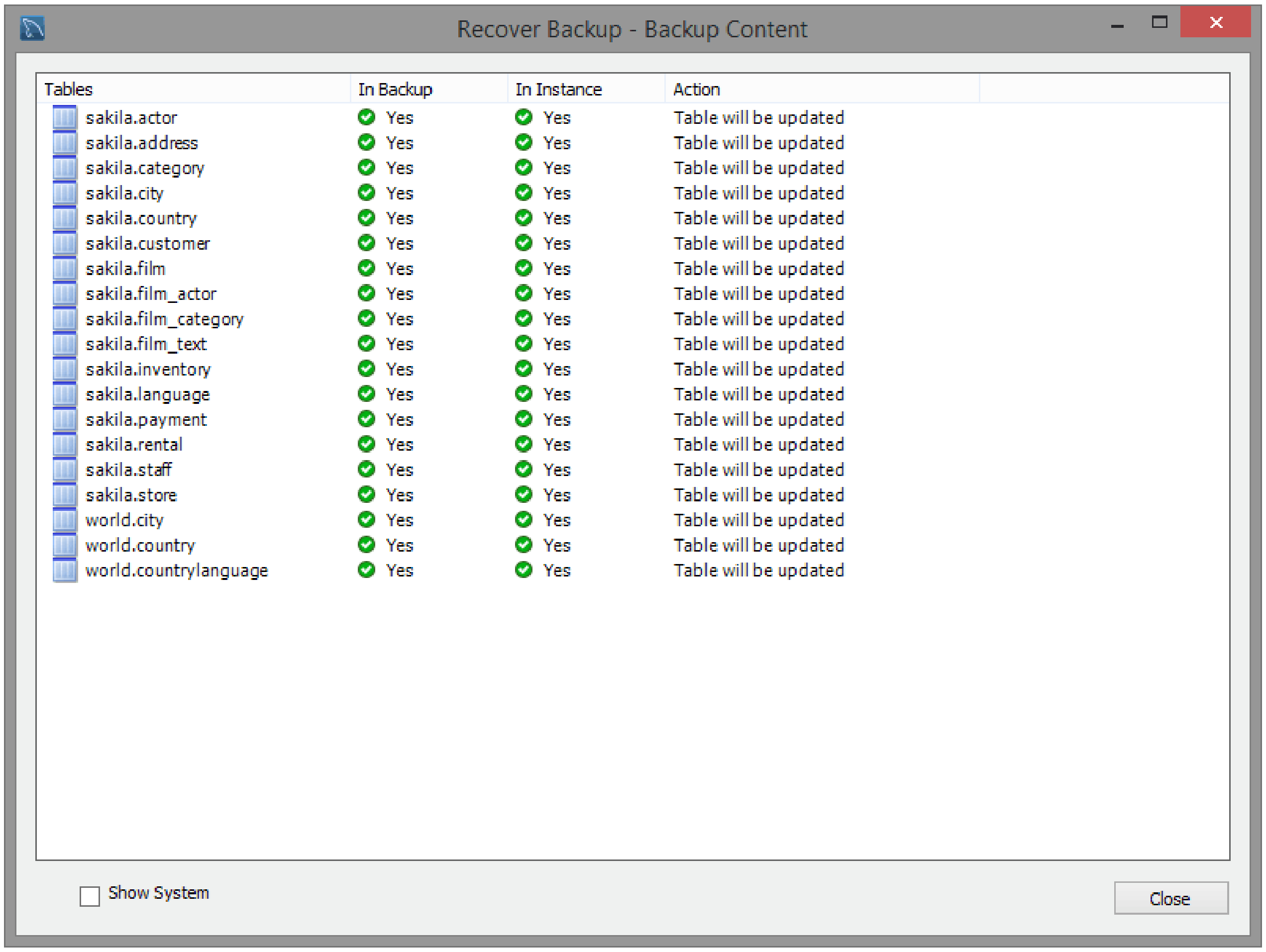
Selected Backup Details:

```
Backup Type: Full
Backup Content: everything

apply-log will be performed on the backup before it is restored.

Backup Start Time: 2015-06-10 20:24:50
Backup Finish Time: 2015-06-10 20:24:56

Backup Variables:
    is_onlyinnodb=0
    is_partial=0
    end_lsn=15303341
    meb_version=3.12.0
    is_compressed=1
    start_lsn=15303168
    apply_log_done=0
    is_incremental_with_redo_log_only=0
    is_skip_unused_pages=0
    is_incremental=0
```

Connection:
Name: Local instance MySQL56
Host: localhost
Port: 3306
Server: MySQL Enterprise Server - Advanced Edition (Commercial)
Version: 5.6.25-enterprise-commercial-advanced-log

Optionally, click **View Backup Content** to view the backup contents for the restoration:

Figure 6.40 Workbench: Backup Recovery: Contents Table View



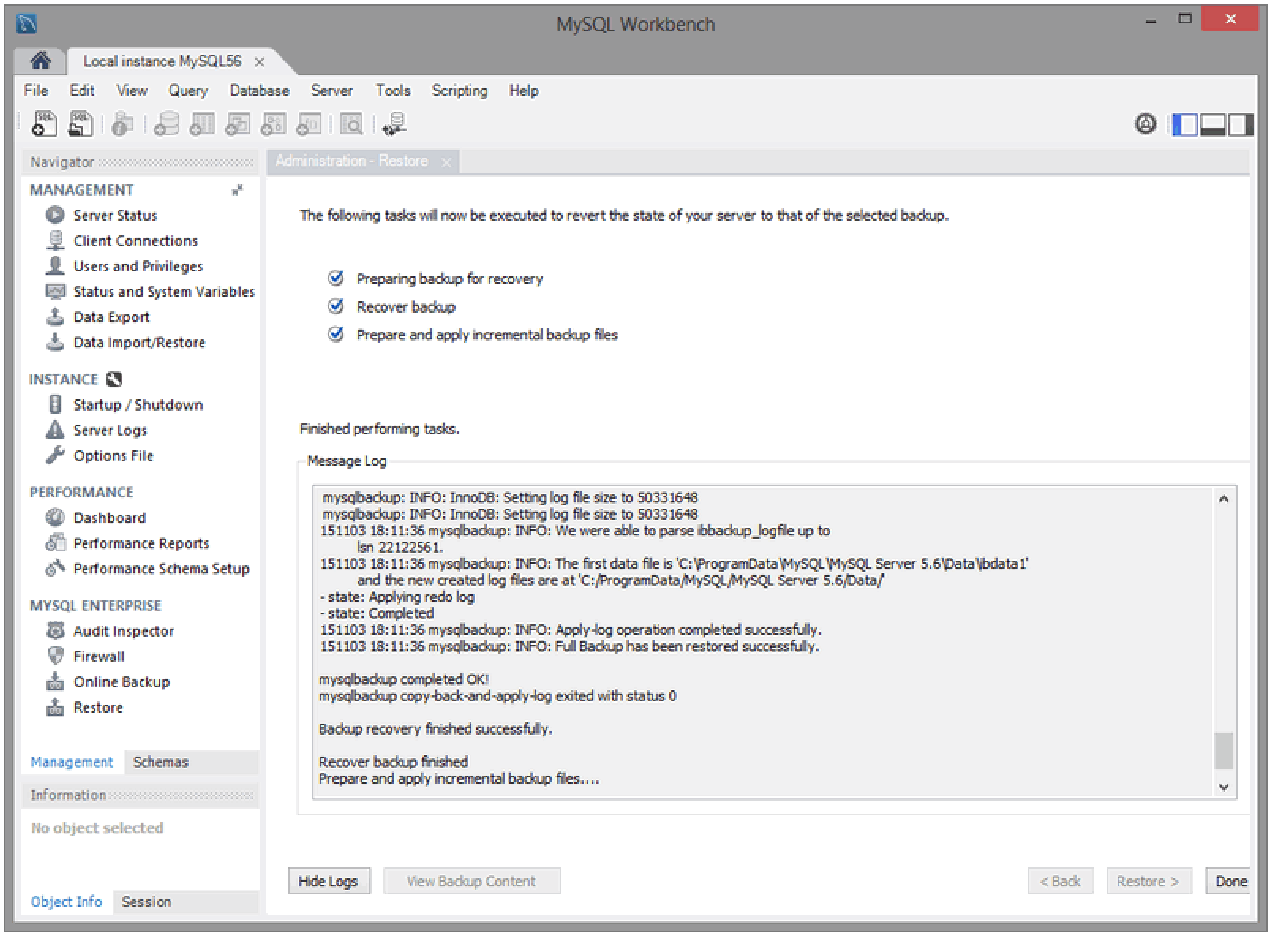
Tables	In Backup	In Instance	Action
sakila.actor	✓ Yes	✓ Yes	Table will be updated
sakila.address	✓ Yes	✓ Yes	Table will be updated
sakila.category	✓ Yes	✓ Yes	Table will be updated
sakila.city	✓ Yes	✓ Yes	Table will be updated
sakila.country	✓ Yes	✓ Yes	Table will be updated
sakila.customer	✓ Yes	✓ Yes	Table will be updated
sakila.film	✓ Yes	✓ Yes	Table will be updated
sakila.film_actor	✓ Yes	✓ Yes	Table will be updated
sakila.film_category	✓ Yes	✓ Yes	Table will be updated
sakila.film_text	✓ Yes	✓ Yes	Table will be updated
sakila.inventory	✓ Yes	✓ Yes	Table will be updated
sakila.language	✓ Yes	✓ Yes	Table will be updated
sakila.payment	✓ Yes	✓ Yes	Table will be updated
sakila.rental	✓ Yes	✓ Yes	Table will be updated
sakila.staff	✓ Yes	✓ Yes	Table will be updated
sakila.store	✓ Yes	✓ Yes	Table will be updated
world.city	✓ Yes	✓ Yes	Table will be updated
world.country	✓ Yes	✓ Yes	Table will be updated
world.countrylanguage	✓ Yes	✓ Yes	Table will be updated

Note

The **Show System** checkbox toggles internal schemas from view, schemas such as the internal performance_schema and mysql tables.

Clicking **Next >** will open the **Restore** wizard. We then clicked **Restore >** to execute the restoration process, and toggled the message logs in our example below:

Figure 6.41 Workbench: Backup Recovery: Restore



The following tasks will now be executed to revert the state of your server to that of the selected backup.

- ✓ Preparing backup for recovery
- ✓ Recover backup
- ✓ Prepare and apply incremental backup files

Finished performing tasks.

Message Log

```
mysqlbackup: INFO: InnoDB: Setting log file size to 50331648
mysqlbackup: INFO: InnoDB: Setting log file size to 50331648
151103 18:11:36 mysqlbackup: INFO: We were able to parse ibbackup_logfile up to
        lsn 22122561.
151103 18:11:36 mysqlbackup: INFO: The first data file is 'C:\ProgramData\MySQL\MySQL Server 5.6\Data\ibdata1'
        and the new created log files are at 'C:/ProgramData/MySQL/MySQL Server 5.6/Data/'
- state: Applying redo log
- state: Completed
151103 18:11:36 mysqlbackup: INFO: Apply-log operation completed successfully.
151103 18:11:36 mysqlbackup: INFO: Full Backup has been restored successfully.

mysqlbackup completed OK!
mysqlbackup copy-back-and-apply-log exited with status 0

Backup recovery finished successfully.

Recover backup finished
Prepare and apply incremental backup files....
```

6.8 MySQL Enterprise Firewall Interface

MySQL Workbench offers a GUI interface to MySQL Enterprise Firewall.

Note

The MySQL Enterprise Firewall interface was added in MySQL Workbench 6.3.4.

For additional information about MySQL Enterprise Firewall, see https://dev.mysql.com/doc/en/firewall.html.

Setup and Configuration

MySQL Workbench can manage the MySQL Enterprise Firewall installation and configuration by installing (or uninstalling) and enabling (or disabling) the plugin.

Note

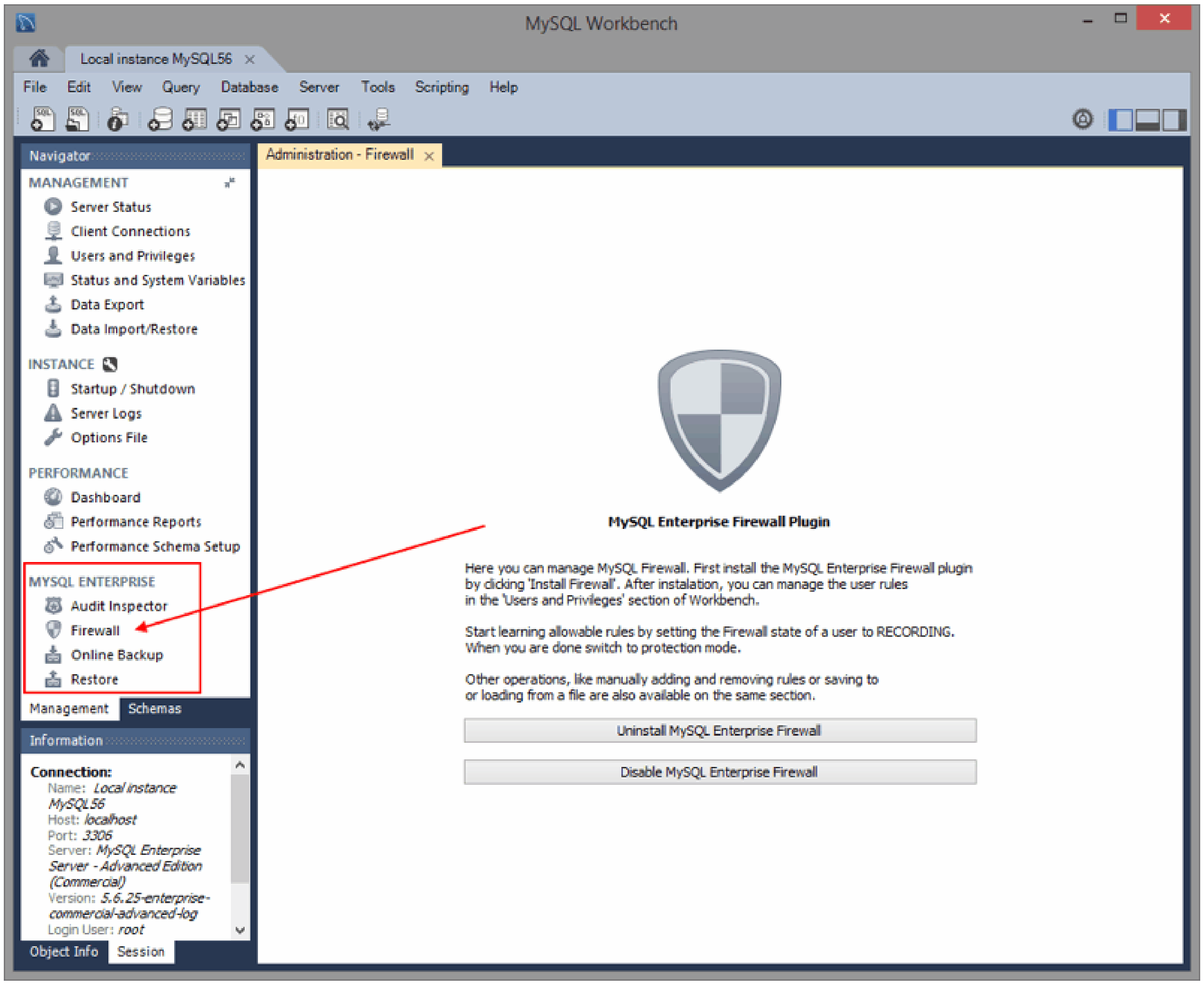
Alternatively, you can manually install and enable MySQL Enterprise Firewall. For additional information, see Installing or Uninstalling MySQL Enterprise Firewall.

- **Enable**: Executes *SET GLOBAL mysql_firewall_mode = ON;* against the connected MySQL server. **Disable** sets it to OFF instead of ON.

This is a runtime operation. Configure the MySQL server configuration file to enable MySQL Enterprise Firewall at startup.

- **Install**: Executes queries to install the new MySQL Enterprise Firewall tables and stored procedure needed to switch the state. **Uninstall** reverses these effects, which also removes the recorded rules.

Figure 6.42 MySQL Enterprise Firewall Installation and Configuration



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*

MySQL Enterprise Firewall Plugin

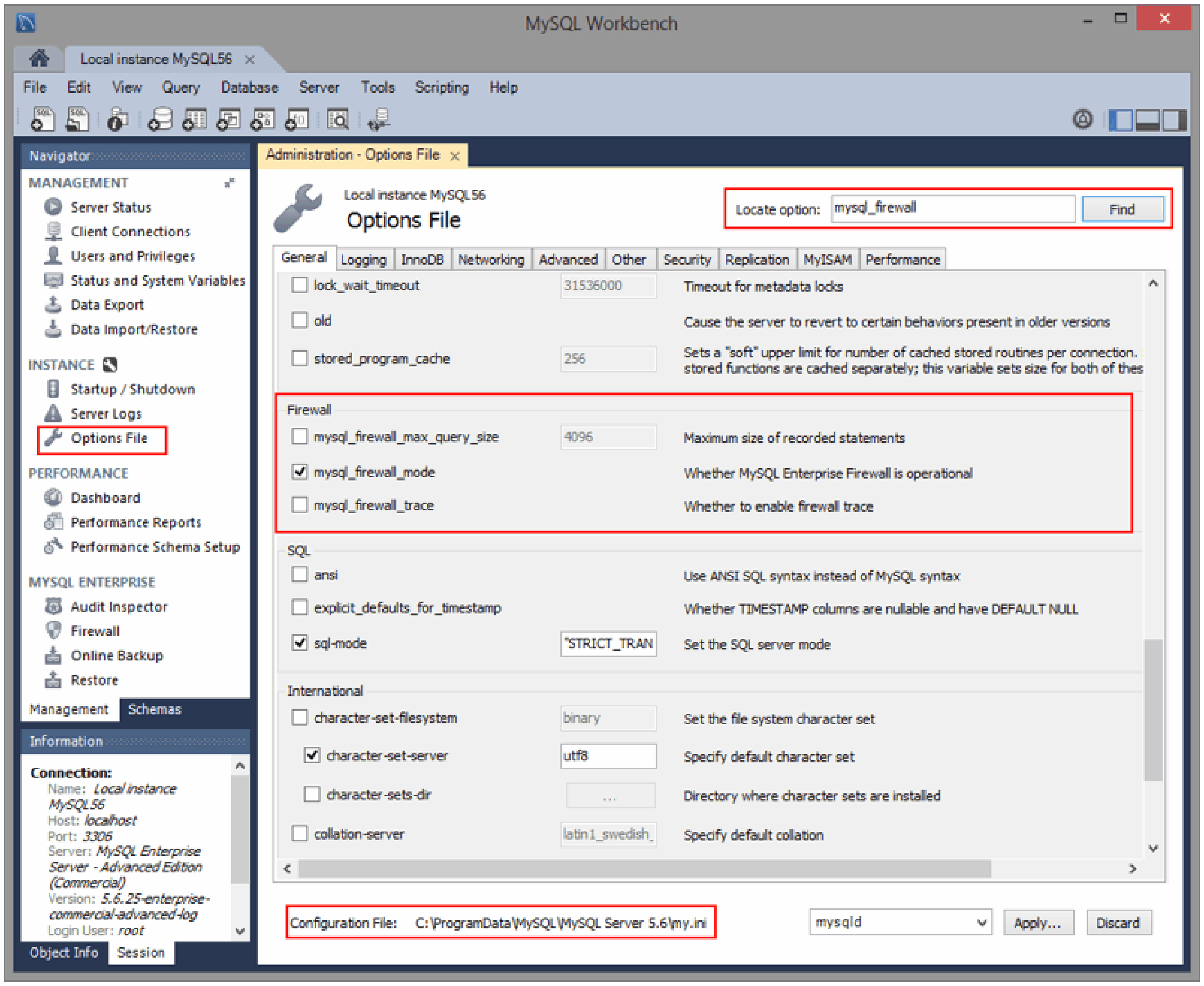
Here you can manage MySQL Firewall. First install the MySQL Enterprise Firewall plugin by clicking 'Install Firewall'. After instalation, you can manage the user rules in the 'Users and Privileges' section of Workbench.

Start learning allowable rules by setting the Firewall state of a user to RECORDING. When you are done switch to protection mode.

Other operations, like manually adding and removing rules or saving to or loading from a file are also available on the same section.

Because clicking **Enable Firewall** from MySQL Workbench is a runtime operation, enabling the `mysql_firewall_mode` option in the configuration option will enable it after a restart. Manually edit the MySQL configuration file, or use MySQL Workbench to edit it.

Figure 6.43 Edit MySQL Enterprise Firewall Options Using Workbench



Locate option: mysql_firewall

☐	lock_wait_timeout	31536000	Timeout for metadata locks
☐	old		Cause the server to revert to certain behaviors present in older versions
☐	stored_program_cache	256	Sets a "soft" upper limit for number of cached stored routines per connection. stored functions are cached separately; this variable sets size for both of thes

Firewall

☐	mysql_firewall_max_query_size	4096	Maximum size of recorded statements
✓	mysql_firewall_mode		Whether MySQL Enterprise Firewall is operational
☐	mysql_firewall_trace		Whether to enable firewall trace

SQL

☐	ansi		Use ANSI SQL syntax instead of MySQL syntax
☐	explicit_defaults_for_timestamp		Whether TIMESTAMP columns are nullable and have DEFAULT NULL
✓	sql-mode	"STRICT_TRAN	Set the SQL server mode

International

☐	character-set-filesystem	binary	Set the file system character set
✓	character-set-server	utf8	Specify default character set
☐	character-sets-dir	...	Directory where character sets are installed
☐	collation-server	latin1_swedish_	Specify default collation

Configuration File: C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

mysqld

Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*

Firewall Rules and Information

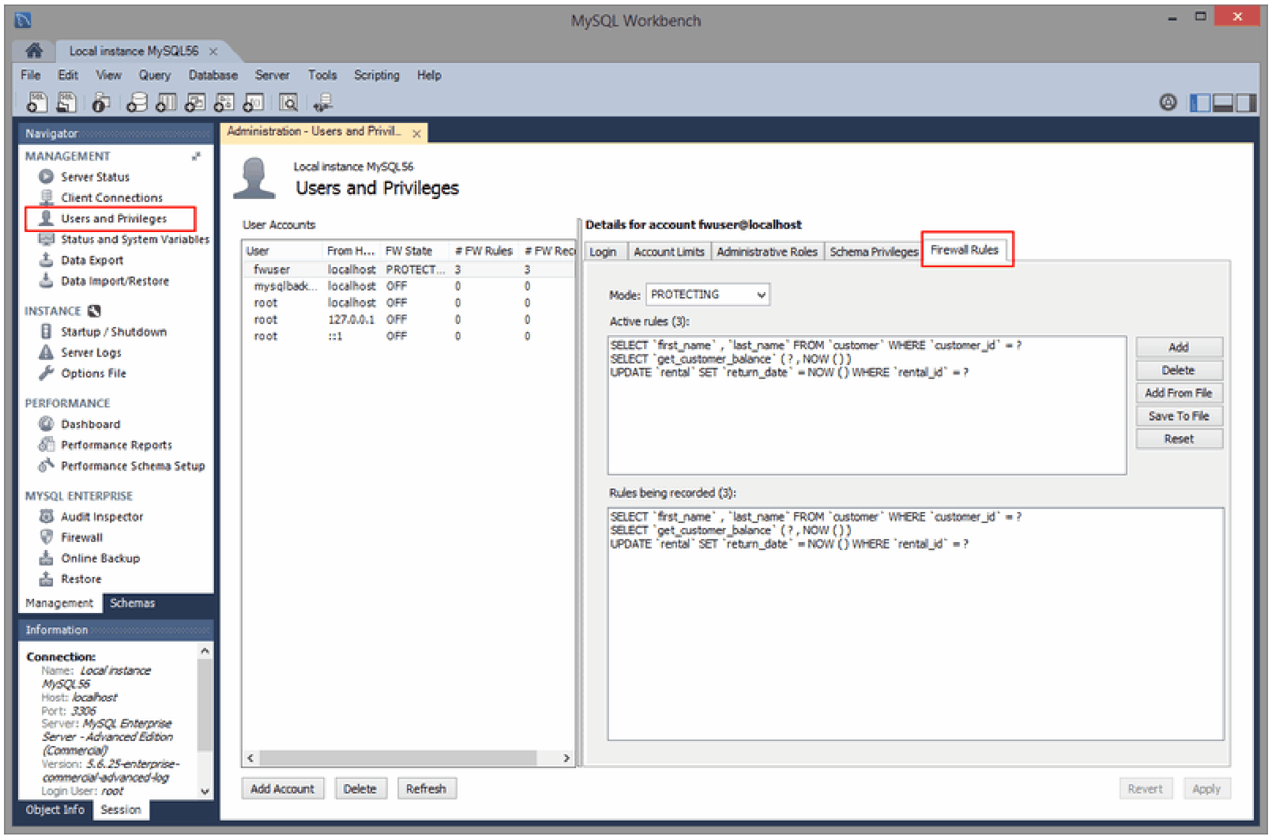
The **Firewall Rules** tab lists the active and recorded rules for a given user, the state of each rule, and includes options to add, delete, and save rules.

- **State** (mode): Options include **OFF** (disables the firewall), **PROTECTING** (enables the whitelist), **RECORDING** (training mode), and **RESET** (removes the rules). For additional information about the meaning of these states, see MySQL Enterprise Firewall Procedures and Functions.
- Administrative actions include **Add** and **Delete** for individual rules, and **Clear** to clear (remove) all rules. **Add From File** prompts for a firewall rules text file (defaults to the `.fwr` extension) that contains one rule per line, and **Save To File** saves the current rules.
- *Active rules* are used in PROTECTIVE mode, and *Rules being recorded* are entries still being RECORDED. Switching from RECORDING to PROTECTING mode copies the recorded rules into the active rule subset.

Note

MySQL Workbench executes queries, gets variables, and performs a lot of checks. For this reason, MySQL Workbench is more useful as an administration tool for MySQL Enterprise Firewall than a tool to record rules. For example, RECORDING rules in MySQL Workbench will record the behind-the-scenes operations performed by MySQL Workbench for the MySQL user. Also, using MySQL Workbench by a MySQL user in PROTECTING mode will attempt to execute operations that a typical firewalled MySQL user might not have access to.

Figure 6.44 MySQL Enterprise Firewall Rules



Connection:
Name: *Local instance MySQL56*
Host: *localhost*
Port: *3306*
Server: *MySQL Enterprise Server - Advanced Edition (Commercial)*
Version: *5.6.25-enterprise-commercial-advanced-log*
Login User: *root*

User	From H...	FW State	# FW Rules	# FW Rec
fwuser	localhost	PROTECT...	3	3
mysqlback...	localhost	OFF	0	0
root	localhost	OFF	0	0
root	127.0.0.1	OFF	0	0
root	::1	OFF	0	0

Mode: PROTECTING

Active rules (3):

```
SELECT `first_name` , `last_name` FROM `customer` WHERE `customer_id` = ?
SELECT `get_customer_balance` ( ? , NOW ( ) )
UPDATE `rental` SET `return_date` = NOW ( ) WHERE `rental_id` = ?
```

Rules being recorded (3):

```
SELECT `first_name` , `last_name` FROM `customer` WHERE `customer_id` = ?
SELECT `get_customer_balance` ( ? , NOW ( ) )
UPDATE `rental` SET `return_date` = NOW ( ) WHERE `rental_id` = ?
```

6.9 The wbcopytables Tool

`wbcopytables` is a command line utility included in MySQL Workbench that allows you to copy table data from a supported source database server to MySQL. It is used by the Workbench Migration wizard to copy data after the schema is migrated and created in the target MySQL server.

`wbcopytables` can connect to the source database using either ODBC, the Python DBAPI, or the native MySQL client library.

The copy executes a `SELECT` statement on the source database, and then `INSERT`'s the retrieved rows into the target MySQL server.

Table 6.2 File Location (Default)

Operating System	Location
Linux	`/usr/bin/wbcopytables`

Operating System	Location
OS X	`/Applications/MySQLWorkbench.app/Contents/MacOS/wbcopytables`
Windows	`C:\Program Files (x86)\MySQL\MySQL Workbench 6.3\wbcopytables.exe`

Connection Parameters

Options for the source connection are:

`--odbc-source=`*`ODBC_connection_string`*: The syntax of the ODBC connection string uses standard ODBC syntax. You can also use a ODBC data source name (DSN).

`--mysql-source=`*`MySQL_connection_string`*: Use for MySQL sources (when doing a MySQL to MySQL migration/copy). It uses the same syntax as the MySQL Utilities:

- For TCP/IP connections: *`username[:password]@host:port`*
- For local socket connections: *`username[:password]@::socket_path`*

You can pass the connection password by using the `--source-password` option.

For the target connection, the option is: `--target=`*`MySQL_connection_string`*.

You can use the `--passwords-from-stdin` option to pass a passwords through STDIN. Source and target passwords must be separated by a tab character.

You can use ODBC specific data source options from the source RDBMS to specify the number of rows to fetch at a time for the source `SELECT` statement.

Table Specification

One or more tables can be specified in the command line for the copy operation. There are two copy types:

- Full table copy: `--table`
- Range copy: `--table-range`

Both table copy types require a set of common arguments:

- **Source schema**: The schema/catalog the table belongs to. If quoting is required, it must be done using the syntax from the source RDBMS. For example, SQL Server uses *[square_brackets]*.
- **Source table**: The table to copy. If the source RDBMS uses a schema name in addition to a catalog, both schema and table must be specified here and separated by a dot. For example, *[dbo].[mytable]*.
- **Target schema**: The name of the MySQL schema. If quoting is needed, it must use the MySQL backtick syntax. For example, \`sakila\`.
- **Target table**: The name of the MySQL table.
- **Select expression**: The list of fields to `SELECT`. This will be inserted verbatim into the source `SELECT` statement.

Caution

Use caution as this expression is copied directly into the source `SELECT` statement.

For the select expression, if both the source and target tables have the same fields in the same order, and use compatible types, you can simply pass `*` here, which will build a query like "*SELECT * FROM [dbo].[mytable]*". If not, you can specify the fields as you would in the `SELECT` statement, which are comma (,) separated and with proper escaping/quoting specific to the source RDBMS. You can also specify typecasts and/or data conversions that the source RDBMS supports. For example:

[client_id], [name], [address], AsText([location])

Because each option must be interpreted as a single option by the `wbcopytables` command, you must perform OS shell specific quoting whenever necessary. Usually, quoting your parameter values with 'single' or "double" quotes is enough. This is in addition to any database specific quoting you use.

Full Table Copy

This performs a full `SELECT` on the source table, fetches records, and then inserts them into the target table.

There are no additional arguments required.

The `--table` syntax is as follows:

```
--table Source_Schema Source_Table Target_Schema Target_Table Select_Expression
```

Range Copy

This performs a `SELECT` copy on the source table for the specified range. The table must have a numeric `UNIQUE NOT NULL` or `PRIMARY KEY` that is used to create a `WHERE` expression for the range.

The `--table-range` syntax is as follows:

```
--table-range Source_Schema Source_Table Target_Schema Target_Table Select_Expression Source_Key Range_Start R
```

The generated expression is:

```
key_column >= range_start AND key_column <= range_end
```

If you specify **-1** for Range_End, then the expression will be:

```
key_column >= range_start
```

Other Options

- `--thread-count=`*`Number`*: If you are copying more than one table, you can use this option to divide the tables across several threads. There is no support for dividing a single table across many threads.
- `--count-only`: Only perform a `COUNT(*)` of the `SELECT` which would be generated by the `--table` option that was used. The target schema and table can be omitted in this case.

- `--truncate-target`: Execute a TRUNCATE TABLE command on each target table that is copied.

Trigger Handling

Because there is no way to temporarily disable triggers in MySQL and they can affect the copy process, MySQL Workbench will backup and drop all triggers from the target MySQL database before the copy process starts, and then these triggers are restored after the copy finishes. The triggers are backed up in the target schema under a table named `wb_tmp_triggers`.

- `--disable-triggers-on=`*`Schema_Name`*: Performs the backup and DROP process for all triggers in the specified schema.
- `--reenable-triggers-on=`*`Schema_Name`*: Restores triggers previously backed up to the `wb_tmp_triggers` table.
- `--dont-disable-triggers`: Bypass the trigger disabling step.

Chapter 7 Performance Tools

Table of Contents

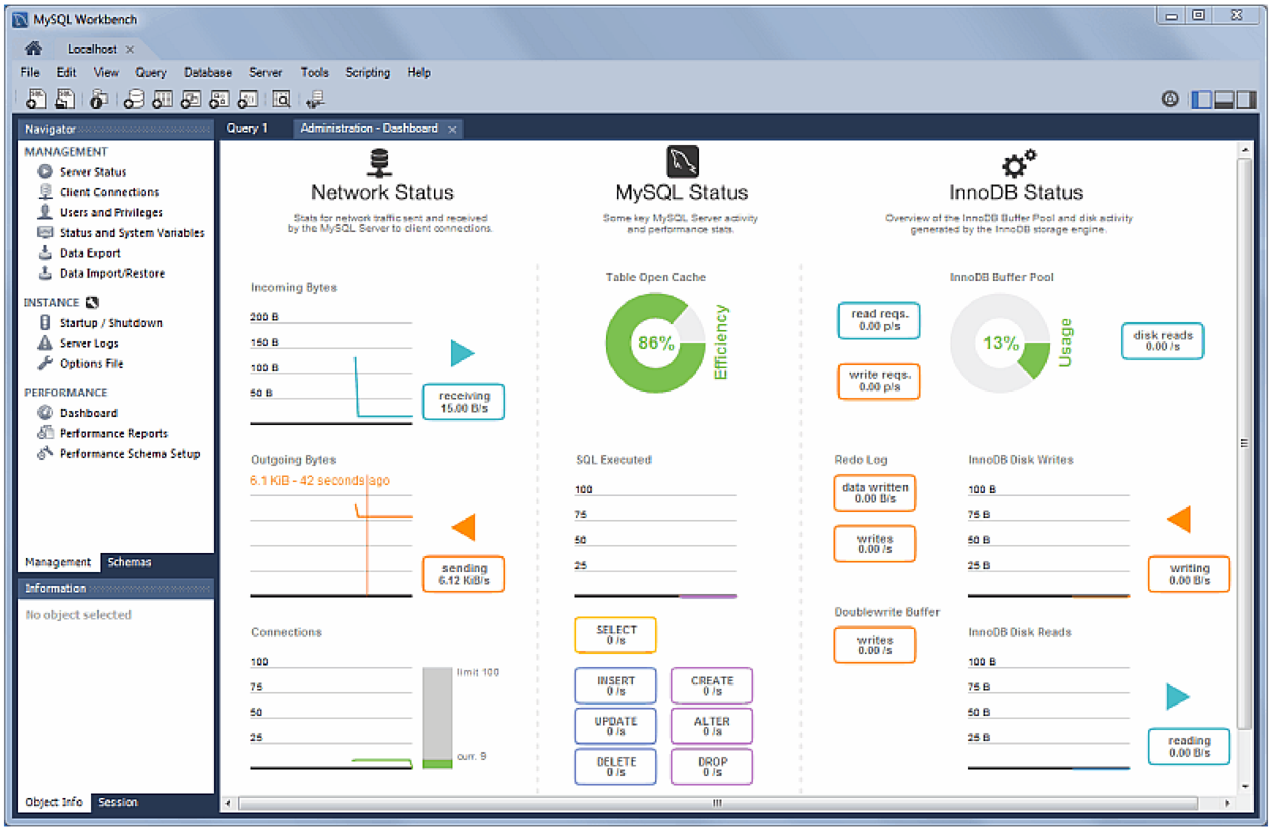
7.1 Performance Dashboard

View server performance statistics in a graphical dashboard.

Note

This feature requires MySQL Server 5.6 or higher.

Figure 7.1 Performance: Dashboard



Network Status

This highlights statistics for network traffic sent and received by the MySQL server over client connections. Data points include the **Incoming Network Traffic**, **Outgoing Network Traffic**, and **Client Connections**.

MySQL Status

This highlights the primary MySQL server activity and performance statistics. Data points include the **Table Open Cache** efficiency, **SQL Statements Executed**, and counts (per second) for SELECT, INSERT, UPDATE, DELETE, CREATE, ALTER, and DROP statements.

InnoDB Status

This provides an overview of the InnoDB Buffer Pool and disk activity that is generated by the InnoDB storage engine. Data points are separated into three groups:

Note

Hover over a graph to see additional information, such as a total count.

- **Usage**
  - *Read Requests*: The number of logical read requests (per second) InnoDB has performed to the buffer pool.
  - *Write Requests*: The number of logical write requests (per second) InnoDB has performed to the buffer pool.
  - *Disk Reads*: The number of logical reads that InnoDB could not satisfy from the buffer pool. As a result, these had to be read from the disk.
  - *InnoDB Buffer Pool Usage*: The percentage of the InnoDB buffer pool that is in use. Hover over the graphic to see additional information, such as Usage Rate and Pages Free.
- **Writes**
  - *Data Written*: The number of writes written to the InnoDB redo log file.
  - *Writes*: The number of physical writes written to the InnoDB redo log file.
  - *InnoDB Disk Writes*: Hover over this dynamic graph to see the number of disk writes over a specific second. The available range includes the last 120 seconds.
  - *Writing*: Total amount of data (in bytes) written using file operations by the InnoDB storage engine.
- **Reads**
  - *Doublewrite Buffer Writes*: The number of doublewrite operations that were performed.
  - *InnoDB Disk Reads*: Hover over this dynamic graph to see the number of disk reads over a specific second. The available range includes the last 120 seconds.
  - *Reading*: Total amount of data (in bytes) read in file operations by the InnoDB storage engine.

7.2 Performance Schema Reports

Performance schema based reports provide insight into the MySQL server operations through helpful high-level reports. MySQL Workbench uses the **SYS** views on the Performance Schema to generate over 20 reports to help analyze the performance of your MySQL databases. Reports help analyze IO hotspots, discover high cost SQL statements, and review wait statistics and InnoDB engine metrics. For additional information about the SYS schema, see MySQL sys Schema.

Note

This feature requires MySQL Server 5.6 or higher.

Installation and Configuration

A GUI for configuring and fine tuning the Performance Schema instrumentation. Initially, this loads an "Easy Setup" page that is enough for most users. Slide the "Performance Schema Full Enabled" slider to **YES** to enable all available Performance Schema instruments.

The SYS schema is bundled with MySQL Server 5.7 and above, and MySQL Workbench uses that version. However, for MySQL Server 5.6, Workbench installs its own bundled version of the SYS schema.

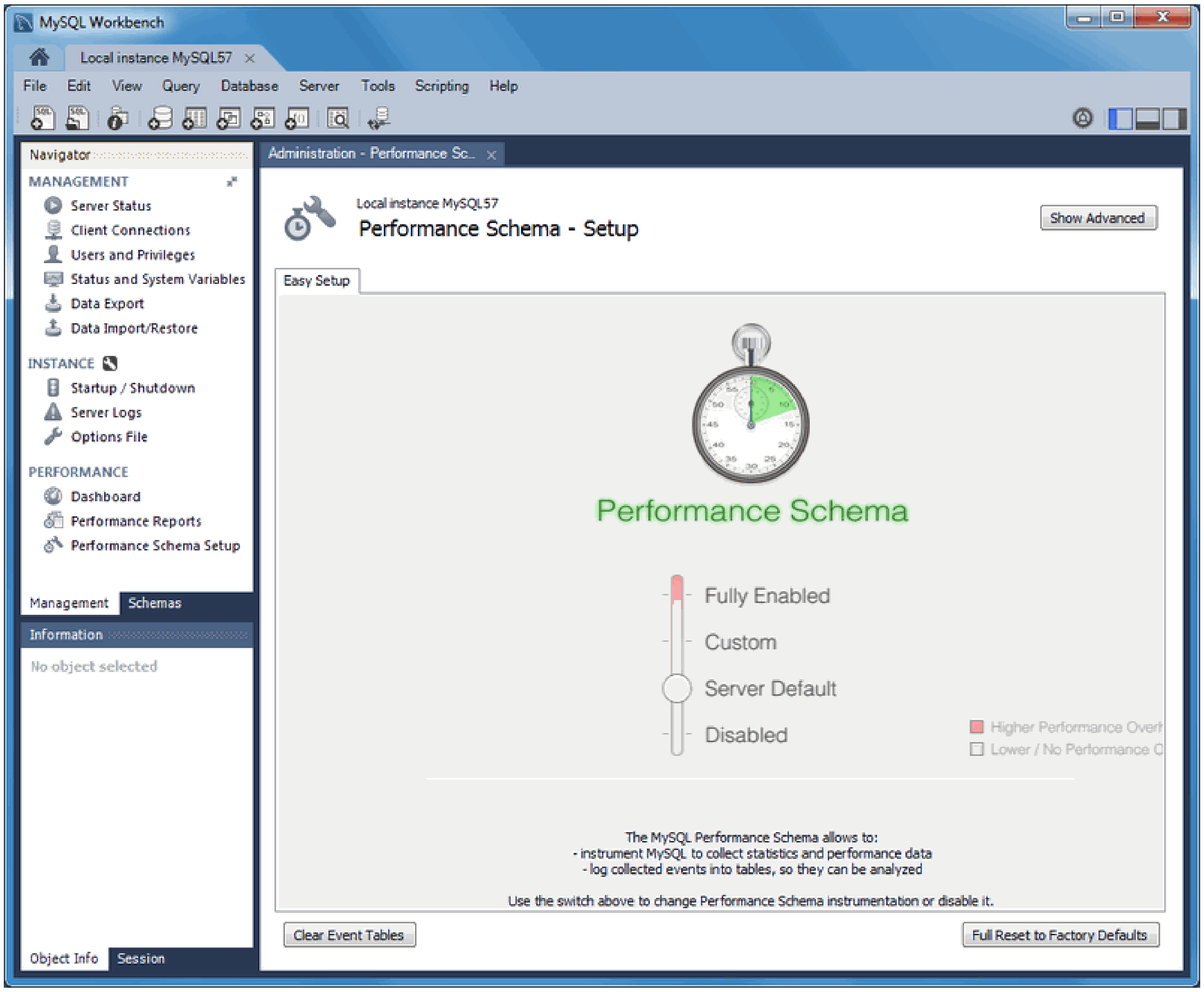
Note

This feature requires MySQL Server 5.6 or higher.

Note

The size of the saved digested query is determined by the MySQL server.

Figure 7.2 Performance Schema Setup: Easy Setup

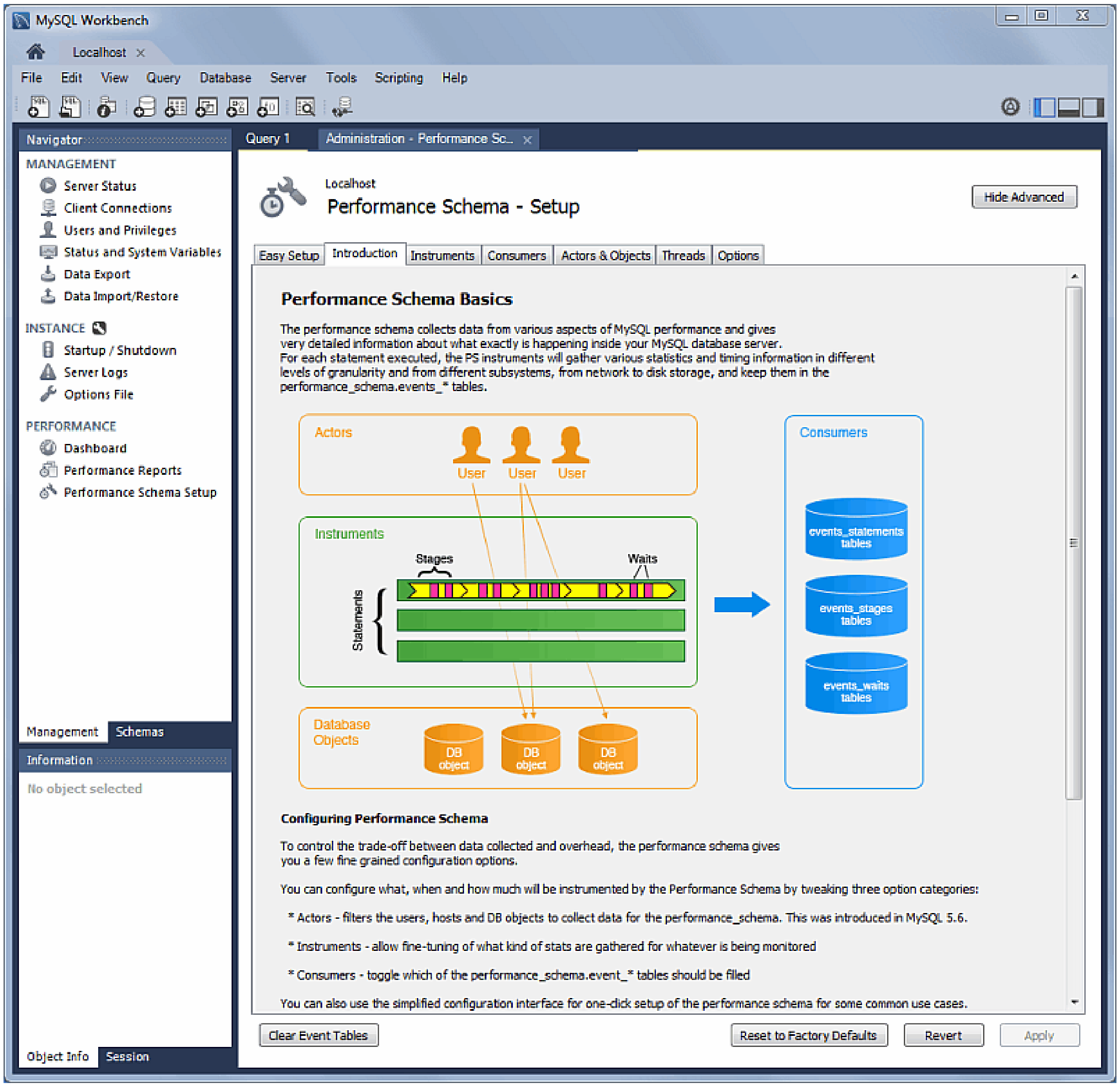


The MySQL Performance Schema allows to:
- instrument MySQL to collect statistics and performance data
- log collected events into tables, so they can be analyzed

Use the switch above to change Performance Schema instrumentation or disable it.

Clicking **Show Advanced** provides methods to fine tune the Performance Schema instrumentation.

Figure 7.3 Performance Schema Setup: Introduction



Performance Schema Basics

The performance schema collects data from various aspects of MySQL performance and gives very detailed information about what exactly is happening inside your MySQL database server. For each statement executed, the PS instruments will gather various statistics and timing information in different levels of granularity and from different subsystems, from network to disk storage, and keep them in the performance_schema.events_* tables.

Configuring Performance Schema

To control the trade-off between data collected and overhead, the performance schema gives you a few fine grained configuration options.

You can configure what, when and how much will be instrumented by the Performance Schema by tweaking three option categories:

* Actors - filters the users, hosts and DB objects to collect data for the performance_schema. This was introduced in MySQL 5.6.

* Instruments - allow fine-tuning of what kind of stats are gathered for whatever is being monitored

* Consumers - toggle which of the performance_schema.event_* tables should be filled

You can also use the simplified configuration interface for one-click setup of the performance schema for some common use cases.

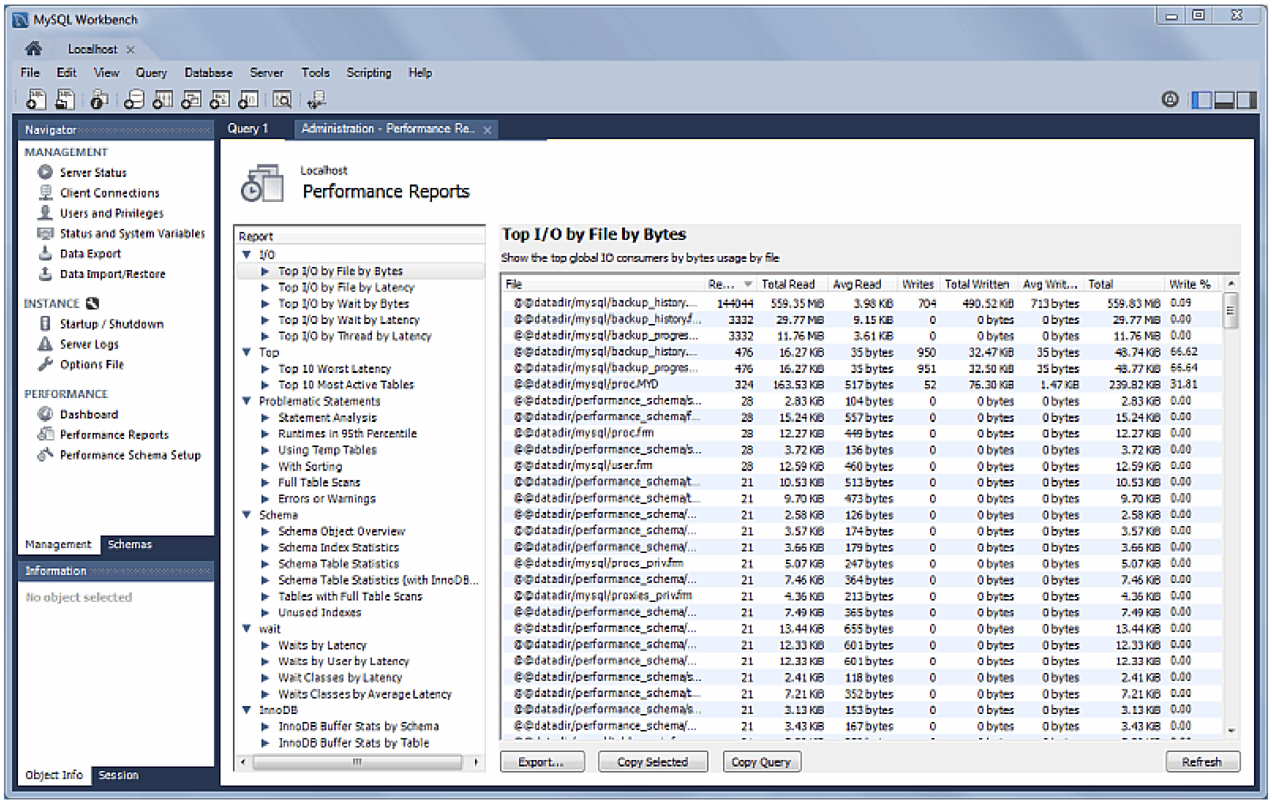
Performance Report Controls

Performance report data can be viewed and exported using the following controls:

- **Export**: Export all entries and associated data (and column headings) from the current performance report, which includes all queries and values. Opens a file dialog for export.
- **Copy Selected**: Copies a single entry and associated data (and column headings) from the current performance report. Saves to the system's clipboard. An example:
- **Copy Query**: Copies the SQL query that generated the performance report. Saves to the system's clipboard.
- **Refresh**: Refreshes (reloads) the performance report.

Performance Report Descriptions

Figure 7.4 Performance Reports: Top I/0 By Bytes



Top I/O by File by Bytes

Show the top global IO consumers by bytes usage by file

File	Re...	Total Read	Avg Read	Writes	Total Written	Avg Writ...	Total	Write %
@@datadir/mysql/backup_history....	144044	559.35 MiB	3.98 KiB	704	490.52 KiB	713 bytes	559.83 MiB	0.09
@@datadir/mysql/backup_history.f...	3332	29.77 MiB	9.15 KiB	0	0 bytes	0 bytes	29.77 MiB	0.00
@@datadir/mysql/backup_progres...	3332	11.76 MiB	3.61 KiB	0	0 bytes	0 bytes	11.76 MiB	0.00
@@datadir/mysql/backup_history....	476	16.27 KiB	35 bytes	950	32.47 KiB	35 bytes	48.74 KiB	66.62
@@datadir/mysql/backup_progres...	476	16.27 KiB	35 bytes	951	32.50 KiB	35 bytes	48.77 KiB	66.64
@@datadir/mysql/proc.MYD	324	163.53 KiB	517 bytes	52	76.30 KiB	1.47 KiB	239.82 KiB	31.81
@@datadir/performance_schema/s...	28	2.83 KiB	104 bytes	0	0 bytes	0 bytes	2.83 KiB	0.00
@@datadir/performance_schema/f...	28	15.24 KiB	557 bytes	0	0 bytes	0 bytes	15.24 KiB	0.00
@@datadir/mysql/proc.frm	28	12.27 KiB	449 bytes	0	0 bytes	0 bytes	12.27 KiB	0.00
@@datadir/performance_schema/s...	28	3.72 KiB	136 bytes	0	0 bytes	0 bytes	3.72 KiB	0.00
@@datadir/mysql/user.frm	28	12.59 KiB	460 bytes	0	0 bytes	0 bytes	12.59 KiB	0.00
@@datadir/performance_schema/t...	21	10.53 KiB	513 bytes	0	0 bytes	0 bytes	10.53 KiB	0.00
@@datadir/performance_schema/t...	21	9.70 KiB	473 bytes	0	0 bytes	0 bytes	9.70 KiB	0.00
@@datadir/performance_schema/...	21	2.58 KiB	126 bytes	0	0 bytes	0 bytes	2.58 KiB	0.00
@@datadir/performance_schema/...	21	3.57 KiB	174 bytes	0	0 bytes	0 bytes	3.57 KiB	0.00
@@datadir/performance_schema/...	21	3.66 KiB	179 bytes	0	0 bytes	0 bytes	3.66 KiB	0.00
@@datadir/mysql/procs_priv.frm	21	5.07 KiB	247 bytes	0	0 bytes	0 bytes	5.07 KiB	0.00
@@datadir/performance_schema/...	21	7.46 KiB	364 bytes	0	0 bytes	0 bytes	7.46 KiB	0.00
@@datadir/mysql/proxies_priv.frm	21	4.36 KiB	213 bytes	0	0 bytes	0 bytes	4.36 KiB	0.00
@@datadir/performance_schema/...	21	7.49 KiB	365 bytes	0	0 bytes	0 bytes	7.49 KiB	0.00
@@datadir/performance_schema/...	21	13.44 KiB	655 bytes	0	0 bytes	0 bytes	13.44 KiB	0.00
@@datadir/performance_schema/...	21	12.33 KiB	601 bytes	0	0 bytes	0 bytes	12.33 KiB	0.00
@@datadir/performance_schema/...	21	12.33 KiB	601 bytes	0	0 bytes	0 bytes	12.33 KiB	0.00
@@datadir/performance_schema/s...	21	2.41 KiB	118 bytes	0	0 bytes	0 bytes	2.41 KiB	0.00
@@datadir/performance_schema/t...	21	7.21 KiB	352 bytes	0	0 bytes	0 bytes	7.21 KiB	0.00
@@datadir/performance_schema/s...	21	3.13 KiB	153 bytes	0	0 bytes	0 bytes	3.13 KiB	0.00
@@datadir/performance_schema/...	21	3.43 KiB	167 bytes	0	0 bytes	0 bytes	3.43 KiB	0.00

The available performance reports include (this is a non-exhaustive list):

- *Top File I/O Activity Report*: Show the files performing the most IOs (in bytes)
- *Top I/O by File by Time*: Show the highest IO usage by file and latency
- *Top I/O by Event Category*: Show the highest IO Data usage by event categories
- *Top I/O in Time by Event Categories*: Show the highest IO time consumers by event categories
- *Top I/O Time by User/Thread*: Show the top IO time consumers by user/thread
- *Statement Analysis*: Lists statements with aggregated statistics
- *Statements in Highest 5 Percent by Runtime*: Lists the top 5% statements with the highest runtime (in microseconds),
- *Using Temp Tables*: Lists all statements that use temporary tables -- accesses the highest number of disk temporary tables, then memory temp tables
- *With Sorting*: List all normalized statements that have done sorts, and accesses them in the following priority order -- sort_merge_passes, sort_scans, then sort_rows
- *Full Table Scans*: Lists statements that performed a full table scan. Accesses query performance and the where clause(s), and if no index is used then it recommends adding indexes for large tables
- *Errors or Warnings*: Lists statements that have raised errors or warnings

- *Schema Object Overview (High Overhead)*: Shows counts by object type for each schema

  Note

  This can take a long time to execute on instances with a large number of objects.

- *Schema Index Statistics*
- *Schema Table Statistics*
- *Schema Table Statistics (with InnoDB buffer)*
- *Tables with Full Table Scans*: Finds tables that are being accessed by full table scans, ordering them by the number of rows scanned (DESC)
- *Unused Indexes*: List of indexes that were never used since the server started or since P_S data collection started
- *Waits by Time*: Lists the top wait events by their total time, ignoring idle (this often contain large values)
- *Waits by User by Time*: Lists the top wait events by their total time, ignoring idle (this often contain large values)
- *Wait Classes by Time*: Lists the top wait classes by total time, ignoring idle (this often contain large values)
- *Waits Classes by Average Time*: Lists the top wait classes by average time, ignoring idle (this often contain large values)
- *InnoDB Buffer Stats by Schema*: Summarizes the output of the `INFORMATION_SCHEMA.INNODB_BUFFER_PAGE` table, aggregating it by schema
- *InnoDB Buffer Stats by Table*: Summarizes the output of the `INFORMATION_SCHEMA.INNODB_BUFFER_PAGE` table, aggregating it by schema and table name

7.3 Visual Explain Plan

The **Visual Explain** feature generates and displays a visual representation of the MySQL `EXPLAIN` statement by using extended information available in the extended JSON format.

Note

The extended EXPLAIN format is available as of MySQL server 5.6.5.

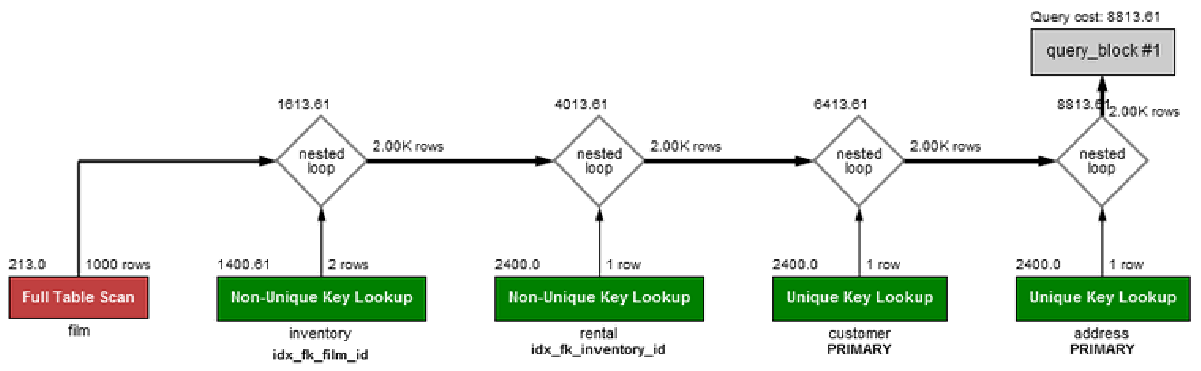
MySQL Workbench provides all of the EXPLAIN formats for executed queries including the raw extended JSON, traditional format, and visual query plan.

Visual Explain Conventions

An example Visual Explain diagram:

```
SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer, address.phone, film.title
FROM rental
INNER JOIN customer ON rental.customer_id = customer.customer_id
INNER JOIN address ON customer.address_id = address.address_id
INNER JOIN inventory ON rental.inventory_id = inventory.inventory_id
INNER JOIN film ON inventory.film_id = film.film_id
WHERE rental.return_date IS NULL
AND rental_date + INTERVAL film.rental_duration DAY < CURRENT_DATE()
LIMIT 5;
```

Figure 7.5 A Visual Explain Example



The order of execution is bottom to top, and left to right.

Graphic Conventions

- Standard Boxes: tables
- Rounded boxes: operations such as GROUP and SORT
- Framed boxes: subqueries
- Diamonds: joins

Textual Conventions

- Standard text below boxes: table (or alias) name
- Bold text below boxes: key/index that was used
- Number in top right of a box: number of rows used from the table after filtering
- Number in top left of a box: relative cost of accessing that table (requires MySQL 5.7 or greater)
- Number to the right of nested loop diamonds: number of rows produced by the JOIN
- Number above the loop diamonds: relative cost of the JOIN (requires MySQL 5.7 or greater)

The associated colors and descriptions used in the Visual Explain diagram:

Table 7.1 Visual Explain Diagram Information

System Name	Color	Text on Visual Diagram	Tooltip related information
SYSTEM	Blue	Single row: system constant	Very low cost

System Name	Color	Text on Visual Diagram	Tooltip related information
CONST	Blue	Single row: constant	Very low cost
EQ_REF	Green	Unique Key Lookup	Low cost -- The optimizer is able to find an index that it can use to retrieve the required records. It is fast because the index search directly leads to the page with all the row data
REF	Green	Non-Unique Key Lookup	Low-medium -- Low if the number of matching rows is small; higher as the number of rows increases
FULLTEXT	Yellow	Fulltext Index Search	Specialized FULLTEXT search. Low -- for this specialized search requirement
REF_OR_NULL	Green	Key Lookup + Fetch NULL Values	Low-medium -- if the number of matching rows is small; higher as the number of rows increases
INDEX_MERGE	Green	Index Merge	Medium -- look for a better index selection in the query to improve performance
UNIQUE_SUBQUERY	Orange	Unique Key Lookup into table of subquery	Low -- Used for efficient Subquery processing
INDEX_SUBQUERY	Orange	Non-Unique Key Lookup into table of subquery	Low -- Used for efficient Subquery processing
RANGE	Orange	Index Range Scan	Medium -- partial index scan
INDEX	Red	Full Index Scan	High -- especially for large indexes
ALL	Red	Full Table Scan	Very High -- very costly for large tables, but less of an impact for small ones. No usable indexes were found for the table, which forces the optimizer to search every row. This could also mean that the search range is so broad that the index would be useless.
UNKNOWN	Black	unknown	Note: This is the default, in case a match cannot be determined

Visual Explain Usage

To view a visual explain execution plan, execute your query from the SQL editor and then choose the **Execution Plan** tab in the query results tab. The execution plan defaults to "Visual Explain" but also has a "Tabular Explain" view that is similar to what you would see when executing EXPLAIN in the MySQL client.

7.4 Query Statistics

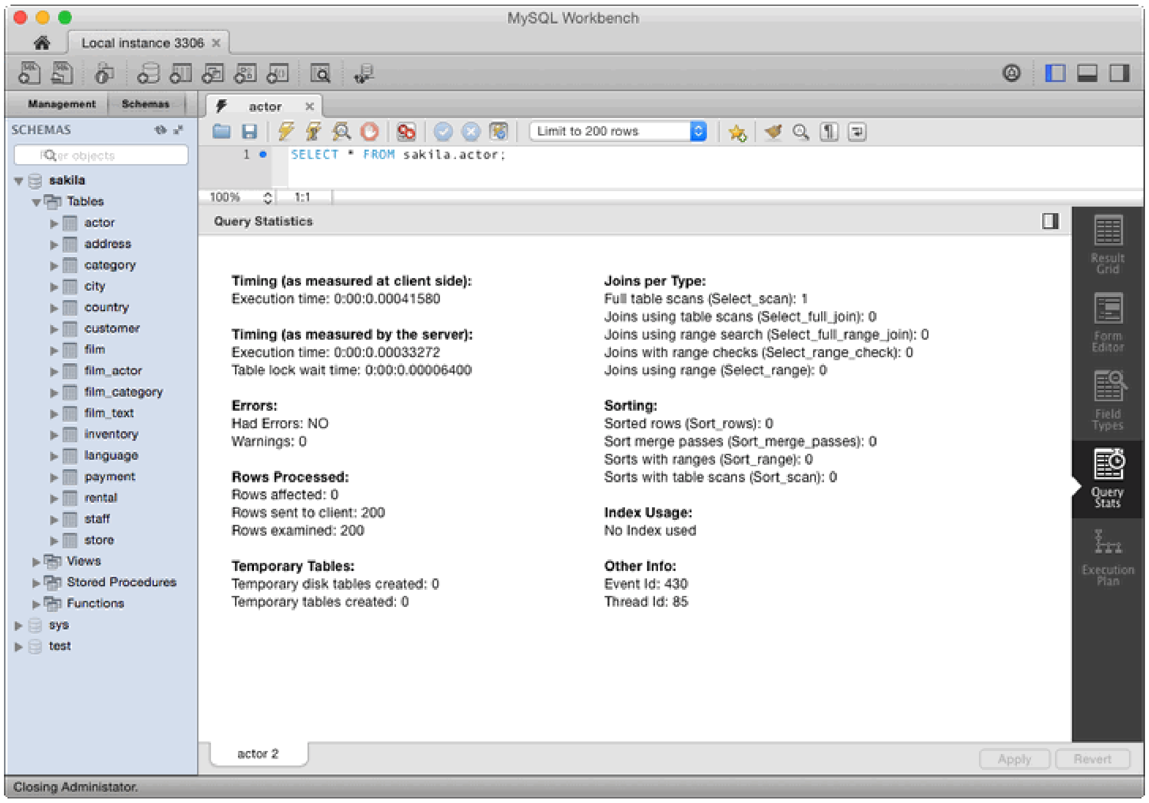
This **Query Stats** SQL editor results tab uses Performance Schema data to gather key statistics collected for executed query, such as timing, temporary tables, indexes, joins, and more.

Requirements

- MySQL server 5.6 or newer

- **Query**, **Collect Performance Schema Stats** enabled.
- The `performance_schema` enabled with statement instrumentation

Figure 7.6 SQL Editor: Query Stats



```
SELECT * FROM sakila.actor;
```

Query Statistics

Timing (as measured at client side):
Execution time: 0:00:0.00041580

Timing (as measured by the server):
Execution time: 0:00:0.00033272
Table lock wait time: 0:00:0.00006400

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 200
Rows examined: 200

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

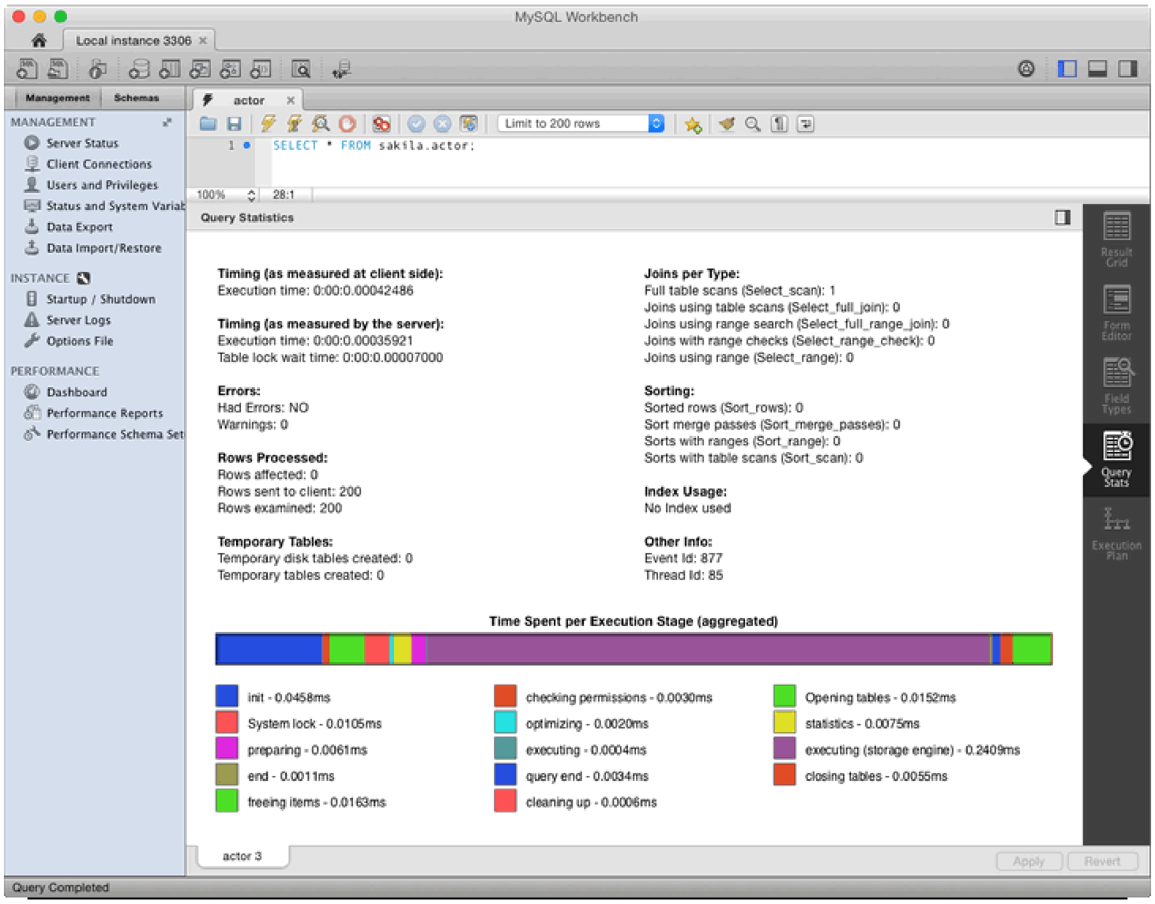
Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 430
Thread Id: 85

Figure 7.7 SQL Editor: Query Stats with Performance Schema Graphs



```
SELECT * FROM sakila.actor;
```

Timing (as measured at client side):
Execution time: 0:00:0.00042486

Timing (as measured by the server):
Execution time: 0:00:0.00035921
Table lock wait time: 0:00:0.00007000

Errors:
Had Errors: NO
Warnings: 0

Rows Processed:
Rows affected: 0
Rows sent to client: 200
Rows examined: 200

Temporary Tables:
Temporary disk tables created: 0
Temporary tables created: 0

Joins per Type:
Full table scans (Select_scan): 1
Joins using table scans (Select_full_join): 0
Joins using range search (Select_full_range_join): 0
Joins with range checks (Select_range_check): 0
Joins using range (Select_range): 0

Sorting:
Sorted rows (Sort_rows): 0
Sort merge passes (Sort_merge_passes): 0
Sorts with ranges (Sort_range): 0
Sorts with table scans (Sort_scan): 0

Index Usage:
No Index used

Other Info:
Event Id: 877
Thread Id: 85

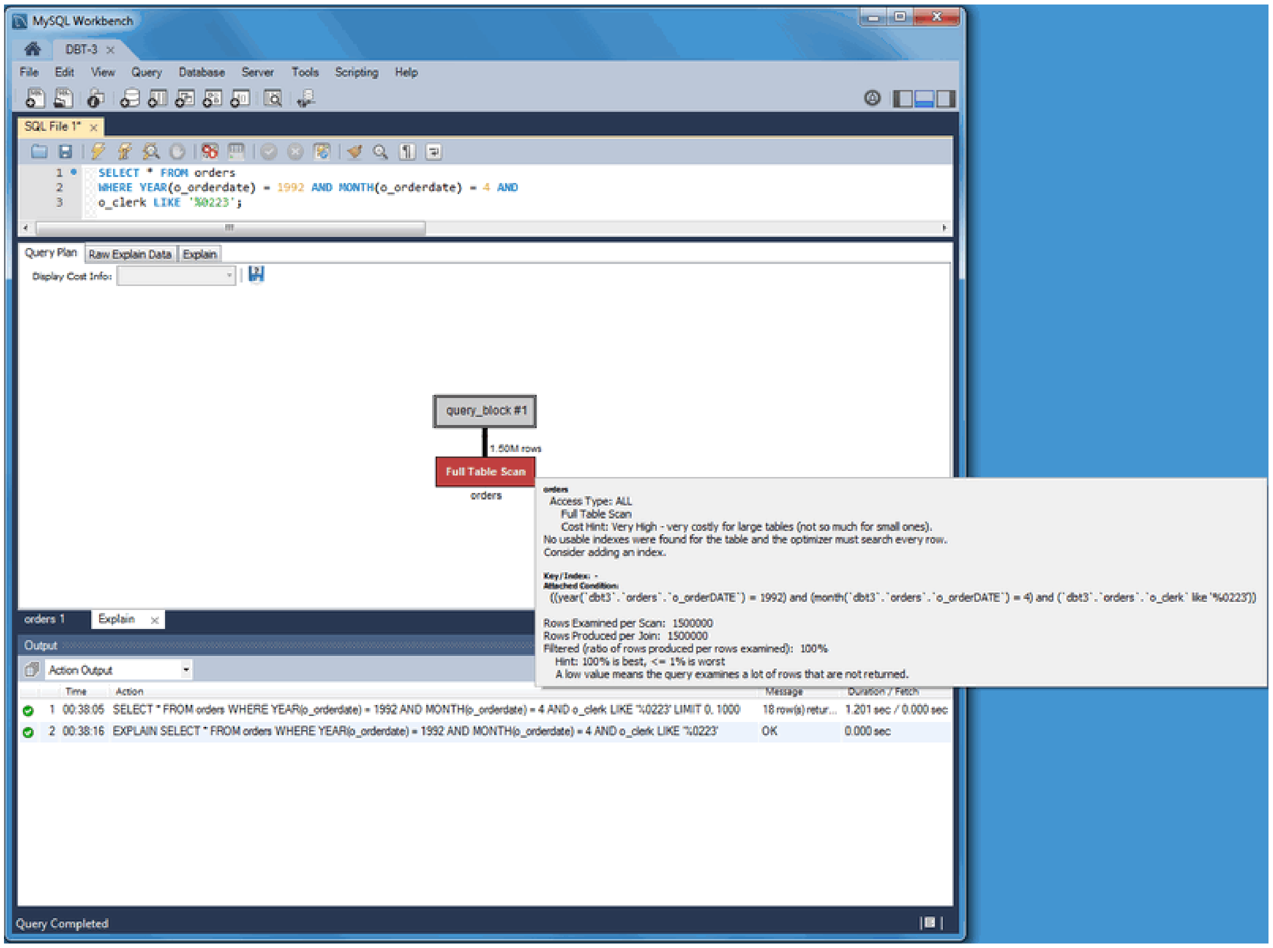
7.5 Tutorial: Using Visual Explain to improve query performance

In this example, Visual Explain helps locate and fix problematic (slow) queries. This tutorial uses the DBT-3 database, and begins with the following query:

```
SELECT * FROM orders
WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4
AND o_clerk LIKE '%0223';
```

In the screenshot below, we executed this query and generated a Visual Explain report by selecting **Query**, **Visual Explain Current Statement** from the main menu.

Figure 7.8 DBT-3 Visual Explain Tutorial: Full Table Scan

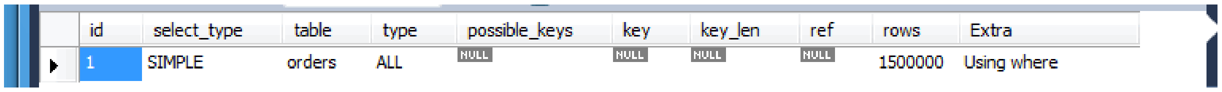


```
SELECT * FROM orders
WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4 AND
o_clerk LIKE '%0223';
```

```
orders
  Access Type: ALL
     Full Table Scan
     Cost Hint: Very High - very costly for large tables (not so much for small ones).
No usable indexes were found for the table and the optimizer must search every row.
Consider adding an index.

Key/Index: -
Attached Condition:
  ((year(`dbt3`.`orders`.`o_orderDATE`) = 1992) and (month(`dbt3`.`orders`.`o_orderDATE`) = 4) and (`dbt3`.`orders`.`o_clerk` like '%0223'))

Rows Examined per Scan:  1500000
Rows Produced per Join:  1500000
Filtered (ratio of rows produced per rows examined):  100%
   Hint: 100% is best, <= 1% is worst
   A low value means the query examines a lot of rows that are not returned.
```

```
   Time      Action                                                                                                   Message            Duration / Fetch
1  00:38:05  SELECT * FROM orders WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4 AND o_clerk LIKE '%0223' LIMIT 0, 1000   18 row(s) retur...   1.201 sec / 0.000 sec
2  00:38:16  EXPLAIN SELECT * FROM orders WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4 AND o_clerk LIKE '%0223'        OK                 0.000 sec
```

Figure 7.9 DBT-3 Visual Explain Tutorial: Full Table Scan: Traditional View



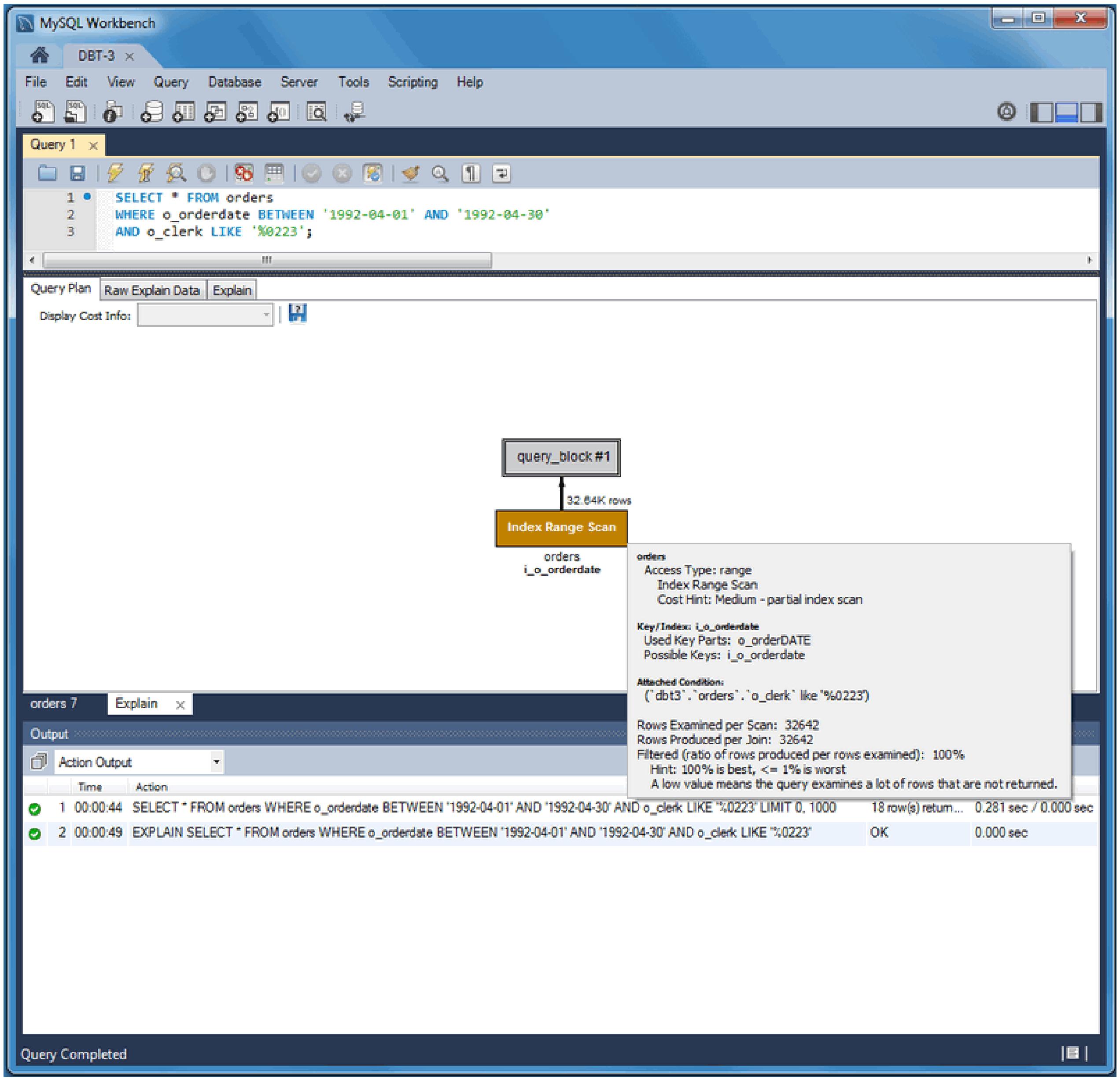
id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	orders	ALL	NULL	NULL	NULL	NULL	1500000	Using where

Notice that ...

Why did this query generate a full table scan? Why is our indexed `o_orderdate` column missing as a possible key? Looking more closely, we notice that our indexed column is being used in an expression as `"WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4"`, so its index is not used. To use the existing index we can adjust the query like so:

```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE '%0223';
```

Figure 7.10 DBT-3 Visual Explain Tutorial: Index Range Scan

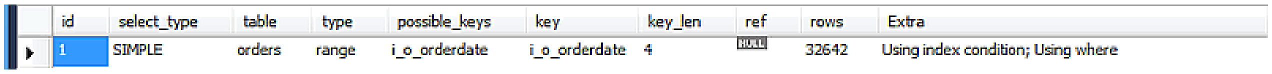


```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE '%0223';
```

```
orders
  Access Type: range
    Index Range Scan
    Cost Hint: Medium - partial index scan

Key/Index: i_o_orderdate
  Used Key Parts: o_orderDATE
  Possible Keys: i_o_orderdate

Attached Condition:
  (`dbt3`.`orders`.`o_clerk` like '%0223')

Rows Examined per Scan: 32642
Rows Produced per Join: 32642
Filtered (ratio of rows produced per rows examined): 100%
   Hint: 100% is best, <= 1% is worst
   A low value means the query examines a lot of rows that are not returned.
```

	Time	Action		
1	00:00:44	SELECT * FROM orders WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30' AND o_clerk LIKE '%0223' LIMIT 0, 1000	18 row(s) return...	0.281 sec / 0.000 sec
2	00:00:49	EXPLAIN SELECT * FROM orders WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30' AND o_clerk LIKE '%0223'	OK	0.000 sec

Figure 7.11 DBT-3 Visual Explain Tutorial: Index Range Scan: Traditional View

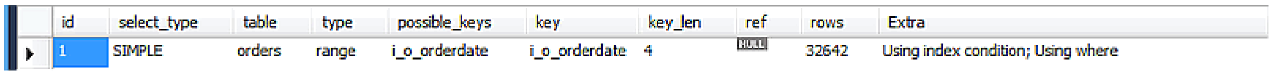


id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	orders	range	i_o_orderdate	i_o_orderdate	4	NULL	32642	Using index condition; Using where

Notice the differences. The Type changed from ALL to range, possible keys (and used key) changed from NULL to i_o_orderdate, and the number of scanned rows changed from 1.5 million to about 33 thousand. Still, scanning 33 thousand rows while returning just 18 is unnecessary, so we focus on the `o_clerk` column. An index here should improve performance: so:

```
CREATE INDEX i_o_clerk ON orders(o_clerk);
```

Figure 7.12 DBT-3 Visual Explain Tutorial: Index Range Scan: Traditional View, After Index

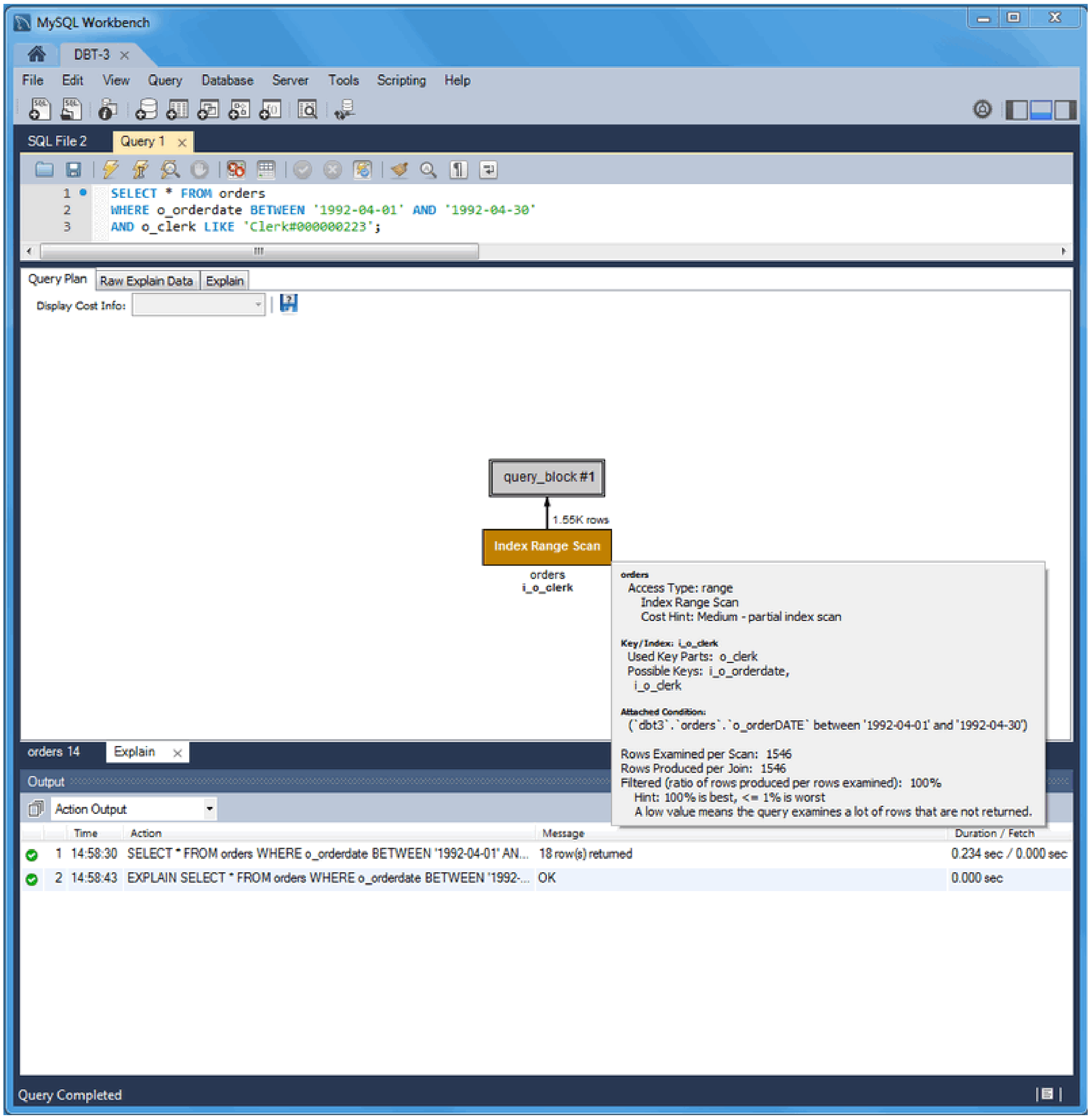


id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	orders	range	i_o_orderdate	i_o_orderdate	4	NULL	32642	Using index condition; Using where

Notice that our new index is not being considered as a possible key. This is because we are searching the suffix of the `o_clerk` column, and indexes do not work with suffixes (although they do work with prefixes). In our simple case, we were being lazy and could have simply used the entire clerk ID. Adjusting the query shows better results:

```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE 'Clerk#000000223';
```

Figure 7.13 DBT-3 Visual Explain Tutorial: Index Range Scan: Better

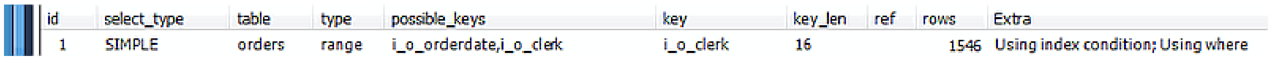


```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE 'Clerk#000000223';
```

```
orders
  Access Type: range
    Index Range Scan
    Cost Hint: Medium - partial index scan

Key/Index: i_o_clerk
  Used Key Parts:  o_clerk
  Possible Keys:  i_o_orderdate,
   i_o_clerk

Attached Condition:
  (`dbt3`.`orders`.`o_orderDATE` between '1992-04-01' and '1992-04-30')

Rows Examined per Scan:  1546
Rows Produced per Join:  1546
Filtered (ratio of rows produced per rows examined):  100%
   Hint:  100% is best, <= 1% is worst
   A low value means the query examines a lot of rows that are not returned.
```

		Time	Action	Message	Duration / Fetch
✓	1	14:58:30	SELECT * FROM orders WHERE o_orderdate BETWEEN '1992-04-01' AN...	18 row(s) returned	0.234 sec / 0.000 sec
✓	2	14:58:43	EXPLAIN SELECT * FROM orders WHERE o_orderdate BETWEEN '1992-...	OK	0.000 sec

Figure 7.14 DBT-3 Visual Explain Tutorial: Index Range Scan: Better (Traditional)



id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	orders	range	i_o_orderdate,i_o_clerk	i_o_clerk	16		1546	Using index condition; Using where

The new `o_clerk` index was considered and used, and our query scanned 1546 rows instead of 32642, and the query execution improved from 0.281 to 0.234 seconds. However, EXPLAIN estimates that this query scans 1546 rows to return 18. After reviewing our query, notice that a multi-column index can meet the conditions of our WHERE clause that is based on both the `o_orderdate` and `o_clerk` columns:

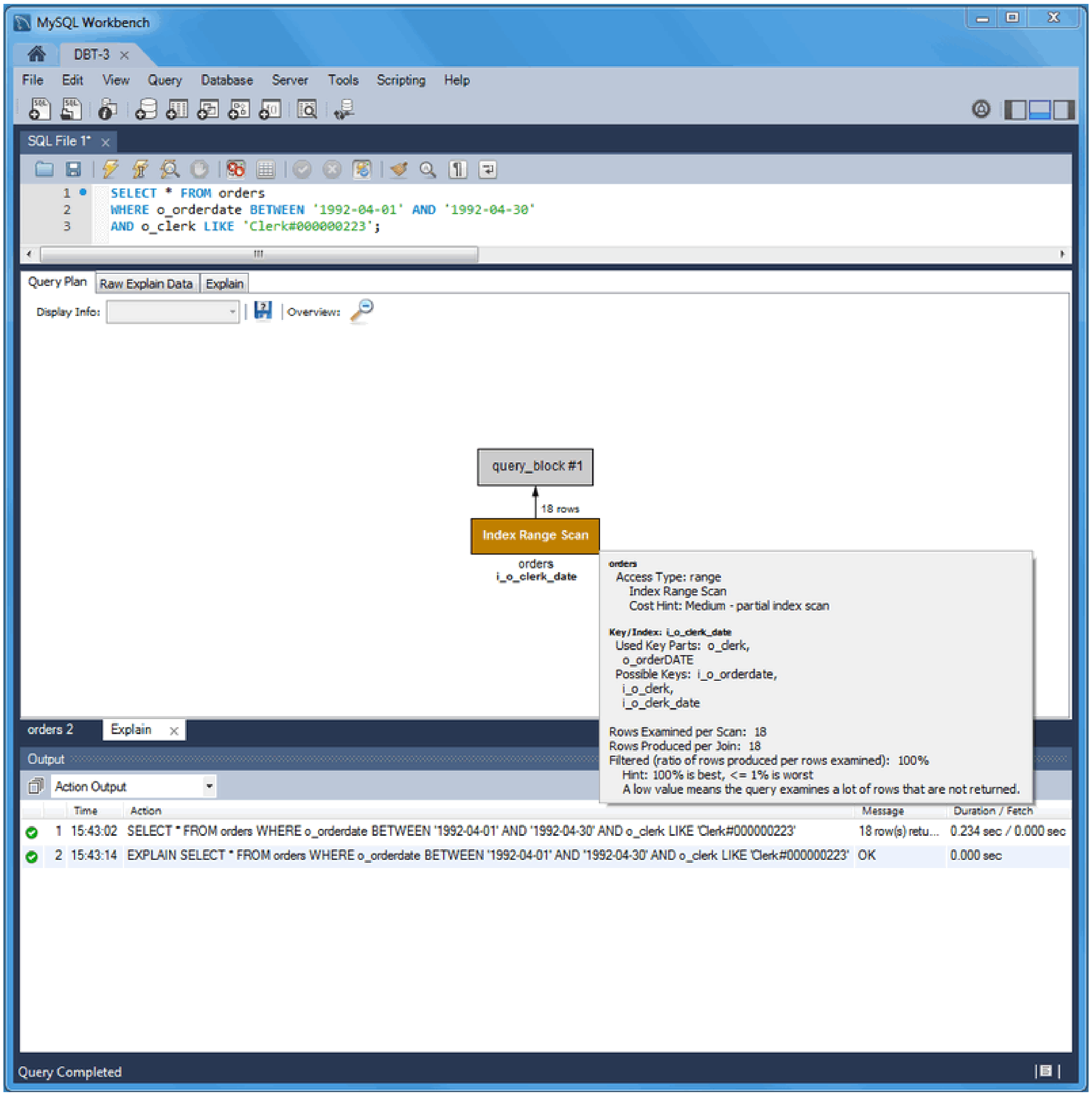
```
CREATE INDEX io_clerk_date ON orders(o_clerk, o_orderdate)
```

Note

We listed `o_clerk` as the first column in our index because `o_orderdate` uses a range. ...

Now, executing the same query shows even better results. An estimated 18 rows are both scanned and returned, and the execution time of our query is 0.234 seconds.

Figure 7.15 DBT-3 Visual Explain Tutorial: Index Range Scan: Best

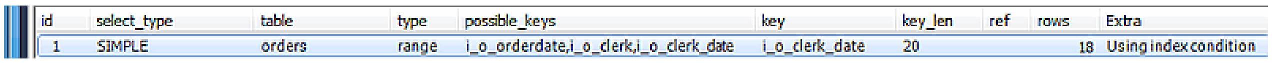


```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE 'Clerk#000000223';
```

```
orders
  Access Type: range
    Index Range Scan
    Cost Hint: Medium - partial index scan

Key/Index: i_o_clerk_date
  Used Key Parts:  o_clerk,
   o_orderDATE
  Possible Keys:  i_o_orderdate,
   i_o_clerk,
   i_o_clerk_date

Rows Examined per Scan:  18
Rows Produced per Join:  18
Filtered (ratio of rows produced per rows examined):  100%
   Hint:  100% is best, <= 1% is worst
   A low value means the query examines a lot of rows that are not returned.
```

```
   Time      Action                                                                                                  Message          Duration / Fetch
1  15:43:02  SELECT * FROM orders WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30' AND o_clerk LIKE 'Clerk#000000223'          18 row(s) retu...  0.234 sec / 0.000 sec
2  15:43:14  EXPLAIN SELECT * FROM orders WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30' AND o_clerk LIKE 'Clerk#000000223'  OK               0.000 sec
```

Figure 7.16 DBT-3 Visual Explain Tutorial: Index Range Scan: Best (traditional)



id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	orders	range	i_o_orderdate,i_o_clerk,i_o_clerk_date	i_o_clerk_date	20		18	Using index condition

To summarize the results:

Table 7.2 DBT-3 Visual Explain Tutorial Query Comparison

Type	Possible keys	Key	Rows Scanned	Duration (seconds)	Extra info	Rows returned
all	NULL	NULL	1.50M	1.201	Using where	18
range	i_o_orderdate	i_o_orderdate	32642	0.281	Using index condition; Using where	18
range	i_o_orderdate, i_o_clerk	i_o_clerk	1546	0.234	Using index condition; Using where	18
range	i_o_orderdate, i_o_clerk, i_o_clerk_date	i_o_clerk_date	18	0.234	Using index condition	18

Chapter 8 Database Development

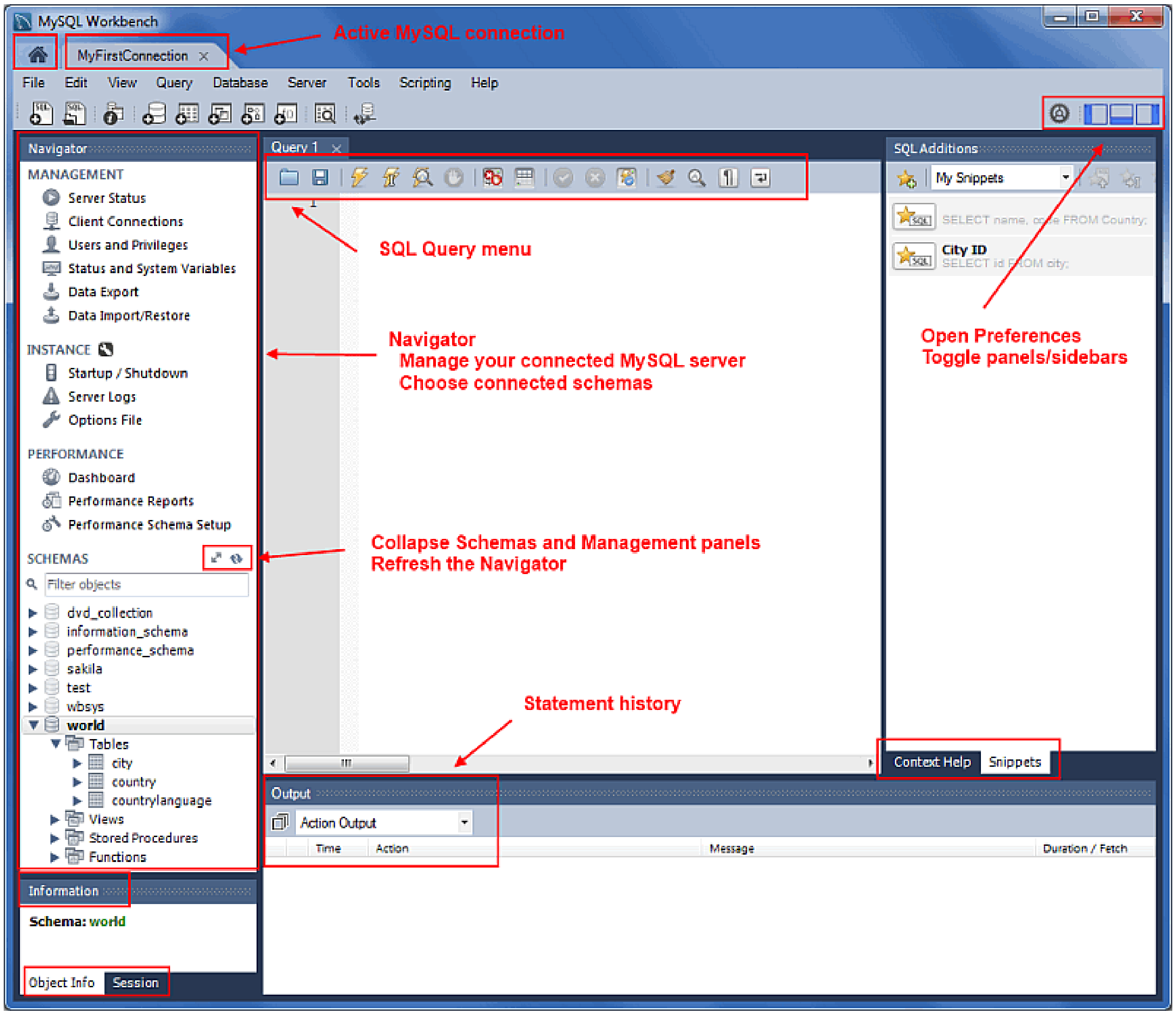
Table of Contents

A set of visual tools to create, edit, and manage SQL queries, database connections, and objects.

8.1 Visual SQL Editor

The visual SQL Editor lets you build, edit and run queries, create and edit data, and view and export results. Color syntax highlighting, context sensitive help and auto-complete helps write and debug SQL statements. The integrated EXPLAIN plans provide data to help optimize the your queries.

Figure 8.1 SQL Editor GUI



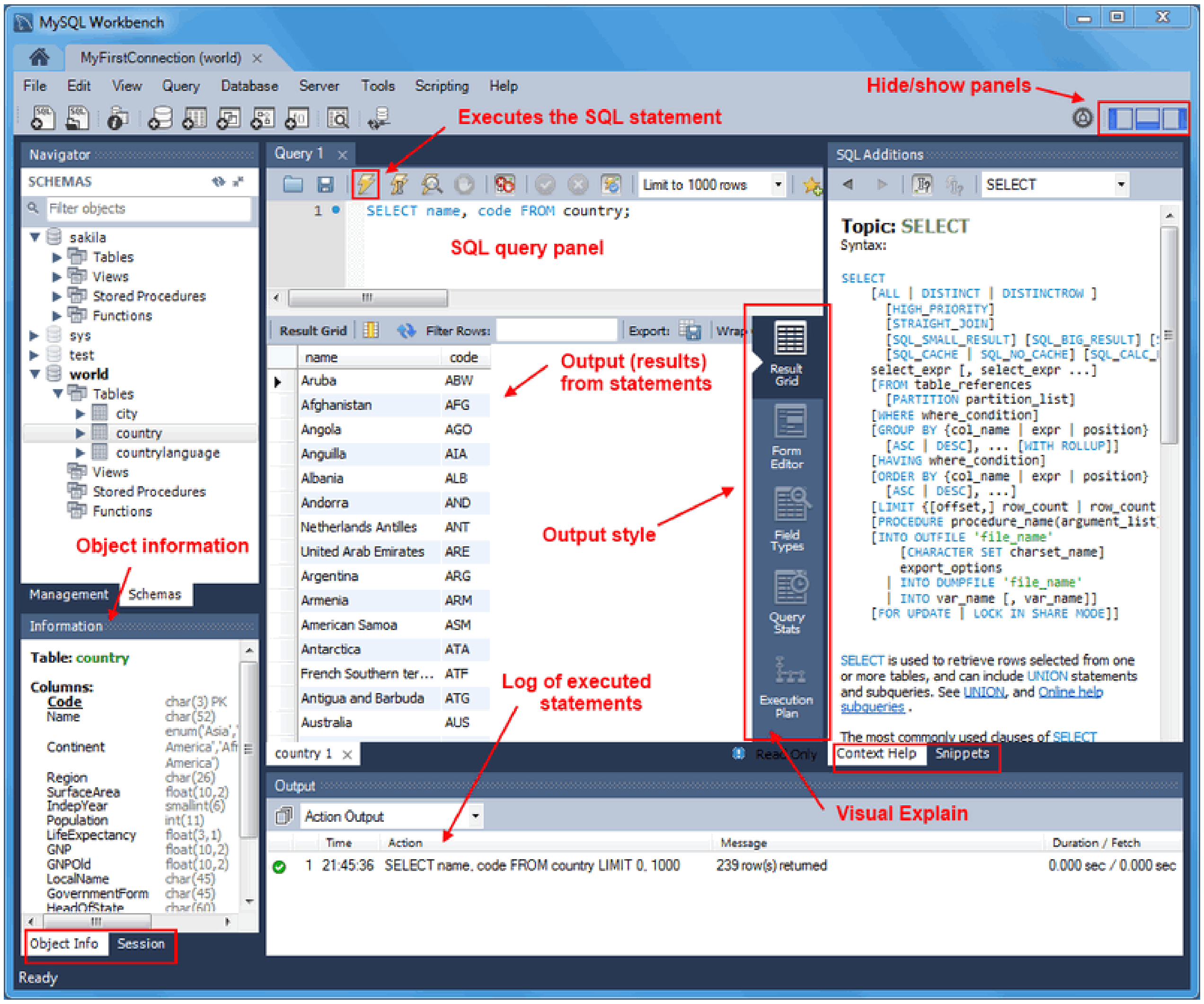
The following sections describe how to use the visual SQL editor.

8.1.1 SQL Query Window

In this area, you can enter SQL statements directly. The statements entered can be saved to a file or snippet for later use. At any point, you can also execute the statements you have entered.

To save a snippet of code entered into the SQL Query panel, click the `Save SQL to Snippets List` icon in the **Snippets** panel, enter a name (optional), and click **OK**. The snippet can be inserted into the SQL Query panel at any time by double-clicking the snippet in the SQL Snippets panel.

Figure 8.2 SQL Editor - SQL Query Panel



```
SELECT name, code FROM country;
```

name	code
Aruba	ABW
Afghanistan	AFG
Angola	AGO
Anguilla	AIA
Albania	ALB
Andorra	AND
Netherlands Antilles	ANT
United Arab Emirates	ARE
Argentina	ARG
Armenia	ARM
American Samoa	ASM
Antarctica	ATA
French Southern ter...	ATF
Antigua and Barbuda	ATG
Australia	AUS

Table: country

Columns:	
Code	char(3) PK
Name	char(52)
Continent	enum('Asia','Europe','North America','Africa'... America')
Region	char(26)
SurfaceArea	float(10,2)
IndepYear	smallint(6)
Population	int(11)
LifeExpectancy	float(3,1)
GNP	float(10,2)
GNPOld	float(10,2)
LocalName	char(45)
GovernmentForm	char(45)
HeadOfState	char(60)

	Time	Action	Message	Duration / Fetch
1	21:45:36	SELECT name, code FROM country LIMIT 0, 1000	239 row(s) returned	0.000 sec / 0.000 sec

Topic: SELECT
Syntax:

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [S
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_
    select_expr [, select_expr ...]
    [FROM table_references
      [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count
    [PROCEDURE procedure_name(argument_list
    [INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]
```

SELECT is used to retrieve rows selected from one or more tables, and can include UNION statements and subqueries. See UNION, and Online help subqueries .

The most commonly used clauses of SELECT

Executing a `SELECT` query will display the associated result set in the SQL View panel, directly below the SQL Query panel. These cells are editable if MySQL Workbench is able to determine how, as for example they are editable if a Primary or Unique key exists within the result set. If not, MySQL Workbench will display a "read-only" icon at the bottom-right corner of the SQL View panel, and hovering the mouse cursor over this icon will provide a hint as to why it's not editable.

Note

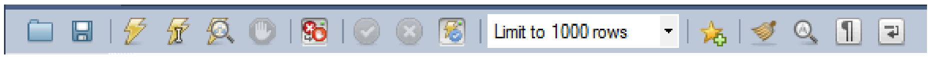
To quickly enter the name of a table, view, or column, double-click the item in the Schemata Palette. The item name will be inserted into the SQL Query panel.

The SQL Editor has several configurable panels and windows, as described in the screenshot above.

8.1.2 SQL Query Window Toolbar

The toolbar features buttons in two locations, in the main toolbar and within the SQL Editor itself. The SQL Editor buttons are described below.

Figure 8.3 SQL Editor - Toolbar



From left to right, these buttons are:

- **Open an SQL Script File**: Loads contents from a saved SQL script into the SQL editor.
- **Save SQL Script to File**: Saves contents from the SQL editor into a file.
- **Execute SQL Script**: Executes the selected portion of the query, or the entire query if nothing is selected.
- **Execute Current SQL script**: Execute the statement under the keyboard cursor.
- **Explain (All or Selection)**: Execute the `EXPLAIN` command on the query under the keyboard cursor.

  A "Results Grid" tab is also displayed when executing an EXPLAIN statement. Clicking it will execute the same query, as if **Execute SQL Script** was selected.

  Alternatively, the Visual Explain plan is already available for all executed queries. Select **Execution Plan** from the results tab to view it.
- **Stop the query being executed**: Halts execution of the currently executing SQL script.

  Note

  The database connection will not be restarted, and open transactions will remain open.
- **Toggle whether execution of SQL script should continue after failed statements**: If the red “breakpoint” circle is displayed, the script terminates on a statement that fails. If the button is depressed so that the green arrow is displayed, execution continues past the failed code, possibly generating additional result sets. In either case, any error generated from attempting to execute the faulty statement is recorded in the Output tabsheet.

  This behavior can also be set from the **SQL Execution** user preferences panel.
- **Commit**: Commits the current transaction.

  Note

  All query tabs in the same connection share the same transactions. To have independent transactions, a new connection must be opened.
- **Rollback**: Rolls back the current transaction.

  Note

  All query tabs in the same connection share the same transactions. To have independent transactions, a new connection must be opened.
- **Toggle Auto-Commit Mode**: If selected, each statement will be committed independently.

  Note

  All query tabs in the same connection share the same transactions. To have independent transactions, a new connection must be opened.

Auto-commit is enabled by default, and this default behavior can be modified (disabled) under the **SQL Execution** user preferences panel.

- **Set Limit for Executed Queries**: The default value is 1000, which appends "LIMIT 0, 1000" to SELECT queries.

  The default (1000) can be changed from the **SQL Execution** user preferences panel.

- **Save Snippet**: Save the current statement or selection to the active snippet list.
- **Beautify SQL**: Beautify/reformat the SQL script.

  By default, SQL keywords are changed to UPPER CASE. This functionality can be changed from the **SQL Editor** user preferences panel.

- **Find panel**: Show the Find panel for the editor.
- **Invisible characters**: Toggle display of invisible characters, such as newlines, tabs, spaces.

  A new line is represented as **[LF]**, a space as a single dot (.), and a tab as a right arrow.

- **Wrapping**: Toggles the wrapping of long lines in the SQL editor window.

8.1.3 Query and Edit Menus

When an SQL Editor tab is selected, the most important items on the main menu bar are the **Query** and **Edit** menus.

SQL Query Menu

The **Query** menu features the following items:

- **Execute (All or Selection)**: Executes all statements in the SQL Query area, or only the selected statements.
- **Execute (All or Selection) to Text**: Executes all statements in the SQL Query area, or only the selected statements, and displays it in plain text like the standard MySQL command line console.
- **Execute Current Statement**: Executes the current SQL statement.
- **Execute Current Statement (Vertical Text Output)**: Executes all statements in the SQL Query area, or only the selected statements, and displays it in plain text like the MySQL command line console does vertically (\G).
- **Explain Current Statement**: Describes the current statement by using the MySQL EXPLAIN statement.
- **Visual Explain Current Statement**: Visually describes the current statement, based on EXPLAIN information provided by MySQL Server 5.6 and above. MySQL Workbench parses the EXPLAIN (JSON) output from MySQL server 5.6+, and outputs a visual representation.

  For additional information about Visual Explain, see Section 7.3, “Visual Explain Plan” and Section 7.5, “Tutorial: Using Visual Explain to improve query performance”.

- **Stop**: Stops executing the currently running script.
- **Stop Script Execution On Errors**: If enabled, MySQL Workbench stops executing the a query if errors are found. It can be enabled/disabled from this menu.

- **Limit Rows**: By default, the number of returned rows (LIMIT) is 1000. Values defined here affects subsequent statements. The number ranges from 10 to 50000, and "Don't Limit".
- **Collect Performance Schema Stats**: Provides data to the **Query Stats** result set view, which includes statement specific information about Timing, Rows processed, Temporary tables, Joins per type, Sorting, and Index usage.
- **Collect Resultset Field Metadata**: Provides data to the **Form Editor** and **Field Types** result set views.
- **Reconnect to Server**: Reconnects to the MySQL server.
- **New Tab to Current Server**: Creates a duplicate of the current SQL Editor tab.
- **Auto-Commit Transactions**: Enable to auto-commit transactions.
- **Commit Transaction**: Commits a database transaction.
- **Rollback Transaction**: Rolls back a database transaction.
- **Commit Result Edits**: Commits any changes you have made to the server.
- **Discard Result Edits**: Discards any changes you have made.
- **Export Results**: Exports result sets to a file. Selecting this option displays the **Export Query Results to File** dialog. The dialog enables you to select which result set you wish to export, the file format (CSV, HTML, XML), and the name and location of the output file. Then click **Export** to export the data.

Edit Menu

The **Edit** menu features the **Format** submenu. The **Format** submenu includes the following menu items:

- **Beautify Query**: Reformats the query selected in the query tab and lays it out in nicely indented fashion.
- **UPCASE Keywords**: Converts keywords to uppercase in the currently selected query in the query tab.
- **lowercase Keywords**: Converts keywords to lowercase in the currently selected query in the query tab.
- **Un/Comment Selection**: Comments the lines currently selected in the query tab. If the lines are already commented, this operation removes the comments.
- **Auto-complete**: Triggers the auto-completion wizard. This is enabled (and triggered) by default, and can be disabled with **Preferences**, **SQL Editor**, **Automatically Start Code Completion**. Auto-completion will list functions, keywords, schema names, table names and column names.

8.1.4 Results Window

The results area of the screen shows the results from executed statements. If the script contains multiple statements, a result tab will be generated for each statemented that returned results.

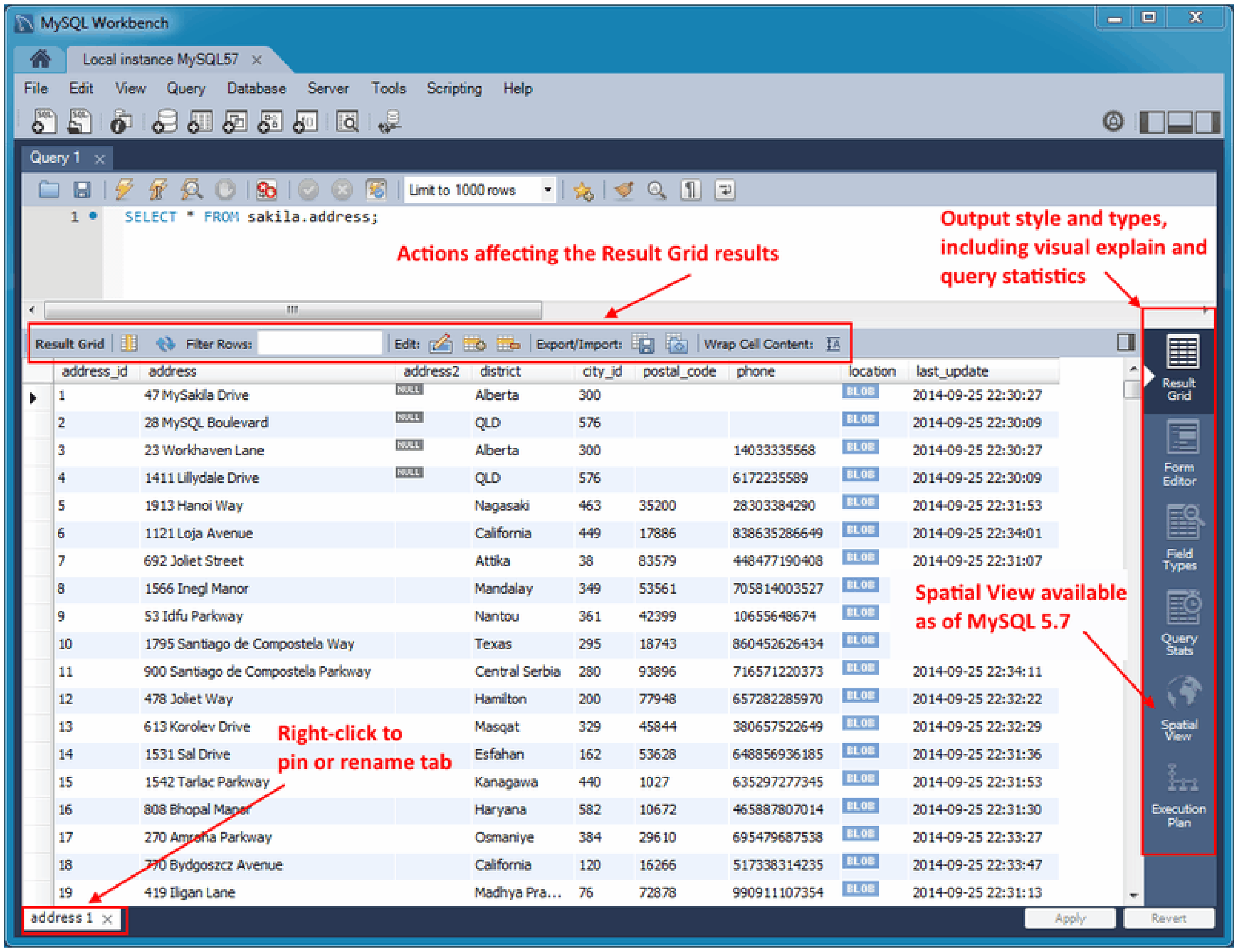
Note

MySQL Workbench handles quoting and escaping for strings entered into the results grid, so adding quotes and proper escaping here is optional.

Note

It is possible to enter a function, or other expression, into a field. Use the prefix `\func` to prevent MySQL Workbench from escaping quotation marks. For example, for the expression `md5('fred')`, MySQL Workbench normally would generate the code `md5(\'fred\')`. To prevent this, enter the expression as `\func md5('fred')` to ensure that the quoting is not escaped.

Figure 8.4 SQL Editor - Result Grid



```
1 ● SELECT * FROM sakila.address;
```

address_id	address	address2	district	city_id	postal_code	phone	location	last_update
1	47 MySakila Drive	NULL	Alberta	300			BLOB	2014-09-25 22:30:27
2	28 MySQL Boulevard	NULL	QLD	576			BLOB	2014-09-25 22:30:09
3	23 Workhaven Lane	NULL	Alberta	300		14033335568	BLOB	2014-09-25 22:30:27
4	1411 Lillydale Drive	NULL	QLD	576		6172235589	BLOB	2014-09-25 22:30:09
5	1913 Hanoi Way		Nagasaki	463	35200	28303384290	BLOB	2014-09-25 22:31:53
6	1121 Loja Avenue		California	449	17886	838635286649	BLOB	2014-09-25 22:34:01
7	692 Joliet Street		Attika	38	83579	448477190408	BLOB	2014-09-25 22:31:07
8	1566 Inegl Manor		Mandalay	349	53561	705814003527	BLOB	
9	53 Idfu Parkway		Nantou	361	42399	10655648674	BLOB	
10	1795 Santiago de Compostela Way		Texas	295	18743	860452626434	BLOB	
11	900 Santiago de Compostela Parkway		Central Serbia	280	93896	716571220373	BLOB	2014-09-25 22:34:11
12	478 Joliet Way		Hamilton	200	77948	657282285970	BLOB	2014-09-25 22:32:22
13	613 Korolev Drive		Masqat	329	45844	380657522649	BLOB	2014-09-25 22:32:29
14	1531 Sal Drive		Esfahan	162	53628	648856936185	BLOB	2014-09-25 22:31:36
15	1542 Tarlac Parkway		Kanagawa	440	1027	635297277345	BLOB	2014-09-25 22:31:53
16	808 Bhopal Manor		Haryana	582	10672	465887807014	BLOB	2014-09-25 22:31:30
17	270 Amroha Parkway		Osmaniye	384	29610	695479687538	BLOB	2014-09-25 22:33:27
18	770 Bydgoszcz Avenue		California	120	16266	517338314235	BLOB	2014-09-25 22:33:47
19	419 Iligan Lane		Madhya Pra...	76	72878	990911107354	BLOB	2014-09-25 22:31:13

The result grid navigation panel offers the following options:

- **Reset**: Resets all sorted columns.
- **Refresh**: Refreshes all data by re-executing the original statement.
- **Filter Rows**: performs a case-insensitive search of all cells. It automatically refreshes, and there is also the refresh button to perform this action manually.
- **Edit Current Row**: Edit the current row.
- **Add New Row**: Adds a new empty row, and highlights it in edit mode. Click **Apply** to execute (and review) the insert row query.
- **Delete Selected Rows**: Deletes the selected rows. Click **Apply** execute (and review) the delete query.
- **Export**: Writes a result set to a CSV, HTML, JSON, SQL INSERT, Excel, XML, or Tab separated file as required.

Note

This exports a result set. To export an entire table or schema, see Data Export.

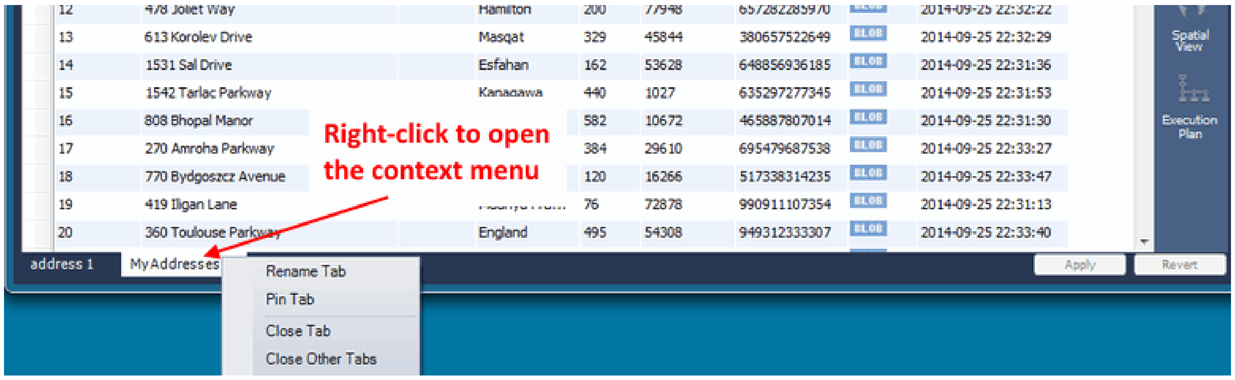
- **Import**: Import records from an external CSV file.
- **Wrap Cell Content**: If the contents of a cell exceeds the cell width, then the data will be cut off with an ellipses. This option will instead wrap the contents within the cell, and adjust the cell height accordingly.

Note

The "Refresh" button automatically adjusts the column width to match the longest string one of its cells. You may also manually adjust the column width.

Right-clicking on a results grid tab opens the following context-menu:

Figure 8.5 SQL Editor - Result Grid Context Menu

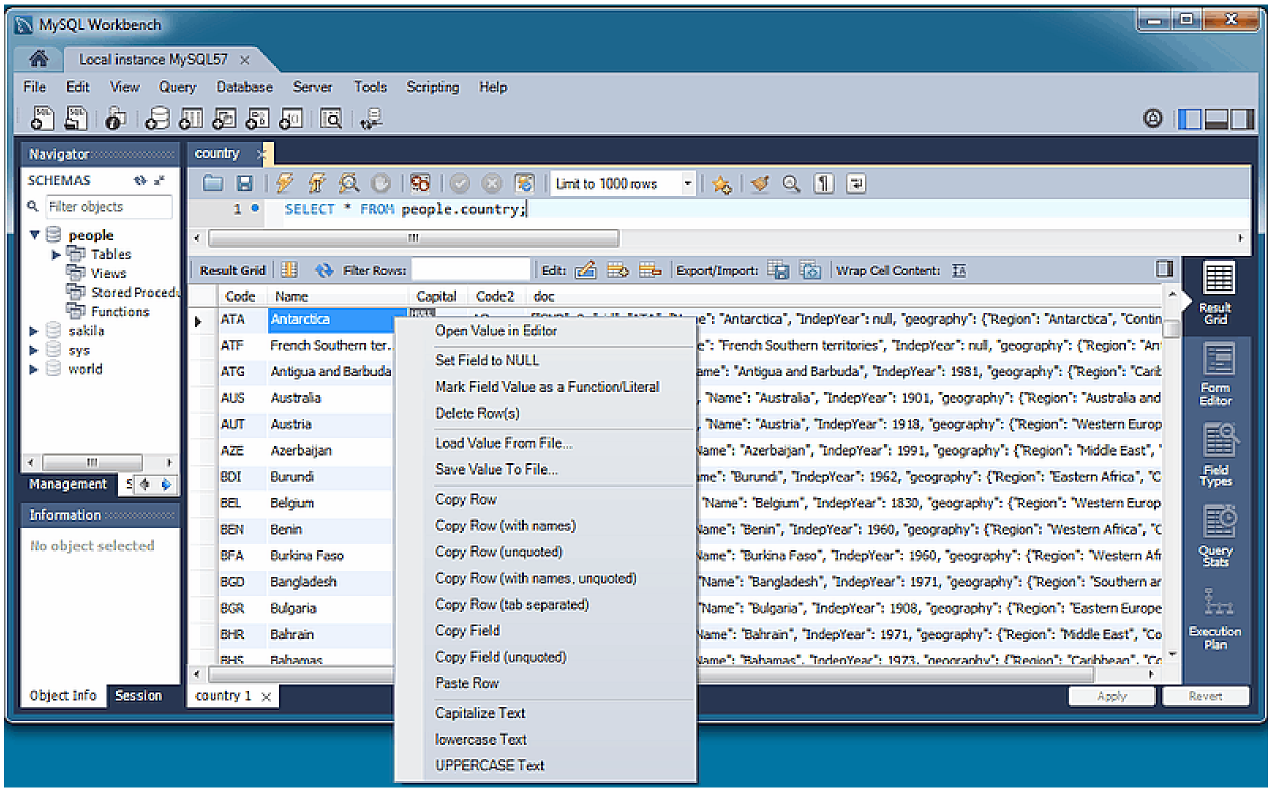


12	478 Joliet Way		Hamilton	200	77948	657282285970	BLOB	2014-09-25 22:32:22
13	613 Korolev Drive		Masqat	329	45844	380657522649	BLOB	2014-09-25 22:32:29
14	1531 Sal Drive		Esfahan	162	53628	648856936185	BLOB	2014-09-25 22:31:36
15	1542 Tarlac Parkway		Kanagawa	440	1027	635297277345	BLOB	2014-09-25 22:31:53
16	808 Bhopal Manor			582	10672	465887807014	BLOB	2014-09-25 22:31:30
17	270 Amroha Parkway			384	29610	695479687538	BLOB	2014-09-25 22:33:27
18	770 Bydgoszcz Avenue			120	16266	517338314235	BLOB	2014-09-25 22:33:47
19	419 Iligan Lane		[illegible]	76	72878	990911107354	BLOB	2014-09-25 22:31:13
20	360 Toulouse Parkway		England	495	54308	949312333307	BLOB	2014-09-25 22:33:40

- **Rename Tab**: Customize the name (title) of this tab.
- **Pin Tab**: Pin the results tab to the results grid. Executing additional SQL statements will create new result grid tabs.
- **Close Tab**: Close this tab.
- **Close Other Tabs**: Close all tabs except this one.

Right-clicking on a results grid field opens the following context-menu:

Figure 8.6 SQL Editor - Result Grid Field Context Menu



```
SELECT * FROM people.country;
```

- **Open Value in Editor**: Opens a new editor window that specializes in editing Binary and JSON data, but can edit text.
- **Set Field to NULL**: Sets the field value to NULL.
- **Mark Field Value as a Function/Literal**: Marks as a function, by prepending \func.
- **Delete Row(s)**: Deletes the entire row.
- **Load Value from File**: Opens a file dialog to insert a value from a file. The entire file contents are inserted into the field.
- **Save Value to File**: Saves a field's value to a file.
- **Copy Row**: Copy the row in escaped CSV format, in a form such as: 'a', 'b','c'. Alternatively, there is **Copy Row (tab separated)** to use tabs instead of commas as the separator, and **Copy Row (unquoted)** to not escape the values.
- **Copy Row (with names)**: Copy an escaped row like "Copy Row", but also adds a #comment containing column names. Alternatively, there is **Copy Row (with names, unquoted)**.
- **Copy Field**: Copies the field name, such as: 'a', or use **Copy Field (unquoted)** to not use single quotes.
- **Paste Row**: Pastes the row over the currently selected row.
- **Capitalize Text**: Capitalizes text in the current row, such as: Hello World.
- **lowercase Text**: Lowercases text in the current row, such as: hello world.
- **UPPERCASE Text**: Alters row to use all capitals, such as: HELLO WORLD.

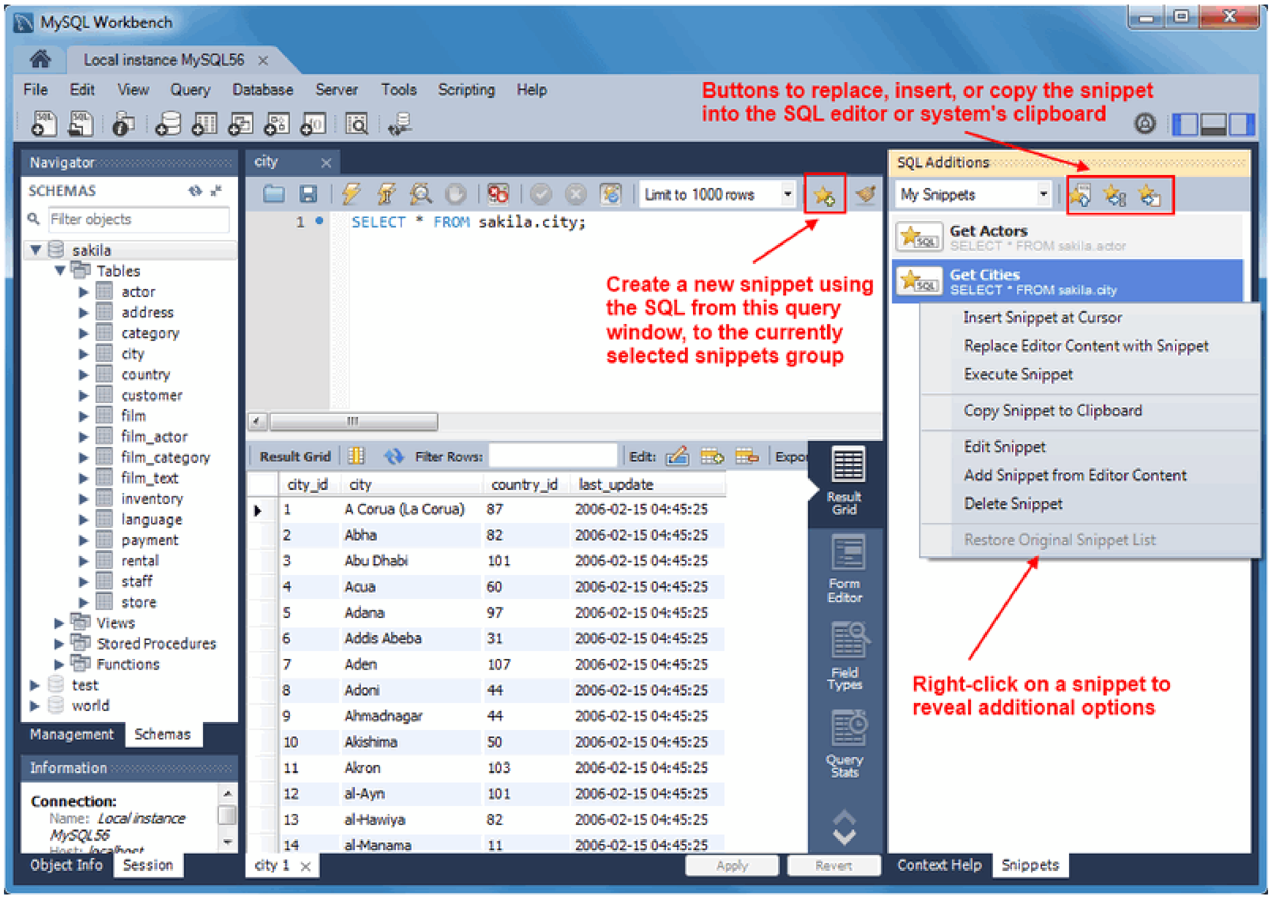
8.1.5 SQL Snippets tab

The Snippets tab includes built-in, local, and shared custom snippets. The **My Snippets** tab stores custom snippets in a file under the MySQL Workbench user's configuration directory. Select the **Shared** option for shared snippets.

Using Snippets

Snippets can be inserted into the SQL editor or the system's clipboard. To insert (use) a snippet, either use the snippet icons or right-click on the desired snippet and choose Insert.

Figure 8.7 SQL Snippets: Usage



Buttons to replace, insert, or copy the snippet into the SQL editor or system's clipboard

Create a new snippet using the SQL from this query window, to the currently selected snippets group

Right-click on a snippet to reveal additional options

```
SELECT * FROM sakila.city;
```

	city_id	city	country_id	last_update
▶	1	A Corua (La Corua)	87	2006-02-15 04:45:25
	2	Abha	82	2006-02-15 04:45:25
	3	Abu Dhabi	101	2006-02-15 04:45:25
	4	Acua	60	2006-02-15 04:45:25
	5	Adana	97	2006-02-15 04:45:25
	6	Addis Abeba	31	2006-02-15 04:45:25
	7	Aden	107	2006-02-15 04:45:25
	8	Adoni	44	2006-02-15 04:45:25
	9	Ahmadnagar	44	2006-02-15 04:45:25
	10	Akishima	50	2006-02-15 04:45:25
	11	Akron	103	2006-02-15 04:45:25
	12	al-Ayn	101	2006-02-15 04:45:25
	13	al-Hawiya	82	2006-02-15 04:45:25
	14	al-Manama	11	2006-02-15 04:45:25

Get Actors
SELECT * FROM sakila.actor

Get Cities
SELECT * FROM sakila.city

Local Snippets (My Snippets)

Local snippets are stored in the MySQL Workbench user's directory. By default, the "My Snippets" SQL snippets are stored here:

Table 8.1 Default Local Snippet File Location

Operating System	File Path
Windows	%AppData%\MySQL\Workbench\User Snippets.txt
OS X	~username/Library/Application Support/MySQL/Workbench/snippets/User Snippets.txt

Operating System	File Path
Linux	~username/.mysql/workbench/snippets/User Snippets.txt

Editing (or adding) snippets to "My Snippets" in MySQL Workbench edits this plain text file. Optionally, you can edit this file outside of MySQL Workbench or create new files that will also be listed under the snippets selector. For example, adding a file named "More Snippets.txt" will add a "More Snippets" section to the snippets selection box.

Shared Snippets

Shared snippets are saved in a `.mysqlworkbench` schema on the connected MySQL server. Selecting "Shared" for the first time will request permission for MySQL Workbench to create this shared `.mysqlworkbench` schema. Users connected to this MySQL server are allowed to create, edit, and use these shared snippets.

Note

Shared snippets were added in MySQL Workbench 6.2.0.

The `.mysqlworkbench` schema is hidden from within MySQL Workbench as it is considered an internal schema that does not need to be seen or edited.

Built-in Snippets

Several built-in SQL snippets are bundled with MySQL Workbench, and typically show the SQL syntax for MySQL operations. They are divided up into the following categories.

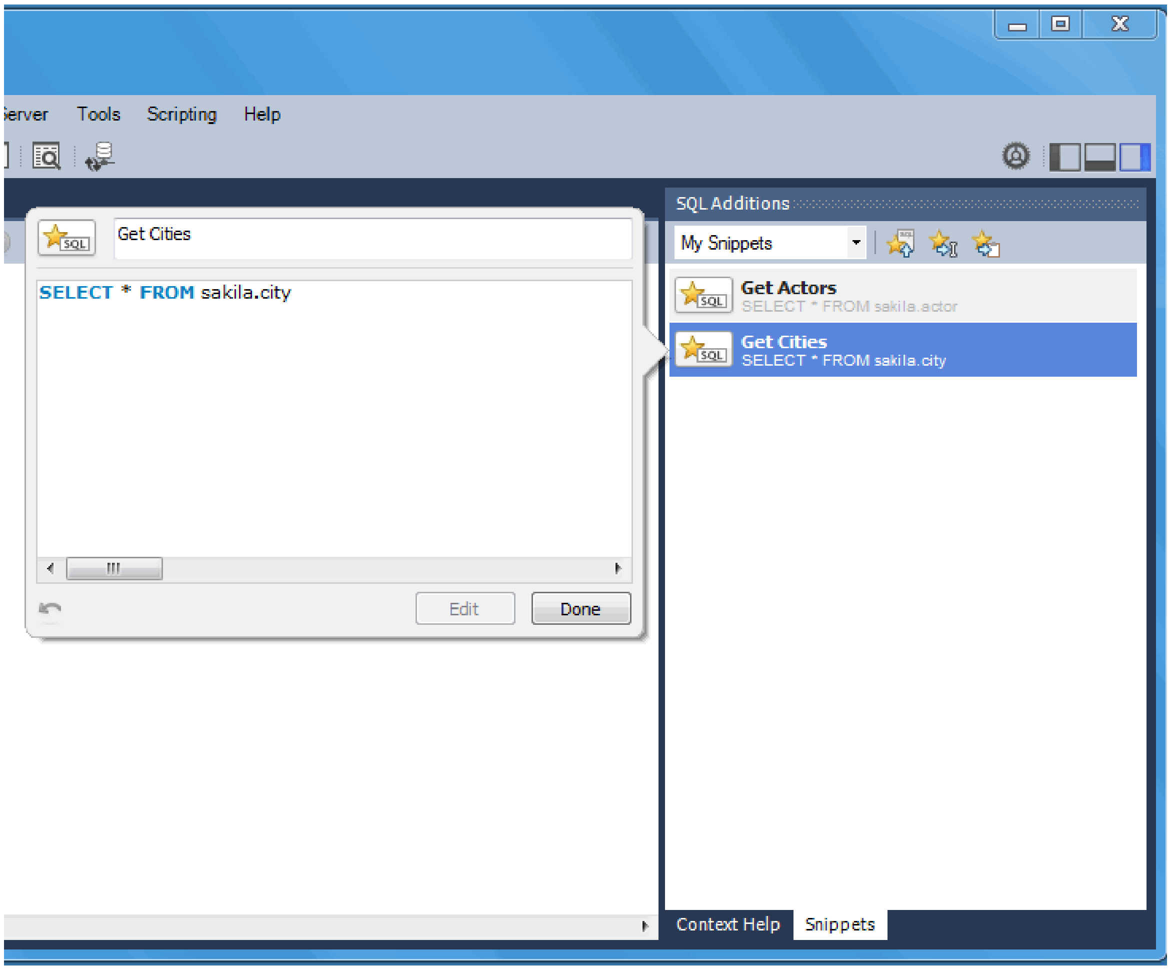
- **DB Mgmt** (Database Management): Syntax examples use `SHOW` in many forms to provide information about databases, tables, columns, or status information about the MySQL server.
- **SQL DDL** (SQL Data Definition Language): Syntax examples include creating, altering, and dropping tables, indexes, views, and procedures.
- **SQL DML** (SQL Data Manipulation Language): Syntax examples for operations such as SELECT, INSERT, and REPLACE.

The built-in operations are stored in text files in the same directory as the custom snippet files.

Saving and Editing Snippets

To save a snippet, choose the Snippets Insert icon ( ) or right-click in the snippet window and choose **Add Snippet from Editor Content** from the context-menu. Double-click a snippet to open it, and choose the snippet editor to edit its body or title. This example shows two snippets with only the first having defined a name.

Figure 8.8 SQL Snippets: Editor



```
SELECT * FROM sakila.city
```

Get Actors
SELECT * FROM sakila.actor

Get Cities
SELECT * FROM sakila.city

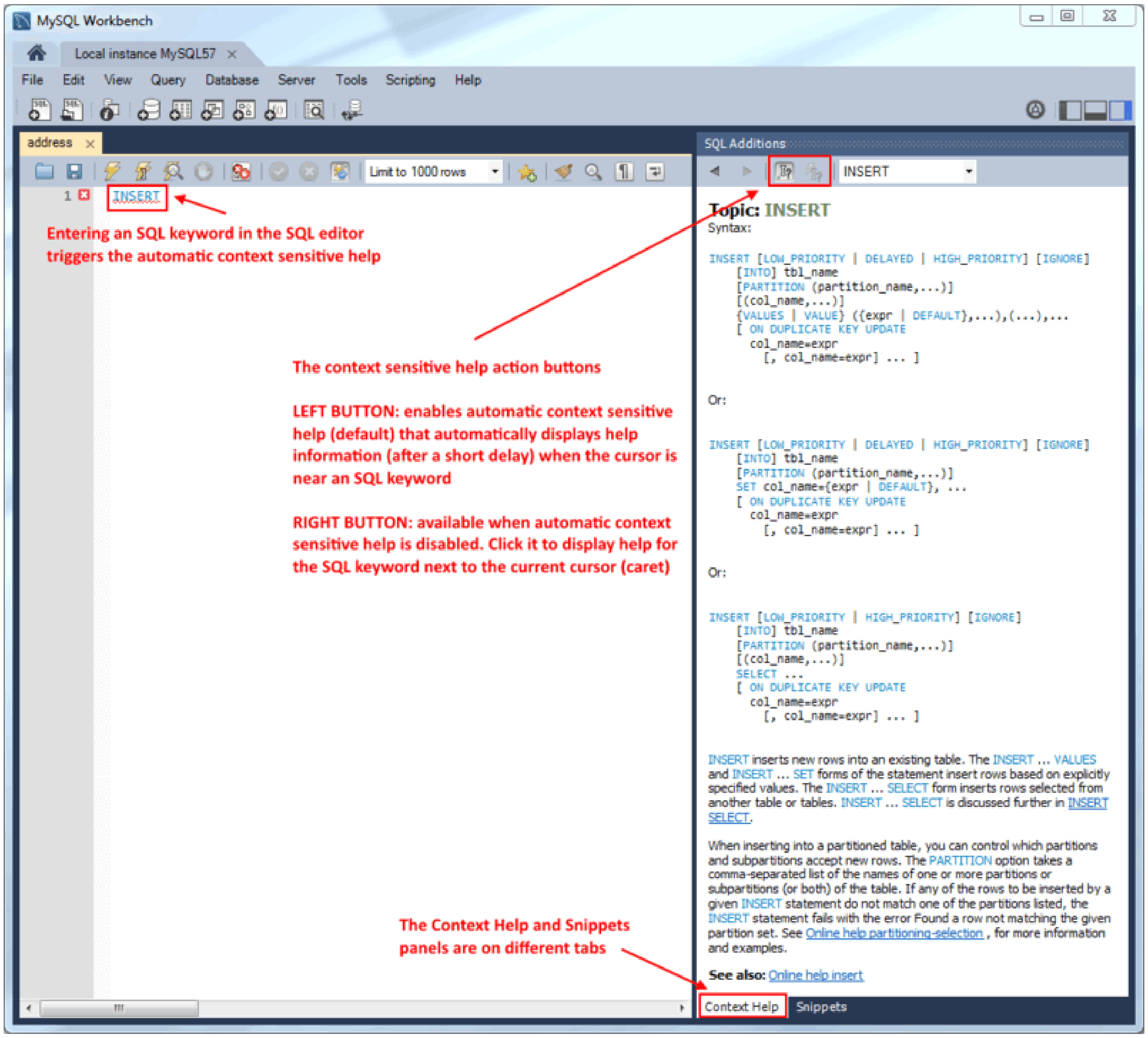
8.1.6 Context Sensitive Help

Select a keyword or function in your query, and after a delay it shows formatted help information from the MySQL Server (equivalent to using the help command from the command-line MySQL Client).

Note

This queries the MySQL Server for the help text, so it can be slow over a slow network

Figure 8.9 SQL Editor: Context Sensitive Help



```
INSERT
```

Entering an SQL keyword in the SQL editor triggers the automatic context sensitive help

The context sensitive help action buttons

LEFT BUTTON: enables automatic context sensitive help (default) that automatically displays help information (after a short delay) when the cursor is near an SQL keyword

RIGHT BUTTON: available when automatic context sensitive help is disabled. Click it to display help for the SQL keyword next to the current cursor (caret)

The Context Help and Snippets panels are on different tabs

Topic: INSERT

Syntax:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    {VALUES | VALUE} ({expr | DEFAULT},...),(...),...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

Or:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    SET col_name={expr | DEFAULT}, ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

Or:

```
INSERT [LOW_PRIORITY | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name,...)]
    [(col_name,...)]
    SELECT ...
    [ ON DUPLICATE KEY UPDATE
      col_name=expr
        [, col_name=expr] ... ]
```

INSERT inserts new rows into an existing table. The INSERT ... VALUES and INSERT ... SET forms of the statement insert rows based on explicitly specified values. The INSERT ... SELECT form inserts rows selected from another table or tables. INSERT ... SELECT is discussed further in INSERT SELECT.

When inserting into a partitioned table, you can control which partitions and subpartitions accept new rows. The PARTITION option takes a comma-separated list of the names of one or more partitions or subpartitions (or both) of the table. If any of the rows to be inserted by a given INSERT statement do not match one of the partitions listed, the INSERT statement fails with the error Found a row not matching the given partition set. See Online help partitioning-selection , for more information and examples.

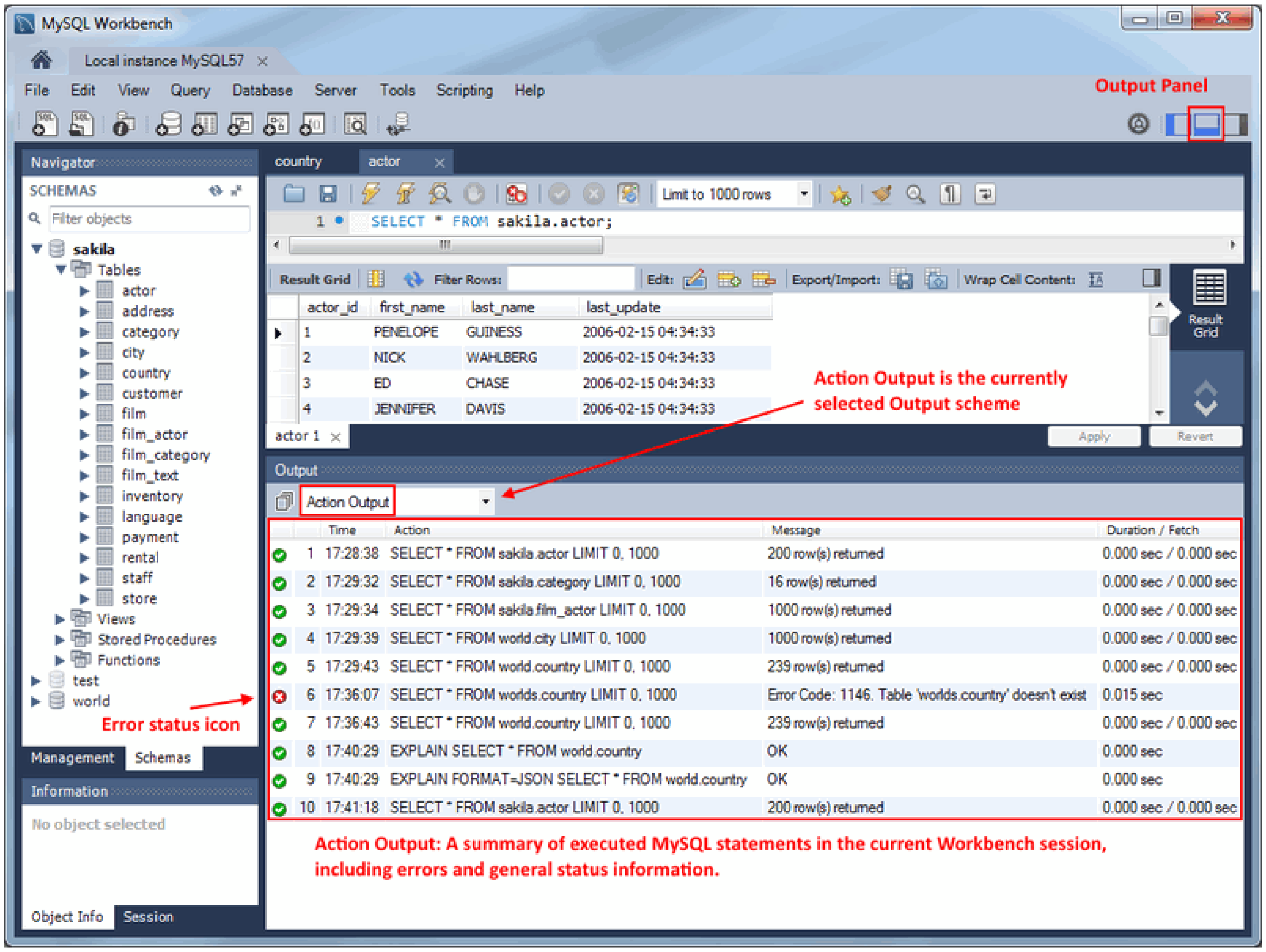
See also: Online help insert

8.1.7 Output History Panel

The **Output** is located at the bottom of MySQL Workbench. Its select box includes the `Action Output`, `History Output`, and `Text Output` options.

The **Action Output** panel displays a summary of the communication between the active MySQL connection in MySQL Workbench and the MySQL server, and can refer to errors or general information. Each message displays the time, action, and server response. This output is useful for troubleshooting scripts.

Figure 8.10 SQL Editor: Output: Action Output



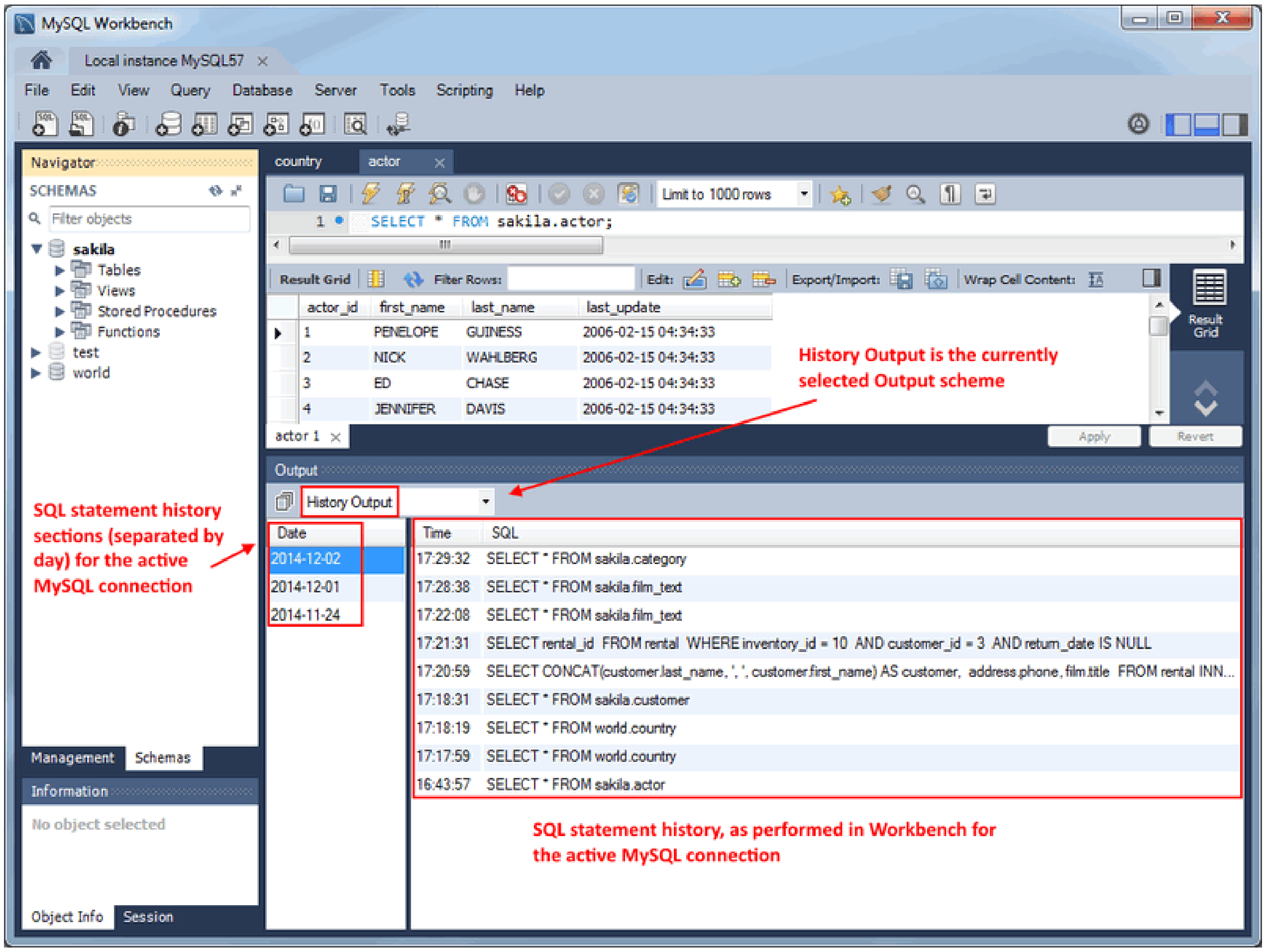
```
1 SELECT * FROM sakila.actor;
```

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15 04:34:33
2	NICK	WAHLBERG	2006-02-15 04:34:33
3	ED	CHASE	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33

		Time	Action	Message	Duration / Fetch
✓	1	17:28:38	SELECT * FROM sakila.actor LIMIT 0, 1000	200 row(s) returned	0.000 sec / 0.000 sec
✓	2	17:29:32	SELECT * FROM sakila.category LIMIT 0, 1000	16 row(s) returned	0.000 sec / 0.000 sec
✓	3	17:29:34	SELECT * FROM sakila.film_actor LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
✓	4	17:29:39	SELECT * FROM world.city LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec
✓	5	17:29:43	SELECT * FROM world.country LIMIT 0, 1000	239 row(s) returned	0.000 sec / 0.000 sec
✗	6	17:36:07	SELECT * FROM worlds.country LIMIT 0, 1000	Error Code: 1146. Table 'worlds.country' doesn't exist	0.015 sec
✓	7	17:36:43	SELECT * FROM world.country LIMIT 0, 1000	239 row(s) returned	0.000 sec / 0.000 sec
✓	8	17:40:29	EXPLAIN SELECT * FROM world.country	OK	0.000 sec
✓	9	17:40:29	EXPLAIN FORMAT=JSON SELECT * FROM world.country	OK	0.000 sec
✓	10	17:41:18	SELECT * FROM sakila.actor LIMIT 0, 1000	200 row(s) returned	0.000 sec / 0.000 sec

The **History Output** panel provides a history of SQL operations carried out in MySQL Workbench for the active MySQL connection. The time and SQL code for each operation is recorded. To view the executed SQL statement, click the time, and the SQL code executed will be displayed in the **SQL** column.

Figure 8.11 SQL Editor: History Output


MySQL Workbench
Local instance MySQL57
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sakila
Tables
Views
Stored Procedures
Functions
test
world
Management Schemas
Information
No object selected
Object Info Session
country actor
Limit to 1000 rows
Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content:
Result Grid
actor 1
Apply Revert
Output
History Output
SQL statement history sections (separated by day) for the active MySQL connection
History Output is the currently selected Output scheme
SQL statement history, as performed in Workbench for the active MySQL connection

```
SELECT * FROM sakila.actor;
```

actor_id	first_name	last_name	last_update
1	PENELOPE	GUINESS	2006-02-15 04:34:33
2	NICK	WAHLBERG	2006-02-15 04:34:33
3	ED	CHASE	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33

Date
2014-12-02
2014-12-01
2014-11-24

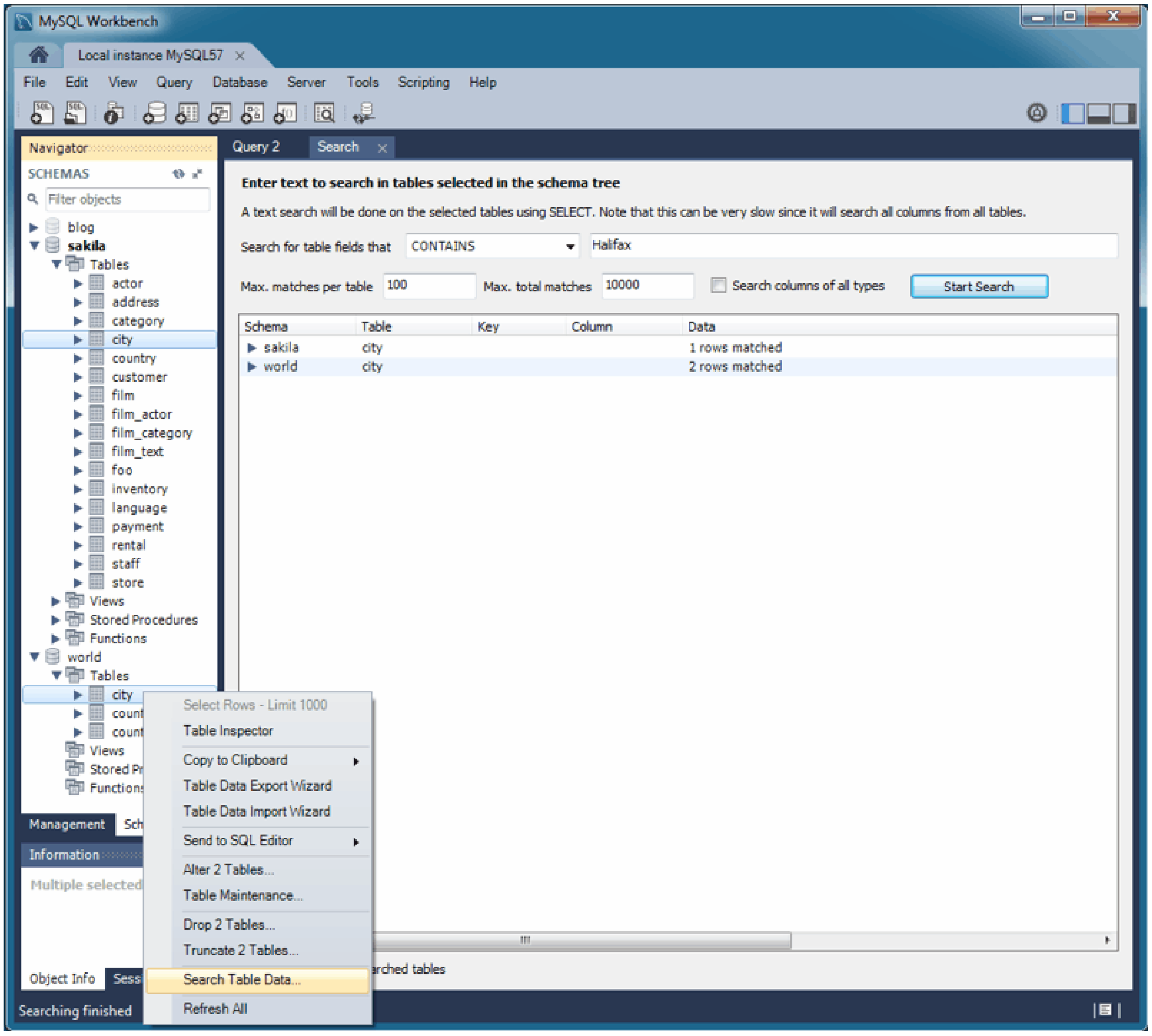
Time	SQL
17:29:32	SELECT * FROM sakila.category
17:28:38	SELECT * FROM sakila.film_text
17:22:08	SELECT * FROM sakila.film_text
17:21:31	SELECT rental_id FROM rental WHERE inventory_id = 10 AND customer_id = 3 AND return_date IS NULL
17:20:59	SELECT CONCAT(customer.last_name, ', ', customer.first_name) AS customer, address.phone, film.title FROM rental INN...
17:18:31	SELECT * FROM sakila.customer
17:18:19	SELECT * FROM world.country
17:17:59	SELECT * FROM world.country
16:43:57	SELECT * FROM sakila.actor

8.1.8 Table Data Search Panel

Find data across a MySQL connection by using the text search feature on any number of tables and schemas.

From the Schema Tree, choose the tables and/or schemas you want to search, and then select **Search Data Table...** from the context menu.

Figure 8.12 Table Search Example: Multiple Tables and Schemas



Enter text to search in tables selected in the schema tree

A text search will be done on the selected tables using SELECT. Note that this can be very slow since it will search all columns from all tables.

Search for table fields that: CONTAINS — Halifax

Max. matches per table: 100 — Max. total matches: 10000 — [ ] Search columns of all types — Start Search

Schema	Table	Key	Column	Data
sakila	city			1 rows matched
world	city			2 rows matched

The search options include:

- Search for table fields that: "CONTAINS", "Search using =", "Search using LIKE", "Search using REGEXP". These search options are case-insensitive.
- Max. matches per table: [`100`]
- Max. total matches: [`1000`]
- [ ] Search columns of all types: If checked, non-text column type columns are cast to CHAR to perform the matches, otherwise only text types (CHAR, VARCHAR, and TEXT) are searched. This is unchecked by default.

8.1.9 Export / Import a Table

Export or Import tables using the table export and import wizard.

Note

These wizards were added in MySQL Workbench 6.3.

Export a Table

Note

Alternatively, use Section 6.5, “Data Export and Import” to export larger sets of data, such as entire tables and databases.

Import into Table

Note

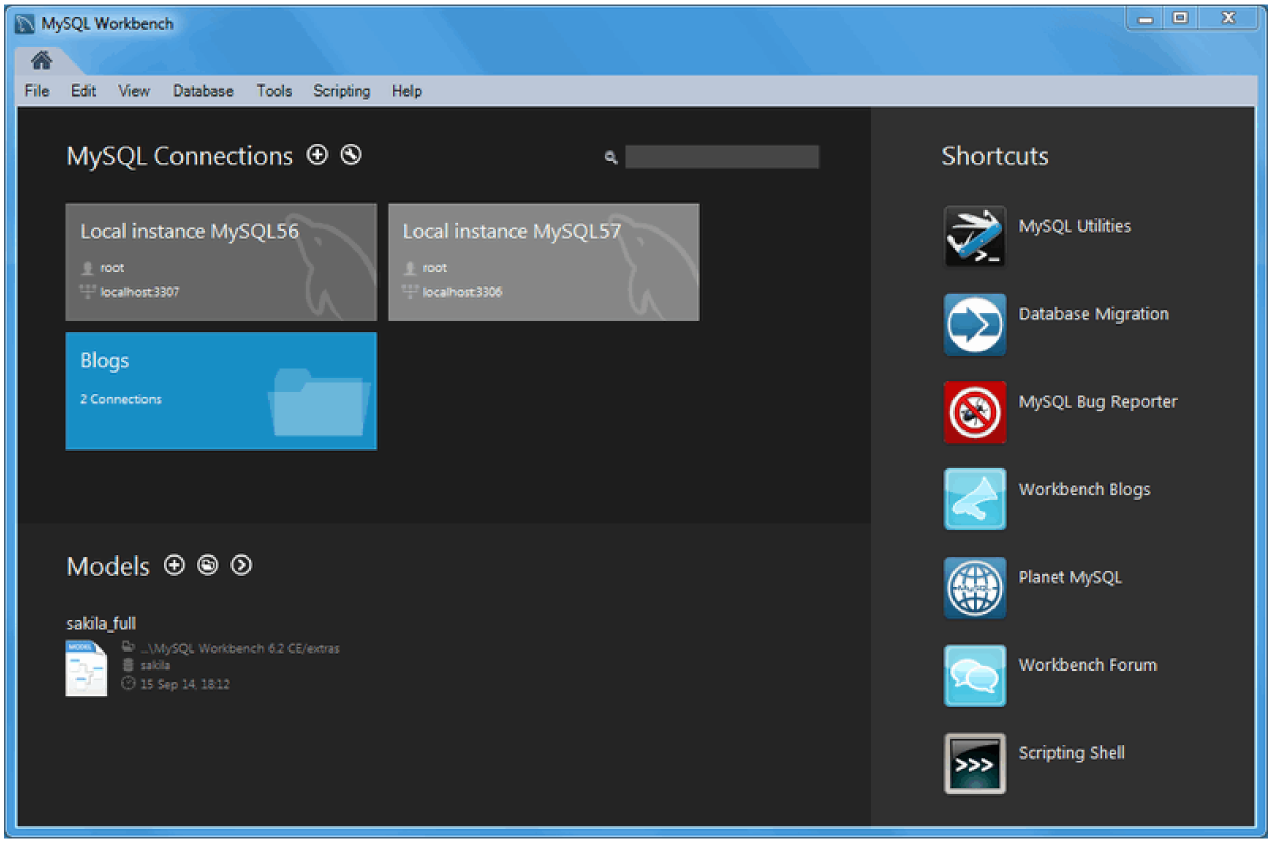
Alternatively, use Section 6.5, “Data Export and Import” to export larger sets of data, such as entire tables and databases.

8.1.10 Tutorial: Adding Data

In the previous section, you created a model, schema, and table. You also forward engineered your model to the live MySQL server. This section uses MySQL Workbench to add data into your MySQL database.

Open a MySQL connection.

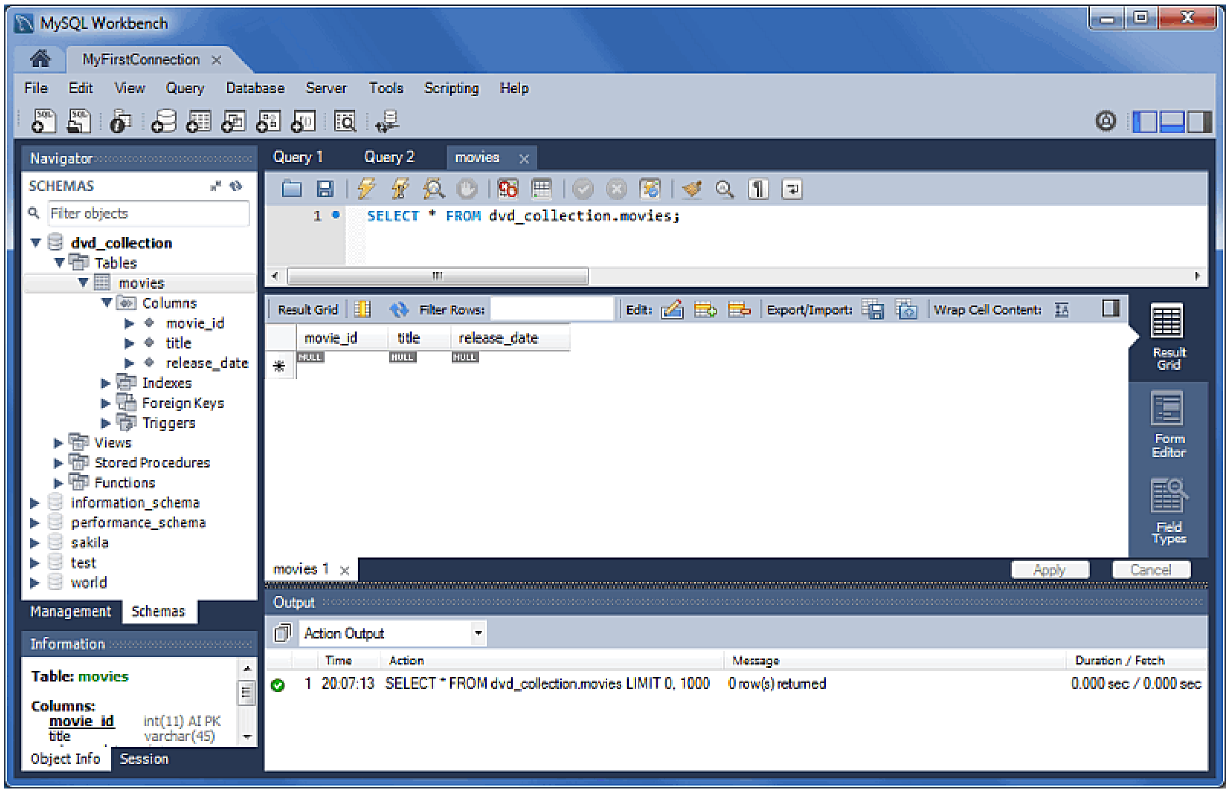
Figure 8.13 Getting Started Tutorial - SQL Editor



1. From the **Navigator** panel on the left, select the `movies` table from the `dvd_collection` schema that we created earlier in this tutorial. Right-click on the `movies` table and choose **Select Rows - Limit 1000** from the context menu.

> **Note**
>
> The **Navigator** panel has both **Management** and **Schemas** tabs.

Figure 8.14 Getting Started Tutorial - Adding Data from the SQL Editor



```
SELECT * FROM dvd_collection.movies;
```

movie_id	title	release_date
NULL	NULL	NULL

Table: movies

Columns:

movie_id	int(11) AI PK
title	varchar(45)

```
     Time      Action                                              Message               Duration / Fetch
1    20:07:13  SELECT * FROM dvd_collection.movies LIMIT 0, 1000   0 row(s) returned     0.000 sec / 0.000 sec
```

2. This displays the query and its associated results grid. The table is empty, and data may be added into the results grid.

> **Note**
>
> The `movie_id` column is set to `AUTO_INCREMENT`, so values are not needed for this column.

Input the following data into the **movies** table:

title	release_date
Gone with the Wind	1939-04-17
The Hound of the Baskervilles	1939-03-31
The Matrix	1999-06-11
Above the Law	1988-04-08
Iron Man 2	2010-05-07

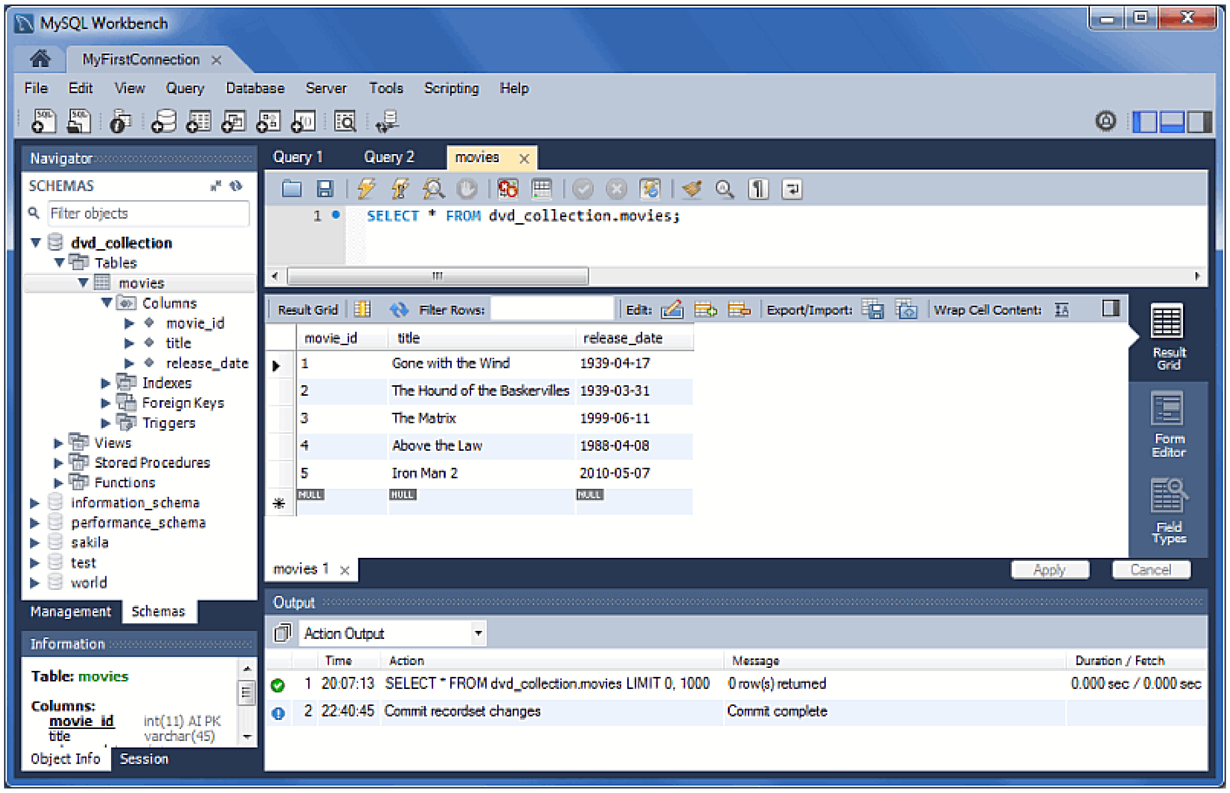
Note

Do not modify `movie_id` column values.

3. Click **Apply** to apply these changes to the live MySQL server.

4. View the data grid again and observe the generated `AUTO_INCREMENT` values.

Figure 8.15 Getting Started Tutorial - Edit Data



```
SELECT * FROM dvd_collection.movies;
```

movie_id	title	release_date
1	Gone with the Wind	1939-04-17
2	The Hound of the Baskervilles	1939-03-31
3	The Matrix	1999-06-11
4	Above the Law	1988-04-08
5	Iron Man 2	2010-05-07
NULL	NULL	NULL

	Time	Action	Message	Duration / Fetch
1	20:07:13	SELECT * FROM dvd_collection.movies LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
2	22:40:45	Commit recordset changes	Commit complete	

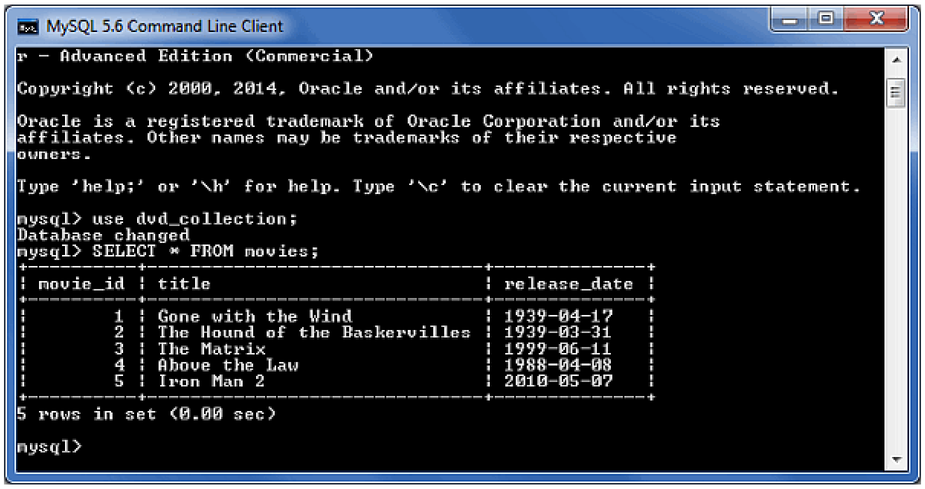
Table: movies

Columns:

movie_id	int(11) AI PK
title	varchar(45)

5. Optionally, you might confirm the changes by checking an external source, such as the MySQL Command Line Client. To check, enter `SELECT * FROM movies;` from the MySQL Command Line Client to confirm that the data was entered.

Figure 8.16 Getting Started Tutorial - View Data From The Command Line



```
r - Advanced Edition (Commercial)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use dvd_collection;
Database changed
mysql> SELECT * FROM movies;
+----------+--------------------------------+--------------+
| movie_id | title                          | release_date |
+----------+--------------------------------+--------------+
|        1 | Gone with the Wind             | 1939-04-17   |
|        2 | The Hound of the Baskervilles  | 1939-03-31   |
|        3 | The Matrix                     | 1999-06-11   |
|        4 | Above the Law                  | 1988-04-08   |
|        5 | Iron Man 2                     | 2010-05-07   |
+----------+--------------------------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

6. You can also use MySQL Workbench to perform a similar check. Close the `MyFirstConnection` tab (or MySQL Workbench) and then open the `MyFirstConnection` connection from the home page. Execute `USE dvd_collection; SELECT * FROM movies;` to display the newly entered data.

In this section of the tutorial, you have learned how to add data to your database, and also how to execute SQL statements using MySQL Workbench.

For additional information about the SQL editor, see Section 8.1, “Visual SQL Editor”.

8.1.11 The MySQL Table Editor

The MySQL Table Editor is a used to create and modify tables. .ou can add or modify a table's columns or indexes, change the engine, add foreign keys, or alter the table's name.

To access the MySQL Table Editor, right-click on a table name in the **Object Viewer** (the left navigator panel that lists all schemas) and choose **ALTER TABLE**. This opens a new tab within the main **SQL Editor** window. You can also access the MySQL Table Editor from an EER Diagram by double-clicking on a table object.

8.1.11.1 The Main Editor Window

Any number of tables may be edited in the MySQL Table Editor at any one time. Adding another table creates a new tab at the top of the editor. By default, the MySQL Table Editor appears docked at the top of the table editor tab, within the SQL editor..

The MySQL Table Editor is shown on top of the following figure.

Figure 8.17 The Table Editor

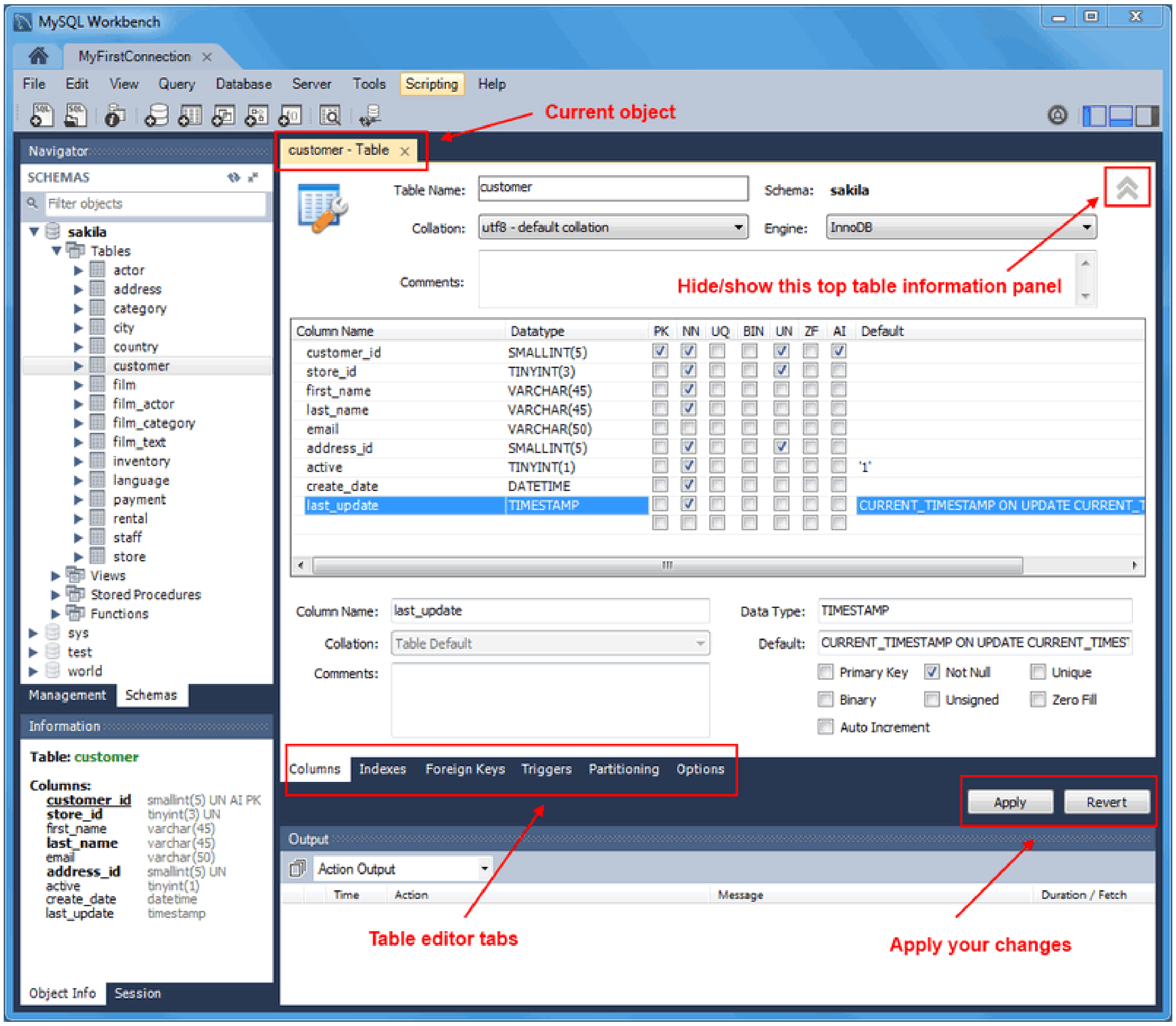


Table Name: customer
Schema: sakila
Collation: utf8 - default collation
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
customer_id	SMALLINT(5)	✓	✓			✓		✓	
store_id	TINYINT(3)		✓			✓			
first_name	VARCHAR(45)		✓						
last_name	VARCHAR(45)		✓						
email	VARCHAR(50)								
address_id	SMALLINT(5)		✓			✓			
active	TINYINT(1)		✓						'1'
create_date	DATETIME		✓						
last_update	TIMESTAMP		✓						CURRENT_TIMESTAMP ON UPDATE CURRENT_T

Column Name: last_update
Data Type: TIMESTAMP
Collation: Table Default
Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMES
Comments:
Primary Key [✓ Not Null] Unique
Binary Unsigned Zero Fill
Auto Increment

Table: customer

Columns:

customer_id	smallint(5) UN AI PK
store_id	tinyint(3) UN
first_name	varchar(45)
last_name	varchar(45)
email	varchar(50)
address_id	smallint(5) UN
active	tinyint(1)
create_date	datetime
last_update	timestamp

The MySQL Table Editor provides a work space that has tabs used to perform these actions:

- Columns: Add or modify columns
- Indexes: Add or modify indexes
- Foreign Keys: Add or modify foreign keys
- Triggers: Add or modify triggers
- Partitioning: Manage partitioning
- Options: Add or modify other options, divided in categories named general, row, storage, and merge

8.1.11.2 The Columns Tab

Use the **Columns** tab to display and edit all the column information for a table. With this tab, you can add, drop, and alter columns.

You can also use the **Columns** tab to change column properties such as name, data type, and default value.

Figure 8.18 The Columns Tab

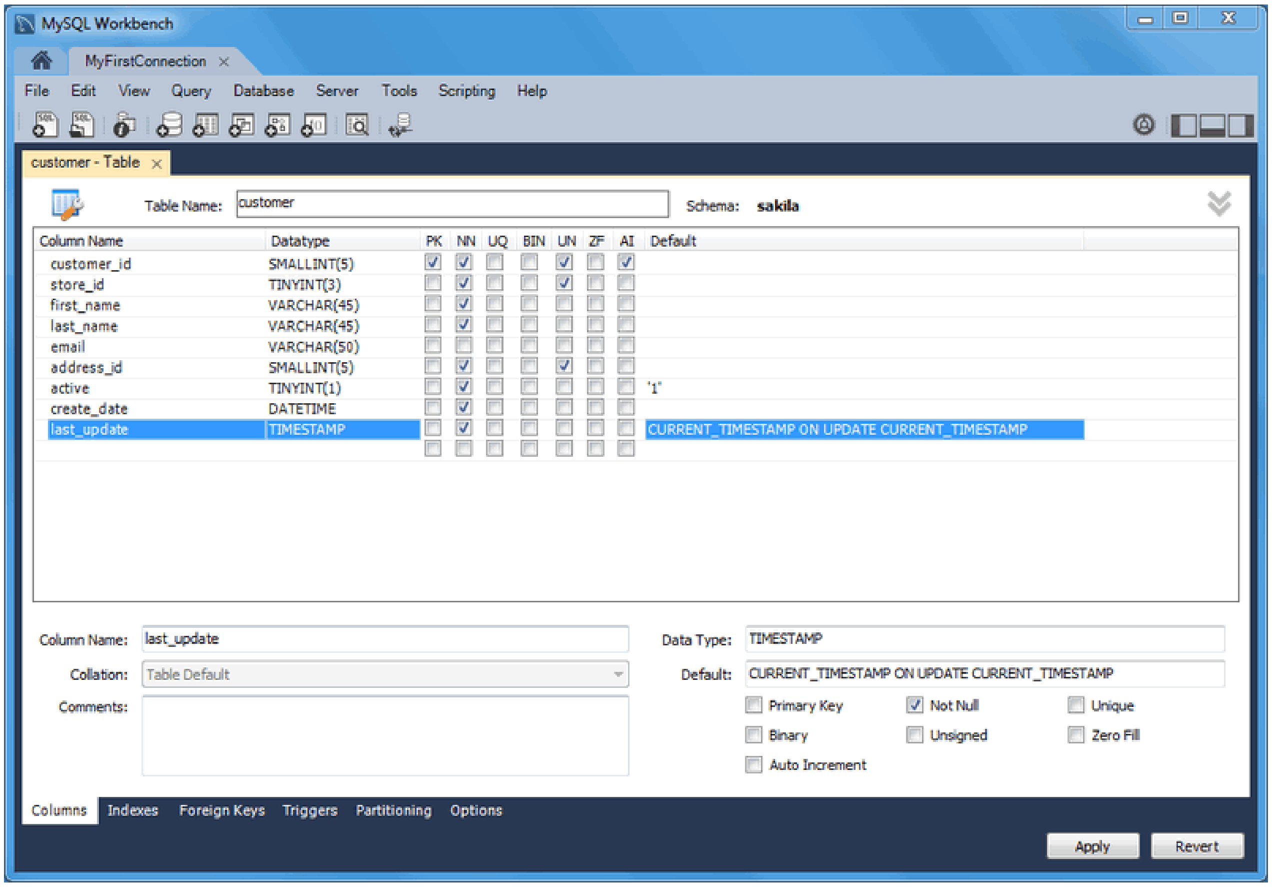


Table Name: customer   Schema: **sakila**

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
customer_id	SMALLINT(5)	✓	✓			✓		✓	
store_id	TINYINT(3)		✓			✓			
first_name	VARCHAR(45)		✓						
last_name	VARCHAR(45)		✓						
email	VARCHAR(50)								
address_id	SMALLINT(5)		✓			✓			
active	TINYINT(1)		✓						'1'
create_date	DATETIME		✓						
last_update	TIMESTAMP		✓						CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

Column Name: last_update
Collation: Table Default
Comments:

Data Type: TIMESTAMP
Default: CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
☐ Primary Key ✓ Not Null ☐ Unique
☐ Binary ☐ Unsigned ☐ Zero Fill
☐ Auto Increment

Right-click a row under the `Column Name` column to open a pop-up menu with the following items:

- **Move Up**: Move the selected column up.
- **Move Down**: Move the selected column down.
- **Copy**: Copies the column for a model.
- **Cut**: Copies and then deletes the column for a model.
- **Paste**: Pastes the column. If a column with the same name already exists, then `_copy1` is appended to the column name.
- **Delete Selected Columns**: Select multiple contiguous columns by right-clicking and pressing the **Shift** key. Use the **Control** key to select separated columns.
- **Refresh**: Update all information in the **Columns** tab.
- **Clear Default**: Clear the assigned default value.
- **Default NULL**: Set the column default value to `NULL`.
- **Default 0**: Set the column default value to `0`.

- **Default CURRENT_TIMESTAMP**: Available for `TIMESTAMP` data types.
- **Default CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP**: Available for `TIMESTAMP` data types.

To add a column, click the `Column Name` field in an empty row and enter an appropriate value. Select a data type from the **Datatype** list. Select the column property check boxes as required according to the list of column properties below, and also read the CREATE TABLE documentation for information about what these options mean.

- **PK**: PRIMARY KEY
- **NN**: NOT NULL
- **UQ**: UNIQUE INDEX
- **BIN**: BINARY
- **UN**: UNSIGNED
- **ZF**: ZEROFILL
- **AI**: AUTO_INCREMENT
- **G**: Generated Column

  This option is available as of MySQL Server 5.7.

To change the name, data type, default value, or comment of a column, double-click the value to edit it.

You can also add column comments to the `Column Comment` field. It is also possible to set the column collation, using the list in the **Column Details** panel.

To the left of the column name is an icon that indicates whether the column is a member of the primary key. If the icon is a small key, that column belongs to the primary key, otherwise the icon is a blue diamond or a white diamond. A blue diamond indicates the column has **NN** set. To add or remove a column from the primary key, double-click the icon. You can also add a primary key by checking the `PRIMARY KEY` check box in the `Column Details` section of the table editor.

If you wish to create a composite primary key you can select multiple columns and check the PK check box. However, there is an additional step that is required, you must click the Indexes tab, then in the Index Columns panel you must set the desired order of the primary keys.

Note

When entering default values, in the case of `CHAR` and `VARCHAR` data types MySQL Workbench will attempt to automatically add quotation marks, if the user does not start their entry with one. For other data types the user must manage quoting if required, as it will not be handled automatically by MySQL Workbench.

Caution

Care must be taken when entering a default value for `ENUM` columns because a non-numeric default will not be automatically quoted. You must manually add single quote characters for the default value. Note that MySQL Workbench will **not** prevent you from entering the default value without the single quotation marks. If a non-numeric default value is entered without quotation marks, this will lead to errors. For example, if the model is reverse engineered, the script will contain unquoted default values for `ENUM` columns and will fail if an attempt is made to run the script on MySQL Server.

Note

ENUM, BIT, and SET must contain at least one value when entering these data types into MySQL Workbench.

8.1.11.3 The Indexes Tab

The **Indexes** tab holds all index information for your table. Use this tab to add, drop, and modify indexes.

Figure 8.19 The Indexes Tab

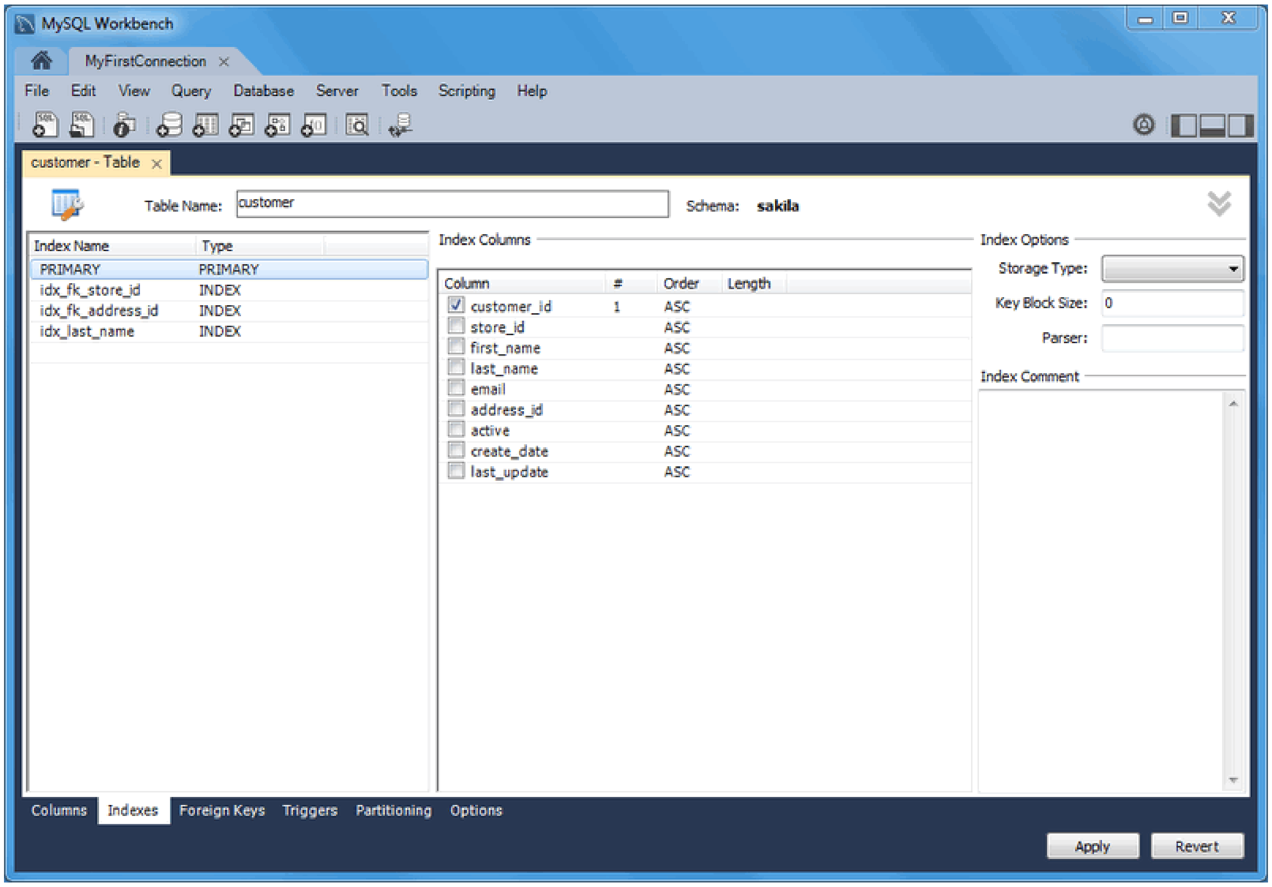


Table Name: customer
Schema: sakila

Index Name	Type
PRIMARY	PRIMARY
idx_fk_store_id	INDEX
idx_fk_address_id	INDEX
idx_last_name	INDEX

Column	#	Order	Length
✓ customer_id	1	ASC	
☐ store_id		ASC	
☐ first_name		ASC	
☐ last_name		ASC	
☐ email		ASC	
☐ address_id		ASC	
☐ active		ASC	
☐ create_date		ASC	
☐ last_update		ASC	

Storage Type:
Key Block Size: 0
Parser:

Select an index by right-clicking it. The **Index Columns** section displays information about the selected index.

To add an index, click the last row in the index list. Enter a name for the index and select the index type from the list. Select the column or columns that you wish to index by checking the column name in the **Index Columns** list. You can remove a column from the index by removing the check mark from the appropriate column.

You can also specify the order of an index by choosing `ASC` or `DESC` under the `Order` column. Create an index prefix by specifying a numeric value under the `Length` column. You cannot enter a prefix value for fields that have a data type that does not support prefixing.

To drop an index, right-click the row of the index you wish to delete, then select the **Delete Selected Indexes** menu item.

8.1.11.4 The Foreign Keys Tab

The **Foreign Keys** tab is organized in much the same fashion as the **Indexes** tab and adding or editing a foreign key is similar to adding or editing an index.

Figure 8.20 The Foreign Keys Tab

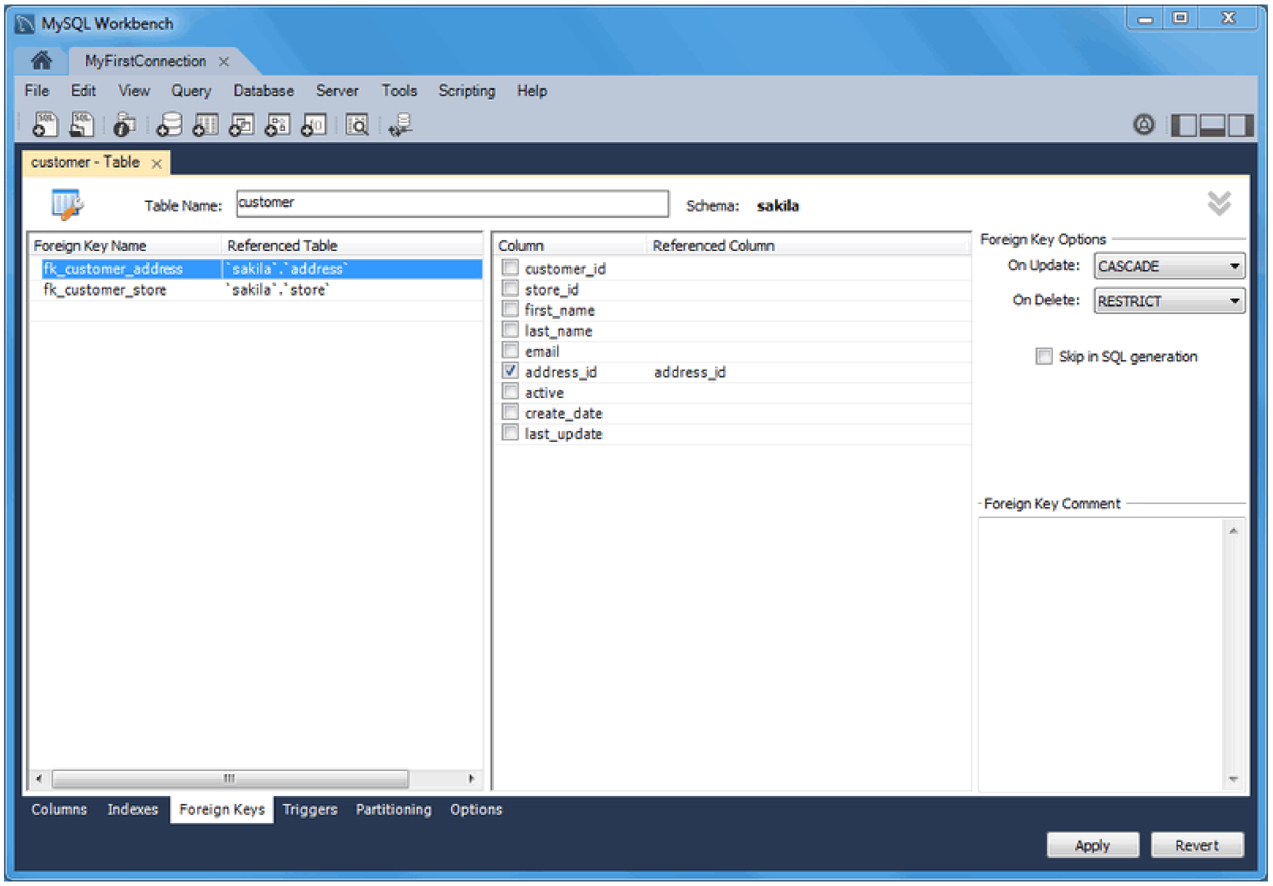

MySQL Workbench
MyFirstConnection
File Edit View Query Database Server Tools Scripting Help
customer - Table
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply
Revert

Table Name: customer   Schema: sakila

Foreign Key Name	Referenced Table
fk_customer_address	`sakila`.`address`
fk_customer_store	`sakila`.`store`

Column	Referenced Column
☐ customer_id	
☐ store_id	
☐ first_name	
☐ last_name	
☐ email	
☑ address_id	address_id
☐ active	
☐ create_date	
☐ last_update	

Foreign Key Options
On Update: CASCADE
On Delete: RESTRICT
☐ Skip in SQL generation

Foreign Key Comment

To add a foreign key, click the last row in the `Foreign Key Name` list. Enter a name for the foreign key and select the column or columns that you wish to index by checking the column name in the **Column** list. You can remove a column from the index by removing the check mark from the appropriate column.

Under **Foreign Key Options**, choose an action for the update and delete events. The options are:

- **RESTRICT**
- **CASCADE**
- **SET NULL**
- **NO ACTION**

To drop a foreign key, right-click the row you wish to delete, then select the **Delete Selected FKs** menu item.

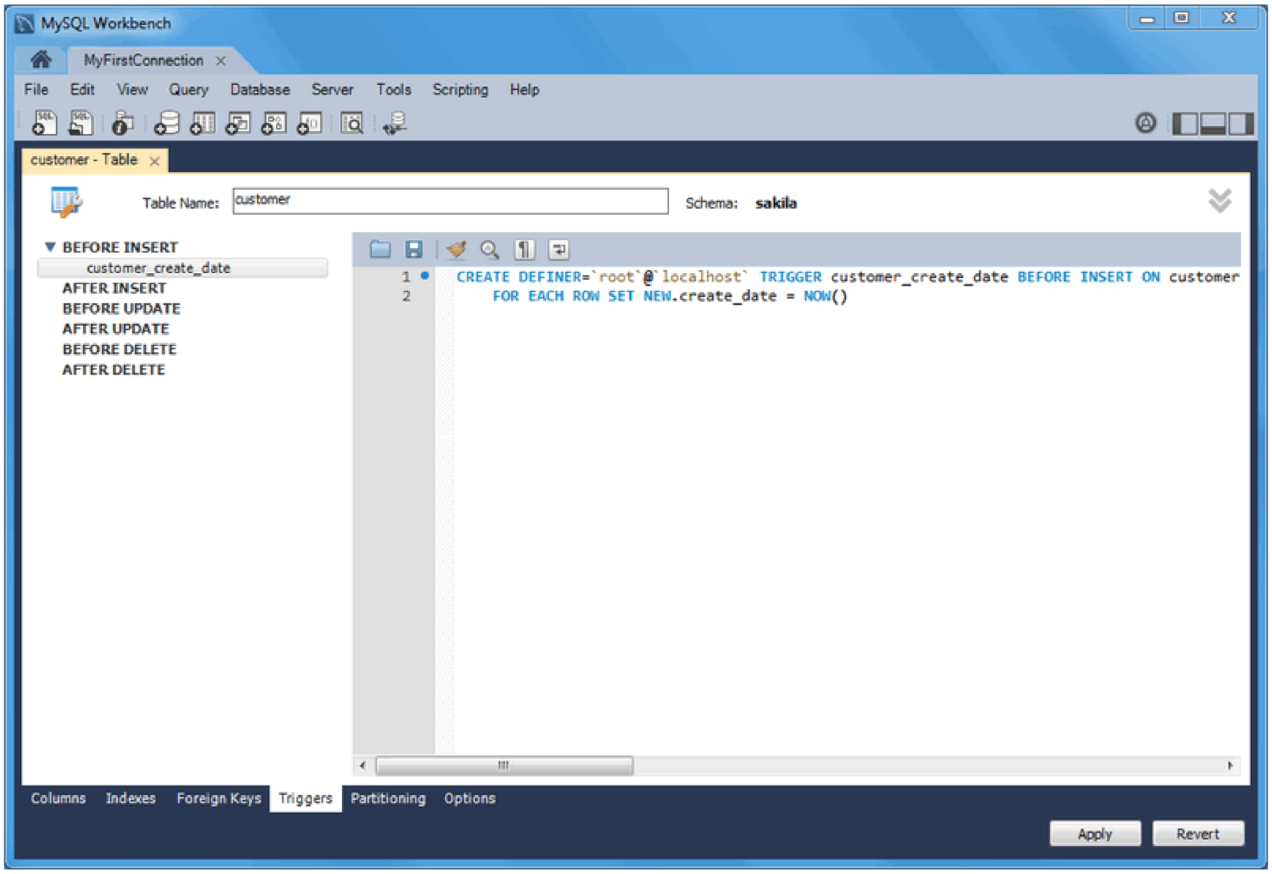
To modify properties of a foreign key, select it and make the desired changes.

8.1.11.5 The Triggers Tab

The **Triggers** tab opens a textbox to create or edit existing triggers.

To add a new trigger, click the **[+]** icon next to the trigger section. To delete a trigger, click the associated **[-]** icon. These icons become visible by hovering over a trigger or trigger section. Click **Apply** to commit your changes.

Figure 8.21 The Triggers Tab



```
CREATE DEFINER=`root`@`localhost` TRIGGER customer_create_date BEFORE INSERT ON customer
    FOR EACH ROW SET NEW.create_date = NOW()
```

8.1.11.6 The Partitioning Tab

To enable partitioning for your table, check the **Enable Partitioning** check box. This enables the partitioning options.

Figure 8.22 The Partitioning Tab

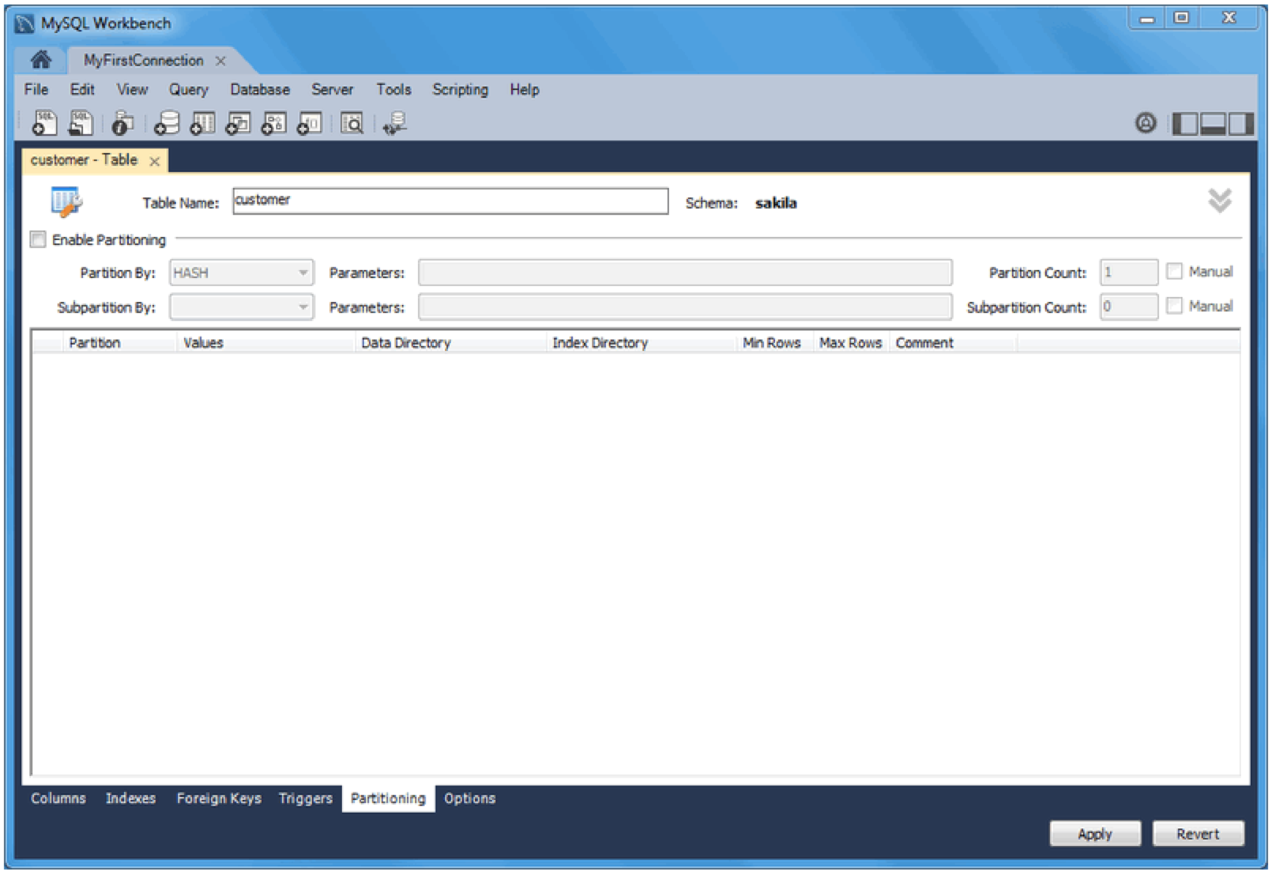

MySQL Workbench
MyFirstConnection
File Edit View Query Database Server Tools Scripting Help
customer - Table
Columns Indexes Foreign Keys Triggers Partitioning Options
Apply
Revert

Table Name: customer
Schema: sakila
Enable Partitioning
Partition By: HASH
Parameters:
Partition Count: 1
Manual
Subpartition By:
Parameters:
Subpartition Count: 0
Manual

Partition	Values	Data Directory	Index Directory	Min Rows	Max Rows	Comment

The **Partition By** pop-up menu displays the types of partitions you can create:

- **HASH**
- **LINEAR HASH**
- **KEY**
- **LINEAR KEY**
- **RANGE**
- **LIST**

Use the **Parameters** field to define any parameters to be supplied to the partitioning function, such as an integer column value.

Choose the number of partitions from the **Partition Count** list. To manually configure your partitions, check the **Manual** check box. This enables entry of values into the partition configuration table. The entries in this table are:

- `Partition`
- `Values`
- `Data Directory`

- `Index Directory`
- `Min Rows`
- `Max Rows`
- `Comment`

Subpartitioning is also available. For more information about partitioning, see Partitioning.

8.1.11.7 The Options Tab

The **Options** tab enables you to set several types of options.

Figure 8.23 The Options Tab

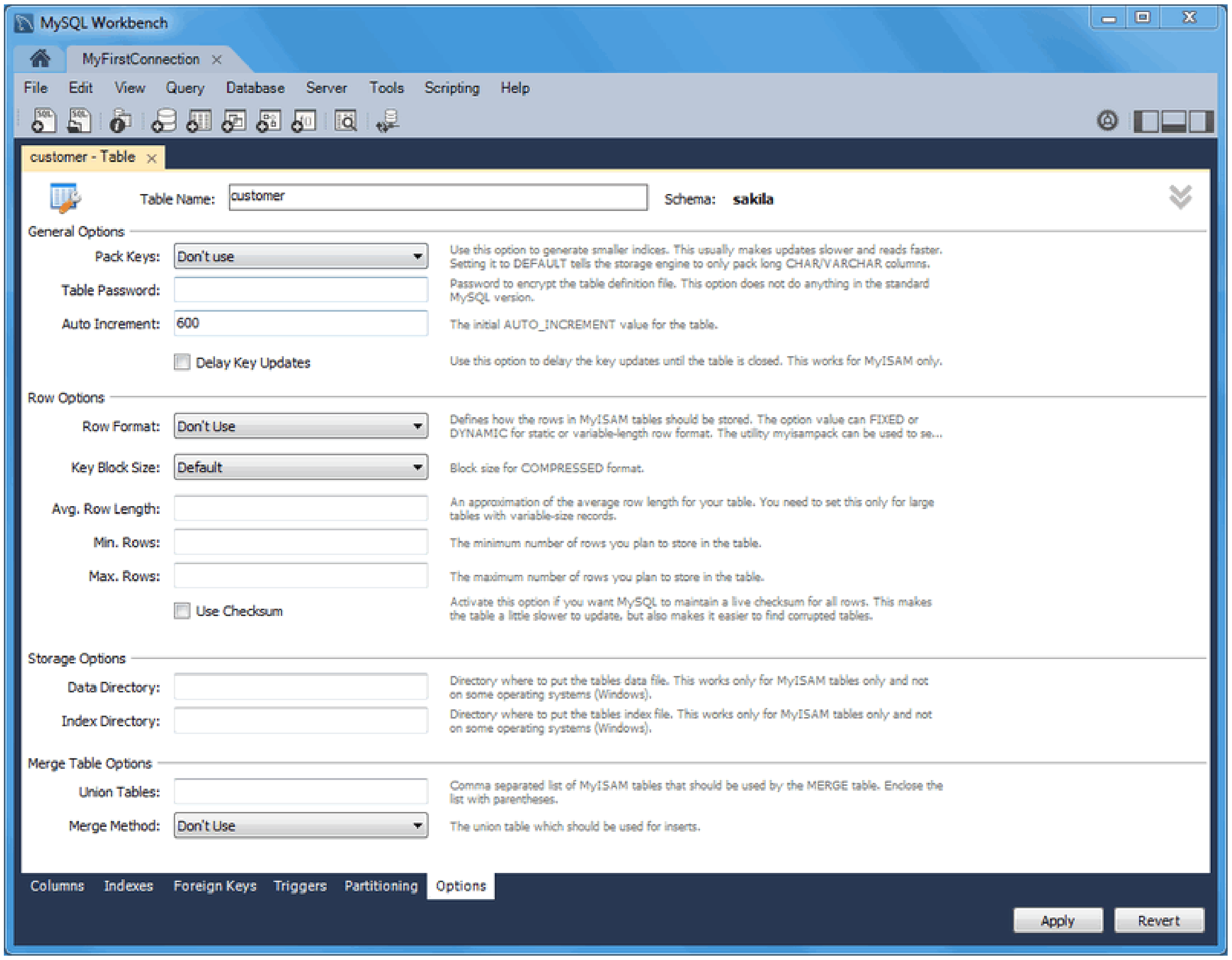


Table Name: customer Schema: sakila

General Options

Pack Keys:	Don't use	Use this option to generate smaller indices. This usually makes updates slower and reads faster. Setting it to DEFAULT tells the storage engine to only pack long CHAR/VARCHAR columns.
Table Password:		Password to encrypt the table definition file. This option does not do anything in the standard MySQL version.
Auto Increment:	600	The initial AUTO_INCREMENT value for the table.
	☐ Delay Key Updates	Use this option to delay the key updates until the table is closed. This works for MyISAM only.

Row Options

Row Format:	Don't Use	Defines how the rows in MyISAM tables should be stored. The option value can FIXED or DYNAMIC for static or variable-length row format. The utility myisampack can be used to se...
Key Block Size:	Default	Block size for COMPRESSED format.
Avg. Row Length:		An approximation of the average row length for your table. You need to set this only for large tables with variable-size records.
Min. Rows:		The minimum number of rows you plan to store in the table.
Max. Rows:		The maximum number of rows you plan to store in the table.
	☐ Use Checksum	Activate this option if you want MySQL to maintain a live checksum for all rows. This makes the table a little slower to update, but also makes it easier to find corrupted tables.

Storage Options

Data Directory:		Directory where to put the tables data file. This works only for MyISAM tables only and not on some operating systems (Windows).
Index Directory:		Directory where to put the tables index file. This works only for MyISAM tables only and not on some operating systems (Windows).

Merge Table Options

Union Tables:		Comma separated list of MyISAM tables that should be used by the MERGE table. Enclose the list with parentheses.
Merge Method:	Don't Use	The union table which should be used for inserts.

which are grouped into the following sections:

- General Options
- Row Options
- Storage Options

- Merge Table options

The following discussion describes these options in more detail.

General Options Section

In the **General Options** section, choose a pack keys option. The options are `Default`, `Pack None`, and `Pack All`. You may also encrypt the definition of a table. The `AUTO_INCREMENT` and delayed key update behaviors apply only to `MyISAM` tables.

Row Options Section

To set the row format, choose the desired row format from the list. For more information about the different row formats that are available, see MyISAM Table Storage Formats.

These options are:

- Default
- Dynamic
- Fixed
- Compressed
- Redundant
- Compact

When you expect a table to be particularly large, use the **Avg. Row**, **Min. Rows**, and **Max. Rows** options to enable the MySQL server to better accommodate your data. See CREATE TABLE Syntax for more information on how to use these options.

Storage Options Section

The `Storage Options` section is available only for `MyISAM` tables. Use it to configure a custom path to the table storage and data files. This can help improve server performance by locating different tables on different hard drives.

Merge Table Options Section

Use the `Merge Table` Options section to configure `MERGE` tables. To create a `MERGE` table, select `MERGE` as your storage engine and then specify the `MyISAM` tables you wish to merge in the **Union Tables** dialog.

You may specify the action the server should take when users attempt to perform `INSERT` statements on the merge table. You may also select the `Merge Method` by selecting from the list. For more information about `MERGE` tables, see The MERGE Storage Engine.

8.1.12 Code Generation Overview

This document provides a quick hands-on introduction to using MySQL Workbench to generate code for later use, for either in or outside of MySQL Workbench.

8.1.12.1 Generating SQL Statements

MySQL Workbench can be used to generate SQL, most typically as either `INSERT` statements or `SELECT` statements.

Below are common methods for generating SQL statements in MySQL Workbench.

Note

All of the MySQL Workbench Export options include the option to export as SQL.

Context-menu options after right-clicking on a `schema` in the schema view, using the `sakila` column as an example:

Create Statement

```
CREATE DATABASE `sakila` /*!40100 DEFAULT CHARACTER SET latin1 */;
```

Name

```
`sakila`
```

Context-menu options after right-clicking on a `table` in the schema view, using the `sakila.actor` column as an example:

Name (Short)

```
`actor`
```

Name (Long)

```
`sakila`.`actor`
```

Select All Statement

```
SELECT `actor`.`actor_id`,
    `actor`.`first_name`,
    `actor`.`last_name`,
    `actor`.`last_update`
FROM `sakila`.`actor`;
```

Select with References

```
SET @actor_id_to_select = <{row_id}>;
SELECT film_actor.*
    FROM film_actor, actor
    WHERE `actor`.`actor_id` = `film_actor`.`actor_id`
          AND actor.actor_id = @actor_id_to_select;
SELECT actor.*
    FROM actor
    WHERE actor.actor_id = @actor_id_to_select;
```

Insert Statement

```
INSERT INTO `sakila`.`actor`
  (`actor_id`,
  `first_name`,
  `last_name`,
  `last_update`)
VALUES
  (<{actor_id: }>,
  <{first_name: }>,
  <{last_name: }>,
  <{last_update: CURRENT_TIMESTAMP}>);
```

Update Statement

```
UPDATE `sakila`.`actor`
SET
`actor_id` = <{actor_id: }>,
`first_name` = <{first_name: }>,
`last_name` = <{last_name: }>,
`last_update` = <{last_update: CURRENT_TIMESTAMP}>
WHERE `actor_id` = <{expr}>;
```

Delete Statement

```
DELETE FROM `sakila`.`actor`
WHERE <{where_expression}>;
```

Delete with References

```
-- All objects that reference that row (directly or indirectly)
-- will be deleted when this snippet is executed.
-- To preview the rows to be deleted, use Select Row Dependencies
START TRANSACTION;
-- Provide the values of the primary key of the row to delete.
SET @actor_id_to_delete = <{row_id}>;

DELETE FROM film_actor
    USING film_actor, actor
    WHERE `actor`.`actor_id` = `film_actor`.`actor_id`
          AND actor.actor_id = @actor_id_to_delete;
DELETE FROM actor
    USING actor
    WHERE actor.actor_id = @actor_id_to_delete;
COMMIT;
```

Create Statement

```
CREATE TABLE `actor` (
  `actor_id` smallint(5) unsigned NOT NULL AUTO_INCREMENT,
  `first_name` varchar(45) NOT NULL,
  `last_name` varchar(45) NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`actor_id`),
  KEY `idx_actor_last_name` (`last_name`)
) ENGINE=InnoDB AUTO_INCREMENT=201 DEFAULT CHARSET=utf8;
```

Context-menu options after right-clicking on a `column` in the schema view, using the `sakila.actor.first_name` column as an example:

Name (short)

```
`first_name`
```

Name (long)

```
`actor`.`first_name`
```

Select Columns Statement

```
SELECT `first_name` FROM `sakila`.`actor`;
```

Insert Statement

```
INSERT INTO `sakila`.`actor`
(`first_name`)
VALUES
(<{first_name}>);
```

Update Statement

```
UPDATE `sakila`.`actor`
SET
`first_name` = <{first_name}>
WHERE <{where_expression}>;
```

Context-menu options after right-clicking on a `field` in the results view, using record #1 in the `sakila.actor` table as an example:

Copy Rows (with names)

```
# actor_id, first_name, last_name, last_update
'1', 'PENELOPE', 'GUINESS', '2006-02-15 04:34:33'
```

Copy Rows (with names, unquoted)

```
# actor_id, first_name, last_name, last_update
1, PENELOPE, GUINESS, 2006-02-15 04:34:33
```

Copy Row (tab separated)

```
1 PENELOPE GUINESS 2006-02-15 04:34:33
```

Copy Field

```
'GUINESS'
```

8.1.12.2 Generating PHP Code

MySQL Workbench can be used to generate PHP code with the bundled PHP plugin, by using the **Tools**, **Utilities**, **Copy as PHP Code** menu option.

Below is an example scenario for how to create PHP code. It is a `SELECT` statement, and optionally uses `SET` to set variables.

SQL @variables generate PHP variables in the code that then bind to the statement before execution.

1. Generate or type in the desired SQL query into the SQL editor. This example will use the `sakila` database, with the query being:

   ```
   SET @last_update = '2006-02-14';

   SELECT  actor_id, first_name, last_name, last_update
     FROM  actor
     WHERE last_update > @last_update;
   ```

2. While in the SQL editor, choose **Tools**, **Utilities**, **Copy as PHP Code (Iterate SELECT Results)** from the main menu. This will copy PHP code to the clipboard.

3. Paste the code to the desired location.

Additionally, PHP code that connects to the MySQL database can also be generated by choosing **Tools**, **Utilities**, **Copy as PHP Code (Connect to Server)**.

After combining the two, the generated PHP code will look like this:

```
<?php

$host     = "localhost";
$port     = 3306;
$socket   = "";
$user     = "nobody";
```

```
$password = "";
$dbname   = "sakila";

$con = new mysqli($host, $user, $password, $dbname, $port, $socket)
    or die ('Could not connect to the database server' . mysqli_connect_error());

//$con->close();

$query = "SELECT actor_id, first_name, last_name, last_update
          FROM   actor
          WHERE  last_update > ?";
$last_update = '';

$stmt->bind_param('s', $last_update);

if ($stmt = $con->prepare($query)) {

    $stmt->execute();
    $stmt->bind_result($actor_id, $first_name, $last_name, $last_update);

    while ($stmt->fetch()) {
        // printf("%s, %s, %s, %s\n",
        //   $actor_id, $first_name, $last_name, $last_update);
    }

    $stmt->close();
}

?>
```

Note

The generated PHP code uses the `mysqli` PHP extension for MySQL. This extension must be enabled in your PHP distribution for this code to work. For additional details about this PHP extension, see MySQL Improved Extension.

8.2 Object Management

The Object Browser allows you to navigate database schemas and objects. From here, you can perform common tasks such as selecting tables and fields to query, edit tables, create new or drop tables and databases, perform searches, and more.

8.2.1 Object Browser and Editor Navigator

The Navigator panel contains options to manage the active MySQL connection, and also lists the schemas available to that connection. To access the Navigator panel, open an existing connection from the Home screen. If the panel is not visible, click **View**, **Panels**, and then **Show Sidebar**.

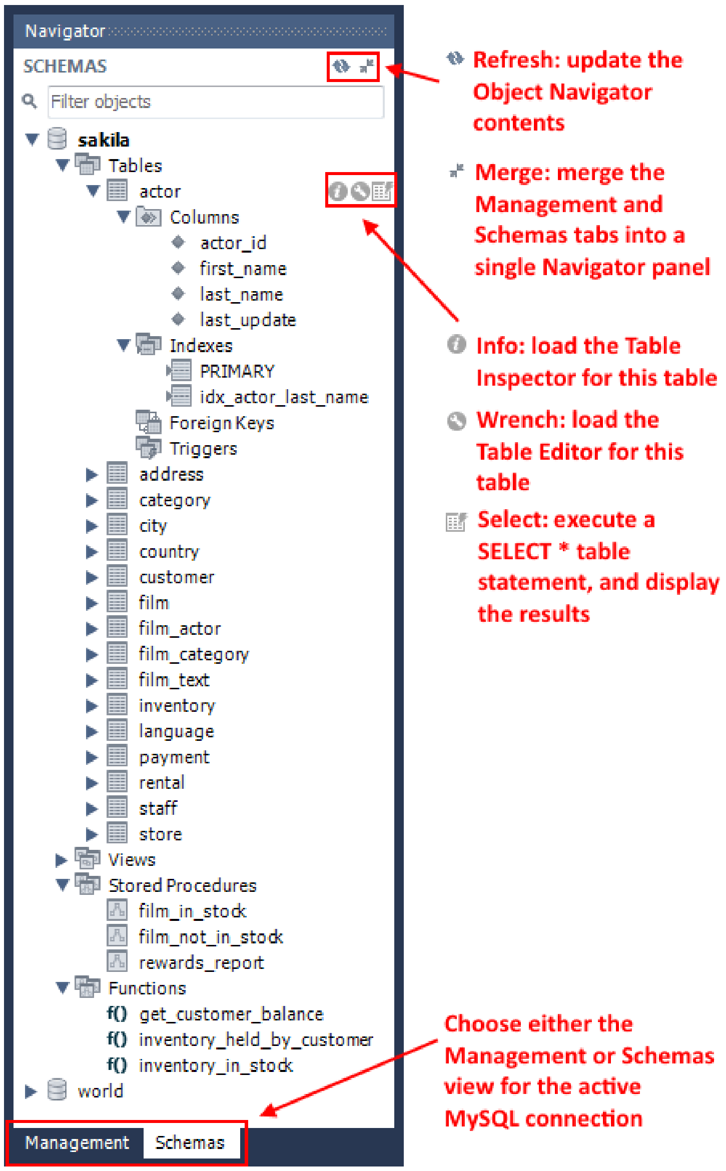
Navigator Schemas Tab

The Schemas list shows available schema on the currently connected server. These can be explored to show tables, views, and routines within the schema.

Note

Internal schemas, such as "performance_schema", "information"schema", "sys", and "mysql", are hidden by default. Toggle the **Show Metadata and Internal Schemas** preference to list them in the object browser. Schemas beginning with a "." are also controlled by this setting.

Figure 8.24 SQL Editor - Navigator Schemas Tab



It is possible to set a schema as the default schema by right-clicking the schema and selecting the **Set As Default Schema** menu item. This executes a `USE schema_name` statement so that subsequent statements without schema qualifiers are executed against this schema. This setting applies only to the query session. To set a default schema for multiple MySQL Workbench sessions, you must set the default schema for the stored connection. From the Home screen, right-click on a MySQL connection, choose **Edit Connection**, and set the desired default schema on the **Default Schema** box.

Note

The selected schema is displayed as `bold` in the Schema navigator.

Double-clicking a table, view, or column name in the schemata explorer inserts the name into the SQL Query area. This reduces typing significantly when entering SQL statements containing references to several tables, views, or columns.

The Schema Navigator also features a context menu which can be displayed by right-clicking an object. For example, right-clicking a table displays the following menu items:

- **Select Rows - Limit 1000**: Pulls up to 1000 rows of table data from the live server into a Results tab, and enables editing. Data can be saved directly to the live server.
- **Table Inspector**: Displays table information, similar to the `Schema Inspector`. This also has a simpler and easier to use interface for analyzing and creating indexes for tables.
- **Table Data Export**: Opens the table export wizard to export the table's data to JSON or customized CSV.
- **Table Data Import**: Opens the table import wizard to import JSON or CSV formatted data to the selected or new table.
- **Copy to Clipboard**: There are various submenus, each of which copies information to the clipboard:
  - Name (short): Copies the table name.
  - Name (long): Copies the qualified table name in the form `` `schema`.`table` ``.
  - Select All Statement: Copies a statement to select all columns in this form:

    ```
    SELECT
    `table`.`column1`,
    `table`.`column2`,
    ...
    FROM `schema`.`table`;
    ```
  - Insert Statement: Copies an `INSERT` statement to insert all columns.
  - Update Statement: Copies an `UPDATE` statement to update all columns.
  - Delete Statement: Copies a `DELETE` statement in the form `` DELETE FROM `world`.`country` WHERE <{where_condition}>; ``.
  - Create Statement: Copies a `CREATE` statement in the form `` DELETE FROM `world`.`country` WHERE <{where_condition}>; ``.
  - Delete with References: Copies a `DELETE` statement, in the form of a transaction, that deletes all objects that reference the row (directly or indirectly).

    Use **Select with References** first to preview this operation.
  - Select with References: Copies a `SELECT` statement that selects all objects that reference the row (directly or indirectly).

    Use **Delete with References** to generate a DELETE statement for this operation.
- **Send to SQL Editor**: Provides functionality similar to Copy to Clipboard. However, this item inserts the SQL code directly into the SQL Query panel, where it can be edited further as required.
- **Create Table**: Launches a dialog to enable you to create a new table.
- **Create Table Like...**: Launches a dialog to enable you to create a new table, and to also apply predefined templates. For additional information, see Section 9.6, “Table Templates”.
- **Alter Table...**: Displays the table editor loaded with the details of the table.

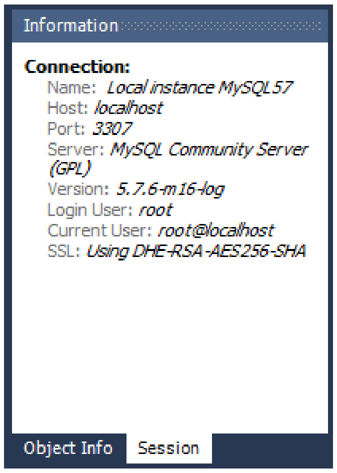
- **Table Maintenance**: Opens a new tab for performing table maintenance operations. Operations include "Analyze Table", "Optimize Table", "Check Table", and "Checksum Table". Additional information about the table may also be viewed from this tab. For additional information, see Schema Inspector.
- **Drop Table...**: Drops the table. All data in the table will be lost if this operation is carried out.
- **Truncate Table...**: Truncates the table.
- **Search Table Data...**: Opens a new tab for performing table searches. It performs a search on all columns, and offers additional options to limit the search.
- **Refresh All**: Refreshes all schemata in the explorer by resynchronizing with the server.

Right-clicking on a schema provides similar options to the table context menu described above, but the operations refer to the Schema. For example, the **Table Maintenance** in the table context menu selects the table in the **Schema Inspector**, which is a schema context menu option.

8.2.2 Session and Object Information Panel

This panel summarizes the current connection to the server.

Figure 8.25 SQL Editor - Connection Information Palette



Connection:
Name: *Local instance MySQL57*
Host: *localhost*
Port: *3307*
Server: *MySQL Community Server (GPL)*
Version: *5.7.6-m16-log*
Login User: *root*
Current User: *root@localhost*
SSL: *Using DHE-RSA-AES256-SHA*

This panel also summarizes information about the object.

Figure 8.26 SQL Editor - Object Info

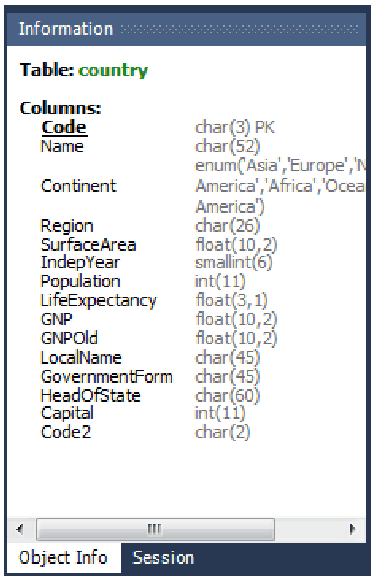


Table: country

Columns:

Code	char(3) PK
Name	char(52)
Continent	enum('Asia','Europe','N America','Africa','Ocea America')
Region	char(26)
SurfaceArea	float(10,2)
IndepYear	smallint(6)
Population	int(11)
LifeExpectancy	float(3,1)
GNP	float(10,2)
GNPOld	float(10,2)
LocalName	char(45)
GovernmentForm	char(45)
HeadOfState	char(60)
Capital	int(11)
Code2	char(2)

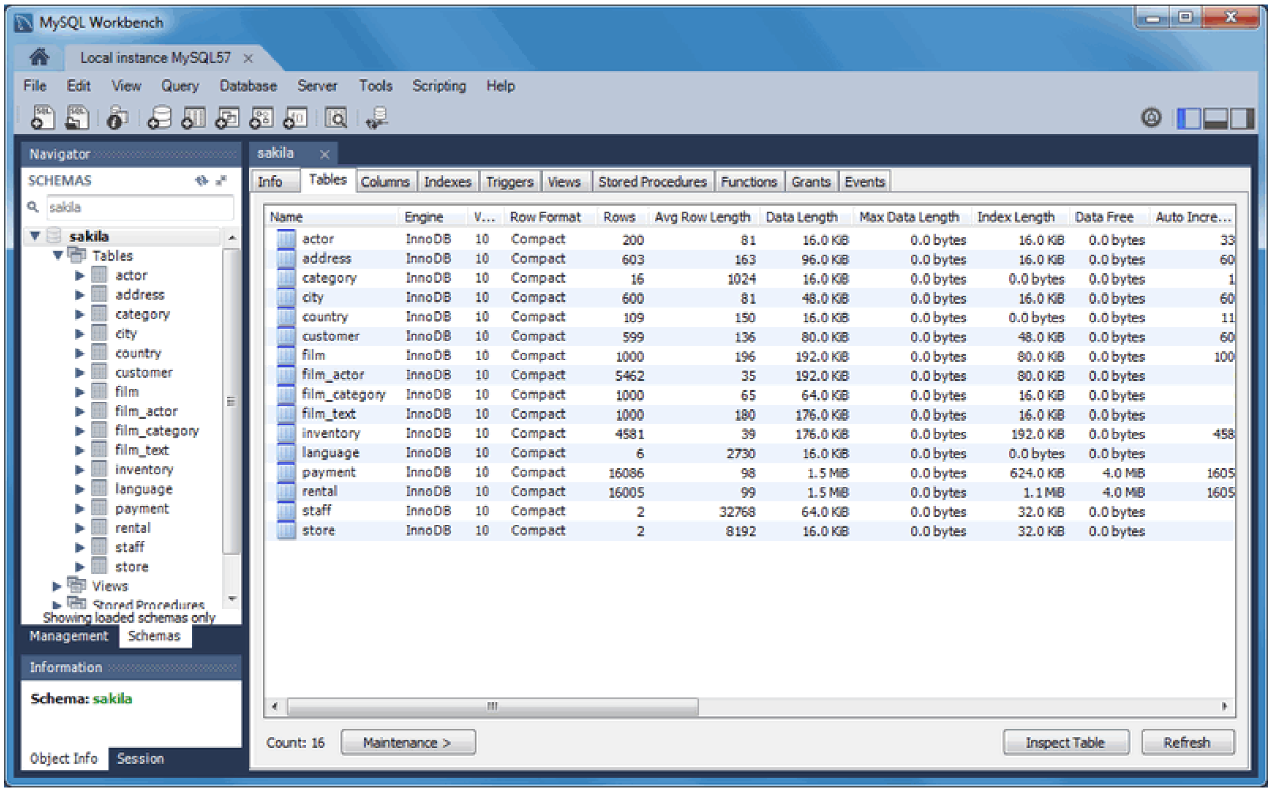
8.2.3 Schema and Table Inspector

The Schema and Table Inspector includes the ability to analyze and repair tables, and also view table metrics.

Schema Inspector

Use the Schema Inspector to browse general information from schema objects. It allows you to perform maintenance tasks on tables such as ANALYZE, OPTIMIZE, CHECK, and CHECKSUM TABLE. To access the inspector, right-click on a schema and select the Schema Inspector

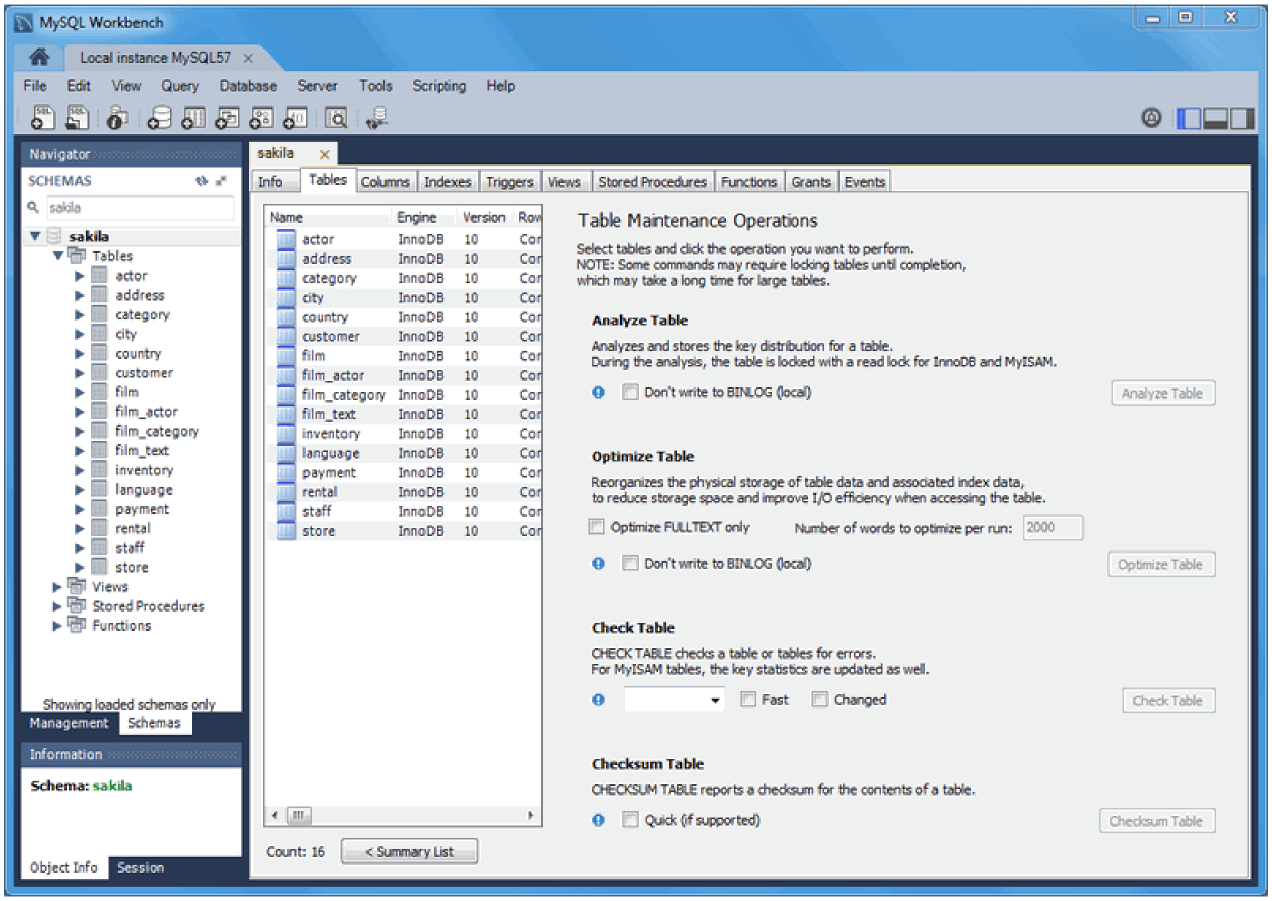
Figure 8.27 Schema Inspector



Name	Engine	V...	Row Format	Rows	Avg Row Length	Data Length	Max Data Length	Index Length	Data Free	Auto Incre...
actor	InnoDB	10	Compact	200	81	16.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	33
address	InnoDB	10	Compact	603	163	96.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	60
category	InnoDB	10	Compact	16	1024	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	1
city	InnoDB	10	Compact	600	81	48.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	60
country	InnoDB	10	Compact	109	150	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	11
customer	InnoDB	10	Compact	599	136	80.0 KiB	0.0 bytes	48.0 KiB	0.0 bytes	60
film	InnoDB	10	Compact	1000	196	192.0 KiB	0.0 bytes	80.0 KiB	0.0 bytes	100
film_actor	InnoDB	10	Compact	5462	35	192.0 KiB	0.0 bytes	80.0 KiB	0.0 bytes	
film_category	InnoDB	10	Compact	1000	65	64.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	
film_text	InnoDB	10	Compact	1000	180	176.0 KiB	0.0 bytes	16.0 KiB	0.0 bytes	
inventory	InnoDB	10	Compact	4581	39	176.0 KiB	0.0 bytes	192.0 KiB	0.0 bytes	458
language	InnoDB	10	Compact	6	2730	16.0 KiB	0.0 bytes	0.0 bytes	0.0 bytes	
payment	InnoDB	10	Compact	16086	98	1.5 MiB	0.0 bytes	624.0 KiB	4.0 MiB	1605
rental	InnoDB	10	Compact	16005	99	1.5 MiB	0.0 bytes	1.1 MiB	4.0 MiB	1605
staff	InnoDB	10	Compact	2	32768	64.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	
store	InnoDB	10	Compact	2	8192	16.0 KiB	0.0 bytes	32.0 KiB	0.0 bytes	

Each tab lists topic oriented information, such as "Tables", "Indexes", and "Triggers". From the **Tables** tab, click **Inspect Table** to open the Table Inspector, or **Maintenance** to open the table maintenance tools:

Figure 8.28 Schema Inspector: Table Maintenance



Schema: sakila

Name	Engine	Version	Row
actor	InnoDB	10	Cor
address	InnoDB	10	Cor
category	InnoDB	10	Cor
city	InnoDB	10	Cor
country	InnoDB	10	Cor
customer	InnoDB	10	Cor
film	InnoDB	10	Cor
film_actor	InnoDB	10	Cor
film_category	InnoDB	10	Cor
film_text	InnoDB	10	Cor
inventory	InnoDB	10	Cor
language	InnoDB	10	Cor
payment	InnoDB	10	Cor
rental	InnoDB	10	Cor
staff	InnoDB	10	Cor
store	InnoDB	10	Cor

Table Maintenance Operations

Select tables and click the operation you want to perform.
NOTE: Some commands may require locking tables until completion,
which may take a long time for large tables.

Analyze Table

Analyzes and stores the key distribution for a table.
During the analysis, the table is locked with a read lock for InnoDB and MyISAM.

Don't write to BINLOG (local)

Optimize Table

Reorganizes the physical storage of table data and associated index data,
to reduce storage space and improve I/O efficiency when accessing the table.

Optimize FULLTEXT only Number of words to optimize per run: 2000

Don't write to BINLOG (local)

Check Table

CHECK TABLE checks a table or tables for errors.
For MyISAM tables, the key statistics are updated as well.

Fast Changed

Checksum Table

CHECKSUM TABLE reports a checksum for the contents of a table.

Quick (if supported)

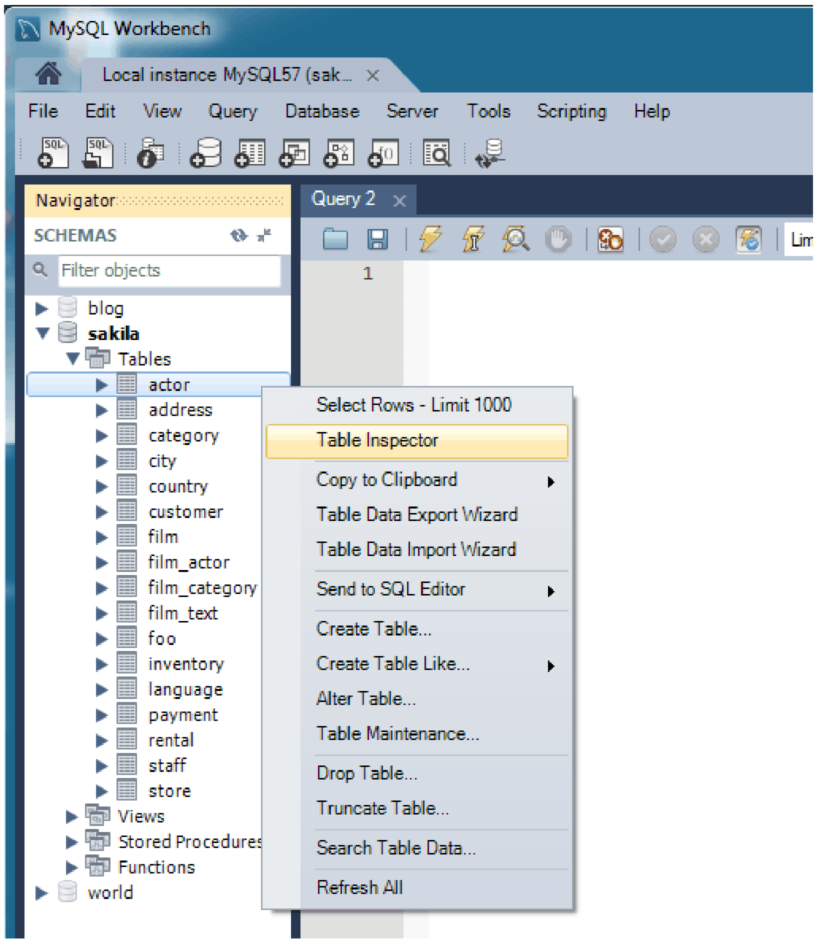
Table Inspector

Table Inspector

View table information, similar to the Schema Inspector. This also has a simpler and easier to use interface for analyzing and creating indexes for your tables.

To open, right-click on a table in the object browser and choose **Table Inspector** from the context menu.

Figure 8.29 Open the Table Inspector



The **Table Inspector** shows information related to the table.

Figure 8.30 Table Inspector: Info Tab

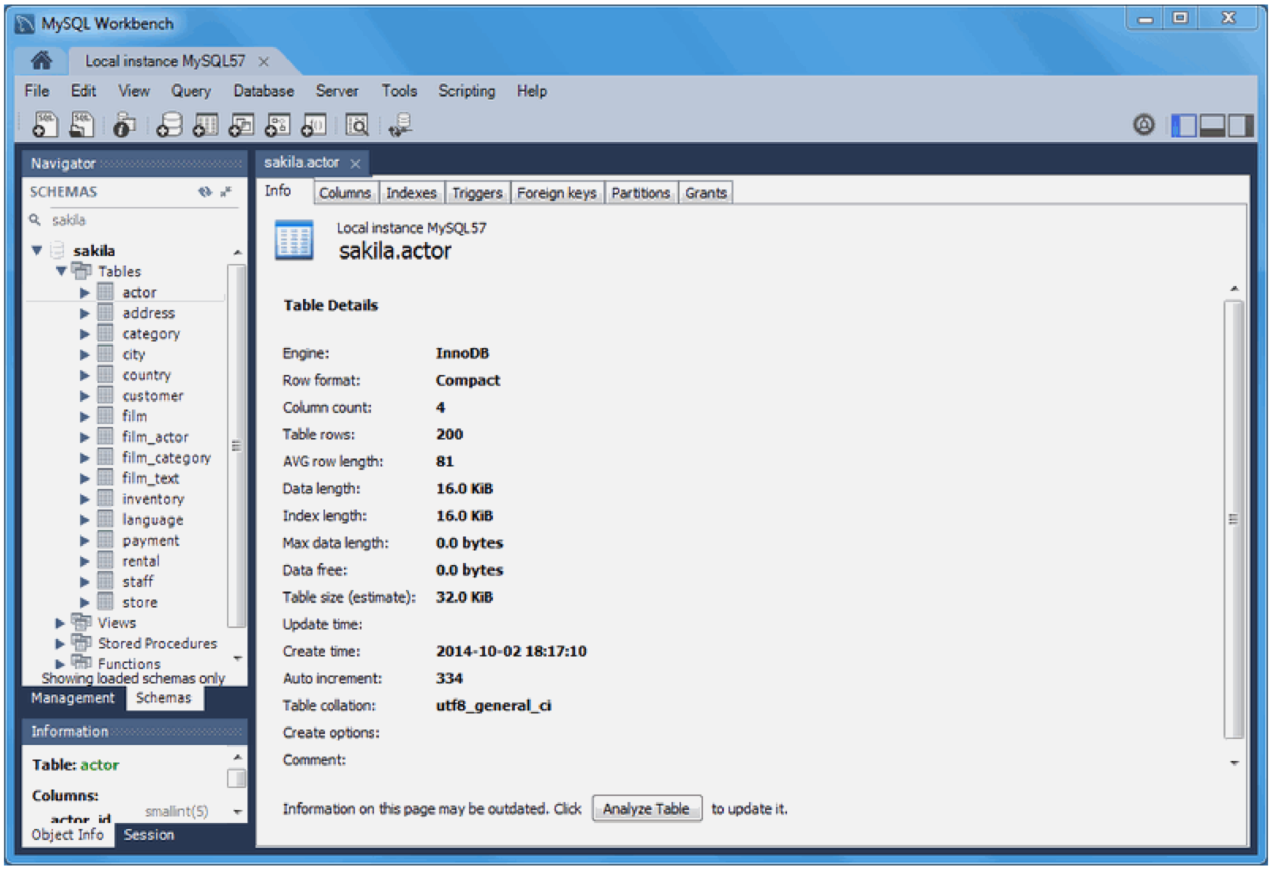


Table: actor

Columns:

actor_id smallint(5)

Local instance MySQL57

sakila.actor

Table Details

Engine:	InnoDB
Row format:	Compact
Column count:	4
Table rows:	200
AVG row length:	81
Data length:	16.0 KiB
Index length:	16.0 KiB
Max data length:	0.0 bytes
Data free:	0.0 bytes
Table size (estimate):	32.0 KiB
Update time:	
Create time:	2014-10-02 18:17:10
Auto increment:	334
Table collation:	utf8_general_ci
Create options:	
Comment:	

Information on this page may be outdated. Click Analyze Table to update it.

Chapter 9 Database Design / Modeling

Table of Contents

Modeling simplifies database design and maintenance by enabling you, the data architect, to visualize requirements and resolve design issues. Model-driven database design is an efficient methodology for creating valid and well-performing databases, while providing the flexibility to respond to evolving data requirements. Models are used to build ER diagrams and physical MySQL databases.

MySQL Workbench provides extensive capabilities for creating and manipulating database models, including these:

- Create and manipulate a model graphically
- Reverse engineer a live database to a model
- Forward engineer a model to a script or live database
- Create and edit tables and insert data

This is not an exhaustive list. The following sections discuss these and additional data-modeling capabilities.

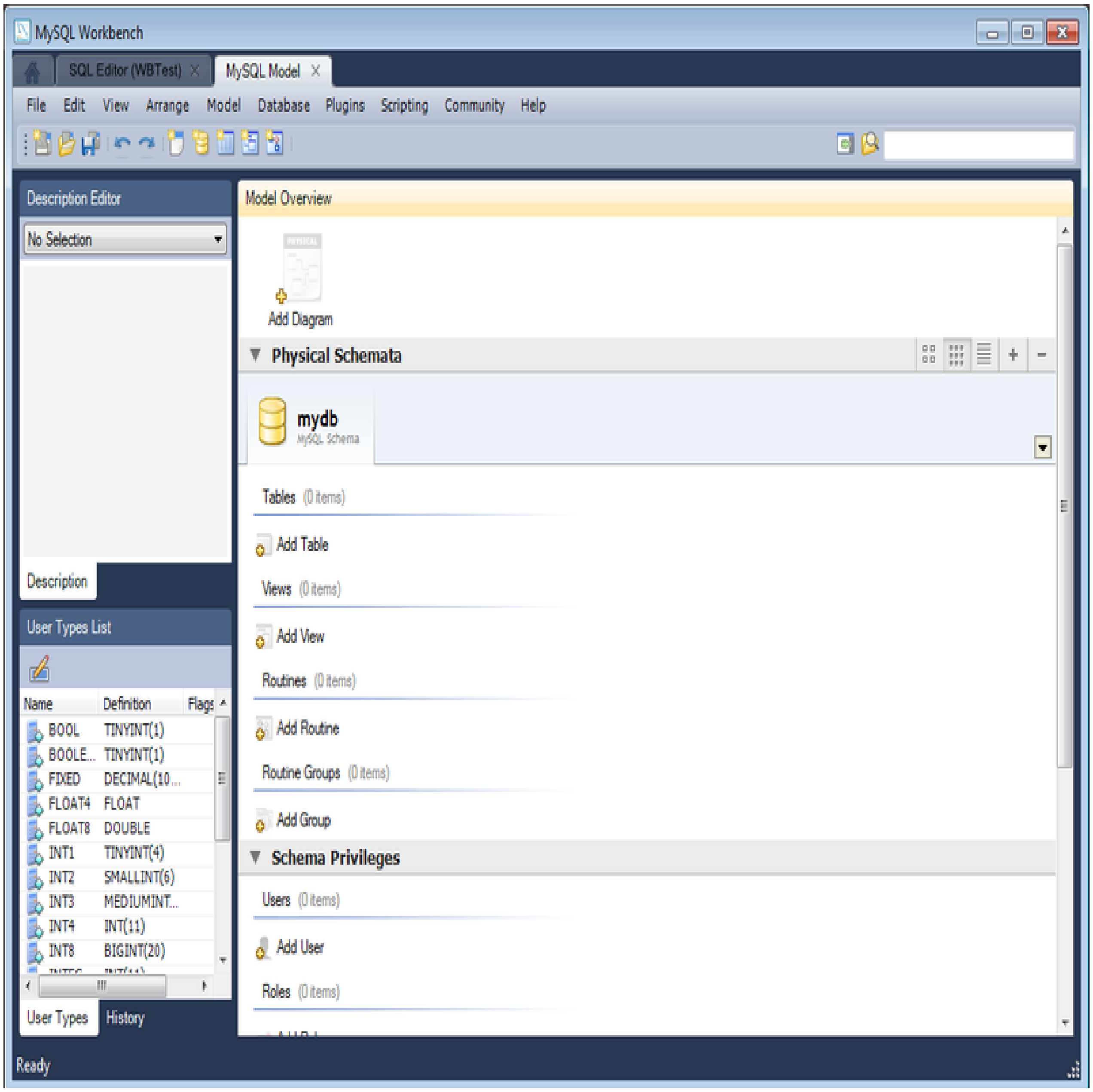
9.1 Modeling Interface

Modeling concepts and interface elements are described below.

9.1.1 Model Editor

When the Model Editor is executed from the Home screen, MySQL Workbench displays the MySQL Model page. The MySQL Model page has three main panels, as shown in the following screenshot: Description Editor, User Types List/History panel, and Model Overview.

Figure 9.1 The MySQL Model Page



Name	Definition	Flags
BOOL	TINYINT(1)	
BOOLE...	TINYINT(1)	
FIXED	DECIMAL(10...	
FLOAT4	FLOAT	
FLOAT8	DOUBLE	
INT1	TINYINT(4)	
INT2	SMALLINT(6)	
INT3	MEDIUMINT...	
INT4	INT(11)	
INT8	BIGINT(20)	

The Description Editor and User Types List/History panel are contained within the Sidebar. The Sidebar is located on the left by default, but can be relocated to the right using a setting in the Workbench Preferences dialog.

The **Model Overview** panel includes the following sections:

- EER Diagrams
- Physical Schemata
- Schema Privileges
- SQL Scripts
- Model Notes

For each of these sections, add objects to a project by clicking the appropriate add-object icon. You may also rename, edit, cut, copy, or delete objects on this page by right-clicking to open a pop-up menu.

The following sections further discuss the MySQL Model page.

9.1.1.1 Modeling Menus

Some menu items are not available in the MySQL Workbench Community edition of this application, and are available only in the MySQL Workbench commercial editions;. This is indicated where applicable.

The File Menu

Use the **File** menu to open a project, begin a new project, or save a project. Choosing **Model** opens the default schema, `mydb`. Choosing **Open Model** opens a file dialog box with the default file type set to MySQL Workbench Models (`mwb` extension). To display a list of recently opened MWB files, choose the **Open Recent** menu item. The keyboard shortcut to create a new project is **Control+N** and the command to open an existing project is **Control+O**.

To close the currently active `MySQL Model` or `EER Diagram` tab, use the **Close Tab** menu item. You can also do this from the keyboard by pressing **Control+W**. To reopen the `MySQL Model` tab, see The View Menu. To reopen an `EER Diagram` tab, double-click the `EER Diagram` icon in the `EER Diagrams` section of the `MySQL Model` page.

Use the **Save Model** or **Save Model As** menu items to save a model. When you save a model, its name appears in the title bar of the application. If you have made changes to a project and have not saved those changes, an asterisk appears in the title bar following the model name. When you save a model, it is saved as a MySQL Workbench file with the extension `mwb`.

Use the **Import** menu item to import a MySQL data definition (DDL) script file. For example, this might be a file created by issuing the command `mysqldump --no-data`. MySQL Workbench handles the script as follows:

- If the script does not contain a `CREATE DATABASE` *`db_name`*`;` statement, the schema objects are copied to the default schema, `mydb`.
- If the script creates a database, a new tab bearing the database name is added to the `Physical Schemata` section of the `MySQL Model` page.
- If the script contains data, the data is ignored.

For details about importing a DDL script, see Section 9.4.2.1, “Reverse Engineering Using a Create Script”.

Under the **Import** submenu, you can also import `DBDesigner4` files.

There are variety of items under the **Export** submenu. You may generate the SQL statements necessary to create a new database or alter an existing one. For more information about these menu items, see Section 9.4.1.1, "Forward Engineering Using an SQL Script".

Using the **Export** submenu, you can also export an EER diagram as a PNG, SVG, PDF, or Postscript file. For an example of a PNG file, see Figure 9.29, "The sakila Database EER Diagram".

The **Page Setup** menu item enables you to set the paper size, orientation, and margins for printing purposes.

The printing options are enabled only if the **EER Diagrams** tab is selected. You have the choice of printing your model directly to your printer, printing it as a PDF file, or creating a PostScript file. For more information, see Section 9.2.1, "Printing Diagrams".

Note

The printing options are available only in commercial versions of MySQL Workbench.

Use the **Document Properties** menu item to set the following properties of your project:

- `Name`: The model name (default is `MySQL Model`)
- `Version`: The project version number
- `Author`: The project author
- `Project`: The project name
- `Created`: Not editable; determined by the MWB file attributes
- `Last Changed`: Not editable; determined by the MWB file attributes
- `Description`: A description of your project

The Edit Menu

Use the **Edit** menu to make changes to objects. The menu item text descriptions change to reflect the name of the selected object.

This menu has items for cutting, copying, and pasting. These actions can also be performed using the **Control+X**, **Control+C**, and **Control+V** key combinations. Undo a deletion using the **Undo Delete '*object_name*'** item. The **Control+Z** key combination can also be used to undo an operation. It is also possible to carry out a **Redo** operation using either the menu item, or the key combination **Control+Y**.

Also find a **Delete '*object_name*'** menu item for removing the currently selected object. The keyboard command for this action is **Control+Delete**. You can also right-click an object and choose the delete option from the pop-up menu.

The **Delete '*object_name*'** menu item behaves differently depending upon circumstances. For example, if an **EER Diagram** is active and a table on the canvas is the currently selected object, a dialog box may open asking whether you want to remove the table from the canvas only or from the database as well. For information about setting the default behavior when deleting from an EER Diagram, see Section 3.2.4, "Modeling Preferences".

Warning

If the `MySQL Model` page is active, the selected object is deleted from the catalog and there will be *no confirmation dialog box*.

Choose **Edit Selected** to edit the currently selected object. You can also perform edits in a new window by selecting **Edit Selected in New Window**. The keyboard shortcuts for **Edit Selected** and **Edit Selected in New Window** are **Control+E** and **Control+Shift+E**, respectively.

The **Select** item has the following submenus:

- **Select All** (Keyboard shortcut, **Control+A**): Selects all the objects on the active EER diagram.
- **Similar Figures** (Objects of the same type): Finds objects similar to the currently selected object.
- **Connected Figures**: Finds all the objects connected to the currently selected object.

These menu items are active only when an **EER Diagram** tab is selected. The **Similar Figures** and the **Connected Figures** menu items are disabled if no object is currently selected on an EER diagram.

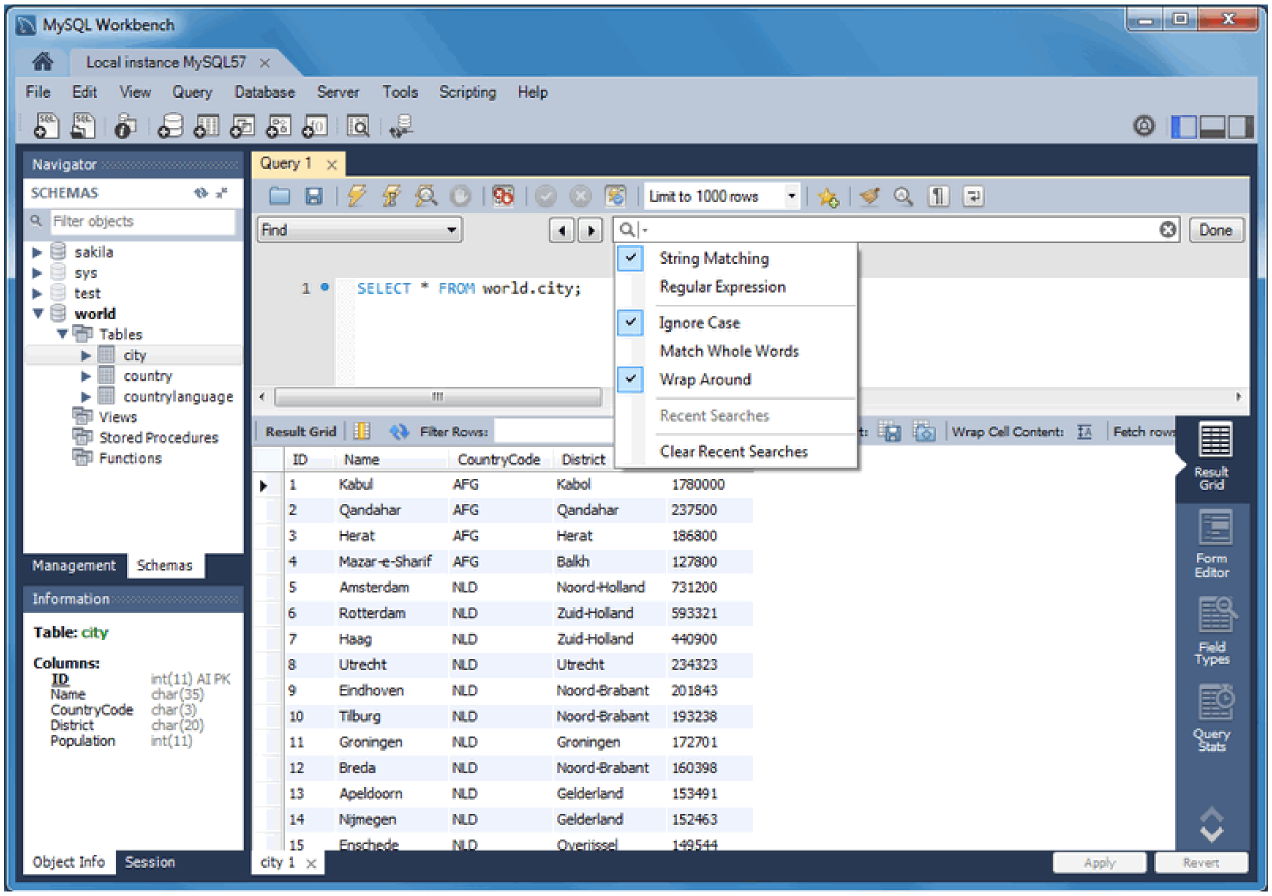
When multiple objects have been selected using one of these menu items, you can navigate between selected items by choosing the **Go to Next Selected** or **Go to previous Selected** menu item.

Selecting objects changes some of the **Edit** menu items. If only one object is selected, that object's name appears after the **Cut**, **Copy** and **Delete** menu items. If more than one object is selected, these menu items show the number of objects selected.

Find Dialog Window

Each MySQL Workbench window includes search functionality. The **Find** panel with **Find & Replace** enabled is shown below:

Figure 9.2 The Find Panel with Find & Replace



String Matching ✓
Regular Expression
Ignore Case ✓
Match Whole Words
Wrap Around ✓
Recent Searches
Clear Recent Searches

```
SELECT * FROM world.city;
```

Table: city

Columns:

ID	int(11) AI PK
Name	char(35)
CountryCode	char(3)
District	char(20)
Population	int(11)

ID	Name	CountryCode	District	
1	Kabul	AFG	Kabol	1780000
2	Qandahar	AFG	Qandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haag	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilburg	NLD	Noord-Brabant	193238
11	Groningen	NLD	Groningen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Nijmegen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544

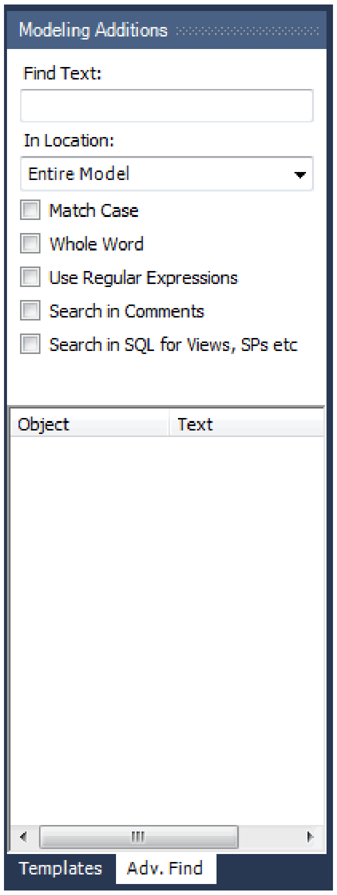
Find options

The Find dialogue options are described below:

- **String Matching** (default) or **Regular Expression**: Search by matching a string, or a PCRE regular expression.
- **Ignore Case**: A case-insensitive search. Works with both the **String Matching** and **Regular Expression** search methods. Enabled by default.
- **Match Whole Words**: If enabled, only whole strings are matched. For example, a search for "home" would not match "home_id". Disabled by default.
- **Wrap Around**: The search will wrap around to the beginning of the document, as otherwise it will only search from the cursor position to the end of the document. Enabled by default.
- And the arrows jump to the discovered search terms, and behave according to the **Wrap Around** option.

The MySQL Workbench commercial editions include an advanced Find facility for models:

Figure 9.3 The Find Window



You can search the following locations:

- Entire Model: Searches the entire model.
- Current View: Searches the current view only. This may be the `MySQL Model` page.
- All Views: Searches the `MySQL Model Page` and all EER diagrams.

- Database Objects: Searches database objects only.
- Selected Figures: Searches the currently selected objects. This feature works only for EER diagrams.

Enter the text you wish to search for in the **Find Text** list. You may also select any or all of the following check boxes:

- Match Case
- Whole Word
- Use Regular Expression
- Search in Comments
- Search in SQL for Views, SPs etc.

Any text you enter into the **Find Text** list is retained for the duration of your session. Use the **Next** or **Previous** buttons to find occurrences of your search criterion.

Clicking the **Find All** button opens a **Find Results** window anchored at the bottom of the application. If you wish, you may undock this window as you would any other.

Use this window to navigate to objects. For example, double-clicking the `Description` of an object located on an EER diagram navigates to the specific diagram and selects the object. Notice that the properties of the object are displayed in the `Properties` palette.

The `Find` dialog window can also be opened using the **Control+F** key combination. Use **Control+G** to find the next occurrence and **Control+Shift+G** to find a previous occurrence. Close the `Find` dialog window by clicking the **x** in the top right corner or by pressing the **Esc** key.

Workbench Preferences

This menu item enables you to set global preferences for the MySQL Workbench application.

For further information, see Section 3.2, “Workbench Preferences”.

The View Menu

This context-aware menu features general options for changing the view in MySQL Workbench. These options change depending on the current tab, and here are the available **View** menu items:

General options:

- **Home**: Selects the Home screen.
- **Panels**: Configure which of the three available panels are open. You may also manage this from the GUI using the panel toggle buttons on the top-right side of MySQL Workbench.
- **Output**: Displays the console output.
- **Select Next Main Tab**: Selects the next (moves to the right, and wraps around) MySQL Workbench tab.
- **Select Next Main Tab**: Selects the previous (moves to the left, and wraps around) MySQL Workbench tab.

Model/EER options:

- **Windows**: A submenu with items that activate (slide open) specific panels. Designated panels include the "Model Navigator", "Catalog", "Layers", "User Datatypes", "Object Descriptions", "Object Properties", and "Undo History".

- **Zoom 100%**: The default level of detail of an EER diagram.
- **Zoom In**: Zooms in on an EER diagram.
- **Zoom Out**: Zooms out from an EER diagram.

  The ability to zoom in on an EER diagram is also available using the slider tool in the `Model Navigator` palette. See Section 9.1.1.9, “The Model Navigator Panel”.
- **Set Marker**: Bookmarks an object. From the keyboard, select the object you wish to bookmark, then use the key combination **Control+Shift** and the number of the marker (1 through 9). You may create up to nine markers.
- **Go To Marker**: Returns to a marker. From the keyboard, use the **Control** key and the number of the marker.
- **Toggle Grid**: Displays grid lines on an EER diagram.
- **Toggle Page Guides**: Toggles Page Guides to help design the EER diagram on a per-page basis.

The Arrange Menu

The **Arrange** menu items apply only to objects on an EER diagram canvas and are enabled only if an EER diagram view is active. The **Arrange** menu has these items:

- **Align to Grid**: Aligns items on the canvas to the grid lines
- **Bring to Front**: Brings objects to the foreground
- **Send to Back**: Sends objects to the background
- **Center Diagram Contents**: Centers objects on the canvas
- **Autolayout**: Automatically arranges objects on the canvas
- **Reset Object Size**: Expands an object on an EER diagram. For example, if a table has a long column name that is not fully displayed, this menu item expands the table to make the column visible. This menu item is not enabled unless an object is selected.
- **Expand All**: Use this item to expand all objects on an EER diagram. This item will display a table's columns if the object notation supports expansion. Some object notations, such as `Classic`, do not permit expansion or contraction. Indexes will not automatically be expanded unless they were previously expanded and have been collapsed using the **Collapse All** menu item.
- **Collapse All**: Undo the operation performed by **Expand All**.

The Model Menu

When a model is opened, this menu features actions to perform against your model, and the **Model** menu has these items:

- **Add Diagram**: Creates a new EER Diagram. The keyboard shortcut is **Control+T**.
- **Create Diagram From Catalog Objects**: Creates an EER diagram from all the objects in the catalog.
- **User Defined Types**: Presents a dialog box that enables you to add and delete user defined data types.
- **DBDoc – Model Reporting...**: For information about this menu item, see The DBDoc Model Reporting Dialog Window (Commercial Version). Commercial version only.

- **Validation**: Checks the validity of the model using ANSI standards. For information about this menu item, see The Validation Submenus (Commercial Version). Commercial version only.
- **Validation (MySQL)**: Checks the validity of the model using MySQL standards. For information about this menu item, see The Validation Submenus (Commercial Version). Commercial version only.
- **Object Notation**: For information about this menu item, see The Object Notation Submenu.
- **Relationship Notation**: For information about this menu item, see The Relationship Notation Submenu.
- **Diagram Properties and Size**: Opens a diagram size dialog box that enables you to adjust the width or height of the canvas. The unit of measure is pages; the default value is two.

  When you have tables with numerous columns, use this menu item to increase the size of the EER.
- **Model Options**: Sets options at the model level. These options should not be confused with the options that are set globally for the Workbench application, and which are referred to as Workbench Preferences. The available model options are a subset of the Workbench Preferences options.

  For more information about Workbench Preferences, see Section 3.2.4, "Modeling Preferences".

The DBDoc Model Reporting Dialog Window (Commercial Version)

This dialog window is found by navigating to the **Model** menu and choosing the **DBDoc - Model Reporting...** item.

Note

The **DBDoc - Model Reporting...** feature is only available in the MySQL Workbench commercial editions.

Use this dialog window to set the options for creating documentation of your database models. For more information, see Section 9.2.2, "DBDoc Model Reporting".

The Validation Submenus (Commercial Version)

The **Model** menu has two validation submenus, **Validation** and **Validation (MySQL)**. Use these submenus for general validation and MySQL-specific validation of the objects and relationships defined in your model.

Note

These items are only available in the MySQL Workbench commercial editions.

The **Validation** submenu has these items:

- **Validate All**: Performs all available validation checks
- **Empty Content Validation**: Checks for objects with no content, such as a table with no columns
- **Table Efficiency Validation**: Checks the efficiency of tables, such as a table with no primary key defined
- **Duplicate Identifiers Validation**: Checks for duplicate identifiers, such as two tables with the same name
- **Consistency Validation**: Checks for consistent naming conventions
- **Logic Validation**: Checks, for example, that a foreign key does not reference a nonprimary key column in the source table

The **Validation (MySQL)** submenu has these items:

- **Validate All**: Performs all available validation checks
- **Integrity Validation**: Checks for invalid references, such as a table name longer than the maximum permitted
- **Syntax validation**: Checks for correct SQL syntax
- **Duplicate Identifiers Validation (Additions)**: Checks for objects with the same name

For detailed information about validation, see Section 9.2.3, “Schema Validation Plugins”.

The Object Notation Submenu

The items under the **Model**, **Object Notation** submenu apply exclusively to an EER diagram. They are not enabled unless an EER diagram tab is selected.

The **Object Notation** submenu has these items:

- **Workbench (Default)**: Displays table columns, indexes, and triggers
- **Workbench (Simplified)**: Shows only a table's columns
- **Classic**: Similar to the `Workbench (Simplified)` style showing only the table's columns
- **IDEF1X**: The ICAM DEFinition language information modeling style

The object notation style that you choose persists for the duration of your MySQL Workbench session and is saved along with your model. When MySQL Workbench is restarted, the object notation reverts to the default.

Note

If you plan to export or print an EER diagram be sure to decide on a notation style first. Changing notation styles after objects have been placed on a diagram can significantly change the appearance of the diagram.

The Relationship Notation Submenu

The items under the **Relationship Notation** submenu apply exclusively to an EER diagram. They are not enabled unless an EER diagram tab is selected.

The **Relationship Notation** submenu has these items:

- **Crow's Foot (IE)**: The default modeling style. For an example, see Figure 9.24, “Adding Tables to the Canvas”.
- **Classic**: Uses a diamond shape to indicate cardinality.
- **Connect to Columns**
- **UML**: Universal Modeling Language style.
- **IDEF1X**: The ICAM DEFinition language information modeling method

To view the different styles, set up a relationship between two or more tables and choose the different menu items.

The relationship notation style that you choose persists for the duration of your MySQL Workbench session and is saved along with your model. When MySQL Workbench is restarted, the relationship notation reverts to the default, the `Crow's Foot` style.

Note

If you plan to export or print an EER diagram, be sure to decide on a notation style first. Changing notation styles after objects have been placed on a diagram can significantly change the appearance of the diagram.

The Database Menu

This menu features actions against the connected MySQL server. The **Database** menu has these items:

- **Query Database**: Launches the SQL Editor, which enables you to create SQL code and execute it on a live server. For more information, see Section 8.1, “Visual SQL Editor”.
- **Manage Connections**: Launches the **Manage Server Connections** dialog, which enables you to create and manage multiple connections. For more information, see Section 5.3, “Manage Server Connections”
- **Reverse Engineer**: Creates a model from an existing database. For more information, see Section 9.4.2.2, “Reverse Engineering a Live Database”.
- **Forward Engineer**: Creates a database from a model. For more information, see Section 9.4.1.2, “Forward Engineering to a Live Server”.
- **Schema Transfer Wizard...**: Executes the database migration wizard for MySQL databases. It is useful for moving from an older MySQL server to the latest MySQL version, and is meant for local development purposes. You should not use this tool on production MySQL instances as they often require more complex data migration techniques.

  For additional information about this wizard, see MySQL Schema Transfer Wizard.
- **Migration Wizard...**: Executes the database migration wizard for most any database, and is meant to migrate tables and data from supported database systems to your MySQL server. For additional information, see Chapter 10, *Database Migration Wizard*.
- **Edit Type Mappings for Generic Migration**: From here you can define custom type mappings, such as migrating the source data type `int8` to the target MySQL data type `BIGINT`.
- **Synchronize Model**: Synchronizes your database model with an existing database. For more information, see Section 9.5.1, “Database Synchronization”.
- **Synchronize with Any Source**: Allows you to compare a target database or script with the open model, external script, or a second database, and apply these changes back to the target. For more information, see Section 9.5.1, “Database Synchronization”.
- **Compare Schemas**: Compares your schema model with a live database or a script file. Section 9.5.2, “Compare and Report Differences in Catalogs”.

The Tools Menu

The **Tools** menu lists tools and utilities that related to MySQL Workbench usage.

- **Browse Audit Log File**: Launches a file browser to open a specific audit log file. MySQL Workbench prompts for sudo access if the MySQL Workbench user is unable to read the audit log file. For additional information about the Audit Inspector, see Section 6.6, “MySQL Audit Inspector Interface”. Commercial only.
- **Configuration**: Backup (or restore) your MySQL Connections, as defined in MySQL Workbench. Connection data is stored in a `connections.xml` file, for additional information about this file, see Section 3.3, “MySQL Workbench Settings and Log Files”.

- **Utilities**: These utilities generate PHP code to either "Connect to the MySQL server" or "Iterate SELECT results", if applicable. For additional information about PHP code generation, see Section 8.1.12.2, “Generating PHP Code”.
- **Start Shell for MySQL Utilities**: Opens the `mysqluc` MySQL Utility. For additional information about MySQL Utilities, see Appendix F, *MySQL Utilities*.

The Scripting Menu

This menu features GRT scripting and plugin options. The **Scripting** menu has these items:

- **Scripting Shell**: Launches the MySQL Workbench Scripting Shell. For additional information, see Section C.5, “The Workbench Scripting Shell”.
- **New Script**: Opens a **New Script File** dialogue, with options to create a **Python Script**, **Python Plugin**, or **Python Module**.
- **Open Script**: Opens a **Open GRT Script** dialogue, which defaults to the Workbench scripts directory. Files are opened into the **Workbench Scripting Shell** window.
- **Run Script File**: Executes the script that is currently open.
- **Run Workbench Script File**: Executes the specified script file.
- **Install Plugin/Module File**: Loads and installs a plugin or module file
- **Plugin Manager**: Displays information about the plugins that are installed, and allows disabling and uninstalling the plugins.

The Help Menu

Use the **Help** menu when you require support, or when you want to help improve MySQL Workbench. This menu has the following items:

- **Help Index**: Opens a window showing a local copy of the MySQL Workbench documentation. Read, search, or print the documentation from this window.
- **MySQL.com Website**: Opens your default browser on the MySQL Web site home page.
- **Workbench Product Page**: Opens your default browser on the MySQL Workbench product page.
- **System Info**: Displays information about your system, which is useful when reporting a bug. For more information, see System Info.
- **Report a Bug**: Opens your default browser to bugs.mysql.com, and automatically fills in several fields such as the Operating System and MySQL Workbench version by passing in additional data via the GET request. The default "Description" requests you to also attach the Workbench log file. For additional information about reporting useful bug reports, see Appendix D, *How To Report Bugs or Problems*.
- **View Reported Bugs**: Opens your default browser to see a list of current bugs.
- **Locate Log Files**: Opens up the directory that contains the MySQL Workbench log files.
- **Show Log File**: Opens up the main MySQL Workbench log file in your default text editor. This file is typically named `wb.log`.
- **Check For Updates**: Checks if you are using the current MySQL Workbench version. If you are, then a popup informs you of this. If not, then a prompt asks you to open the MySQL Workbench download page.

- **About Workbench**: Displays the MySQL Workbench `About` window. This also displays the MySQL Workbench version.

System Info

Use the **Help**, **System Info** menu item to display information about your system. This item is especially useful for determining your rendering mode. Sample output follows.

```
MySQL Workbench Community (GPL) for Windows version 6.1.4  revision 11773 build 1454
Configuration Directory: C:\Users\philip\AppData\Roaming\MySQL\Workbench
Data Directory: C:\Users\philip\Desktop\MySQL\MySQL Workbench 6.1.4 CE
Cairo Version: 1.8.8
OS: Microsoft Windows 7  Service Pack 1 (build 7601), 64-bit
CPU: 4x Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz, 8.0 GiB RAM
Active video adapter NVIDIA GeForce GT 610
Installed video RAM: 1024 MB
Current video mode: 1920 x 1080 x 4294967296 colors
Used bit depth: 32
Driver version: 9.18.13.2049
Installed display drivers: nvd3dumx.dll,nvwgf2umx.dll,nvwgf2umx.dll,nvd3dum,nvwgf2um,nvwgf2um
Current user language: English (United States)
```

9.1.1.2 The Toolbar

The MySQL Workbench toolbar is located immediately below the menu bar. Click the tools in the toolbar to perform the following actions:

- The new document icon: Creates a new document
- The folder icon: Opens a MySQL Workbench file (`mwb` extension)
- The save icon: Saves the current MySQL Workbench project
- The right and left arrows: The left arrow performs an “Undo” operation. The right arrow performs a “Redo” operation.

Other tools appear on the toolbar depending upon the context.

Tool-Specific Toolbar Items

When an EER diagram canvas is selected, the following icons appear to the right of the arrow icons:

- The toggle grid icon: Turns the grid on and off
- The grid icon: Aligns objects on the canvas with the grid
- The new EER diagram icon: Creates a new EER diagram tab.

The toolbar also changes depending upon which tool from the vertical toolbar is active. For discussion of these tools, see Section 9.1.2.1, “The Vertical Toolbar”.

If the `Table` tool is active, schemata lists, engine types, and collations appear on the toolbar. The table properties can be modified using the Properties Editor.

When an object is selected, the object's properties, such as color, can be changed in the Properties Editor.

9.1.1.3 EER Diagrams

Use the `Add new Diagram` icon in the `MySQL Model` area to create EER diagrams. When you add an EER diagram, a new tab appears below the toolbar. Use this tab to navigate to the newly created EER diagram. For further discussion of EER Diagrams, see Section 9.1.2, “EER Diagram Editor”.

9.1.1.4 The Physical Schemata Panel

The `Physical Schemata` panel of the `MySQL Model` page shows the active schemata and the objects that they contain.

Expand and contract the `Physical Schemata` section by double-clicking the arrow on the left of the `Physical Schemata` title bar. When the `Physical Schemata` section is expanded, it displays all currently loaded schemata.

Each schema shows as a tab. To select a specific schema, click its tab. When MySQL Workbench is first opened, a default schema, `mydb`, is selected. You can start working with this schema or you can load a new MySQL Workbench Model file (models use the `.mwb` extension.)

There are a variety of ways to add schema to the `Physical Schemata` panel. You can open an MWB file, reverse engineer a MySQL create script, or, if you are using a commercial version of MySQL Workbench, you can reverse engineer a database by connecting to a MySQL server.

You can also add a new schema by clicking the **+** button on the top right of the `Physical Schemata` panel. To remove a schema, click its tab and use the **-** button found to the immediate left of the **+** button. To the left of these buttons are three buttons that control how database object icons are displayed:

- The left button displays database objects as large icons.
- The middle button displays small icons in multiple rows.
- The right button displays small icons in a single list.

The Schema Objects Panel

The `Physical Schemata` panel has the following sections:

- Tables
- Views
- Routines
- Routine Groups

Each section contains the specified database objects and an icon used for creating additional objects.

Any database objects added to an EER diagram canvas also show up in the `Physical Schemata` section. For information about adding objects to an EER diagram canvas, see Section 9.1.2, “EER Diagram Editor”.

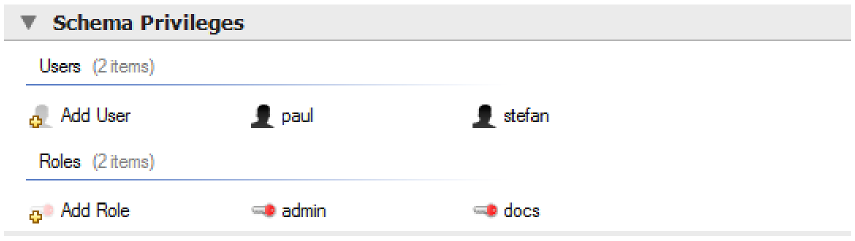
9.1.1.5 The Schema Privileges Panel

The `Schema Privileges` panel has the following sections, used to create users for your schemata and to define roles —:

- Users
- Roles

The following image displays the `Schema Privileges` section of the `MySQL Model` tab.

Figure 9.4 Roles and Privileges



Adding Roles

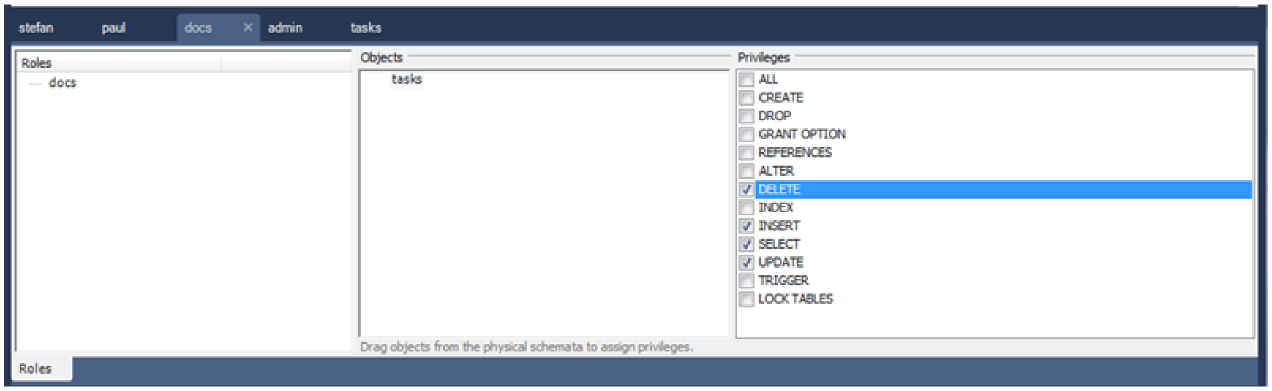
To add a role, double-click the `Add Role` icon. This creates a role with the default name `role1`. Right-clicking a role opens a pop-up menu with the following items:

- **Cut '*`role_name`*'**: Cuts the role
- **Copy '*`role_name`*'**: Copies the role
- **Edit Role...**: Opens the role editor
- **Edit in New Window...**: Opens the role editor in a new editor window
- **Delete '*`role_name`*'**: Removes the role
- **Copy SQL to Clipboard**: Currently not implemented

To rename a role, click the role name. Then you will be able to edit the text.

All defined roles are listed under **Roles** on the left side of the role editor. Double-clicking a role object opens the role editor docked at the bottom of the page.

Figure 9.5 Role Editor



Drag objects from the physical schemata to assign privileges.

- [ ] ALL
- [ ] CREATE
- [ ] DROP
- [ ] GRANT OPTION
- [ ] REFERENCES
- [ ] ALTER
- [✓] DELETE
- [ ] INDEX
- [✓] INSERT
- [✓] SELECT
- [✓] UPDATE
- [ ] TRIGGER
- [ ] LOCK TABLES

Select the role to which you wish to add objects. You may drag and drop objects from the `Physical Schemata` to the `Objects` section of the role editor. To assign privileges to a role, select it from the `Roles` section, then select an object in the `Objects` section. In the `Privileges` section, check the rights you wish to assign to this role. For example, a `web_user` role might have only `SELECT` privileges and only for database objects exposed through a web interface. Creating roles can make the process of assigning rights to new users much easier.

Adding Users

To add a user, double-click the `Add User` icon. This creates a user with the default name `user1`. Double-clicking this user opens the user editor docked at the bottom of the application.

In the User Editor, set the user's name and password using the **Name** and **Password** fields. Assign one role or a number of roles to the user by selecting the desired roles from the field on the right and then clicking the **<** button. Roles may be revoked by moving them in the opposite direction.

Right-clicking a user opens a pop-up menu. The items in the menu function as described in Adding Roles.

9.1.1.6 The SQL Scripts Panel

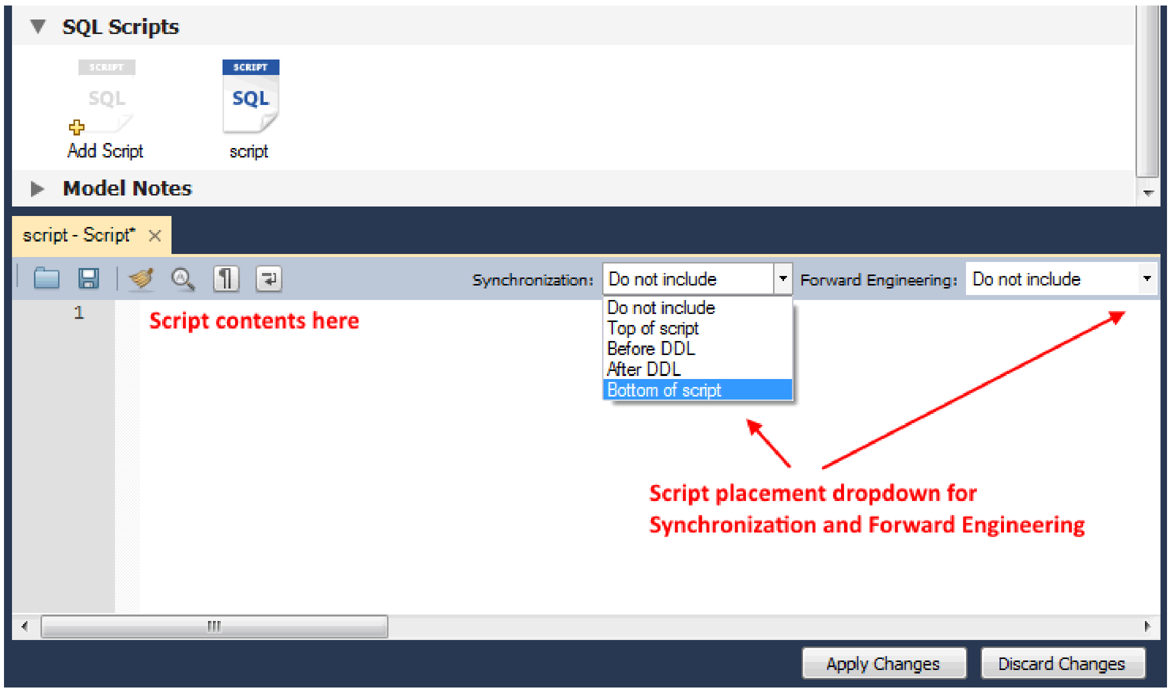
Use the SQL Scripts panel to attach SQL scripts to the model for documentation and organizational purposes, and optionally these attachments can be included in the output script when performing forward engineering or model/schema synchronization.

If you created your project from an SQL script and plan to create an ALTER script, you may want to add the original script here, since it will be needed to create an ALTER script. For more information, see Altering a Schema.

Note

The ability to use the attachments when performing forward engineering and synchronization was added in MySQL Workbench 6.2.0.

Figure 9.6 SQL Scripts Editor



9.1.1.7 The Model Notes Panel

Use the Model Notes panel to write project notes. Any scripts or notes added will be saved with your project.

9.1.1.8 The History Palette

Use the History palette to review the actions that you have taken. Left-clicking an entry opens a pop-up menu with the item, **Copy History Entries to Clipboard**. Choose this item to select a single entry. You can select multiple contiguous entries by pressing the **Shift** key and clicking the entries you wish to copy. Select non-contiguous entries by using the **Control** key.

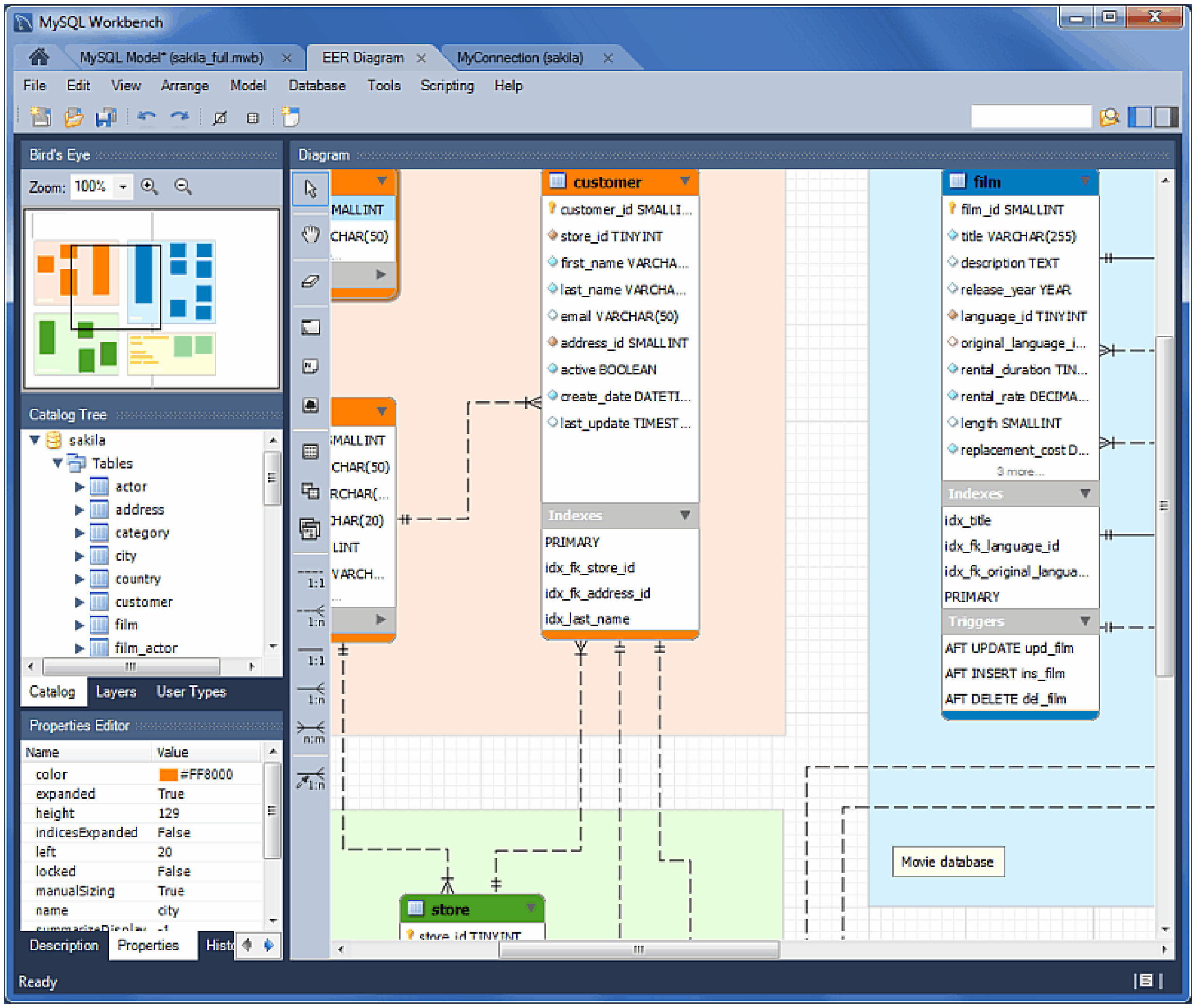
Only actions that alter the MySQL model or change an EER diagram are captured by the `History` palette.

9.1.1.9 The Model Navigator Panel

Docked at the top left of the application is the **Model Navigator**, or **Bird's Eye** panel. This panel provides an overview of the objects placed on an EER diagram canvas and for this reason it is most useful when an EER diagram is active. Any objects that you have placed on the canvas should be visible in the navigator.

The Model Navigator shows the total area of an EER diagram. A black rectangular outline indicates the view port onto the visible area of the canvas. To change the view port of an EER diagram, left-click this black outline and drag it to the desired location. You can zoom in on selected areas of an EER diagram by using the slider tool at the bottom of this window. The dimensions of the view port change as you zoom in and out. If the slider tool has the focus, you can also zoom using the arrow keys.

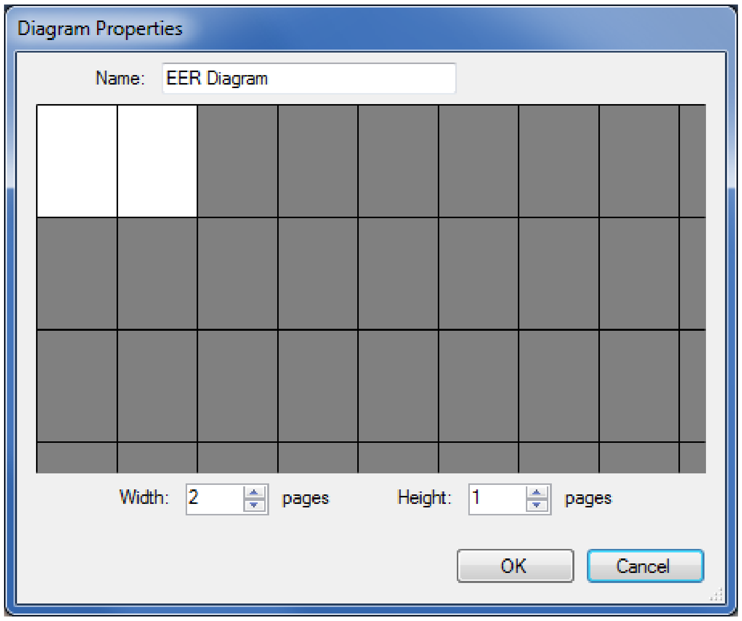
Figure 9.7 The Model Navigator: Example



Name	Value
color	🟧 #FF8000
expanded	True
height	129
indicesExpanded	False
left	20
locked	False
manualSizing	True
name	city

The default size of the `Model Navigator` is two pages. Use the **Model**, **Diagram Properties and Size** page to change the size and diagram name.

Figure 9.8 The Model Navigator Palette



9.1.1.10 The Catalog Tree Palette

The `Catalog Tree` palette shows all the schemata that are present in the `Physical Schemata` section of the `MySQL Model` page. Expand the view of the objects contained in a specific schema by clicking the **>** button to the left of the schema name. This displays the following folder icons:

- Tables
- Views
- Routine Groups

Expand each of these in turn by clicking the **>** button to the left of the folder icon.

The Catalog Tree palette is primarily used to drag and drop objects onto an EER diagram canvas.

You can toggle the sidebar on and off using the **Toggle Sidebar** button, which is located in the top right of the application.

9.1.1.11 The Layers Palette

This palette shows all of the layers and figures on an EER diagram. If a layer or figure is currently selected, an `X` appears beside the name of the object and its properties are displayed in the `Properties` palette. This is useful when determining the selected objects multiple objects were selected using the options under the **Select** menu item. For more information on this topic, see The Edit Menu.

Selecting an object in the `Layers` palette also adjusts the view port to the area of the canvas where the object is located.

Finding Invisible Objects Using the Layers Palette

In some circumstances, you may want to make an object on an EER diagram invisible. Select the object and, in the `Properties` palette, set the `visible` property to `False`.

The `Layer` palette provides an easy way to locate an object, such as a relationship, that has been set to `hidden`. Open the `Layers` palette and select the object by double-clicking it. You can then edit the object and change its visibility setting to `Fully Visible`.

9.1.1.12 The Properties Palette

The `Properties` palette is used to display and edit the properties of objects on an EER diagram. It is especially useful for editing display objects such as layers and notes.

Selecting an object in the EER diagram displays its properties in the `Properties` palette.

All objects except connections have the following properties except as noted:

- `color`: The color accent of the object, displayed as a hexadecimal value. Change the color of the object by changing this value. Only characters that are legal for hexadecimal values may be entered. You can also change the color by clicking the **...** button to open a color changing dialog box.
- `description`: Applicable to layers only. A means of documenting the purpose of a layer.
- `expanded`: This attribute applies to objects such as tables that can be expanded to show columns, indexes, and triggers.
- `height`: The height of the object. Depending upon the object, this property may be read only or read/write.
- `left`: The number of pixels from the object to the left side of the canvas.
- `locked`: Whether the object is locked. The value for this attribute is either `true` or `false`.
- `manualSizing`: Whether the object was manually sized. The value for this attribute is either `true` or `false`.
- `name`: The name of the object.
- `top`: The number of pixels from the object to the top of the canvas.
- `visible`: Whether the object shows up on the canvas. Use `'1'` for true and `'0'` for false. It is currently used only for relationships.
- `width`: The width of the object. Depending upon the object, this property may be read only or read/write.

Tables have the following additional properties:

- `indexesExpanded`: Whether indexes are displayed when a table is placed on the canvas. Use `'1'` for true and `'0'` for false.
- `triggersExpanded`: Whether triggers are displayed when a table is placed on the canvas. Use `'1'` for true and `'0'` for false.

For a discussion of connection properties, see Section 9.1.4.3, “Connection Properties”.

9.1.2 EER Diagram Editor

EER diagrams are created by double-clicking the `Add Diagram` icon. You may create any number of EER diagrams just as you may create any number of physical schemata (databases). Each EER diagram shows as a tab below the toolbar; a specific EER diagram is selected by clicking its tab.

Clicking an EER diagram tab navigates to the canvas used for graphically manipulating database objects. The `Vertical Toolbar` is on the left side of this page.

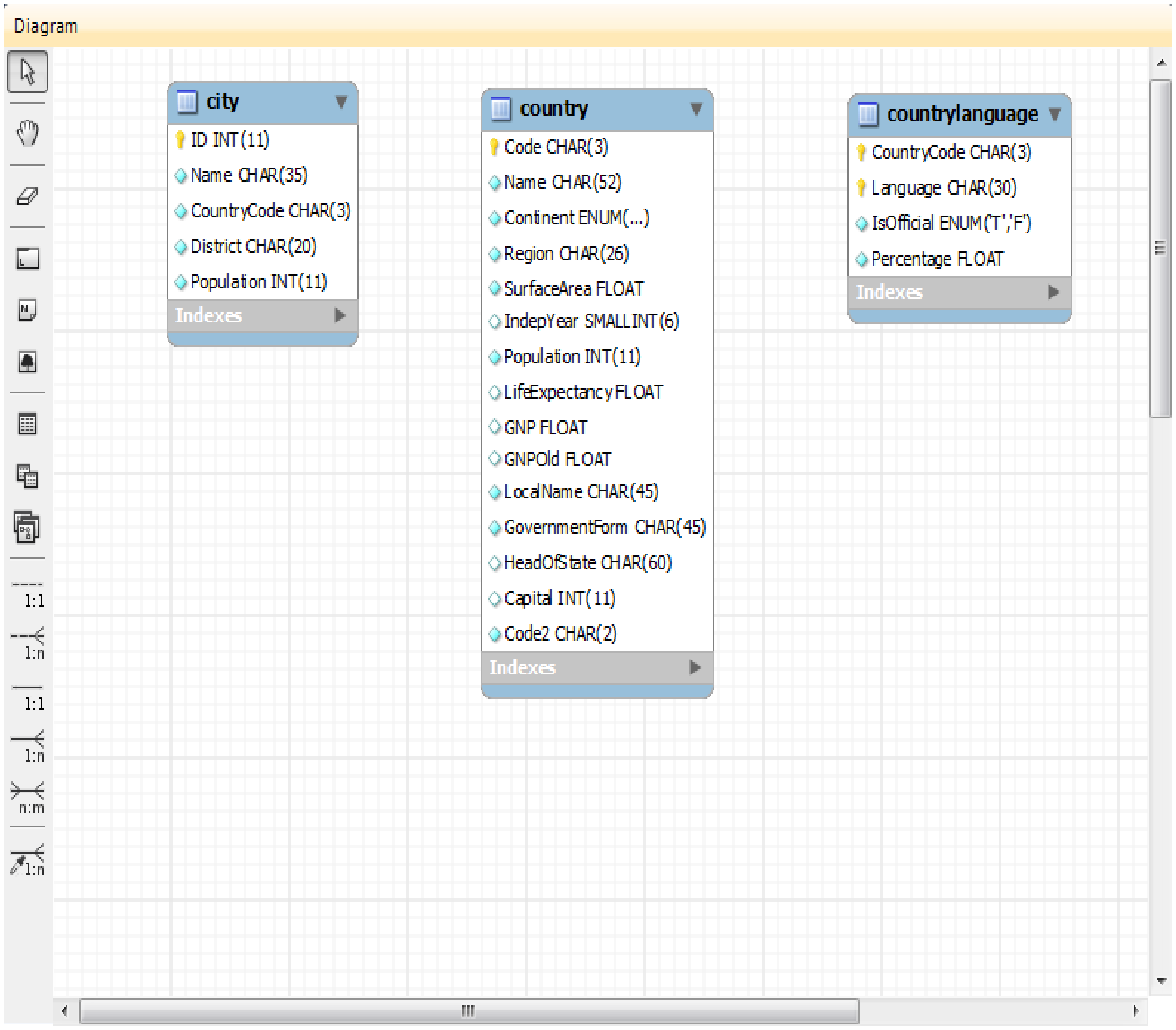
Note

This tool is for creating and editing EER diagrams for a model. To edit an existing database, either reverse engineer the database to create a model, or synchronize your model to a database. For additional information, see Section 9.4.2.2, “Reverse Engineering a Live Database” and Section 9.5, “Schema Synchronization and Comparison”.

9.1.2.1 The Vertical Toolbar

The vertical toolbar shows on the left sidebar when an EER diagram tab is selected. The tools on this toolbar assist in creating EER diagrams.

Figure 9.9 The Vertical Toolbar



Clicking a tool changes the mouse pointer to a pointer that resembles the tool icon, indicating which tool is active. These tools can also be activated from the keyboard by pressing the key associated with the tool. Hover the mouse pointer over a toolbar icon to display a description of the tool and its shortcut key.

A more detailed description of each of these tools follows.

The Standard Mouse Pointer

The standard mouse pointer, located at the top of the vertical toolbar, is the default mouse pointer for your operating system. Use this tool to revert to the standard mouse pointer after using other tools.

To revert to the default pointer from the keyboard, use the **Esc** key.

The Hand Tool

The hand tool is used to move the entire EER diagram. Left-click on this tool and then left-click anywhere on the EER diagram canvas. Moving the mouse while holding down the mouse button changes the view port of the canvas.

To determine your position on the canvas, look at the `Model Navigator` panel on the upper right. If the `Model Navigator` panel is not open, use **View**, **Windows**, **Model Navigator** to open it.

To activate the hand tool from the keyboard, use the **H** key.

You can also change the view port of an EER diagram using the `Model Navigator` panel. See Section 9.1.1.9, "The Model Navigator Panel".

The Eraser Tool

Use the eraser tool to delete objects from the EER Diagram canvas. Change the mouse pointer to the eraser tool, then click the object you wish to delete. Depending upon your settings, the delete dialog box should open, asking you to confirm the type of deletion.

Note

The delete action of the `eraser` tool is controlled by the general option setting for deletion. Before using the eraser tool, be sure that you understand the available options described in Section 3.2.4, "Modeling Preferences".

To activate the eraser tool from the keyboard, use the **D** key.

You can also delete an object by selecting it and pressing **Control+Delete** or by right-clicking it and choosing **Delete** from the pop up menu.

The Layer Tool

The layer tool is the rectangular icon with a capital `L` in the lower left corner. Use the layer tool to organize the objects on an EER Diagram canvas. It is useful for grouping similar objects. For example, you may use it to group all your views.

Click the layer tool and use it to draw a rectangle on the canvas. Change to the standard mouse pointer tool and pick up any objects you would like to place on the newly created layer.

To change the size of a layer, first select it by clicking it. When a layer is selected, small rectangles appear at each corner and in the middle of each side. Adjust the size by dragging any of these rectangles.

You can also make changes to a layer by selecting the layer and changing properties in the `Properties` panel. Using the `Properties` panel is the only way to change the name of a layer.

To activate the layer tool from the keyboard, use the **L** key. For more information about layers, see Section 9.1.7, "Creating Layers".

The Text Tool

The text tool is the square icon with a capital `N` in the top left corner. Use this tool to place text objects on the EER diagram canvas. Click the tool, then click the desired location on the canvas. After a text object has been dropped on the canvas, the mouse pointer reverts to its default.

To add text to a text object, right-click the text object and choose **Edit Note...** or **Edit in New Window...** from the pop-up menu.

You can manipulate the properties of a text object by selecting it and then changing its properties in the `Properties` panel.

To activate the text tool from the keyboard, use the **N** key. For more information about text objects, see Section 9.1.9, "Creating Text Objects".

The Image Tool

Use the image tool to place an image on the canvas. When this tool is selected and you click the canvas, a dialog box opens enabling you to select the desired graphic file.

To activate the image tool from the keyboard, use the **I** key. For more information about images, see Section 9.1.10, "Creating Images".

The Table Tool

Use this tool to create a table on the EER Diagram canvas.

Clicking the canvas creates a table. To edit the table with MySQL Table Editor, right-click it and choose **Edit Table...** or **Edit in New Window...** from the pop-up menu. You can also double-click the table to load it into the table editor.

To activate the table tool from the keyboard, use the **T** key.

For more information about creating and editing tables, see Section 8.1.11, "The MySQL Table Editor".

The View Tool

Use this tool to create a view on an EER Diagram canvas. When the table tool is activated, a schema list appears on the toolbar below the main menu, enabling you to associate the new view with a specific schema. You can also select a color for the object by choosing from the color list to the right of the schema list.

After selecting this tool, clicking the canvas creates a new view. To edit this view, right-click it and choose **Edit View...** or **Edit in New Window...** from the pop-up menu.

To activate the view tool from the keyboard, use the **V** key.

For more information about creating and editing views, see Section 9.1.5, "Creating Views".

The Routine Group Tool

Use this tool to create a routine group on the EER Diagram canvas. When this tool is activated, a schema list appears on the toolbar below the main menu, enabling you to associate the routine group with a specific schema. You can also select a color for the routine group by choosing from the color list to the right of the schema list.

After selecting this tool, clicking the canvas creates a new group. To edit this view, right-click it and choose **Edit Routine Group...** or **Edit in New Window...** from the pop-up menu.

To activate the routine group tool from the keyboard, use the **G** key.

For more information about creating and editing routine groups, see Section 9.1.6.2, “Routine Groups”.

The Relationship Tools

The five relationship tools are used to represent the following relationships:

- One-to-many non-identifying relationships
- One-to-one non-identifying relationships
- One-to-many identifying relationships
- One-to-one identifying relationships
- Many-to-many identifying relationships

These tools appear at the bottom of the vertical tool bar. Hover the mouse pointer over each tool to see a text hint that describes its function.

For more information about relationships, see Section 9.1.4, “Creating Foreign Key Relationships”.

9.1.3 Creating Tables

9.1.3.1 Adding Tables to the Physical Schemata

Double-clicking the `Add table` icon in the `Physical Schemata` section of the `MySQL Model` page adds a table with the default name of `table1`. If a table with this name already exists, the new table is named `table2`.

Adding a new table automatically opens the table editor docked at the bottom of the application. For information about using the table editor, see Section 8.1.11, “The MySQL Table Editor”.

Right-clicking a table opens a context menu with the following items:

- **Cut '*table_name*'**: Cut a table to optionally paste it into another schema.
- **Copy '*table_name*'**: Copy a table to optionally paste it into another schema.
- **Paste '*table_name*'**: Paste a cut or copied table. The Paste option is also accessible from the main **Edit** menu.
- **Edit Table...**: Changes the docked table editor to the selected table.
- **Edit in New Tab...**: Opens the table in a new table editor tab.
- **Copy SQL to Clipboard**: Copies a `CREATE TABLE` statement for the table.
- **Copy Column Names to Clipboard**: Copies a comma-separated list of column names.
- **Copy Insert to Clipboard**: Copies `INSERT` statements based on the model's inserts. Nothing is copied to the clipboard if the table has no inserts defined.
- **Copy Insert Template to Clipboard**: Copies a generic `INSERT` statement that is based on the model.
- **Delete '*table_name*'**: Remove a table from the database.

Warning

This immediately deletes the table without a confirmation dialog box.

If the table editor is not open, the **Edit Table...** item opens it. If it is already open, the selected table replaces the previous one. **Edit in New Tab...** opens an additional table editor tab.

Any tables added to the `Physical Schemata` section also show up in the `Catalog` palette on the right side of the application. They may be added to an EER Diagram by dragging and dropping them from this palette.

9.1.3.2 Adding Tables to an EER Diagram

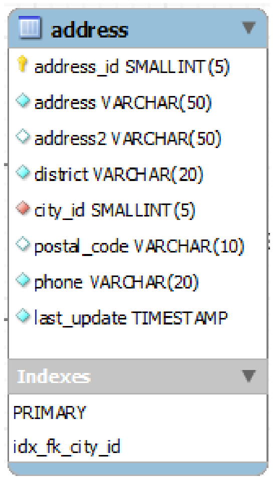
Tables can also be added to an EER Diagram using the `table` tool on the vertical toolbar. Make sure that the **EER Diagram** tab is selected, then right-click the table icon on the vertical toolbar. The table icon is the rectangular tabular icon.

Clicking the mouse on this icon changes the mouse pointer to a table pointer. You can also change the mouse pointer to a table pointer by pressing the **T** key.

Choosing the `table` tool changes the contents of the toolbar that appears immediately below the menu bar. When the `Tables` pointer is active, this toolbar contains a schemata list, an engines list, a collations list, and a color chart list. Use these lists to select the appropriate schema, engine, collation, and color accent for the new table. Make sure that you associate the new table with a database. The engine and collation of a table can be changed using the table editor. The color of your table can be changed using the `Properties` palette. The `Default Engine` and `Default Collation` values refer to the database defaults.

Create a table by clicking anywhere on the EER Diagram canvas. This creates a new table with the default name `table1`. To revert to the default mouse pointer, click the arrow icon at the top of the vertical toolbar.

Figure 9.10 A Table on an EER Diagram


address
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
PRIMARY
idx_fk_city_id

As shown in the preceding diagram, the primary key is indicated by a key icon and indexed fields are indicated by a different colored diamond icon. Click the arrow to the right of the table name to toggle the display of the fields. Toggle the display of indexes and triggers in the same way.

Right-clicking a table opens a pop-up menu with the following items:

- **Cut '*`table_name`*'**

- **Copy '*`table_name`*'**
- **Paste**
- **Edit '*`table_name`*'**
- **Edit '*`table_name`*' in New Tab...**
- **Copy SQL to Clipboard**
- **Copy Column Names to Clipboard**
- **Copy Inserts to Clipboard**
- **Copy Insert Template to Clipboard**
- **Delete '*`table_name`*'**
- **Remove Figure '*`table_name`*'**

With the exception of the deletion item, these menu items function as described in Section 9.1.3.1, "Adding Tables to the Physical Schemata". The behavior of the delete option is determined by your MySQL Workbench options settings. For more information, see Section 3.2.4, "Modeling Preferences".

9.1.4 Creating Foreign Key Relationships

Foreign key constraints are supported for the `InnoDB` storage engine only. For other storage engines, the foreign key syntax is correctly parsed but not implemented. For more information, see Foreign Key Differences.

Using MySQL Workbench you may add a foreign key from within the table editor or by using the relationship tools on the vertical toolbar of an EER Diagram. This section deals with adding a foreign key using the foreign key tools. To add a foreign key using the table editor, see Section 8.1.11.4, "The Foreign Keys Tab".

The graphical tools for adding foreign keys are most effective when you are building tables from the ground up. If you have imported a database using an SQL script and need not add columns to your tables, you may find it more effective to define foreign keys using the table editor.

9.1.4.1 Adding Foreign Key Relationships Using an EER Diagram

The vertical toolbar on the left side of an EER Diagram has six foreign key tools:

- `one-to-one non-identifying relationship`
- `one-to-many non-identifying relationship`
- `one-to-one identifying relationship`
- `one-to-many identifying relationship`
- `many-to-many identifying relationship`
- `Place a Relationship Using Existing Columns`

Differences include:

- An **identifying relationship**: identified by a solid line between tables

An identifying relationship is one where the child table cannot be uniquely identified without its parent. Typically this occurs where an intermediary table is created to resolve a many-to-many relationship. In such cases, the primary key is usually a composite key made up of the primary keys from the two original tables.

- A **non-identifying relationship**: identified by a broken (dashed) line between tables

Create or drag and drop the tables that you wish to connect. Ensure that there is a primary key in the table that will be on the "one" side of the relationship. Click on the appropriate tool for the type of relationship you wish to create. If you are creating a one-to-many relationship, first click the table that is on the "many" side of the relationship, then on the table containing the referenced key. This creates a column in the table on the many side of the relationship. The default name of this column is *`table_name_key_name`* where the table name and the key name both refer to the table containing the referenced key.

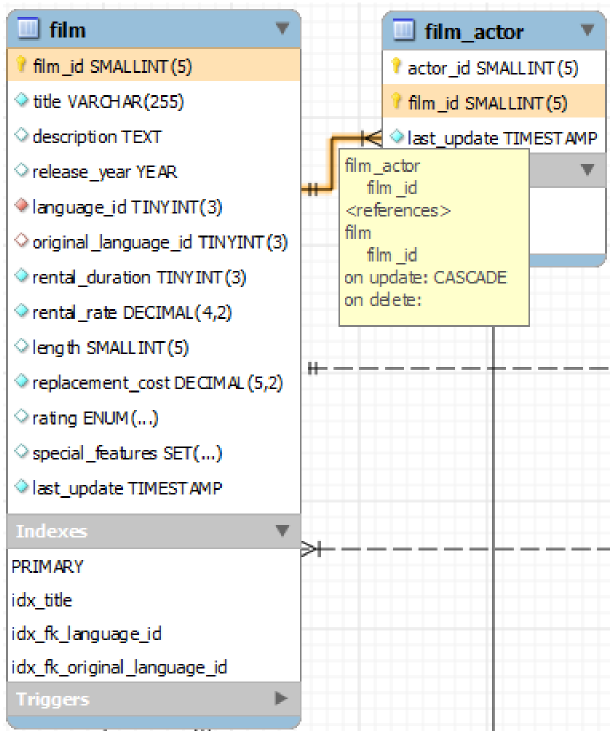
When the many-to-many tool is active, double-clicking a table creates an associative table with a many-to-many relationship. For this tool to function there must be a primary key defined in the initial table.

Use the **Model** menu, **Menu Options** menu item to set a project-specific default name for the foreign key column (see The Relationship Notation Submenu). To change the global default, see Section 3.2.4, "Modeling Preferences".

To edit the properties of a foreign key, double-click anywhere on the connection line that joins the two tables. This opens the relationship editor.

Mousing over a relationship connector highlights the connector and the related keys as shown in the following figure. The `film` and the `film_actor` tables are related on the `film_id` field and these fields are highlighted in both tables. Since the `film_id` field is part of the primary key in the `film_actor` table, a solid line is used for the connector between the two tables. After mousing over a relationship for a second, a yellow box is displayed that provides additional information.

Figure 9.11 The Relationship Connector



```
film_actor
    film_id
<references>
film
    film_id
on update: CASCADE
on delete:
```

If the placement of a connection's caption is not suitable, you can change its position by dragging it to a different location. If you have set a secondary caption, its position can also be changed. For more information about secondary captions, see Section 9.1.4.3, “Connection Properties”. Where the notation style permits, `Classic` for example, the cardinality indicators can also be repositioned.

The relationship notation style in Figure 9.11, “The Relationship Connector” is the default, crow's foot. You can change this if you are using a commercial version of MySQL Workbench. For more information, see The Relationship Notation Submenu.

You can select multiple connections by holding down the **Control** key as you click a connection. This can be useful for highlighting specific relationships on an EER diagram.

9.1.4.2 The Relationship Editor

Double-clicking a relationship on the EER diagram canvas opens the relationship editor. This has two tabs: **Relationship**, and **Foreign Key**.

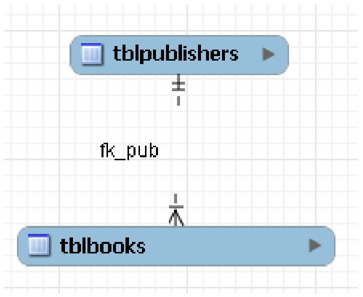
The Relationship Tab

In the **Relationship** tab, you can set the caption of a relationship using the **Caption** field. This name displays on the canvas and is also the name used for the constraint itself. The default value for this name is *`fk_source_table_destination_table`*. Use the **Model** menu, **Menu Options** menu item to set a project-specific default name for foreign keys. To change the global default, see Section 3.2.4, “Modeling Preferences”.

You can also add a secondary caption and a caption to a relationship.

The **Visibility Settings** section is used to determine how the relationship is displayed on the EER Diagram canvas. `Fully Visible` is the default but you can also choose to hide relationship lines or to use split lines. The split line style is pictured in the following figure.

Figure 9.12 The Split Connector



> **Note**
>
> A broken line connector indicates a non-identifying relationship. The split line style can be used with either an identifying relationship or a non-identifying relationship. It is used for display purposes only and does not indicate anything about the nature of a relationship.

To set the notation of a relationship use the **Model** menu, **Relationship Notation** menu item. For more information, see The Relationship Notation Submenu.

The Foreign Key Tab

The **Foreign Key** tab contains several sections: **Referencing Table**, **Cardinality** and **Referenced Table**.

The **Mandatory** check boxes are used to select whether the referencing table and the referenced table are mandatory. By default, both of these constraints are `true` (indicated by the check boxes being checked).

The **Cardinality** section has a set of radio buttons that enable you to choose whether the relationship is one-to-one or one-to-many. There is also a check box that enables you to specify whether the relationship is an identifying relationship.

9.1.4.3 Connection Properties

Right-click a connection to select it. When a connection is selected, it is highlighted and its properties are displayed in the properties palette. Connection properties are different from the properties of other objects. The following list describes them:

- `caption`: The name of the connection. By default, the name is the name of the foreign key and the property is centered above the connection line.
- `captionXOffs`: The X offset of the caption.
- `captionYOffs`: The Y offset of the caption.
- `comment`: The comment associated with the relationship.
- `drawSplit`: Whether to show the relationship as a continuous line.
- `endCaptionXOffs`: The X termination point of the caption offset.
- `endCaptionYOffs`: The Y termination point of the caption offset.
- `extraCaption`: A secondary caption. By default, this extra caption is centered beneath the connection line.
- `extraCaptionXOffs`: The X offset of the secondary caption.
- `extraCaptionYOffs`: The Y offset of the secondary caption.
- `mandatory`: Whether the entities are mandatory. For more information, see Section 9.1.4.2, “The Relationship Editor”.
- `many`: False if the relationship is a one-to-one relationship.
- `middleSegmentOffset`: The offset of the middle section of the connector.
- `modelOnly`: Set when the connection will not be propagated to the DDL. It is just a logical connection drawn on a diagram. This is used, for example, when drawing `MyISAM` tables with a visual relationship, but with no foreign keys.
- `name`: The name used to identify the connection on the EER Diagram canvas. Note that this is **not** the name of the foreign key.
- `referredMandatory`: Whether the referred entity is mandatory.
- `startCaptionXOffs`: The start of the X offset of the caption.
- `startCaptionYOffs`: The start of the Y offset of the caption.

In most cases, you can change the properties of a relationship using the relationship editor rather than the `Properties` palette.

If you make a relationship invisible by hiding it using the relationship editor's **Visibility Settings**, and then close the relationship editor, you will no longer be able to select the relationship to bring up its relationship editor. To make the relationship visible again, you must expand the table object relating to the relationship in the **Layers** palette and select the relationship object. To edit the selected object, right-click it, then select **Edit Object**. You can then set the **Visibility Settings** to **Fully Visible**. The relationship will then be visible in the **EER Diagram** window.

9.1.5 Creating Views

You can add views to a database either from the `Physical Schemata` section of the `MySQL Model` page or from the EER Diagram.

9.1.5.1 Adding Views to the Physical Schemata

To add a view, double-clicking the `Add View` icon in the `Physical Schemata` section of the `MySQL Model` page. The default name of the view is `view1`. If a view with this name already exists, the new view is named `view2`.

Adding a new view automatically opens the view editor docked at the bottom of the application. For information about using the view editor, see Section 9.1.5.3, “The View Editor”.

Right-clicking a view opens a pop-up menu with the following items:

- **Cut '*`view_name`*'**

  The '*`view_name`*' is only cut from the EER canvas, and not removed from the schema.

- **Copy '*`view_name`*'**
- **Paste**
- **Edit View...**
- **Edit in New Window...**
- **Copy SQL to Clipboard**
- **Delete '*`view_name`*'**: deletes from both the EER canvas and schema.
- **Remove '*`view_name`*'**: deletes from the EER canvas, but not the schema.

If the table editor is not open, the **Edit View...** item opens it. If it is already open, the selected table replaces the previous one. **Edit in New Window...** opens a new view editor tab.

The cut and copy items are useful for copying views between different schemata. **Copy SQL to Clipboard** copies the `CREATE VIEW` statement to the clipboard.

Warning

Use the **Delete '*`view_name`*'** item to remove a view from the database. There will be **no** confirmation dialog box.

Any views added to the `Physical Schemata` section also show up in the `Catalog` palette on the left side of the application. They may be added to an EER Diagram, when in the EER Diagram tab, by dragging and dropping them from this palette.

9.1.5.2 Adding Views to an EER Diagram

Views can also be added to an EER Diagram using the `View` tool on the vertical toolbar. Make sure that the **EER Diagram** tab is selected, then left-click the view icon on the vertical toolbar. The view icon is the two overlapping rectangles found below the table icon.

Clicking this icon changes the mouse pointer to a view pointer. To change the mouse pointer to a view pointer from the keyboard, use the **V** key.

Choosing the `View` tool changes the contents of the toolbar that appears immediately below the main menu bar. When the `Views` pointer is active, this toolbar contains a schemata list and a color chart list. Use these lists to select the appropriate schema and color accent for the new view. Make sure that you associate the new view with a database. The color of your view can be changed using the `Properties` palette.

Create a view by clicking anywhere on the EER Diagram canvas. This creates a new view with the default name `view1`. To revert to the default mouse pointer, click the arrow icon at the top of the vertical toolbar.

Right-clicking a view opens a pop-up menu. With the exception of the delete item, these menu items function as described in Section 9.1.5.1, “Adding Views to the Physical Schemata”. The behavior of the delete option is determined by your MySQL Workbench options settings. For more information, see Section 3.2.4, “Modeling Preferences”.

9.1.5.3 The View Editor

To invoke the view editor, double-click a view object on the EER Diagram canvas or double-click a view in the `Physical Schemata` section on the `MySQL Model` page. This opens the view editor docked at the bottom of the application. Double-clicking the title bar undocks the editor. Do the same to redock it. Any number of views may be open at the same time. Each additional view appears as a tab at the top of the view editor.

There are three tabs at the bottom of the view editor: **View**, **Comments**, and **Privileges**. Navigate between different tabs using the mouse or from the keyboard by pressing **Control+Alt+Tab**.

The View Tab

Use the **View** tab to perform the following tasks:

- Rename the view using the **Name** text box.
- Enter the SQL to create a view using the **SQL** field.
- Comment a view using the **Comments** text area.

The Comments Tab

This tab enables you to enter comments for a particular view.

The Privileges Tab

The **Privileges** tab of the view editor functions in exactly the same way as the **Privileges** tab of the routine editor. For more information, see The Privileges Tab.

9.1.5.4 Modifying a View Using the Properties Palette

When you select a view on the EER Diagram canvas, its properties are displayed in the `Properties` palette. Most of the properties accessible from the `Properties` palette apply to the appearance of a view on the EER Diagram canvas.

For a list of properties accessible through the `Properties` palette, see Section 9.1.1.12, “The Properties Palette”.

9.1.6 Creating Routines and Routine Groups

You can add Routine Groups to a database either from the **Physical Schemata** section of the **MySQL Model** page or from an EER Diagram. Routines may be added only from the **Physical Schemata** section of the **MySQL Model** page.

To view an existing schema, along with its Routines and Routine Groups, choose **Database**, **Reverse Engineer...** from the main menu. After the schema has been added to the current model, you can see the schema objects on the **Physical Schemata** panel on the **MySQL Model** page. The Routines and Routine Groups are listed there.

MySQL Workbench unifies both stored procedures and stored functions into one logical object called a Routine. **Routine Groups** are used to group related routines. Define Routine with the **Routine Group Editor** to assign specific routines to a group, using a drag and drop interface.

When designing an EER Diagram, you can place the Routine Groups on the canvas by dragging them from the **Catalog Palette**. Placing individual routines on the diagram is not permitted, as it would clutter the canvas.

9.1.6.1 Routines

Adding Routines to the Physical Schemata

To add a routine, double-click the `Add Routine` icon in the `Physical Schemata` section of the `MySQL Model` page. The default name of the routine is `routine1`. If a routine with this name already exists, the new routine is named `routine2`.

Adding a new routine automatically opens the routine editor docked at the bottom of the application. For information about using the routine editor, see The Routine Editor.

Right-clicking a routine opens a pop-up menu with the following items:

- **Rename**
- **Cut '*routine_name*'**
- **Copy '*routine_name*'**
- **Paste**
- **Edit Routine...**
- **Edit in New Window...**
- **Copy SQL to Clipboard**
- **Delete '*routine_name*'**

The **Edit Routine...** item opens the routine editor.

The cut and paste items are useful for copying routines between different schemata.

Note

Deleting the code for a routine from the **Routines** tab of the Routine Group Editor results in removal of the routine object from the model.

Note

To remove a routine from a routine group, use the controls on the **Routine Group** tab of the Routine Group Editor.

The action of the delete option varies depending upon how you have configured MySQL Workbench. For more information, see Section 3.2.4, “Modeling Preferences”.

The Routine Editor

To invoke the routine editor, double-click a routine in the `Physical Schemata` section on the `MySQL Model` page. This opens the routine editor docked at the bottom of the application. Any number of routines may be open at the same time. Each additional routine appears as a tab at the top of the routine editor.

Routine and **Privileges** tabs appear at the bottom of the routine editor. Navigate between different tabs using the mouse or from the keyboard by pressing **Control+Alt+Tab**.

The Routine Tab

Use the **Routine** tab of the routine editor to perform the following tasks:

- Rename the routine using the **Name** field.
- Enter the SQL to create a routine using the **SQL** field.

The Privileges Tab

The **Privileges** tab of the routine editor allows you to assign specific roles and privileges. You may also assign privileges to a role using the role editor. For a discussion of this topic, see Adding Roles.

When this tab is first opened, all roles that have been created are displayed in the list on the right. Move the roles you wish to associate with this table to the **Roles** list on the left. Do this by selecting a role and then clicking the **<** button. Use the **Shift** key to select multiple contiguous roles and the **Control** key to select noncontiguous roles.

To assign privileges to a role, click the role in the **Roles** list. This displays all available privileges in the **Assigned Privileges** list. The privileges that display are:

- `ALL`
- `CREATE`
- `DROP`
- `GRANT OPTION`
- `REFERENCES`
- `ALTER`
- `DELETE`
- `INDEX`
- `INSERT`
- `SELECT`
- `UPDATE`

- `TRIGGER`

You can choose to assign all privileges to a specific role or any other privilege as listed previously. Privileges irrelevant to a specific table, such as the `FILE` privilege, are not shown.

If a role has already been granted privileges on a specific table, those privileges show as already checked in the **Assigned Privileges** list.

9.1.6.2 Routine Groups

Adding Routine Groups to the Physical Schemata

Double-clicking the `Add Routine Group` icon in the `Physical Schemata` section of the `MySQL Model` page adds a routine group with the default name of `routines1`. If a routine group with this name already exists, the new routine group is named `routines2`.

Adding a new routine group automatically opens the routine groups editor docked at the bottom of the application. For information about using the routine groups editor, see The Routine Group Editor.

Right-clicking a routine group opens a pop-up menu with the following items:

- **Rename**
- **Cut '*routine_group_name*'**
- **Copy '*routine_group_name*'**
- **Edit Routine...**
- **Edit in New Window...**
- **Copy SQL to Clipboard**
- **Delete '*routine_group_name*'**

The **Edit Routine Group...** item opens the routine group editor, which is described in The Routine Group Editor.

The cut and paste items are useful for copying routine groups between different schemata.

Deleting a routine group from the `MySQL Model` page removes the group but does not remove any routines contained in that group.

Any routine groups added to the `Physical Schemata` also show up in the `Catalog` palette on the right side of the application. They may be added to an EER diagram by dragging and dropping them from this palette.

Adding Routine Groups to an EER Diagram

To add routine groups to an EER Diagram, use the `Routine Groups` tool on the vertical toolbar. Make sure that the **EER Diagram** tab is selected, then right-click the routine groups icon on the vertical toolbar. The routine groups icon is immediately above the lowest toolbar separator.

Clicking the mouse on this icon changes the mouse pointer to a routine group pointer. You can also change the mouse pointer to a routine pointer by pressing the **G** key.

Choosing the `Routine Group` tool changes the contents of the toolbar that appears immediately below the menu bar. When the `Routine Groups` pointer is active, this toolbar contains a schemata list and a color chart list. Use these lists to select the appropriate schema and color accent for the new routine group.

Make sure that you associate the new routine group with a database. The color of your routine group can be changed later using the `Properties` palette.

Create a routine group by clicking anywhere on the EER Diagram canvas. This creates a new routine group with the default name `routines1`. To revert to the default mouse pointer, click the arrow icon at the top of the vertical toolbar.

Right-clicking a routine group opens a pop-up menu. With the exception of the delete option and rename options, these menu options function as described in Adding Routine Groups to the Physical Schemata. There is no rename option, and the behavior of the delete option is determined by your MySQL Workbench options settings. For more information, see Section 3.2.4, “Modeling Preferences”.

The Routine Group Editor

To invoke the routine group editor, double-click a routine group object on the EER Diagram canvas or double-click a routine group in the `Physical Schemata` section on the `MySQL Model` page. This opens the routine group editor docked at the bottom of the application. Double-clicking the title bar undocks the editor. Do the same to redock it. Any number of routine groups may be open at the same time. Each additional routine group appears as a tab at the top of the routine editor,

Routine group and **Privileges** tabs appear at the bottom of the routine editor. Navigate between different tabs using the mouse or from the keyboard by pressing **Control+Alt+Tab**.

The Routine Groups Tab

Use the **Routine Groups** tab of the routine groups editor to perform the following tasks:

- Rename the routine group using the **Name** field.
- Add routines to the group by dragging and dropping them.
- Add comments to the routine group.

The Privileges Tab

The **Privileges** tab of the routine group editor functions in exactly the same way as the **Privileges** tab of the table editor. For more information, see The Privileges Tab.

Modifying a Routine Group Using the Properties Palette

When you select a routine group on the EER Diagram canvas, its properties are displayed in the `Properties` palette. All of the properties accessible from the `Properties` palette apply to the appearance of a routine group on the EER Diagram canvas.

For a list of properties accessible through the `Properties` palette, see Section 9.1.1.12, “The Properties Palette”.

9.1.7 Creating Layers

You can add layers to a database only from an EER Diagram. Layers are used to help organize objects on the canvas. Typically, related objects are added to the same layer; for example, you may choose to add all your views to one layer.

9.1.7.1 Adding Layers to an EER Diagram

To add layers to an EER Diagram, use the `Layer` tool on the vertical toolbar. Select an **EER Diagram** tab and left-click the layer icon on the vertical toolbar. The layer icon is the rectangle with an `‘L’` in the lower left corner and it is found below the eraser icon.

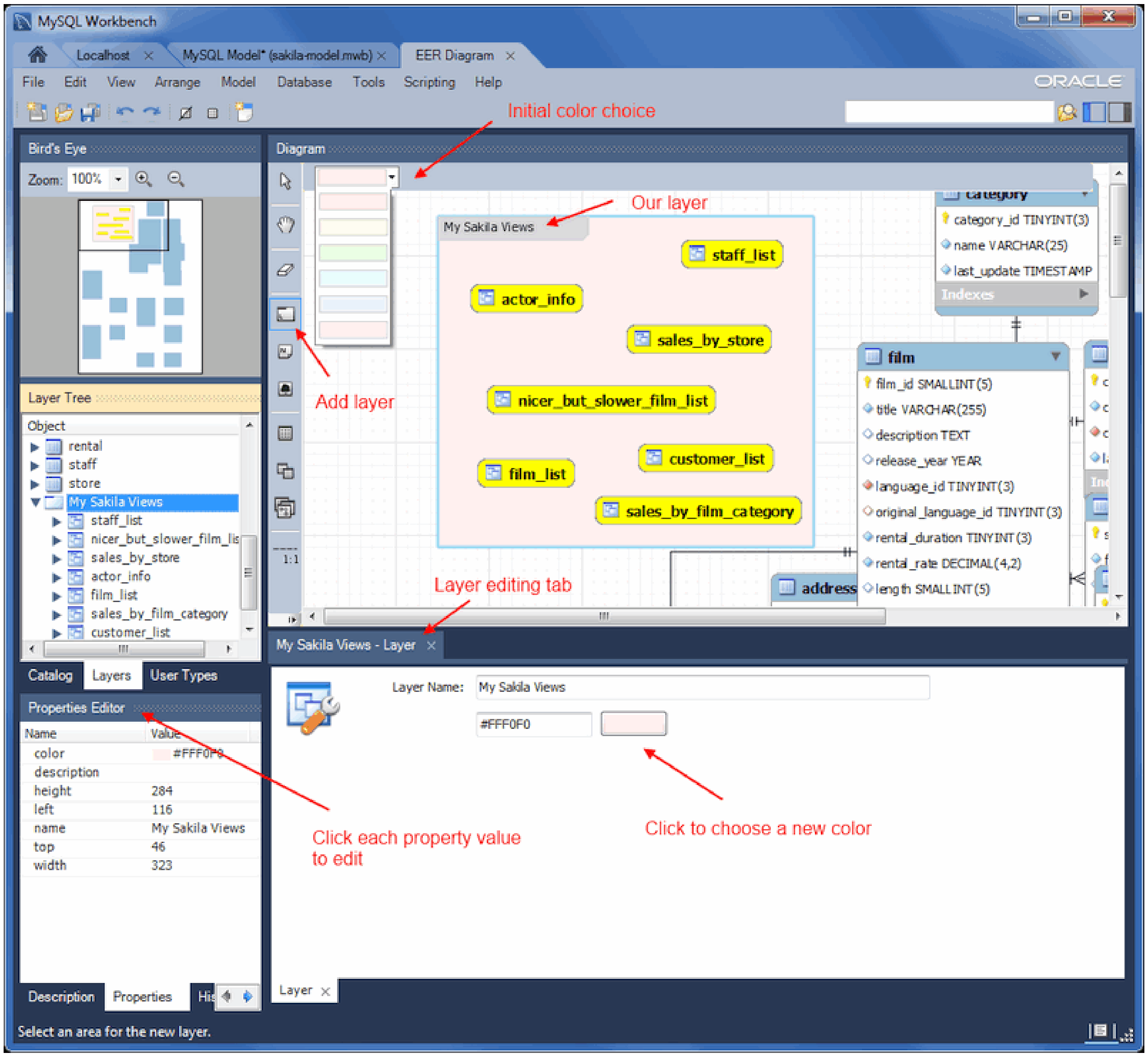
Clicking the mouse on this icon changes the mouse pointer to a layer pointer. You can also change the mouse pointer to a layer pointer by pressing the **L** key.

Choosing the `Layer` tool changes the contents of the toolbar that appears immediately below the menu bar. When the `Layers` pointer is active, this toolbar contains a color chart list. Use this list to select the color accent for the new layer. The color of your layer can be changed later using the `Properties` palette.

Create a layer by clicking anywhere on the EER Diagram canvas and, while holding the left mouse button down, draw a rectangle of a suitable size. This creates a new layer with the default name `layer1`. To revert to the default mouse pointer, click the arrow icon at the top of the vertical toolbar.

The following image shows a layer named "My Sakila Views" with several views:

Figure 9.13 The Layer Object



Layer Name: My Sakila Views
#FFF0F0

Name	Value
color	#FFF0F0
description	
height	284
left	116
name	My Sakila Views
top	46
width	323

To open the layer editor, either double-click the layer or right-click the layer and choose the edit option. The available context-menu options are:

- **Cut '*layer_name*'**
- **Copy '*layer_name*'**
- **Paste '*a_table_name*'**
- **Edit '*layer_name*'**
- **Delete '*layer_name*'**

Note

A layer may also be edited via **Properties Editor** on the left panel, and it offers additional edit options.

The cut and copy items are useful for copying layers between different schemata.

Since layers are not schema objects, no confirmation dialog box opens when you delete a layer regardless of how you have configured MySQL Workbench. Deleting a layer does **not** delete schema objects from the catalog.

Adding Objects to a Layer

To add an object to a layer, drag and drop it directly from the `Catalog` palette onto a layer. If you pick up an object from an EER diagram, you must press **Control** as you drag it onto the layer, otherwise it will not be “locked” inside the layer.

Locking objects to a layer prevents their accidental removal. You cannot remove them by clicking and dragging; to remove an object, you also must press the **Control** key while dragging it.

As a visual cue that the object is being “locked”, the outline of the layer is highlighted as the object is dragged over it.

If you drag a layer over a table object, the table object will automatically be added to the layer. This also works for multiple table objects.

Layers cannot be nested. That is, a layer cannot contain another layer object.

9.1.7.2 Modifying a Layer Using the Properties Palette

Choosing "Edit" allows you to edit the layer name and layer background color, and the "Properties Editor" offers additional edit options.

When you select a layer on the EER Diagram canvas, its properties are displayed in the `Properties` palette. The properties accessible from the `Properties` palette apply to the appearance of a layer on the EER Diagram canvas.

In some circumstances, you may want to make a layer invisible. Select the layer and, in the `Properties` palette, set the `visible` property to `False`. To locate an invisible object, open the `Layers` palette and select the object by double-clicking it. After an object is selected, you can reset the `visible` property from the `Properties` palette.

For a list of properties accessible through the `Properties` palette, see Section 9.1.1.12, “The Properties Palette”. In addition to the properties listed there, a layer also has a `description` property. Use this property to document the purpose of the layer.

9.1.8 Creating Notes

You can add notes to a database only from the `Model Notes` section of the `MySQL Model` page. Notes are typically used to help document the design process.

9.1.8.1 Adding Notes

Double-clicking the `Add Note` icon in the `Model Notes` section of the `MySQL Model` page adds a note with the default name of `note1`. If a note with this name already exists, the new note is named `note2`.

Adding a new note automatically opens the note editor docked at the bottom of the application. For information about using the note editor, see Section 9.1.8.2, "The Note Editor".

Right-clicking a note opens a pop-up menu with the following items:

- **Rename**
- **Cut '*`note_name`*'**
- **Copy '*`note_name`*'**
- **Delete '*`note_name`*'**

The **Edit Note...** item opens the note editor. For information about using the note editor, see Section 9.1.8.2, "The Note Editor".

The cut and copy items are useful for copying notes between different schemata.

Notes can be added only on the `MySQL Model` page.

9.1.8.2 The Note Editor

To invoke the note editor, double-click a note object in the `Model Note` section on the `MySQL Model` page. This opens the note editor docked at the bottom of the application. Double-clicking the note tab undocks the editor. Double-click the title bar to redock it. Any number of notes may be open at the same time. Each additional note appears as a tab at the top of the note editor.

Use the editor to change the name of a note or its contents.

9.1.9 Creating Text Objects

Text objects are applicable only to an EER diagram. They can be used for documentation purposes; for example, to explain a grouping of schema objects. They are also useful for creating titles for an EER diagram should you decide to export a diagram as a PDF or PNG file.

9.1.9.1 Adding Text Objects to an EER Diagram

To add text objects to an EER Diagram, use the `Text Object` tool on the vertical toolbar. Make sure that the **EER Diagram** tab is selected, then right-click the text object icon on the vertical toolbar. The text object icon is the rectangular icon found below the label icon.

Clicking the mouse on this icon changes the mouse pointer to a text object pointer. You can also change the mouse pointer to a text object pointer by pressing the **N** key.

Choosing the `Text Object` tool changes the contents of the toolbar that appears immediately below the menu bar. When the `Text Object` pointer is active, this toolbar contains a color chart list. Use this list to select the color accent for the new text object. The color of your text object can be changed later using the `Properties` palette.

Create a text object by clicking anywhere on the EER Diagram canvas. This creates a new text object with the default name `text1`. To revert to the default mouse pointer, click the arrow icon at the top of the vertical toolbar.

Right-clicking a text object opens a pop-up menu. These menu options are identical to the options for other objects. However, since a text object is not a database object, there is no confirmation dialog box when you delete a text object.

9.1.9.2 The Text Object Editor

To invoke the text object editor, double-click a text object on the EER Diagram canvas. This opens the editor docked at the bottom of the application. Double-clicking the text object table undocks the editor. Double-click the title bar to redock it. Any number of text objects may be open at the same time. Each additional text objects appears as a tab at the top of the text editor.

Use the editor to change the name of a text object or its contents.

Modifying a Text Object Using the Properties Palette

When you select a text object on the EER Diagram canvas, its properties are displayed in the `Properties` palette. Most of the properties accessible from the `Properties` palette apply to the appearance of a view on the EER Diagram canvas.

For a list of properties accessible through the `Properties` palette, see Section 9.1.1.12, “The Properties Palette”.

There is no property in the `Properties` palette for changing the font used by a text object. To do so, choose the **Appearance** tab of the Workbench Preferences dialog. For more information, see Preferences: Modeling: Appearance.

9.1.10 Creating Images

Images exist only on the EER Diagram canvas; you can add them only from the EER Diagram window.

9.1.10.1 Adding Images to an EER Diagram

To add images to an EER Diagram, use the `Image` tool on the vertical toolbar. Make sure that the **EER Diagram** tab is selected, then right-click the image icon on the vertical toolbar. The image icon is the icon just above the table icon.

Clicking the mouse on this icon changes the mouse pointer to an image pointer. You can also change the mouse pointer to an image pointer by pressing the **I** key.

Create an image by clicking anywhere on the EER Diagram canvas. This opens a file open dialog box. Select the desired image, then close the dialog box to create an image on the canvas. To revert to the default mouse pointer, click the arrow icon at the top of the vertical toolbar.

Right-clicking this object opens a pop-up menu with the following items:

- **Cut** ***`'Image'`***
- **Copy** ***`'Image'`***

- **Edit Image...**
- **Edit in New Window...**
- **Delete** `'Image'`

These menu items function in exactly the same way as they do for other objects on an EER diagram. However, images are not database objects so there is no confirmation dialog box when they are deleted.

9.1.10.2 The Image Editor

To invoke the image editor, double-click an image object on an EER Diagram canvas. This opens the image editor docked at the bottom of the application. Double-clicking the image editor tab undocks the editor. Double-click the title bar to redock it. Any number of images may be open at the same time. Each additional image appears as a tab at the top of the image editor.

The Image Tab

Use the **Image** tab of the image editor to perform the following tasks:

- Rename the image using the **Name** text box.
- Browse for an image using the **Browse** button.

Modifying a Image using the Properties Palette

When you select an image on the EER Diagram canvas, its properties are displayed in the **Properties** palette. Most of the properties accessible from the **Properties** palette apply to the appearance of an image on the EER Diagram canvas.

For a list of properties accessible through the **Properties** palette, see Section 9.1.1.12, “The Properties Palette”.

9.2 Additional Modeling Tools

Additional modeling design tools and features.

9.2.1 Printing Diagrams

The printing options used to create printouts of your EER Diagrams are found under the **File** menu. To create *documentation* of your models, see The DBDoc Model Reporting Dialog Window (Commercial Version).

9.2.1.1 Printing Options

The printing menu items not enabled unless an EER Diagram is active. These items are available:

- **Page Setup...**

  Enables you to choose the paper size, orientation, and margins.

- **Print**

  Sends your EER Diagram directly to the printer. This option generates a preview before printing. From the preview you can adjust the scale of the view and also choose a multi-page view. Clicking the printer icon at the top left of this window, prints the currently selected EER Diagram. Close the print preview window if you need to adjust the placement of objects on the EER Diagram canvas.

- **Print to PDF...**

  Creates a PDF file of your EER Diagram.

- **Print to PS...**

  Creates a PostScript file of your EER Diagram.

9.2.2 DBDoc Model Reporting

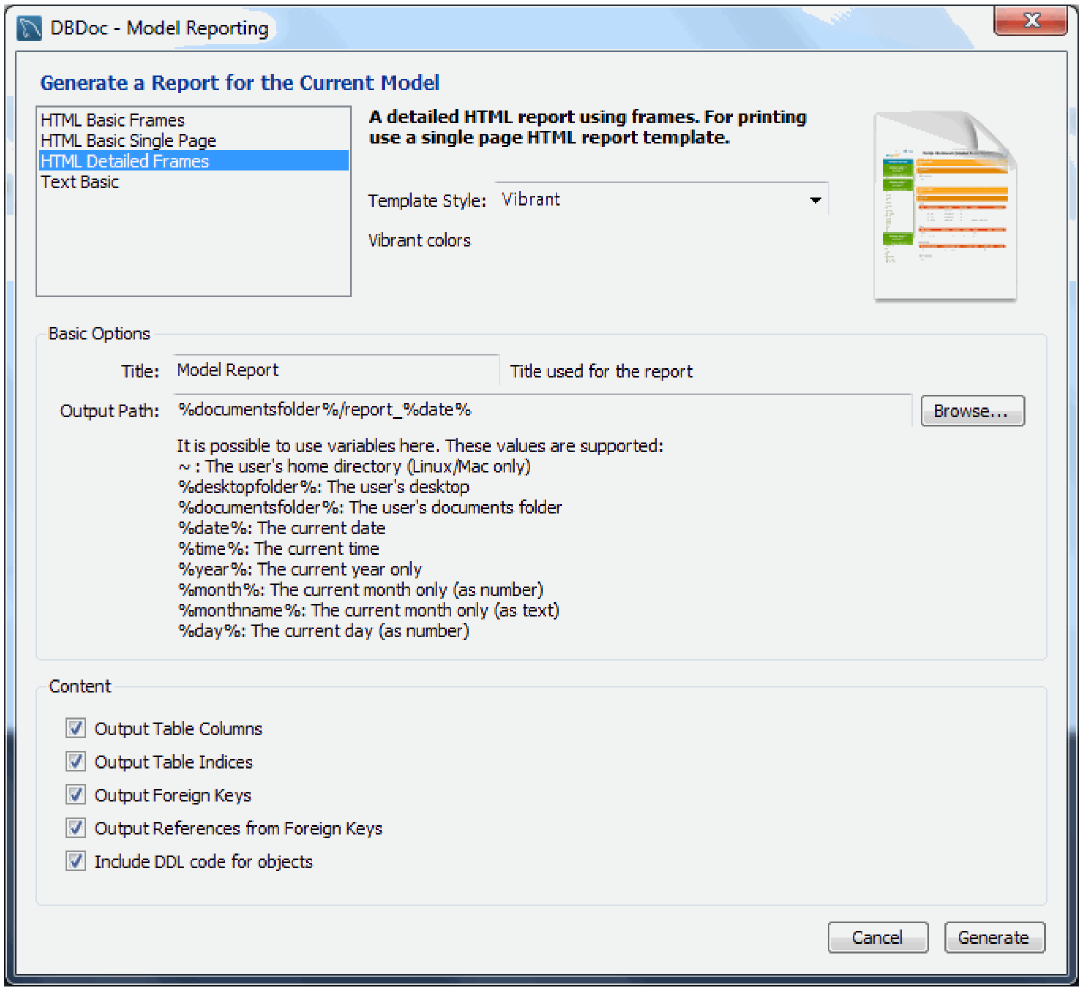
This dialog window is found by navigating to the **Model** menu and choosing the **DBDoc - Model Reporting...** item.

Note

This functionality is only available in the MySQL Workbench commercial editions.

Use this dialog window to set the options for creating documentation of your database models.

Figure 9.14 The DBDoc Model Reporting Main Wizard



Generate a Report for the Current Model

HTML Basic Frames
HTML Basic Single Page
HTML Detailed Frames
Text Basic

A detailed HTML report using frames. For printing use a single page HTML report template.

Template Style: Vibrant

Vibrant colors

Basic Options

Title: Model Report — Title used for the report

Output Path: %documentsfolder%/report_%date%

It is possible to use variables here. These values are supported:
~ : The user's home directory (Linux/Mac only)
%desktopfolder%: The user's desktop
%documentsfolder%: The user's documents folder
%date%: The current date
%time%: The current time
%year%: The current year only
%month%: The current month only (as number)
%monthname%: The current month only (as text)
%day%: The current day (as number)

Content

- [✓] Output Table Columns
- [✓] Output Table Indices
- [✓] Output Foreign Keys
- [✓] Output References from Foreign Keys
- [✓] Include DDL code for objects

You can choose from four available templates:

- `HTML Basic Frames`: Model documentation in HTML format that makes use of frames
- `HTML Basic Single Page`: Single Page HTML documentation, not using frames
- `HTML Detailed Frames`: Detailed HTML documentation, using frames
- `Text Basic`: Text file documentation

When you click a template, a preview image displays on the right side of the page. For the `HTML Basic Frames` template, you can select either the `Colorful` or the `Restrained Colors` option from the **Style** list. The `HTML Basic Single Page` template offers only the `Colorful` style. The `HTML Detailed Frames` template offers the `Vibrant` style, and also the more subdued `Coated` style. The `Text Basic` template offers only the `Fixed Size Font` style.

From the **Base Options** frame choose the report title and the output directory for the report files.

The following variables may be used to configure the output path:

- **~**: The user's home directory. Available on Linux and OS X versions only.
- **%desktopfolder%**: The user's desktop.
- **%documentsfolder%**: The user's Documents folders. The following table shows typical values for common platforms:

Platform	**Typical Default Documents Folder**
Windows	`C:\Documents and Settings \`*`user_name`*`\My Documents`
Linux	`~/Documents`
OS X	`Users/`*`user_name`*`/Documents`

- **%date%**: The date in the format YYYY-MM-DD.
- **%time%**: The time in the format HHMM.
- **%year%**: The year in the format YYYY.
- **%month%**: The month in the format MM. January is 01 and December is 12.
- **%monthname%**: The name of the month, rather than the number.
- **%day%**: The day number in the format DD. For example, the 12th would be 12.

Content options can also be set:

- `Render Table Columns`: Display all the columns.
- `Render Table Indices`: Display all the indexes.
- `Render Foreign Keys`: Display all the foreign keys.
- `List Foreign Keys that refer to that table`: Display the tables that foreign keys reference.
- `Include DDL code for objects`: Generates DDL code.

Clicking the **Generate** button creates the directory defined in the **Output directory** text box. If you chose to create `HTML Basic Frames`, you will find the following files in this directory:

- `basic.css`: The style sheet for the `overview.html` page.
- `index.html`: The main page.
- `overview.html`: The model overview, the navigation links shown in the sidebar.
- `restrained.css`: The CSS file used if the `Restrained Colors` style option was chosen.
- `table_details.html`: The main frame of the model report.

Choosing the `HTML Basic Single Page` option creates a style sheet and an `index.html` file.

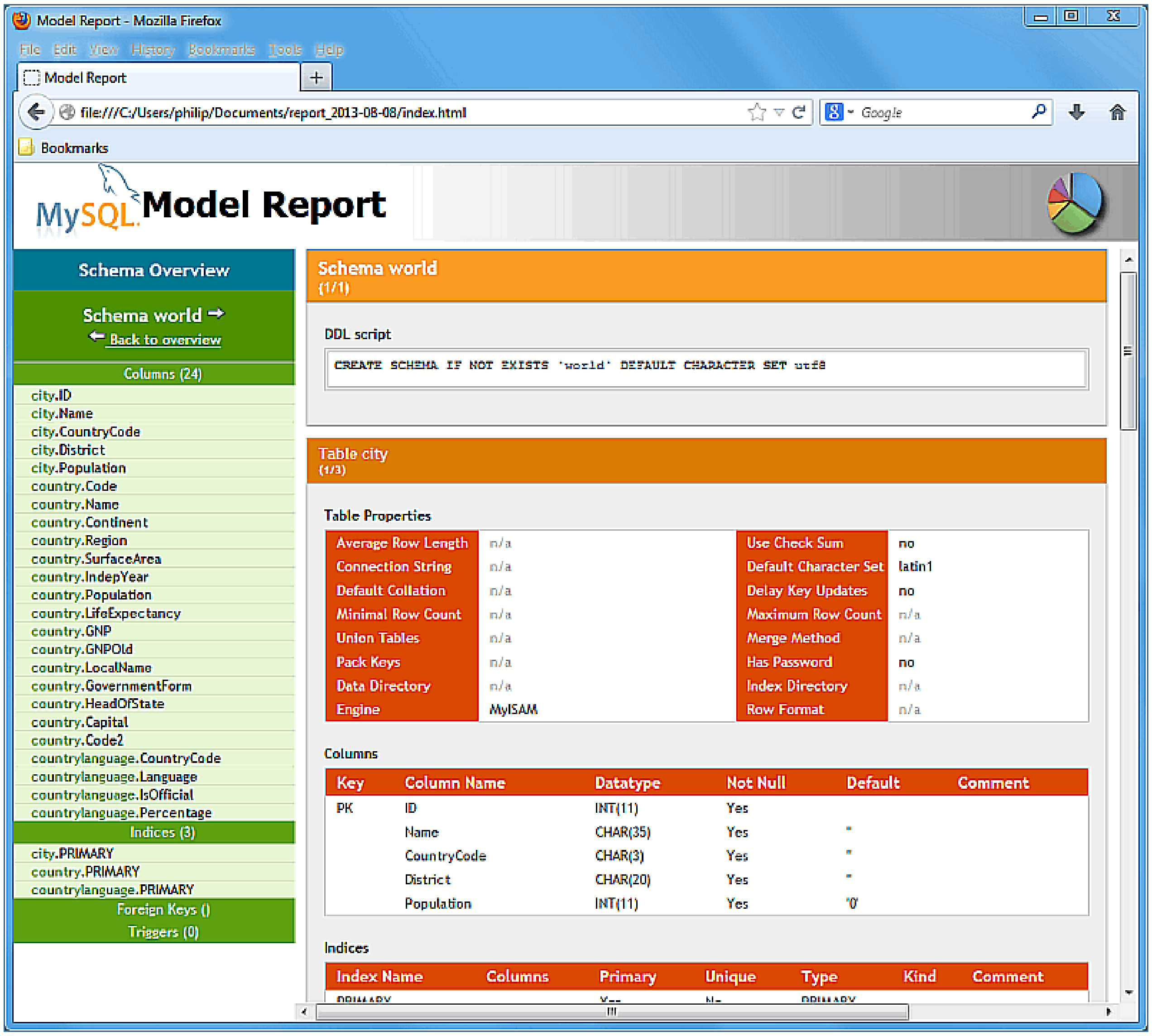
Choosing the `HTML Detailed Frames` option creates the following files:

- `basic.css`: The style sheet for the `overview.html` page. This is used if the `vibrant` style is chosen.
- `coated.css`: The CSS file used if the `Coated` style option was chosen.
- `index.html`: The main page.
- `overview.html`: Overview information for the report such as report title, project name and author.
- `overview_list.html`: A summary of schema in the model along with a list of tables contained in each schema.
- `routine_details.html`: List of all routines for the schema.
- `table_details.html`: The main report details.
- `table_details_list.html`: A Schema overview along with details of columns, indexes and foreign keys for each schema.
- `table_element_details.html`: The details for every element of the table.
- `top.html`: The top frame of the report.
- `view_details.html`: List of all columns and indexes for the schema.

Choosing the `Text Basic` option creates a directory containing one text file.

You can click `index.html` to view a report. The following screenshot shows the `HTML Detailed Frames` report being displayed:

Figure 9.15 The DBDoc Model Report


Model Report - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Model Report
file:///C:/Users/philip/Documents/report_2013-08-08/index.html
Google
Bookmarks
MySQL Model Report
Schema Overview
Schema world
Back to overview

Columns (24)

city.ID
city.Name
city.CountryCode
city.District
city.Population
country.Code
country.Name
country.Continent
country.Region
country.SurfaceArea
country.IndepYear
country.Population
country.LifeExpectancy
country.GNP
country.GNPOld
country.LocalName
country.GovernmentForm
country.HeadOfState
country.Capital
country.Code2
countrylanguage.CountryCode
countrylanguage.Language
countrylanguage.IsOfficial
countrylanguage.Percentage

Indices (3)

city.PRIMARY
country.PRIMARY
countrylanguage.PRIMARY

Foreign Keys ()
Triggers (0)

Schema world
(1/1)

DDL script

```
CREATE SCHEMA IF NOT EXISTS `world` DEFAULT CHARACTER SET utf8
```

Table city
(1/3)

Table Properties

Average Row Length	n/a	Use Check Sum	no
Connection String	n/a	Default Character Set	latin1
Default Collation	n/a	Delay Key Updates	no
Minimal Row Count	n/a	Maximum Row Count	n/a
Union Tables	n/a	Merge Method	n/a
Pack Keys	n/a	Has Password	no
Data Directory	n/a	Index Directory	n/a
Engine	MyISAM	Row Format	n/a

Columns

Key	Column Name	Datatype	Not Null	Default	Comment
PK	ID	INT(11)	Yes		
	Name	CHAR(35)	Yes	''	
	CountryCode	CHAR(3)	Yes	''	
	District	CHAR(20)	Yes	''	
	Population	INT(11)	Yes	'0'	

Indices

Index Name	Columns	Primary	Unique	Type	Kind	Comment
PRIMARY		Yes	No	PRIMARY		

If you wish to create custom templates please refer to Section 9.7, “Customizing DBDoc Model Reporting Templates”.

9.2.3 Schema Validation Plugins

MySQL Workbench provides validation modules so that you can test your models before implementing them.

Note

This functionality is only available in the MySQL Workbench commercial editions.

The validation plugins are accessed from the **Model** menu. One plugin performs general validation for any Relational Database Management System (RDMS) and the other is MySQL specific. Beneath these menu items are a number of specific validation tests. Running any one of these tests opens an output window docked at the bottom of the application. Warning messages are displayed on the left side of this window and the tests performed are displayed on the right.

The following sections outline the tasks performed by the validation modules.

9.2.3.1 General Validation

The following list names the general validation types and gives examples of specific violations:

- **Empty Content Validation**
  - A table with no columns
  - A routine or view with no SQL code defined
  - A routine group containing no routines
  - A table, view, or routine not referenced by at least one role
  - A user with no privileges
  - Objects such as tables that do not appear on at least one EER Diagram
- **Table Efficiency Validation**
  - A table with no primary key
  - A primary key that does not use an integer-based data type
  - A foreign key that refers to a column with a different data type
- **Duplicated Identifiers Validation**
  - Duplicate object names
  - Duplicate role or user names
  - Duplicate index or routine names
- **Consistency Validation**
  - Use of the same column with columns of differing data types
- **Logic Validation**
  - A foreign key that refers to a column other than the primary key in the source table
  - Any object that is object is either read only or write only by role definition
  - Placeholder objects left over from reverse engineering

9.2.3.2 MySQL-Specific Validation

The following list names the MySQL-specific validation types and gives examples of specific violations:

- **Integrity Violation**
  - An object name longer than the maximum permitted

    - A foreign key defined for an engine type that does not support foreign keys (not yet implemented)
    - A view or routine that references a nonexistent table (not yet implemented)
    - A default value that does not match a column's data type
    - An invalid partitioning scheme
- **Syntax Violation**
    - A routine, trigger, or view with incorrect SQL syntax
    - A reserved keyword used as an identifier
    - Use of an invalid character

9.3 Modeling Tutorials

This chapter contains three short tutorials intended to familiarize you with the basics of MySQL Workbench. These tutorials show how MySQL Workbench can be used both to design and to document databases.

Creating a database from scratch is the focus of Section 9.3.4, "Using the Default Schema" and exploring the graphic design capabilities of MySQL Workbench is touched upon in Section 9.3.2, "Basic Modeling". Both these tutorials show the database design capabilities of MySQL Workbench.

Importing an SQL data definition script is probably the quickest way to familiarize yourself with MySQL Workbench—this tutorial makes use of the `sakila` database and emphasizes the use of MySQL Workbench as a documentation tool. Examples taken from the `sakila` database are used throughout the documentation, so this installation procedure is recommended.

9.3.1 Creating a Model

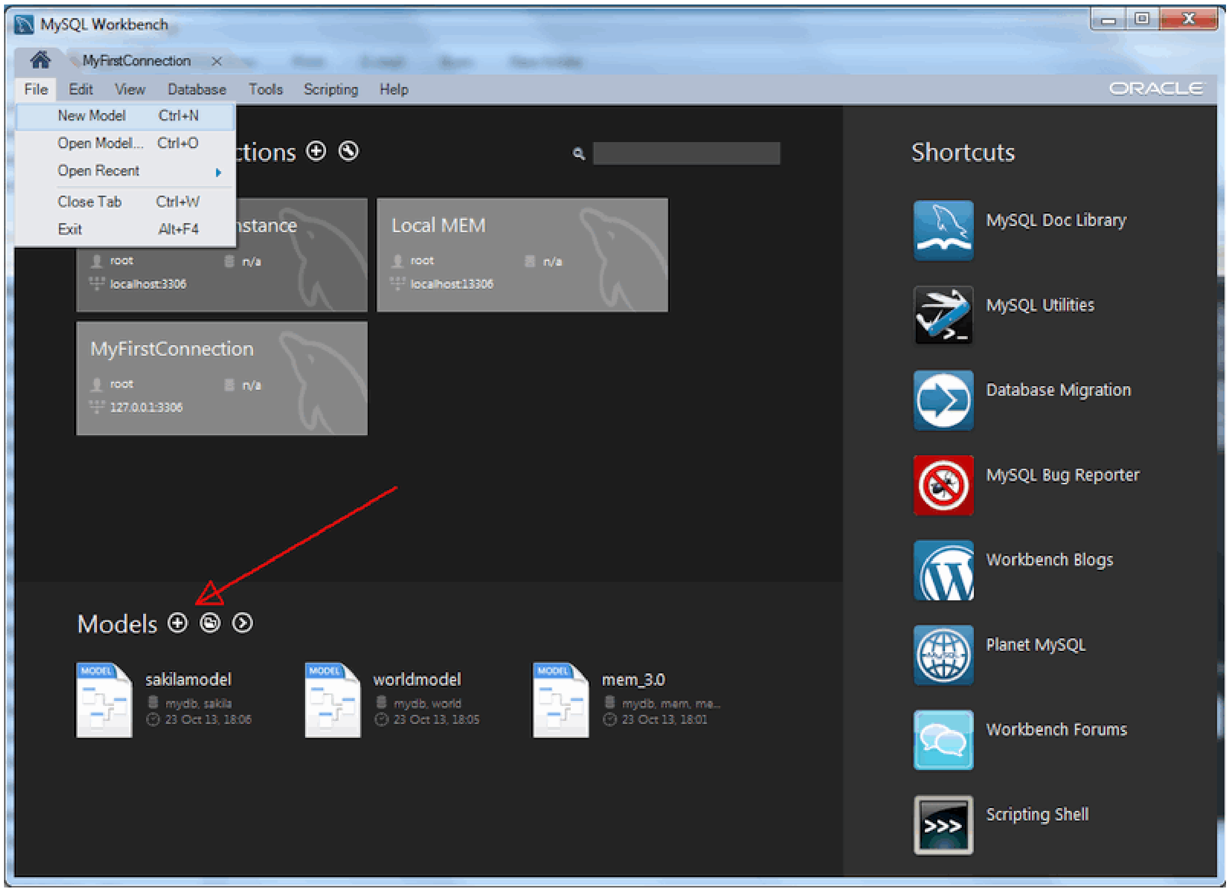
This section provides a tutorial introduction to MySQL models by showing you how to create a new database model, and how to forward engineer a model to a live MySQL server.

Note

Alternatively, you can create a model from a database by using the reverse engineering wizard. For additional information, see Section 9.4.2.2, "Reverse Engineering a Live Database".

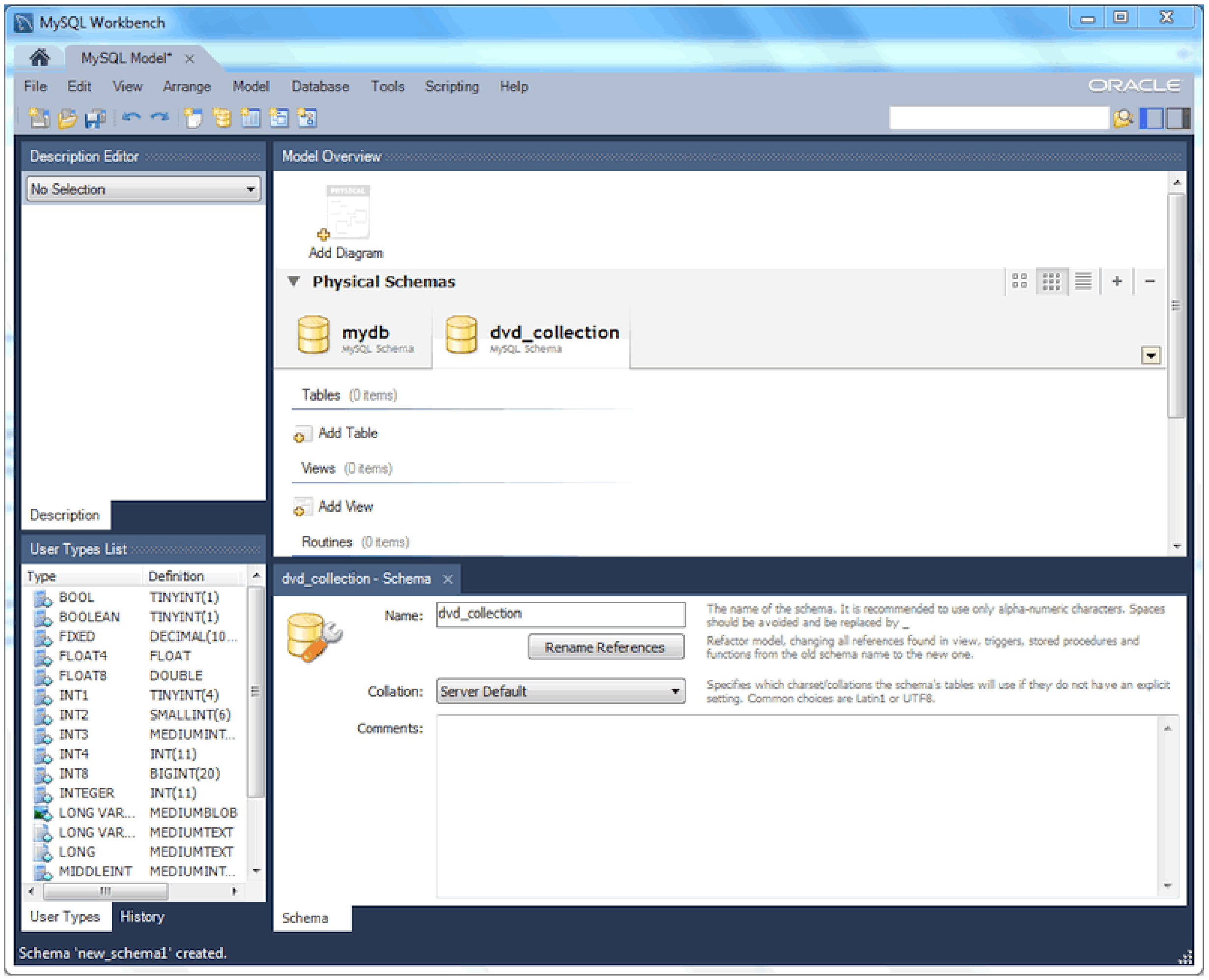
1. Start MySQL Workbench. On the Home screen, click the **[+]** icon next to the **Models** section on the bottom of the page, or select **File**, **New Model**. A model can contain multiple schemata. Note that when you create a new model, it contains the `mydb` schema by default. You can change the name of this schema to serve your own purposes, or delete it.

Figure 9.16 Getting Started Tutorial - Home Screen



2. Click the **+** button on the right side of the **Physical Schemata** toolbar to add a new schema. The default schema name is "new_schema1", now change it to “dvd_collection” by modifying its **Name** field. Confirm this change in the **Physical Schemata** panel. Now you are ready to add a table.

Figure 9.17 Getting Started Tutorial - New Schema



Type	Definition
BOOL	TINYINT(1)
BOOLEAN	TINYINT(1)
FIXED	DECIMAL(10...
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT(4)
INT2	SMALLINT(6)
INT3	MEDIUMINT...
INT4	INT(11)
INT8	BIGINT(20)
INTEGER	INT(11)
LONG VAR...	MEDIUMBLOB
LONG VAR...	MEDIUMTEXT
LONG	MEDIUMTEXT
MIDDLEINT	MEDIUMINT...

Name: dvd_collection

The name of the schema. It is recommended to use only alpha-numeric characters. Spaces should be avoided and be replaced by _

Refactor model, changing all references found in view, triggers, stored procedures and functions from the old schema name to the new one.

Collation: Server Default

Specifies which charset/collations the schema's tables will use if they do not have an explicit setting. Common choices are Latin1 or UTF8.

Comments:

Schema 'new_schema1' created.

3. Double-click **Add Table** in the **Physical Schemata** section.

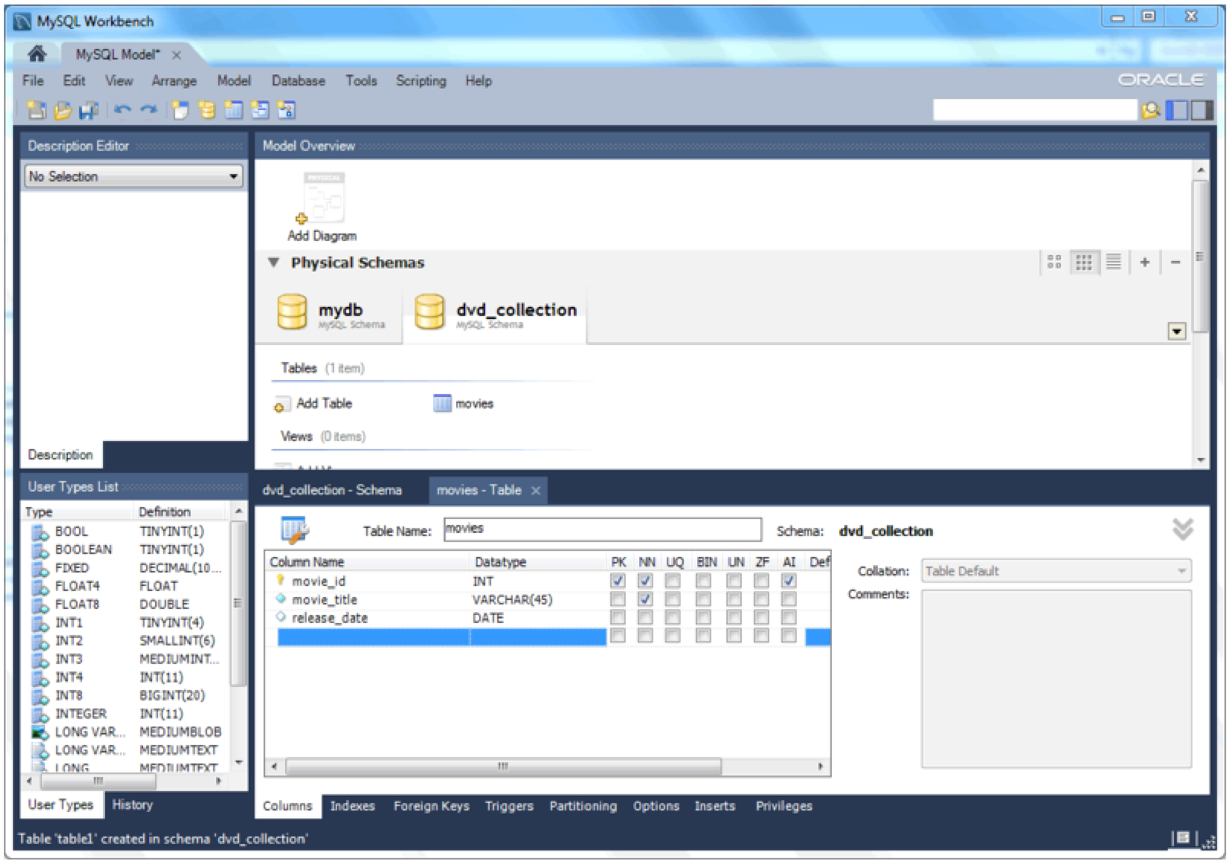
4. This automatically loads the table editor with the default table name **table1**. Edit its **Table Name** field and change the table name from “table1” to “movies”.

5. Next, add columns to your table. Double-click a **Column Name** cell, and the first field defaults to “moviesid” because (by default) MySQL Workbench appends “id” to the table name for the initial field. Change “moviesid” to “movie_id” and keep the **Datatype** as `INT`, and also select the **PK** (PRIMARY KEY), **NN** (NOT NULL), and **AI** (AUTO_INCREMENT) check boxes.

6. Add two additional columns using the same method as described above:

Column Name	Data Type	Column Properties
movie_title	VARCHAR(45)	NN
release_date	DATE (YYYY-MM-DD)	None

Figure 9.18 Getting Started Tutorial - Editing table columns



Type	Definition
BOOL	TINYINT(1)
BOOLEAN	TINYINT(1)
FIXED	DECIMAL(10...
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT(4)
INT2	SMALLINT(6)
INT3	MEDIUMINT...
INT4	INT(11)
INT8	BIGINT(20)
INTEGER	INT(11)
LONG VAR...	MEDIUMBLOB
LONG VAR...	MEDIUMTEXT
LONG	MEDIUMTEXT

Table Name: movies

Schema: dvd_collection

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Def
movie_id	INT	✓	✓					✓	
movie_title	VARCHAR(45)		✓						
release_date	DATE								

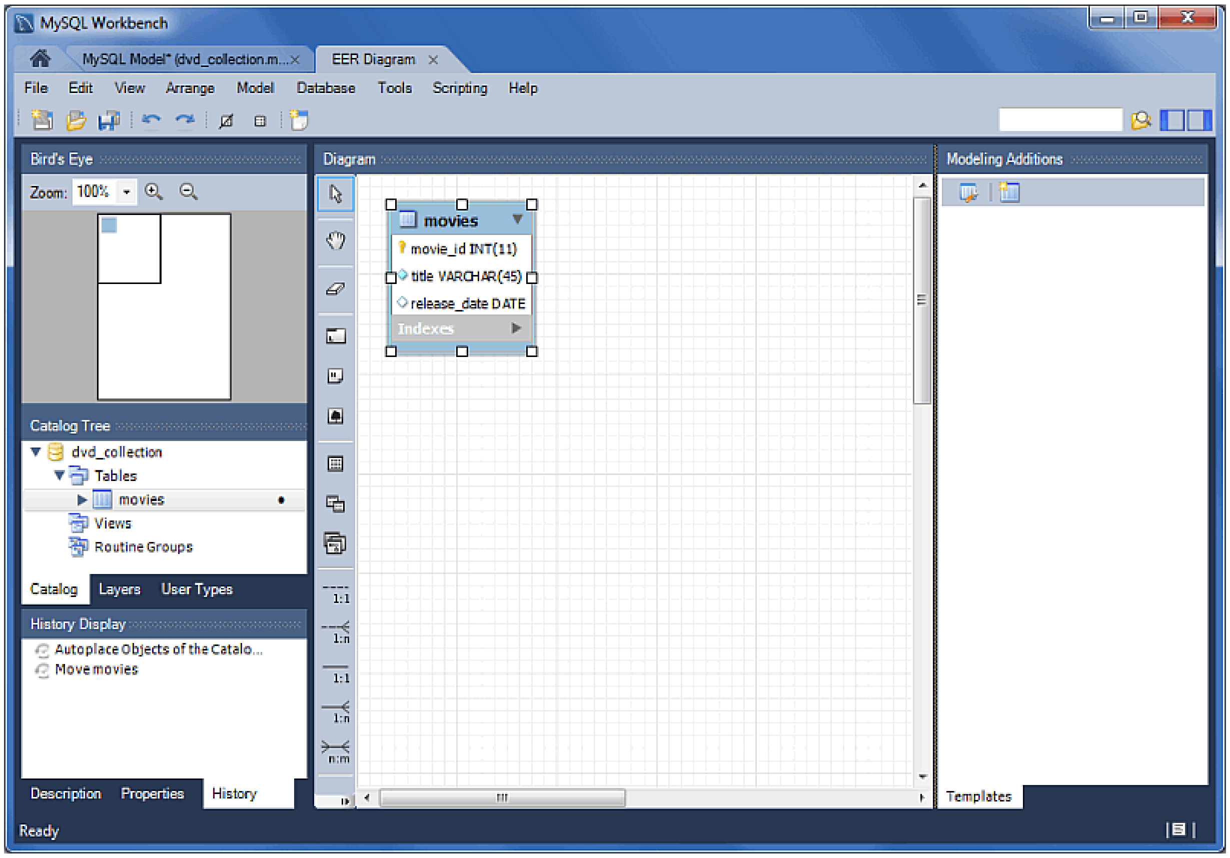
Collation: Table Default

Comments:

Table 'table1' created in schema 'dvd_collection'

7. For a visual representation (EER diagram) of this schema, select **Model**, **Create Diagram from Catalog Objects** to create the EER Diagram for the model.

Figure 9.19 Getting Started Tutorial - EER Diagram



8. In the table editor, change the name of the column “movie_title” to “title”. Note that the EER Diagram is automatically updated to reflect this change.

> **Note**
>
> To open the table editor, either change back to the **MySQL Model** tab and right-click on the `movies` table, or right-click on `movies` in the EER diagram and select an **Edit 'movies'** option.

9. Save the model by choosing **File**, **Save Model** from the main menu, or click **Save Model to Current File** on the toolbar. Enter a model name at the file prompt. For this tutorial, enter “Home_Media” and then **Save** the model.

10. Before synchronizing your new model with the live MySQL server, confirm that you already created a MySQL connection. This tutorial assumes you followed the previous Section 5.2, “Creating A New MySQL Connection (Tutorial)” tutorial to create a MySQL connection named **MyFirstConnection**, although an alternative connection can also work.

11. Now forward engineer your model to the live MySQL server. Select **Database**, **Forward Engineer...** from the main menu to open the **Forward Engineer to Database** wizard.

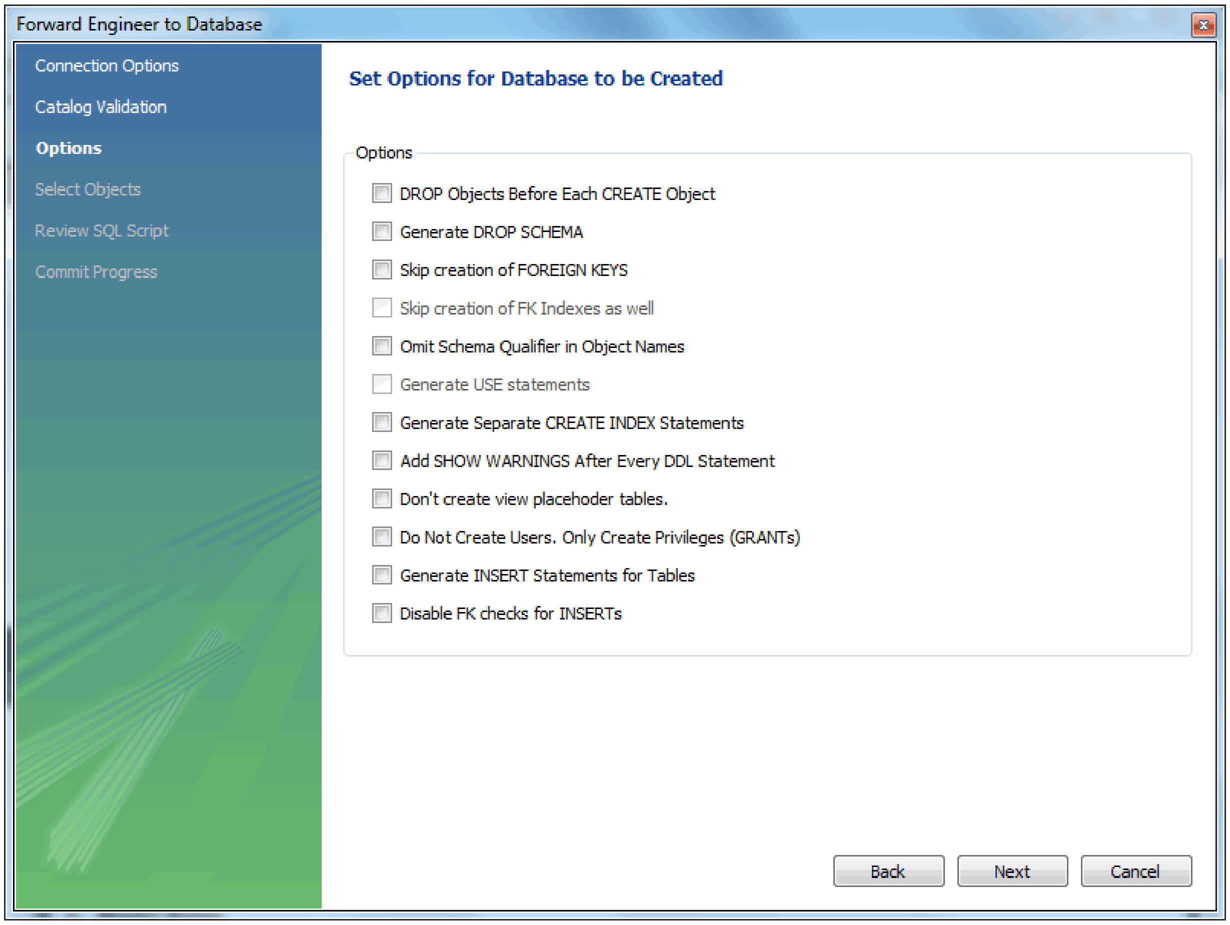
12. The **Connection Options** page selects the MySQL connection and optionally sets additional options for the selected MySQL connection. We do not require connection changes so click **Next**.

Note

You may decided to choose a different MySQL connection here, but this tutorial uses **MyFirstConnection**.

13. The **Options** page lists optional advanced options. For this tutorial, you can ignore these and click **Next**.

Figure 9.20 Getting Started Tutorial - Options



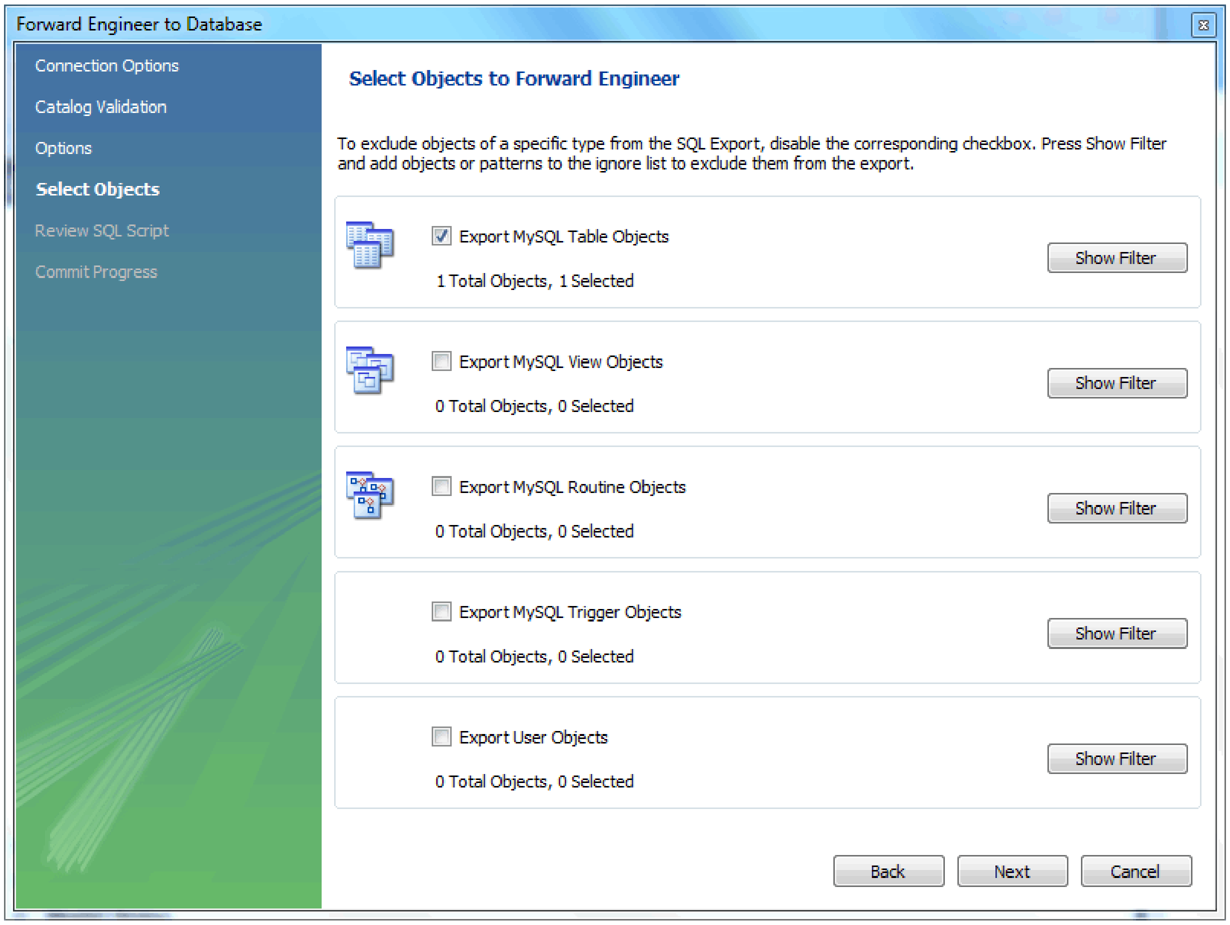
Set Options for Database to be Created

Options

- [ ] DROP Objects Before Each CREATE Object
- [ ] Generate DROP SCHEMA
- [ ] Skip creation of FOREIGN KEYS
- [ ] Skip creation of FK Indexes as well
- [ ] Omit Schema Qualifier in Object Names
- [ ] Generate USE statements
- [ ] Generate Separate CREATE INDEX Statements
- [ ] Add SHOW WARNINGS After Every DDL Statement
- [ ] Don't create view placehoder tables.
- [ ] Do Not Create Users. Only Create Privileges (GRANTs)
- [ ] Generate INSERT Statements for Tables
- [ ] Disable FK checks for INSERTs

14. Select an object to export to the live MySQL server. In this case, we only have one table (dvd_collection), so select `dvd_collection` and click **Next**.

Figure 9.21 Getting Started Tutorial - Select Objects



Select Objects to Forward Engineer

To exclude objects of a specific type from the SQL Export, disable the corresponding checkbox. Press Show Filter and add objects or patterns to the ignore list to exclude them from the export.

[✓] Export MySQL Table Objects
1 Total Objects, 1 Selected

[ ] Export MySQL View Objects
0 Total Objects, 0 Selected

[ ] Export MySQL Routine Objects
0 Total Objects, 0 Selected

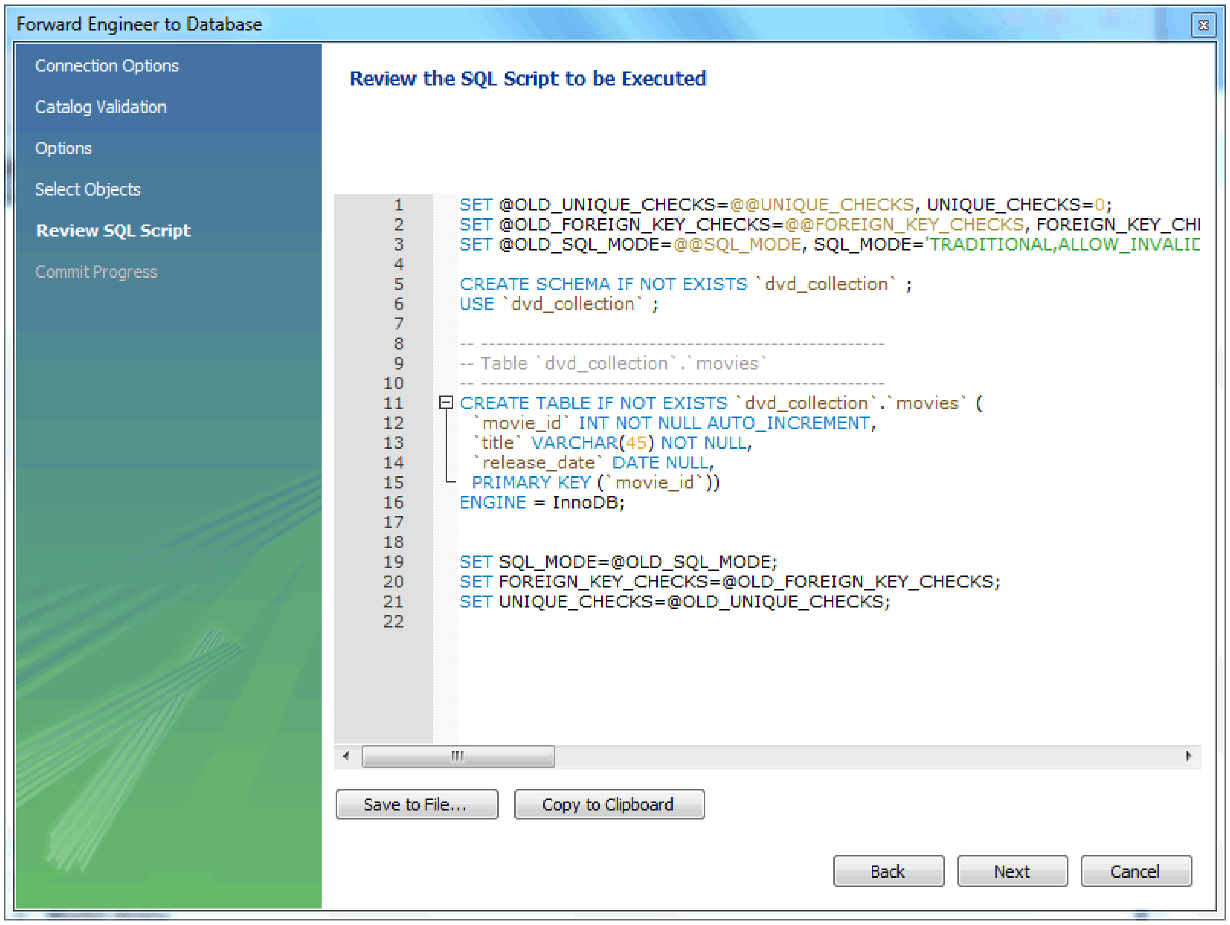
[ ] Export MySQL Trigger Objects
0 Total Objects, 0 Selected

[ ] Export User Objects
0 Total Objects, 0 Selected

15. The **Review SQL Script** page displays the SQL script that will be executed on the live server to create your schema. Review the script to make sure that you understand the operations that will be carried out.

    Click **Next** to execute the Forward Engineering process.

Figure 9.22 Getting Started Tutorial - Review SQL Script



```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHI
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALIE

CREATE SCHEMA IF NOT EXISTS `dvd_collection` ;
USE `dvd_collection` ;

-- -----------------------------------------------------
-- Table `dvd_collection`.`movies`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `dvd_collection`.`movies` (
  `movie_id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(45) NOT NULL,
  `release_date` DATE NULL,
  PRIMARY KEY (`movie_id`))
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

16. The **Commit Progress** page confirms that each step was executed. Click **Show Logs** to view the logs. If no errors are present, click **Close** to close the wizard.

17. The new `dvd_collection` database is now present on the MySQL server. Confirm this by opening the MySQL connection and viewing the schema list, or by executing `SHOW DATABASES` from the MySQL Command Line Client (`mysql`).

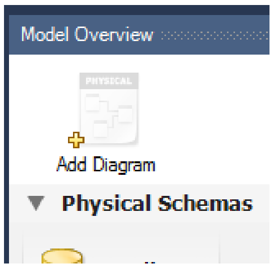
18. Ensure that your model is saved. Click **Save Model to Current File** on the main toolbar.

For additional information about data modeling, see Chapter 9, *Database Design / Modeling*.

9.3.2 Basic Modeling

On the `MySQL Model` page, double-click the **Add Diagram** icon. This creates and opens a new `EER Diagram` canvas.

Figure 9.23 Adding an EER Diagram



From an EER diagram page you can graphically design a database.

9.3.2.1 Adding a Table

The tools in the vertical toolbar on the left of the **EER Diagram** tab are used for designing an EER diagram. Start by creating a table using the table tool. The table tool is the rectangular grid in the middle of the vertical toolbar. Mousing over it shows the message, `Place a New Table (T)`.

Clicking this tool changes the mouse pointer to a hand with a rectangular grid. Create a table on the canvas by clicking anywhere on the `EER Diagram` grid.

Right-click the table and choose **Edit in New Window** from the pop-up menu. This opens the table editor, docked at the bottom of the application.

The table name defaults to `table1`. Change the name by entering `invoice` into the **Name:** field. Changes here affect the name of the tab in the table editor and the name of the table on the canvas.

Pressing **Tab** or **Enter** while the cursor is in the table name field selects the **Columns** tab of the table editor and creates a default column named `idinvoice`.

Pressing **Tab** or **Enter** again sets the focus on the `Datatype` list with `INT` selected. Notice that a field has been added to the table on the EER canvas.

Pressing **Tab** yet again and the focus shifts to adding a second column. Add a `Description` and a `Customer_id` column. When you are finished, close the table editor, by clicking the **x** button on the top left of the table editor.

9.3.2.2 Creating a Foreign Key

Select the table tool again and place another table on the canvas. Name this table `invoice_item`. Next click the `1:n Non-Identifying Relationship` tool.

First, click the `invoice_item` table; notice that a red border indicates that this table is selected. Next, click the `invoice` table. This creates a foreign key in the `invoice_item` table, the table on the “many” side of the relationship. This relationship between the two tables is shown graphically in crow's foot notation.

Revert to the default mouse pointer by clicking the arrow at the top of the vertical toolbar. Click on the `invoice_item` table and select the **Foreign keys** tab.

Click the **Foreign key Name** field. The referenced table should show in the **Referenced Table** column and the appropriate column in the **Referenced Column** column.

To delete the relationship between two tables, click the line joining the tables and then press **Control +Delete**.

Experiment with the other tools on the vertical toolbar. Delete a relationship by selecting the eraser tool and clicking the line joining two tables. Create a view, add a text object, or add a layer.

Save your changes to a `MySQL Workbench Models` file (`mwb` extension) by choosing **Save** from the **File** menu or by using the keyboard command **Control+S**.

9.3.3 Importing a Data Definition SQL Script

For this tutorial, use the `sakila` database script, which you can find by visiting the http://dev.mysql.com/doc/ page, selecting the `Other Docs` tab, and looking in the `Example Databases` section

After downloading the file, extract it to a convenient location. Open MySQL Workbench and find the **Reverse Engineer MySQL Create Script** menu item by first choosing **File** and then **Import**. Find and import the `sakila-schema.sql` file. This is the script that contains the data definition statements for the `sakila` database. The file filter for the file open dialog window defaults to `*.sql` so you should be able to view only files with the `sql` extension.

If the file was successfully imported, the application's status bar reads, `Import MySQL Create Script done.` To view the newly imported script, expand the `Physical Schemata` section by double-clicking the arrow on the left of the `Physical Schemata` title bar. Select the tab labeled **sakila**.

You may also wish to remove the default schema tab, `mydb`. Select this tab, then click the **-** button on the upper right in the **Physical Schemata** panel.

To view all the objects in the `sakila` schema, you may need to expand the **Physical Schemata** window. Move the mouse pointer anywhere over the gray area that defines the lower edge of the **Physical Schemata** window. Hold down the right mouse button and move the mouse to adjust the size of the window.

After you have expanded the window, all the objects in the `sakila` database should be visible. Tables appear at the top followed by views and then routines. There are no routine groups in this schema, but you should see the **Routine Groups** section and an `Add Group` icon.

For a complete description of importing a MySQL create script, see Section 9.4.2.1, “Reverse Engineering Using a Create Script”.

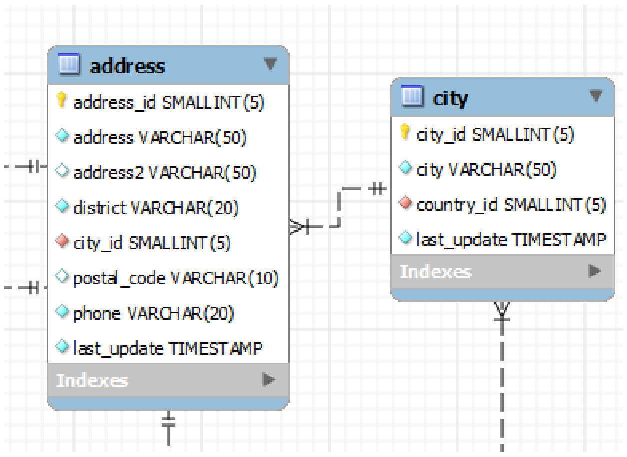
9.3.3.1 Adding an EER Diagram

To create an EER diagram for the `sakila` database, first add an EER diagram by double-clicking the `Add Diagram` icon in the **EER Diagrams** panel to create and open a new `EER Diagram` editor.

The `EER Diagram` canvas is where object modeling takes place. To add a table to the canvas, select the **Catalog** tab in the middle panel on the right side of the application to display any schemata that appear in the **MySQL Model** tab. Find the `sakila` schema and expand the view of its objects by clicking the **+** button to the left of the schema name. Expand the tables list in the same way.

You can add tables to the EER canvas by dragging them from the **Catalog** panel dropping them onto the canvas. Drop the `address` table and the `city` table onto the canvas.

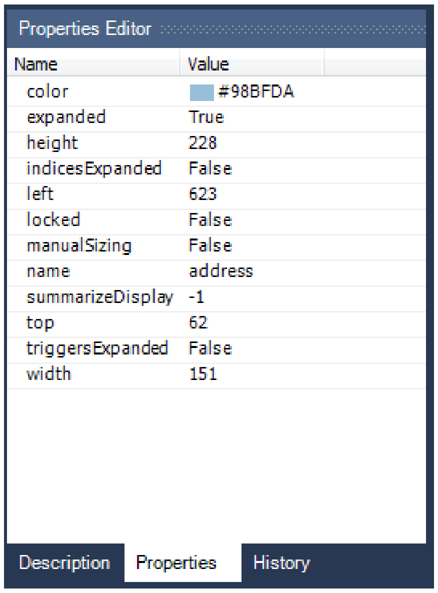
Figure 9.24 Adding Tables to the Canvas



MySQL Workbench automatically discovers that `address.city_id` has been defined as a foreign key referencing the `city.city_id` field. Drop the `country` table onto the canvas and immediately you should see the relationship between the `country` table and the `city` table. (To view all the relationships in the `sakila` database, see Figure 9.29, “The sakila Database EER Diagram”.)

Click the **Properties** tab of the panel on the lower left, then click one of the tables on the canvas. This displays the properties of the table in the `Properties` window. While a table is selected, you can use the `Properties` window to change a table's properties. For example, entering `#FF0000` for the color value will change the color accent to red.

Figure 9.25 Viewing The Properties



Name	Value
color	🟦 #98BFDA
expanded	True
height	228
indicesExpanded	False
left	623
locked	False
manualSizing	False
name	address
summarizeDisplay	-1
top	62
triggersExpanded	False
width	151

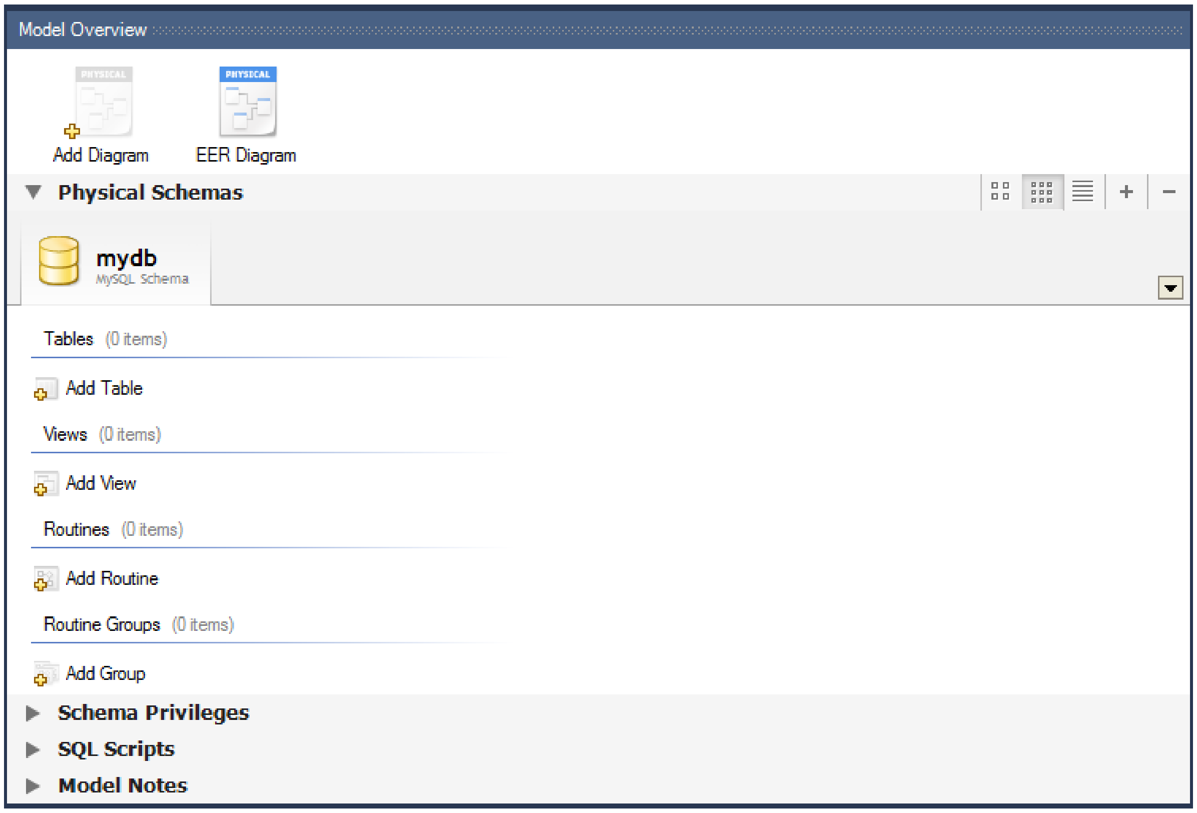
Changing the color of a table is a good way to identify a table quickly—something that becomes more important as the number of tables increases. Changing the color of a table is also an easy way to identify a table in the `Model Navigator` panel. This panel, the uppermost panel on the left side of the page, gives a bird's eye view of the entire EER canvas.

Save your changes to a `MySQL Workbench Models` file (`mwb` extension) by choosing **Save** from the **File** menu or by using the keyboard command **Control** + **S**.

9.3.4 Using the Default Schema

When you first open MySQL Workbench a default schema, `mydb` appears as the leftmost tab of the **Physical Schemata** section of MySQL Workbench. You can begin designing a database by using this default schema.

Figure 9.26 The Default Schema



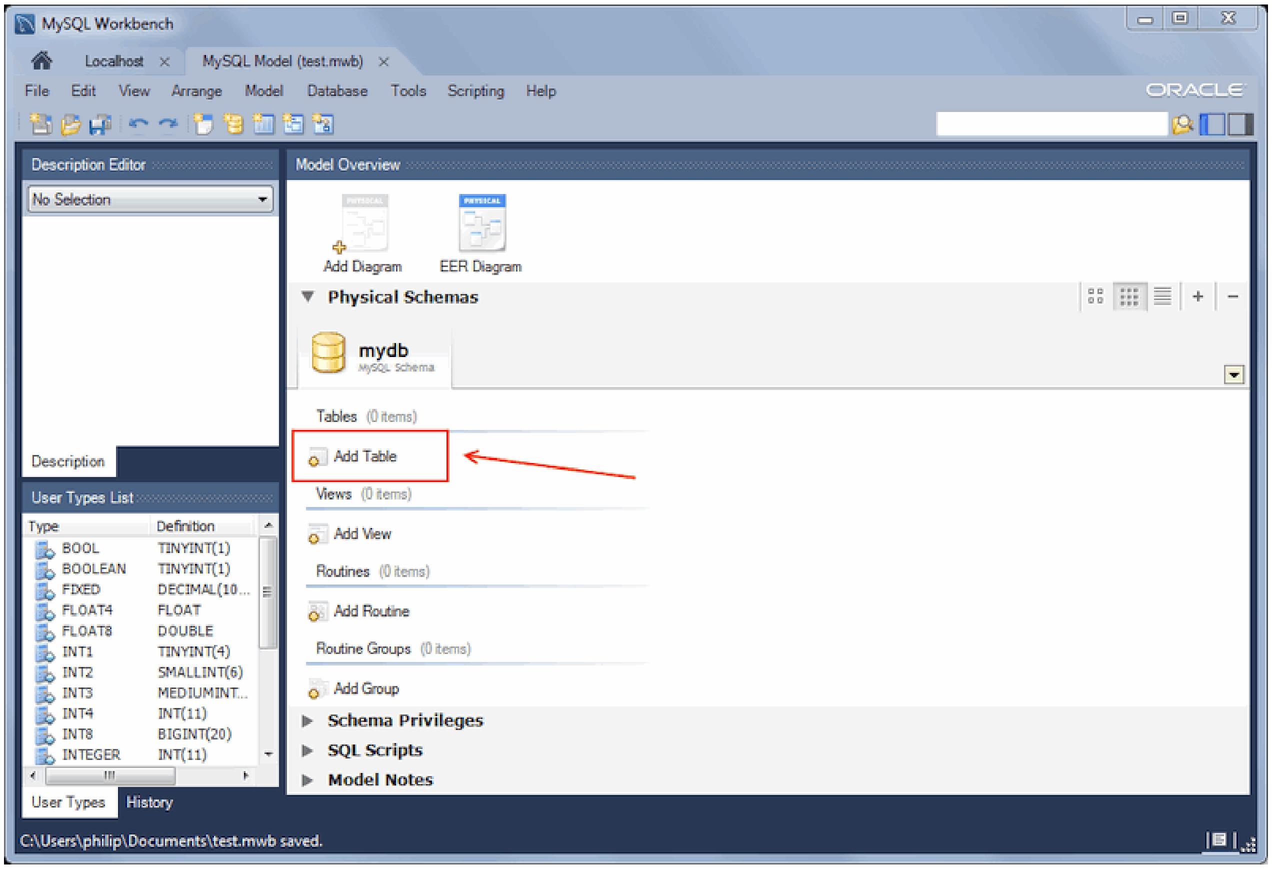
To change the name of the default schema, double-click the schema tab. This opens a schema editor window docked at the bottom of the application. To undock or redock this window, double-click anywhere in the editor title bar.

To rename the schema, use the field labeled **Name**. After you have renamed the schema, a lightning bolt icon appears right aligned in the **Name** field, indicating that other changes are pending. Click the **Comments** field and a dialog box opens asking if you wish to rename all schema occurrences. Clicking **Yes** ensures that your changes are propagated throughout the application. Add comments to the database and change the collation if you wish. Close the schema editor by clicking the **x** button.

9.3.4.1 Creating a New Table

Create a new table by double-clicking the **Add Table** icon in the `Physical Schemata` panel. This opens the table editor docked at the bottom of the application. If you wish, you can undock or dock this editor in exactly the same way as the schema editor window.

Figure 9.27 Model: Creating A New Table



Type	Definition
BOOL	TINYINT(1)
BOOLEAN	TINYINT(1)
FIXED	DECIMAL(10...
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT(4)
INT2	SMALLINT(6)
INT3	MEDIUMINT...
INT4	INT(11)
INT8	BIGINT(20)
INTEGER	INT(11)

C:\Users\philip\Documents\test.mwb saved.

Initially, the table name defaults to 'table1' in the table editor. The following screenshot describes the available actions:

Figure 9.28 Model: Editing Table Values

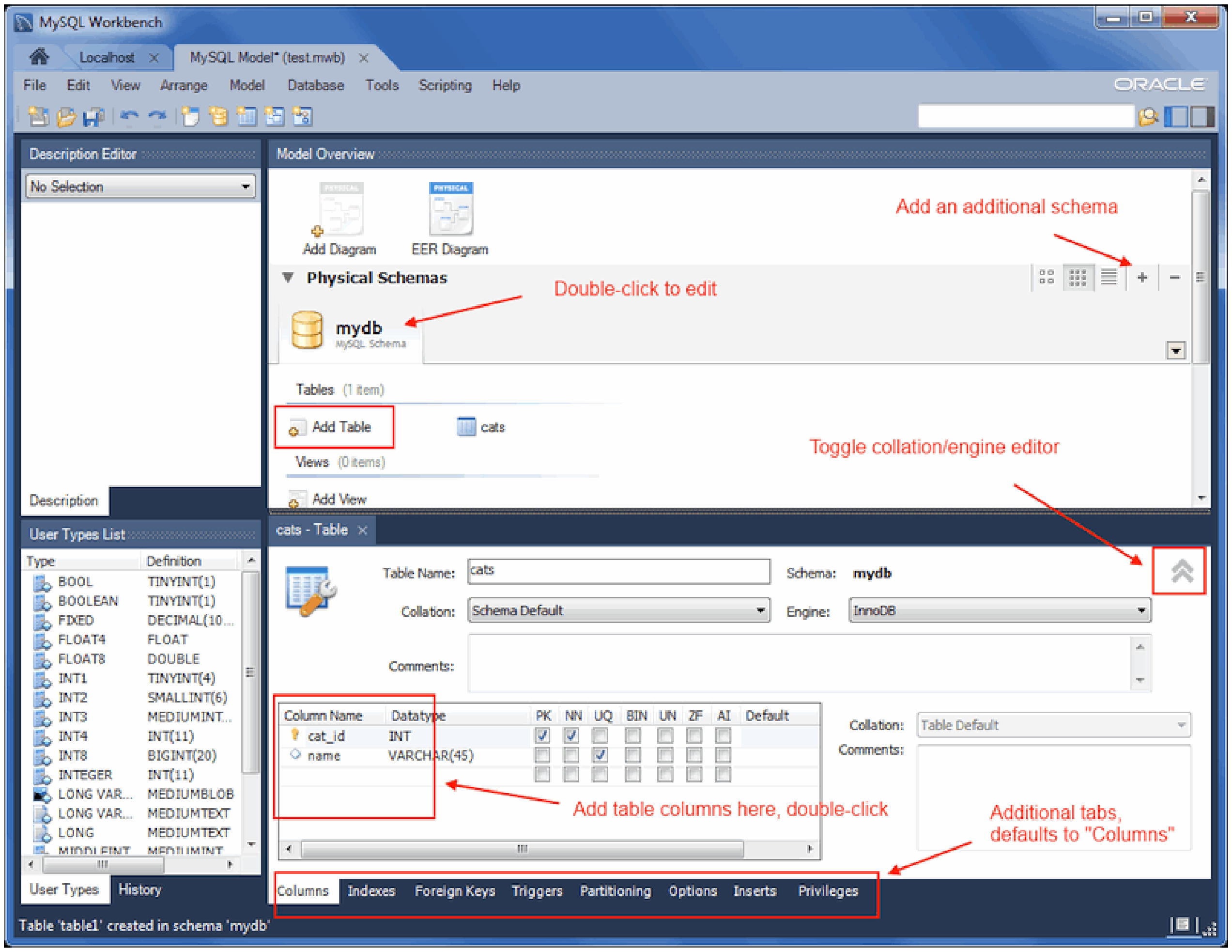


Table Name: cats
Schema: mydb
Collation: Schema Default
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
cat_id	INT	✓	✓						
name	VARCHAR(45)			✓					

Collation: Table Default
Comments:

Type	Definition
BOOL	TINYINT(1)
BOOLEAN	TINYINT(1)
FIXED	DECIMAL(10...
FLOAT4	FLOAT
FLOAT8	DOUBLE
INT1	TINYINT(4)
INT2	SMALLINT(6)
INT3	MEDIUMINT...
INT4	INT(11)
INT8	BIGINT(20)
INTEGER	INT(11)
LONG VAR...	MEDIUMBLOB
LONG VAR...	MEDIUMTEXT
LONG	MEDIUMTEXT
MIDDLEINT	MEDIUMINT

Table 'table1' created in schema 'mydb'

In our example above, columns were added using the **Columns** tab. Clicking an empty row will add a new column, and clicking an existing column starts edit mode. Click the **Tab** key to move to the next column and set the column's data type.

Altering the table by adding indexes or other features is also possible using the table editor by clicking each tab within the table editor.

9.3.4.2 Creating Other Schema Objects

Additional objects such as views or routines can be added in the same way as tables.

Objects are listed under the **Catalog** palette on the right. To view these schema objects, select the **Catalog** tab in the middle palette on the right. View all the objects by clicking the **+** button to the left of the schema name.

Save your changes to a `MySQL Workbench Models` file (`mwb` extension) by choosing **Save** from the **File** menu or by using the keyboard command **Control+S**.

9.3.5 Documenting the sakila Database

This chapter highlights the capabilities of MySQL Workbench as a documentation tool using the `sakila` database as an example. This is a sample database provided by MySQL that you can find by visiting the http://dev.mysql.com/doc/ page, selecting the `Other Docs` tab, and looking in the `Example Databases` section

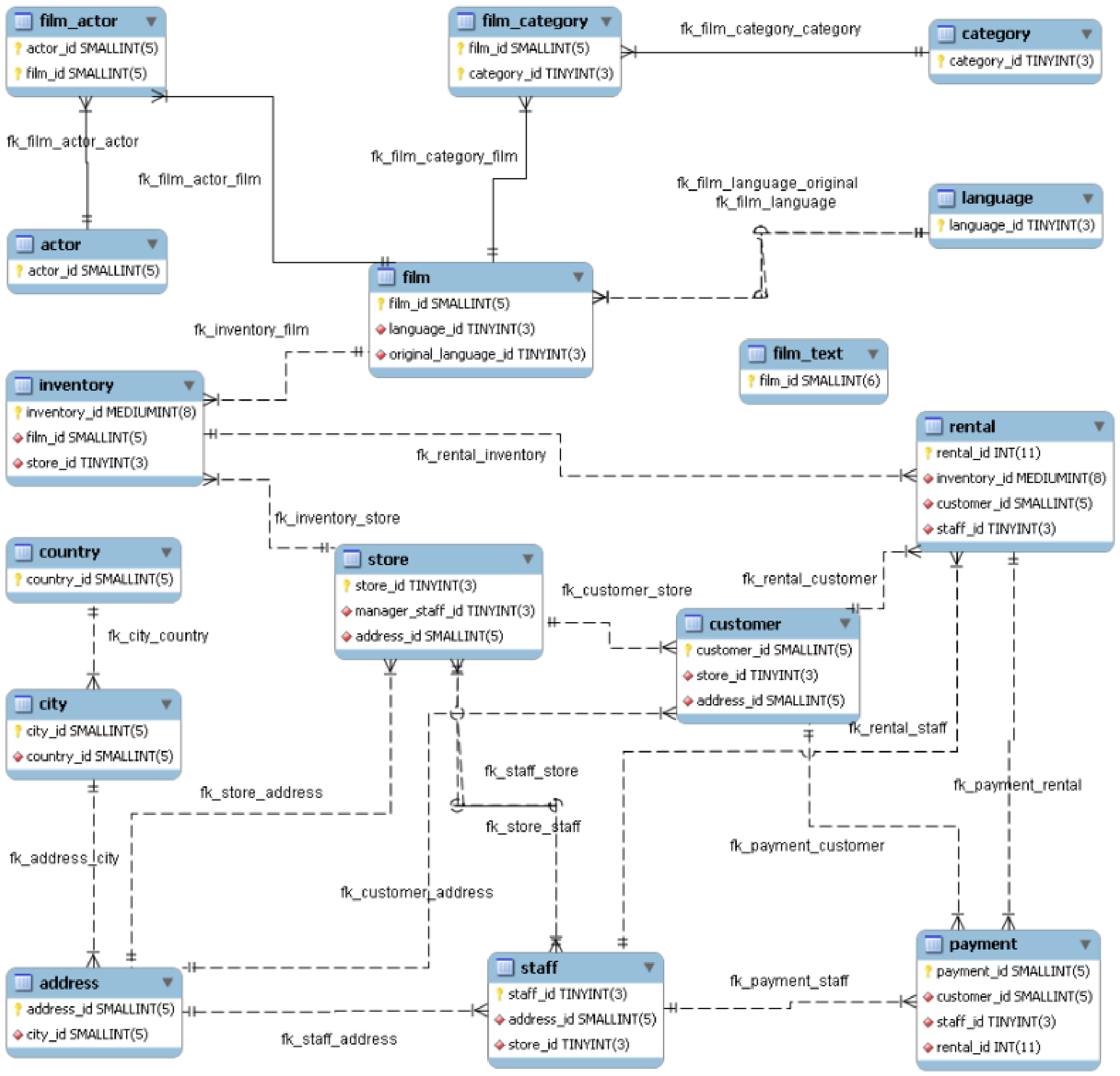
An EER diagram provides a quick overview and understanding of a database. Rather than reading through table definition statements, a quick glance at an EER diagram indicates how tables are related.

You can also see how tables are related; what the foreign keys are and what the nature of the relationship is.

9.3.5.1 A PNG File of the sakila Database

Find following an EER diagram showing all the tables in the `sakila` database. This image was created using the **File**, **Export**, **Export as PNG...** menu item.

Figure 9.29 The sakila Database EER Diagram



The object notation style used in Figure 9.29, “The sakila Database EER Diagram” is `Workbench (PKs only)`. This notation shows only primary keys and no other columns, which is especially useful where space is at a premium. The relationship notation is the default, Crow's Foot.

As the connection lines show, each table is related to at least one other table in the database (with the exception of the `film_text` table). Some tables have two foreign keys that relate to the same table. For example the `film` table has two foreign keys that relate to the `language` table, namely `fk_film_language_original` and `fk_film_language`. Where more than one relationship exists between two tables, the connection lines run concurrently.

Identifying and non-identifying relationships are indicated by solid and broken lines respectively. For example, the foreign key `category_id` is part of the primary key in the `film_category` table so its relationship to the `category` table is drawn with a solid line. On the other hand, in the `city` table, the foreign key, `country_id`, is not part of the primary key so the connection uses a broken line.

9.4 Forward and Reverse Engineering

MySQL Workbench provides capabilities to forward engineering physical database designs. A visual data model can be transformed into a physical database on a target MySQL Server by executing the forward engineering wizard. All SQL code is automatically generated to help eliminate the normal error-prone process of manually writing complex SQL code. MySQL Workbench also enables you to reverse engineer an existing database or packaged application to get better insight into its database design. In addition to forward and reverse engineering existing databases, it can also import SQL scripts to build models, and export models to DDL scripts to execute at a later time.

9.4.1 Forward Engineering

It is possible to forward engineer a database using an SQL script or by connecting to a live database.

9.4.1.1 Forward Engineering Using an SQL Script

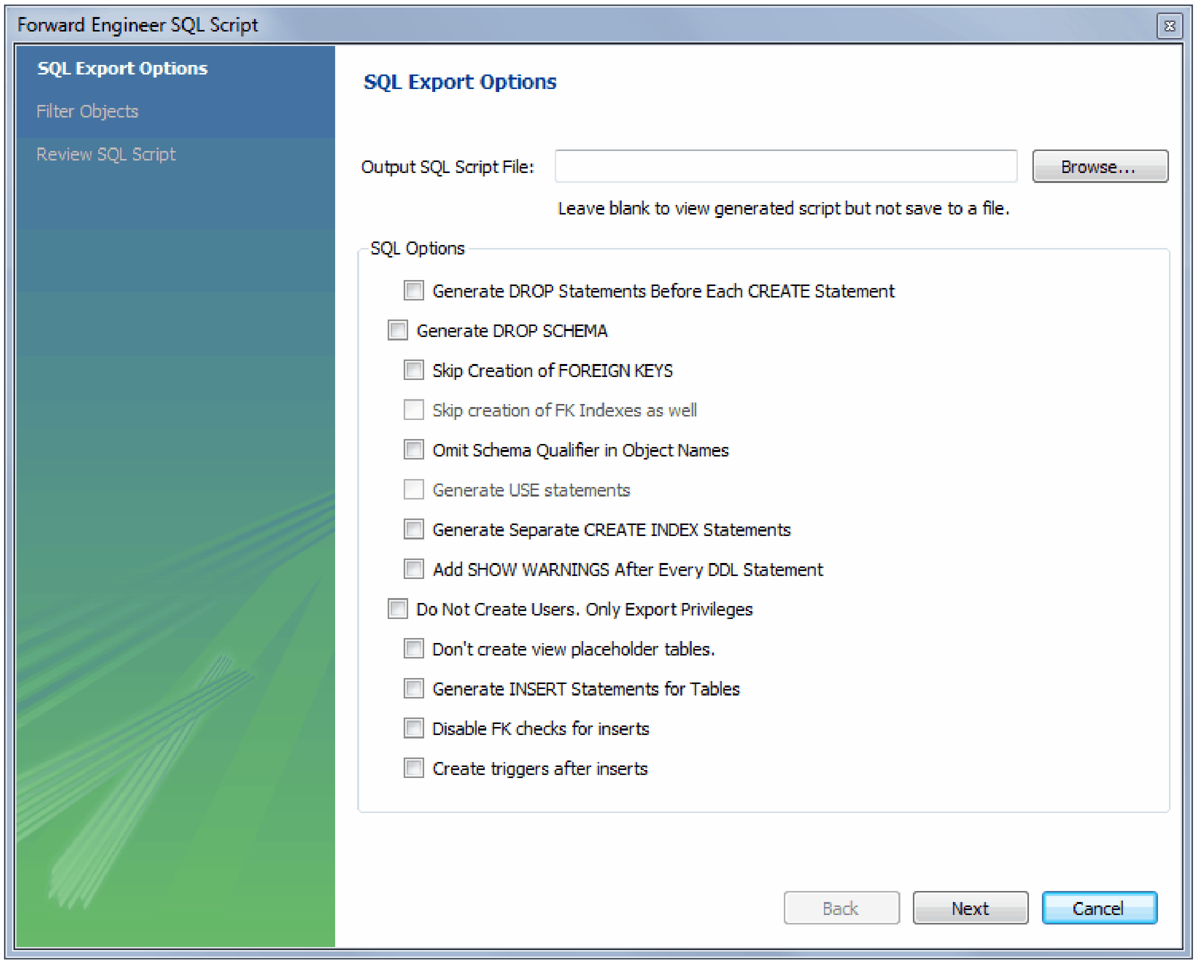
To create a script of your database model, choose the **Export** item from the **File** menu. You may export a script to alter an existing database or create a new database. The script to create a database is similar to the one created using the `mysqldump` *`db_name`* command.

Choosing to create a database yields additional options.

Creating a Schema

Select the **File**, **Export**, **Forward Engineer SQL CREATE Script** menu item to start the Forward Engineer SQL Script wizard. The following figure shows the first page of the wizard.

Figure 9.30 SQL Export Options


Forward Engineer SQL Script
SQL Export Options
Filter Objects
Review SQL Script
SQL Export Options
Output SQL Script File:
Browse...
Leave blank to view generated script but not save to a file.
SQL Options
Generate DROP Statements Before Each CREATE Statement
Generate DROP SCHEMA
Skip Creation of FOREIGN KEYS
Skip creation of FK Indexes as well
Omit Schema Qualifier in Object Names
Generate USE statements
Generate Separate CREATE INDEX Statements
Add SHOW WARNINGS After Every DDL Statement
Do Not Create Users. Only Export Privileges
Don't create view placeholder tables.
Generate INSERT Statements for Tables
Disable FK checks for inserts
Create triggers after inserts
Back
Next
Cancel

The SQL Export Options displays the following facilities:

- `Output SQL Script File`

  To specify the output file name, enter it into the **Output SQL Script File** field, or use the **Browse** button to select a file. Leave this field blank to view, but not save, the generated output.

- `Generate DROP Statements Before Each CREATE Statement`

  Select this option to generate a statement to drop each object before the statement that creates it. This ensures that any existing instance of each object is removed when the output is executed.

- `Generate DROP SCHEMA`
- `Skip creation of FOREIGN KEYS`
- `Skip creation of FK Indexes as well`
- `Omit Schema Qualifier in Object Names`

  Select this option to generate unqualified object names in SQL statements.

- `Generate USE statements`

- `Generate Separate CREATE INDEX Statements`

  Select this option to create separate statements for index creation instead of including index definitions in `CREATE TABLE` statements.

- `Add SHOW WARNINGS after every DDL statement`

  Select this option to add `SHOW WARNINGS` statements to the output. This causes display of any warnings generated when the output is executed, which can be useful for debugging.

- `Do Not Create Users. Only Export Privileges`

  Select this option to update the privileges of existing users, as opposed to creating new users. Exporting privileges for nonexistent users will result in errors when you execute the `CREATE` script. Exporting users that already exist will also result in an error.

- `Don't create view placeholder tables`

- `Generate INSERT Statements for Tables`

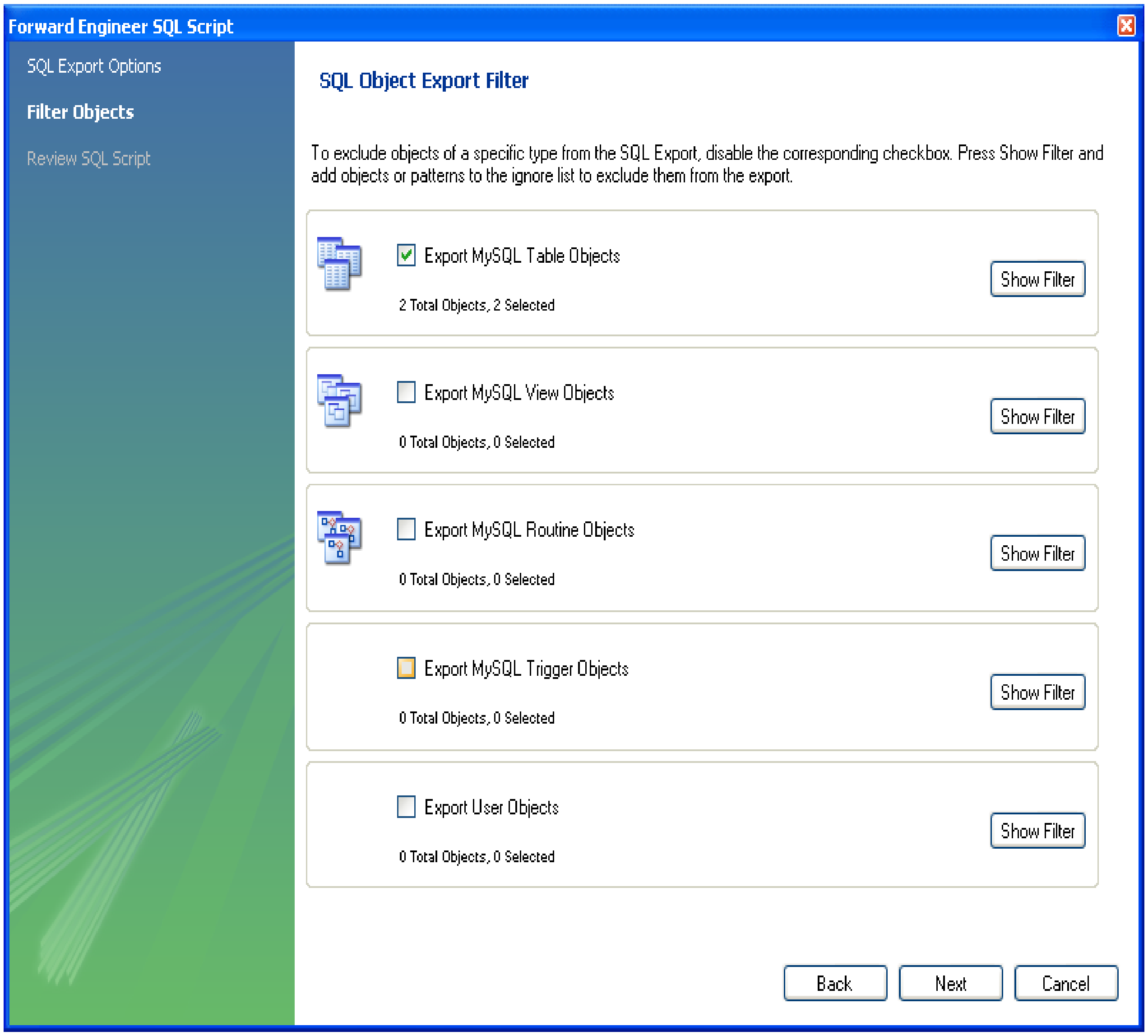
  Select this option if you have added any rows to a table. For more information about inserting rows, see Section 8.1.1, “SQL Query Window”.

- `Disable FK checks for inserts`

- `Create triggers after inserts`

Clicking **Next** takes you to the **SQL Object Export Filter** page where you select the objects you wish to export.

Figure 9.31 SQL Object Export Filter



Forward Engineer SQL Script

SQL Export Options

Filter Objects

Review SQL Script

SQL Object Export Filter

To exclude objects of a specific type from the SQL Export, disable the corresponding checkbox. Press Show Filter and add objects or patterns to the ignore list to exclude them from the export.

[✓] Export MySQL Table Objects
2 Total Objects, 2 Selected
Show Filter

[ ] Export MySQL View Objects
0 Total Objects, 0 Selected
Show Filter

[ ] Export MySQL Routine Objects
0 Total Objects, 0 Selected
Show Filter

[ ] Export MySQL Trigger Objects
0 Total Objects, 0 Selected
Show Filter

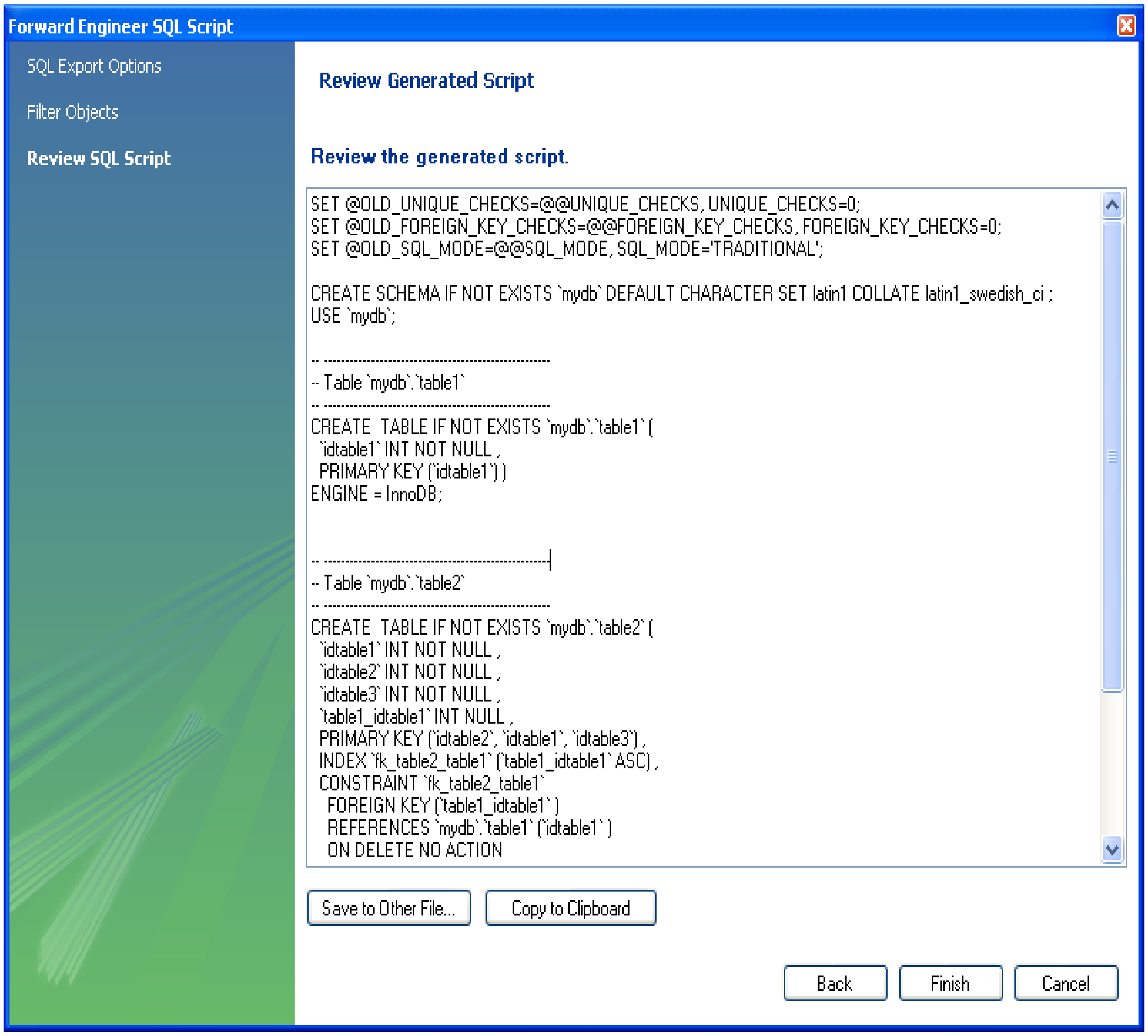
[ ] Export User Objects
0 Total Objects, 0 Selected
Show Filter

Back Next Cancel

Use **Show Filter** to fine tune (filter) the objects for export. After selecting the objects to export, click **Hide Filter** to hide the filter panel.

After selecting the objects to export, click **Next** to review the generated script.

Figure 9.32 Review Generated Script



```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci ;
USE `mydb`;

-- -----------------------------------------------------
-- Table `mydb`.`table1`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `mydb`.`table1` (
  `idtable1` INT NOT NULL ,
  PRIMARY KEY (`idtable1`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`table2`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `mydb`.`table2` (
  `idtable1` INT NOT NULL ,
  `idtable2` INT NOT NULL ,
  `idtable3` INT NOT NULL ,
  `table1_idtable1` INT NULL ,
  PRIMARY KEY (`idtable2`, `idtable1`, `idtable3`) ,
  INDEX `fk_table2_table1` (`table1_idtable1` ASC) ,
  CONSTRAINT `fk_table2_table1`
    FOREIGN KEY (`table1_idtable1` )
    REFERENCES `mydb`.`table1` (`idtable1` )
    ON DELETE NO ACTION
```

You may return to the previous page using the **Back** button.

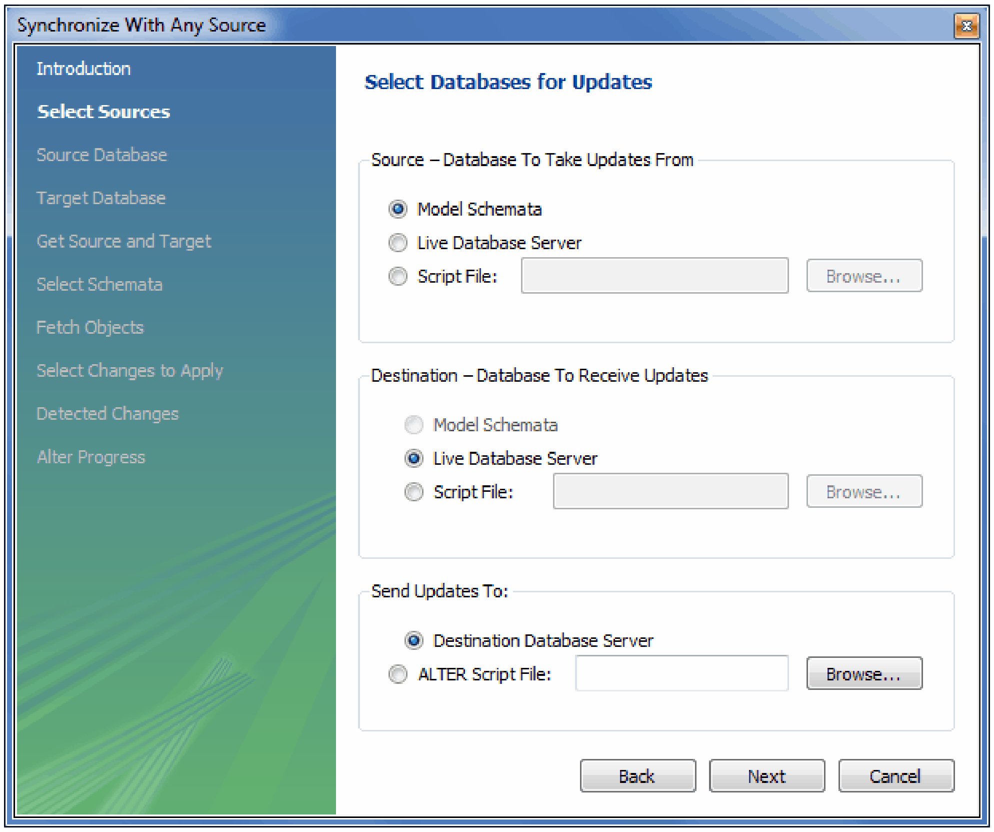
The **Finish** button saves the script file and exits. You can then use the saved script to create a database.

Altering a Schema

The menu item for creating an **ALTER Script File** is **Database**, **Synchronize With Any Source**. Typically, this option is used when the SQL script of a database has been imported into MySQL Workbench and changed, and then you want to create a script that can be executed against the database to alter it to reflect the adjusted model. For instructions on importing a DDL script, see Section 9.4.2.1, "Reverse Engineering Using a Create Script".

Select the **Database**, **Synchronize With Any Source** menu item to start the wizard. You will be presented with the first page showing the introduction, and then the available options:

Figure 9.33 Synchronize With Any Source: Options


Synchronize With Any Source

Introduction
Select Sources
Source Database
Target Database
Get Source and Target
Select Schemata
Fetch Objects
Select Changes to Apply
Detected Changes
Alter Progress

Select Databases for Updates

Source – Database To Take Updates From
[✓ Model Schemata] Live Database Server Script File: Browse...

Destination – Database To Receive Updates
Model Schemata [✓ Live Database Server] Script File: Browse...

Send Updates To:
[✓ Destination Database Server] ALTER Script File: Browse...

Back Next Cancel

For additional information, see Section 9.5.1, “Database Synchronization”.

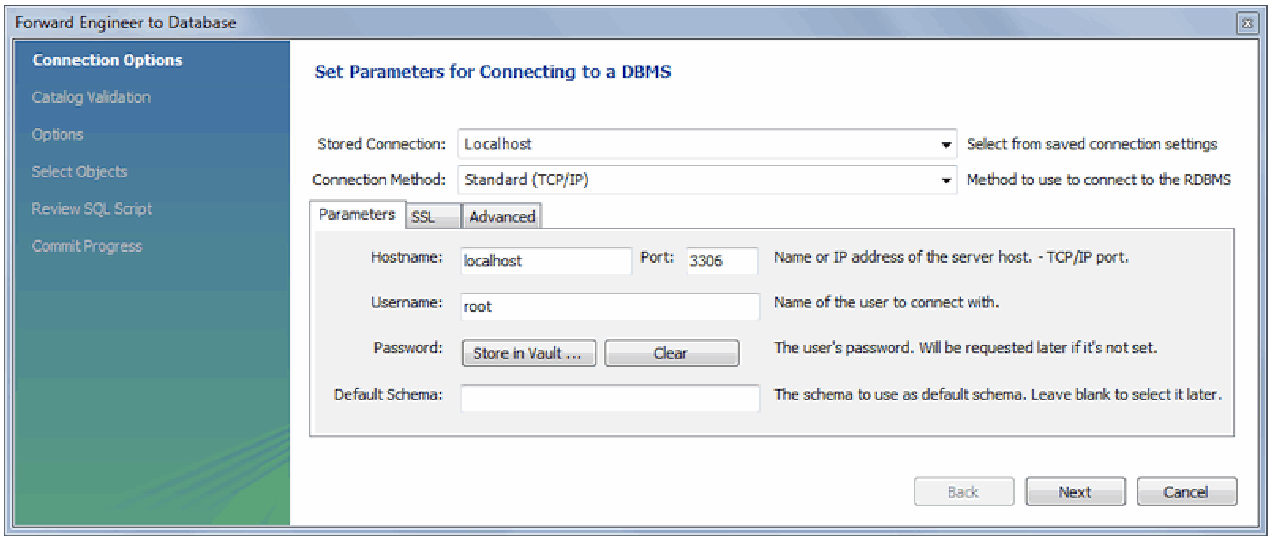
9.4.1.2 Forward Engineering to a Live Server

Use forward engineering to export your schema design to a MySQL server.

Select the model that you wish to forward engineer and then choose the **Database**, **Forward Engineer...** menu item from the main menu.

The first step of the process is to connect to a MySQL server to create the new database schema. This page enables you to use a previously stored connection, or enter the connection parameters.

Figure 9.34 Set Parameters for Connecting to a DBMS



Set Parameters for Connecting to a DBMS

Stored Connection: Localhost — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: localhost Port: 3306 — Name or IP address of the server host. - TCP/IP port.

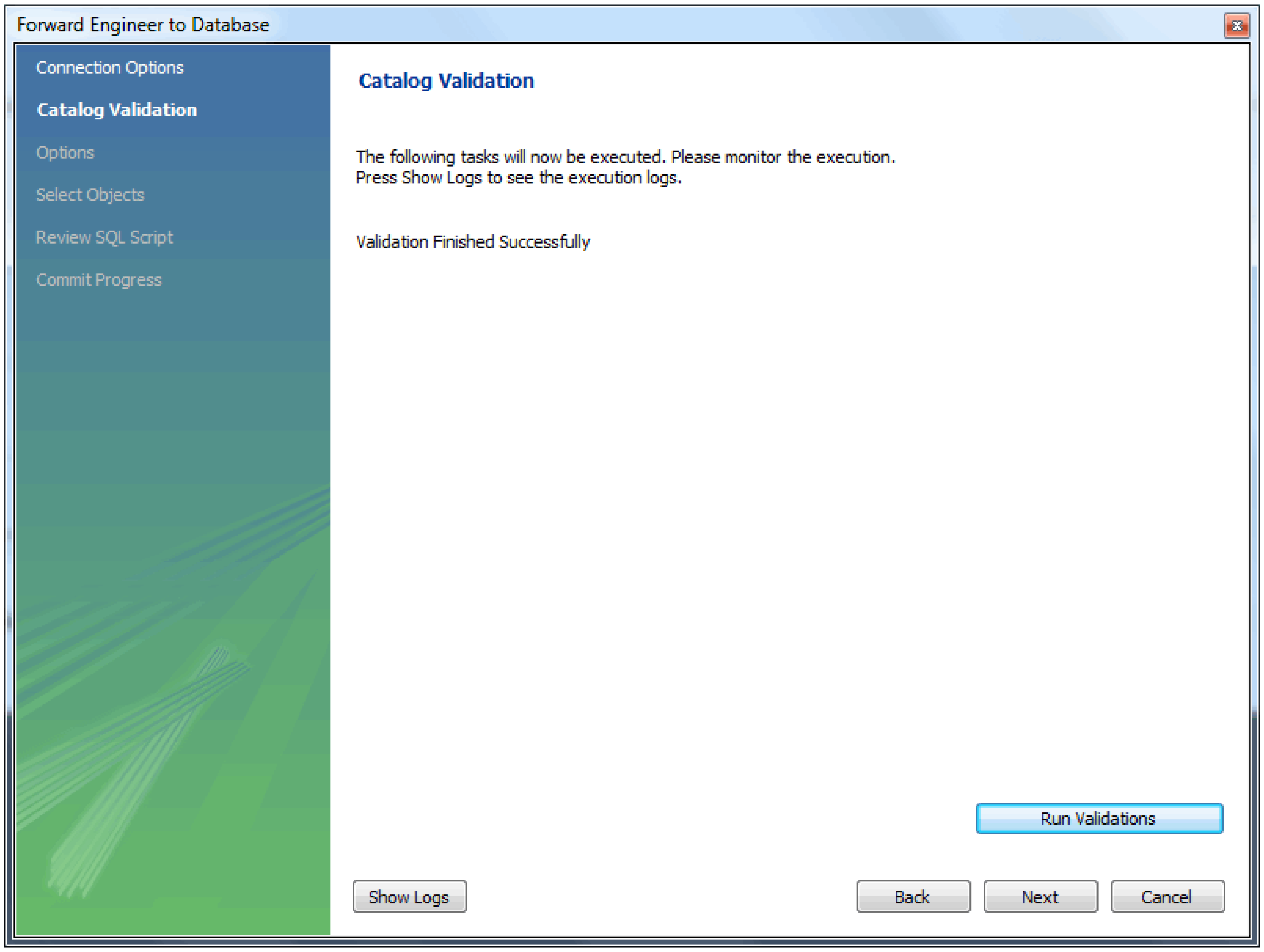
Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

Click **Next** after setting the connection parameters. The next page of the wizard displays is Catalog Validation (validation is available only in the Commercial Edition).

Figure 9.35 Catalog Validation



Catalog Validation

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

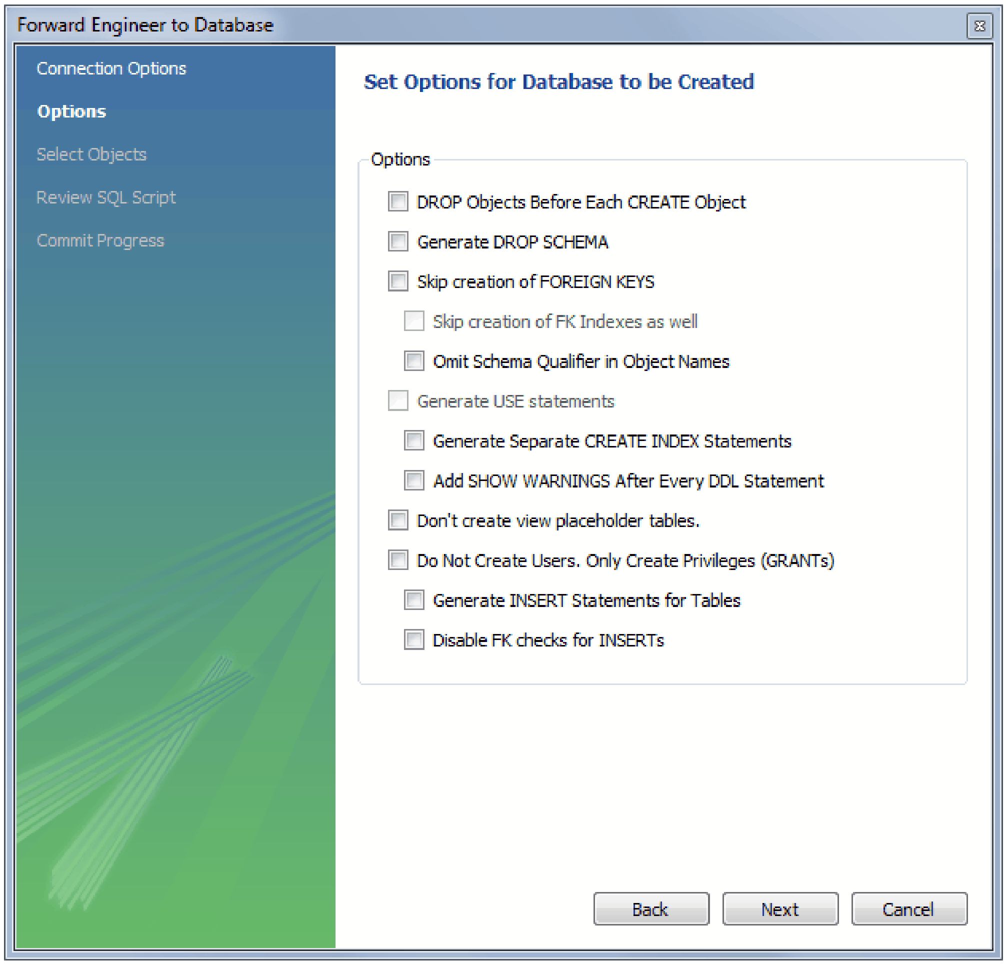
Validation Finished Successfully

Click **Run Validations** to validate the catalog.

Click **Next** to continue.

The next page enables you to set options for the database to be created. These options are as described in Creating a Schema.

Figure 9.36 Options



Set Options for Database to be Created

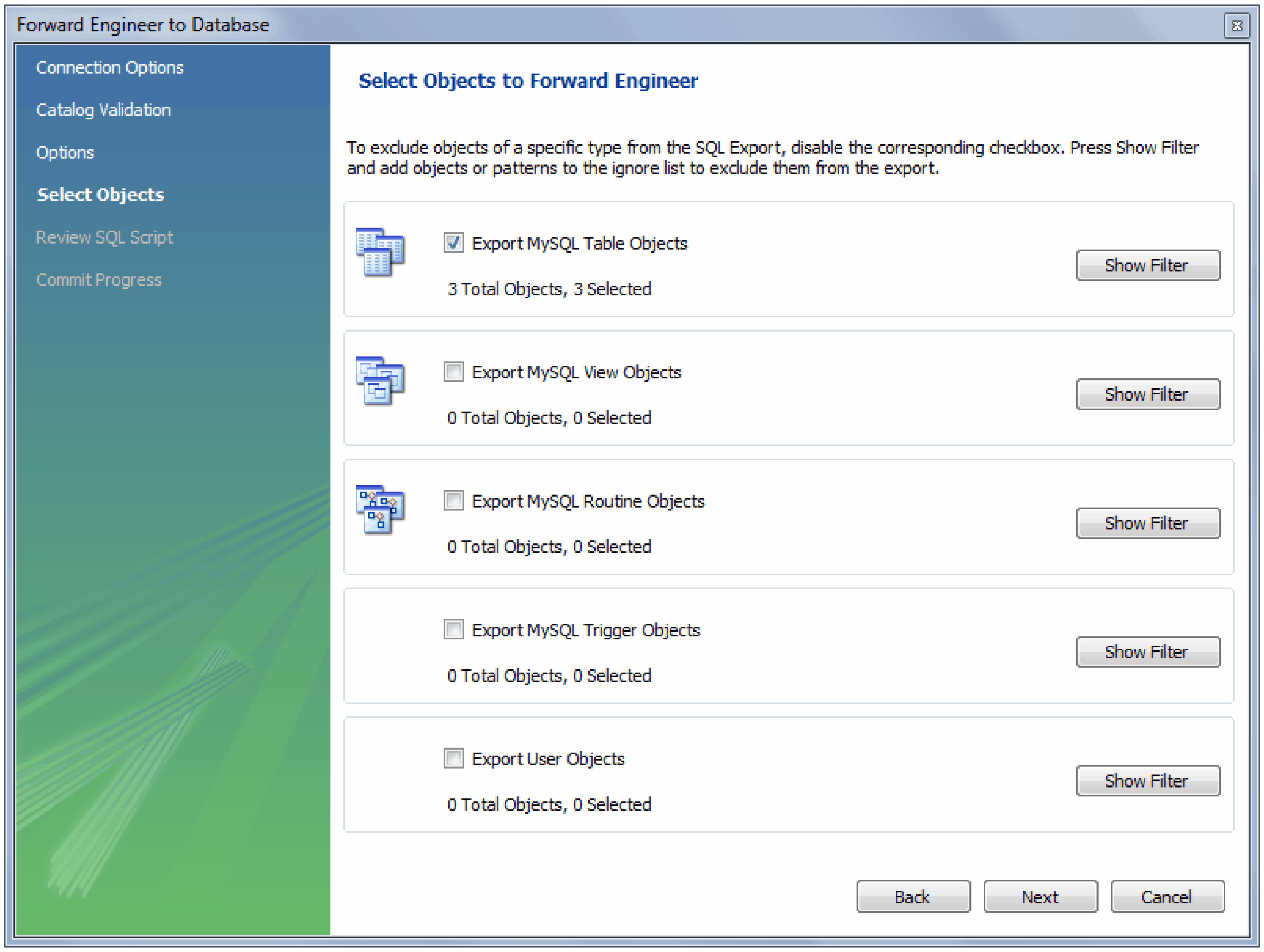
Options

- [ ] DROP Objects Before Each CREATE Object
- [ ] Generate DROP SCHEMA
- [ ] Skip creation of FOREIGN KEYS
- [ ] Skip creation of FK Indexes as well
- [ ] Omit Schema Qualifier in Object Names
- [ ] Generate USE statements
- [ ] Generate Separate CREATE INDEX Statements
- [ ] Add SHOW WARNINGS After Every DDL Statement
- [ ] Don't create view placeholder tables.
- [ ] Do Not Create Users. Only Create Privileges (GRANTs)
- [ ] Generate INSERT Statements for Tables
- [ ] Disable FK checks for INSERTs

Select the required options and then click **Next.**

The next page enables you to select the objects to forward engineer.

Figure 9.37 Select Objects to Forward Engineer



Forward Engineer to Database

- Connection Options
- Catalog Validation
- Options
- **Select Objects**
- Review SQL Script
- Commit Progress

Select Objects to Forward Engineer

To exclude objects of a specific type from the SQL Export, disable the corresponding checkbox. Press Show Filter and add objects or patterns to the ignore list to exclude them from the export.

[✓] Export MySQL Table Objects
3 Total Objects, 3 Selected
Show Filter

[ ] Export MySQL View Objects
0 Total Objects, 0 Selected
Show Filter

[ ] Export MySQL Routine Objects
0 Total Objects, 0 Selected
Show Filter

[ ] Export MySQL Trigger Objects
0 Total Objects, 0 Selected
Show Filter

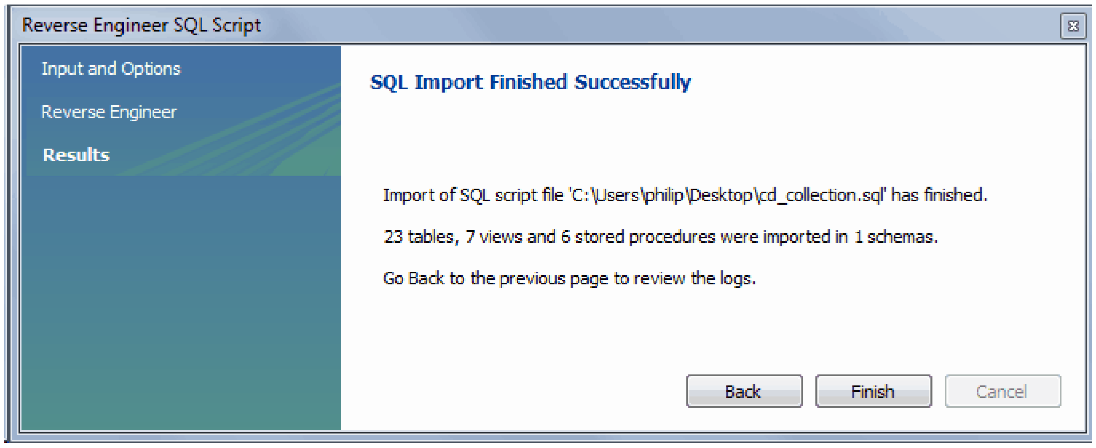
[ ] Export User Objects
0 Total Objects, 0 Selected
Show Filter

Back Next Cancel

To select a subset of objects to forward engineer, use the **Show Filter**/**Hide Filter** button, then select specific objects. After you have selected your objects, click **Next** to continue

On the **Review Script** page you may review and edit the SQL script that will be executed.

Figure 9.38 Review Script



Reverse Engineer SQL Script

- Input and Options
- Reverse Engineer
- **Results**

SQL Import Finished Successfully

Import of SQL script file 'C:\Users\philip\Desktop\cd_collection.sql' has finished.

23 tables, 7 views and 6 stored procedures were imported in 1 schemas.

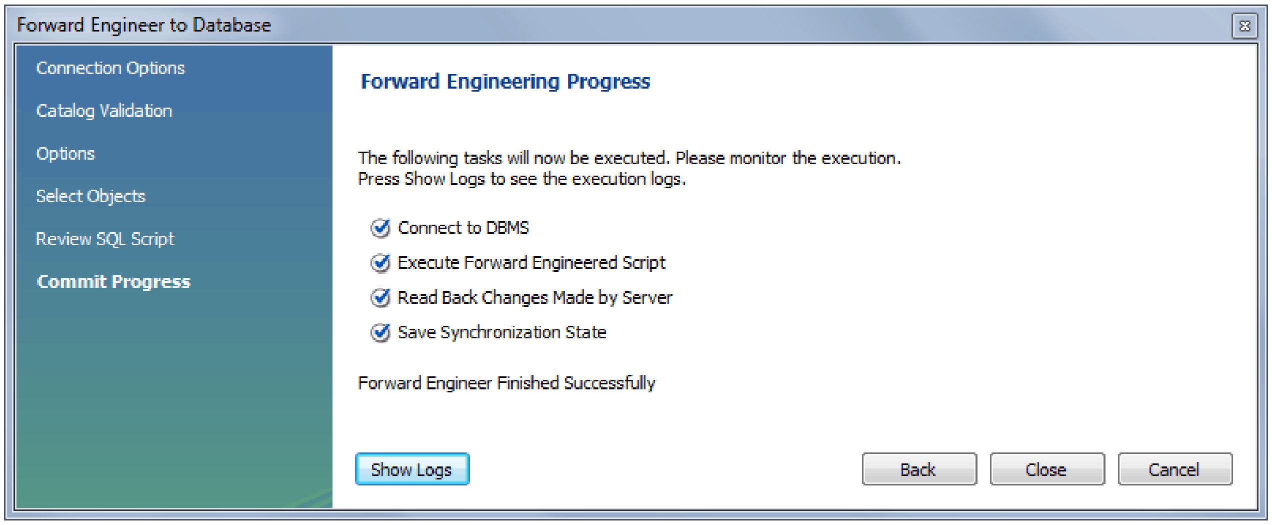
Go Back to the previous page to review the logs.

Back Finish Cancel

Click **Next** to continue if you are satisfied with the generated script.

The next page of the wizard displays the results of the forward engineering process.

Figure 9.39 Forward Engineering Progress



The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Connect to DBMS
✓ Execute Forward Engineered Script
✓ Read Back Changes Made by Server
✓ Save Synchronization State

Forward Engineer Finished Successfully

You can confirm that the script created the schema by connecting to the target MySQL server and issuing a `SHOW DATABASES` statement.

9.4.2 Reverse Engineering

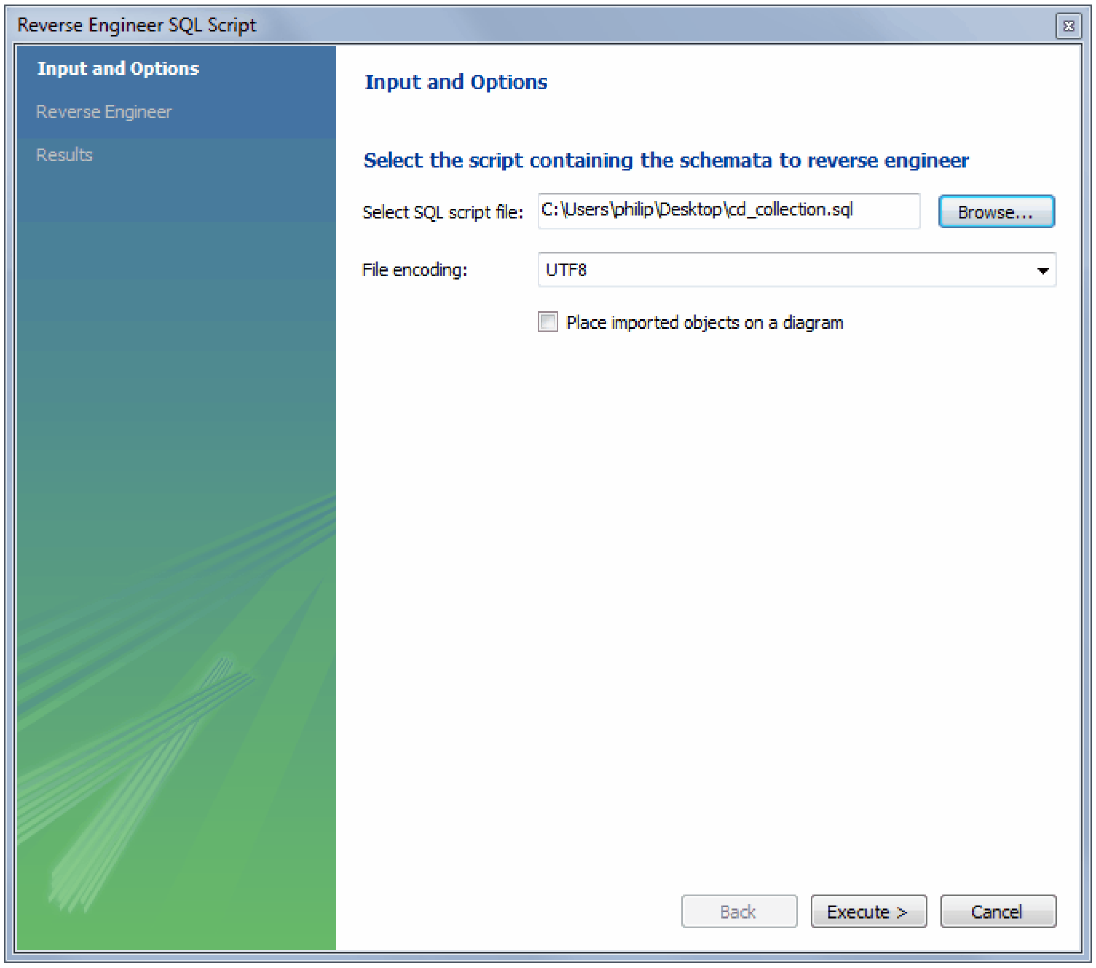
With MySQL Workbench, you can reverse engineer a database using a MySQL create script or you can connect to a live MySQL server and import a single database or a number of databases.

9.4.2.1 Reverse Engineering Using a Create Script

To reverse engineer using a create script, choose the **File**, **Import**, **Reverse Engineer MySQL Create Script...** menu item for a model.

Tables, views, routines, routine groups, indexes, keys, and constraints can be imported from an SQL script file. Objects imported using an SQL script can be manipulated within MySQL Workbench the same as other objects.

Figure 9.40 Reverse Engineer SQL Script: Input



Input and Options

Select the script containing the schemata to reverse engineer

Select SQL script file: C:\Users\philip\Desktop\cd_collection.sql

File encoding: UTF8

Place imported objects on a diagram

- **Select SQL script file**: Open a file open dialog box with the default file type set to an SQL script file, a file with the extension `sql`.
- **File encoding**: Defaults to UTF8.
- **Place imported objects on a diagram**: Also create an EER diagram in MySQL Workbench.

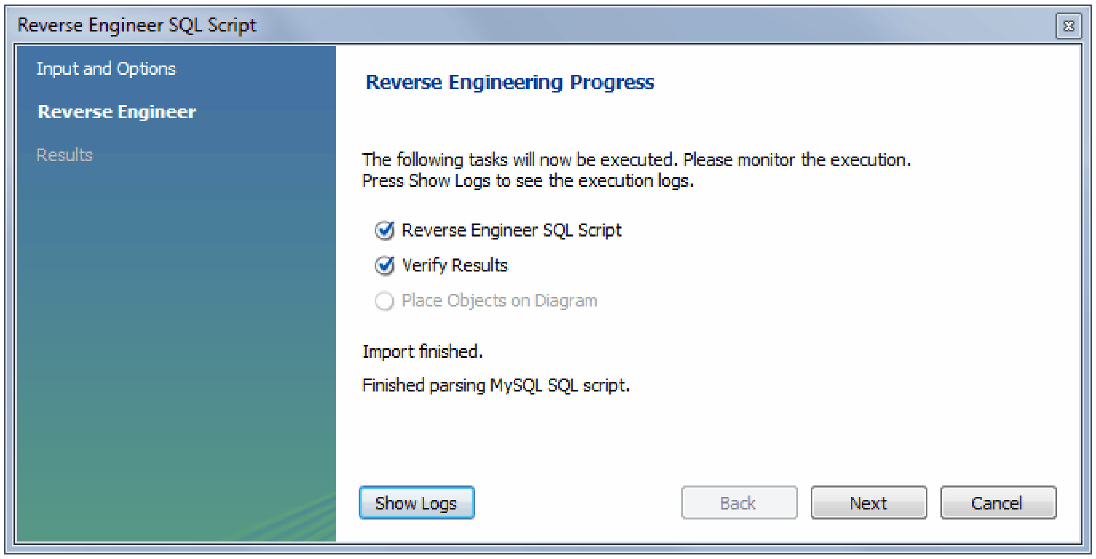
Note

Importing a large number (about 300+) objects could fail to create an EER diagram and instead emit a resource warning with the text "Too many objects are selected for auto placement. Select fewer elements to create the EER diagram." In this case, execute the reverse engineering wizard with this option disabled, manually create the EER diagram, and then import the 300+ objects using the EER diagram catalog viewer.

If your script creates a database, MySQL Workbench creates a new physical schemata tab on the `MySQL Model` page.

Click **Execute** to reverse engineer the SQL script, verify its results, and optionally place the objects in a new EER diagram.

Figure 9.41 Reverse Engineer SQL Script: Execution



Reverse Engineering Progress

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

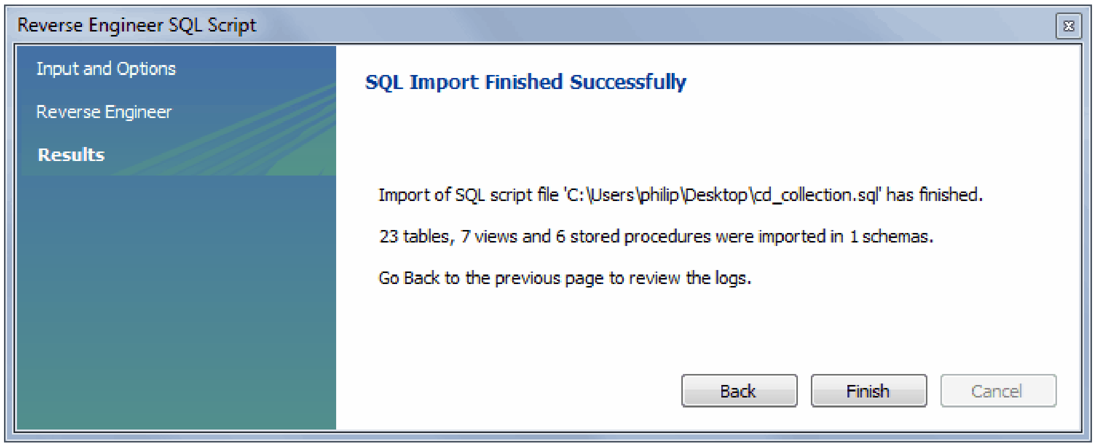
- ✓ Reverse Engineer SQL Script
- ✓ Verify Results
- Place Objects on Diagram

Import finished.

Finished parsing MySQL SQL script.

Click **Next** to view a summary of the results, and then **Finish** to close the wizard.

Figure 9.42 Reverse Engineer SQL Script: Results



SQL Import Finished Successfully

Import of SQL script file 'C:\Users\philip\Desktop\cd_collection.sql' has finished.

23 tables, 7 views and 6 stored procedures were imported in 1 schemas.

Go Back to the previous page to review the logs.

Before exiting MySQL Workbench, be sure to save the schema. Choose the **File**, **Save** menu item and the reverse-engineered database will be saved as a MySQL Workbench file with the extension `mwb`.

See Section 9.3.3, "Importing a Data Definition SQL Script", for a tutorial on reverse engineering the `sakila` database.

Creating a DDL script

You can create a data definition (DDL) script by executing the **`mysqldump`** ***`db_name`*** **`--no-data >`** ***`script_file.sql`*** command. Using the `--no-data` option ensures that the script contains only DDL statements. However, if you are working with a script that also contains DML statements you need not remove them; they will be ignored.

Note

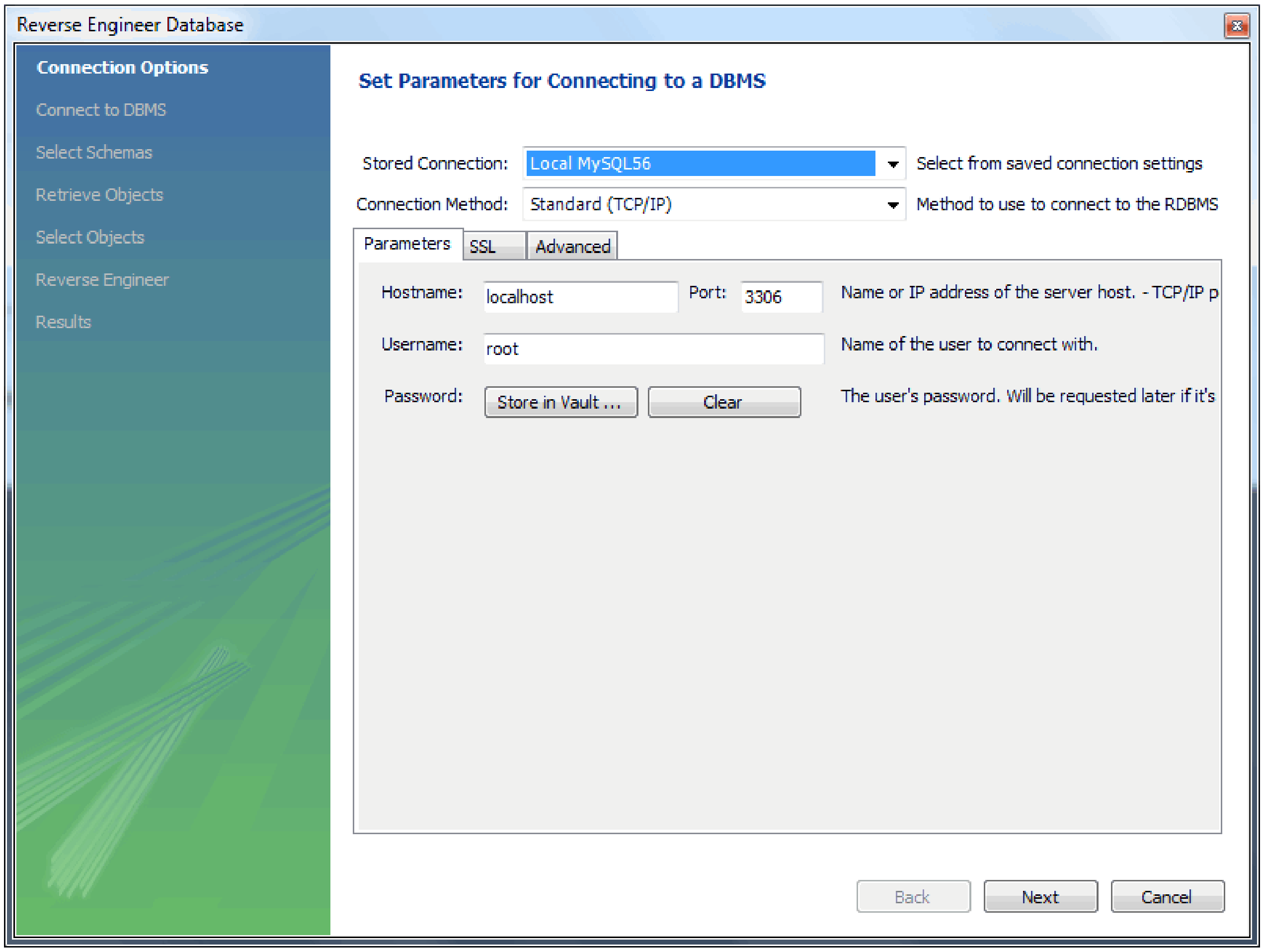
If you plan to redesign a database within MySQL Workbench and then export the changes, be sure to retain a copy of the original DDL script. You will need the original script to create an ALTER script. For more information, see Altering a Schema.

Use the --databases option with mysqldump if you wish to create the database as well as all its objects. If there is no CREATE DATABASE *db_name* statement in your script file, you must import the database objects into an existing schema or, if there is no schema, a new unnamed schema is created.

9.4.2.2 Reverse Engineering a Live Database

To reverse engineer a live database, choose the **Database**, **Reverse Engineer...** menu item from the main menu. This opens the Reverse Engineer Database wizard.

Figure 9.43 Reverse Engineer Database Wizard



Set Parameters for Connecting to a DBMS

Stored Connection: Local MySQL56 — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: localhost Port: 3306 — Name or IP address of the server host. - TCP/IP p

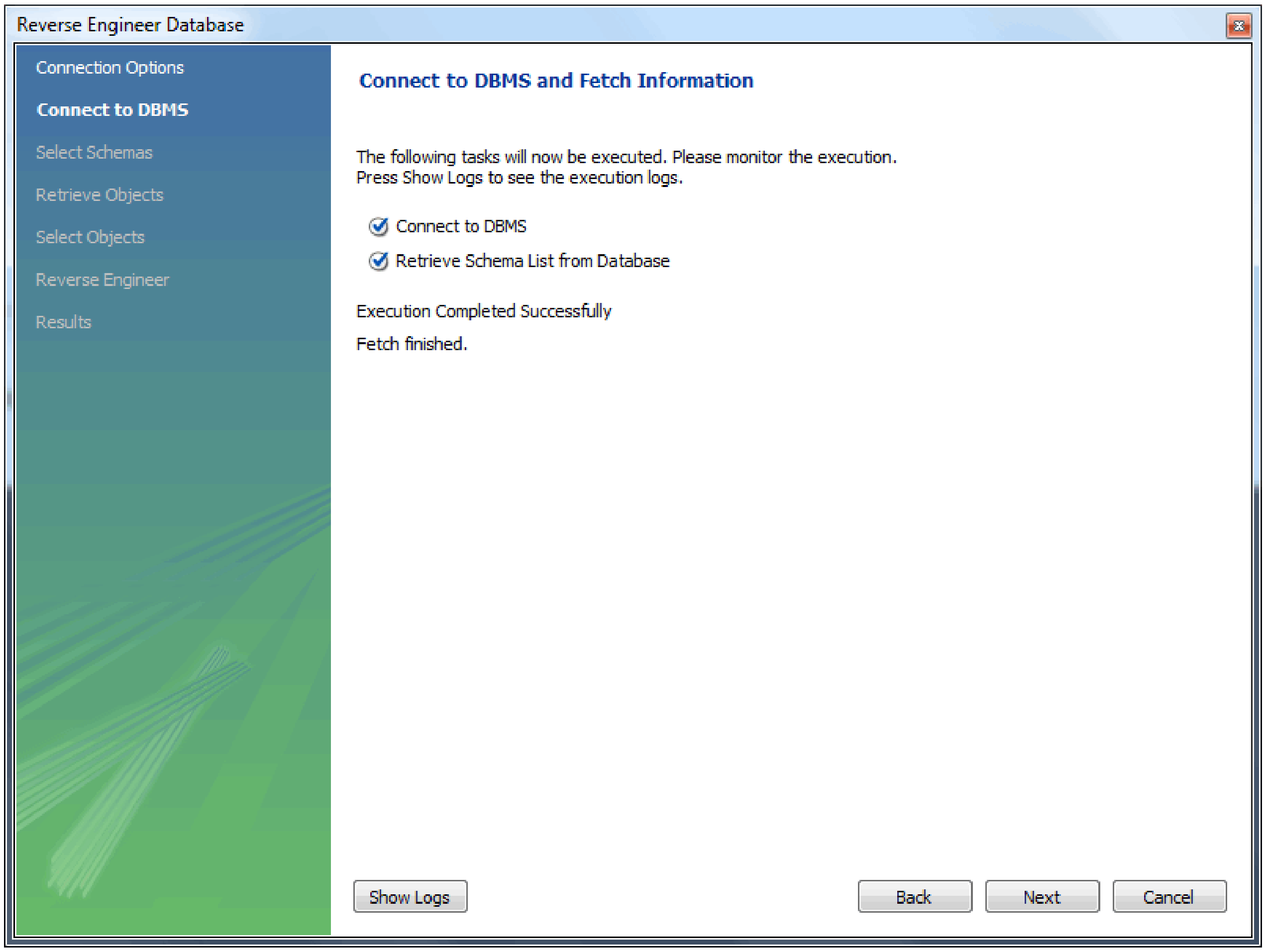
Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's

The first page of the wizard enables you to set up a connection to the live database you wish to reverse engineer. You can set up a new connection or select a previously created stored connection. Typical information required for the connection includes host name, user name and password.

After this information has been entered, or you have selected a stored connection, click the **Next** button to proceed to the next page.

Figure 9.44 Connect to DBMS



Connect to DBMS and Fetch Information

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Connect to DBMS

✓ Retrieve Schema List from Database

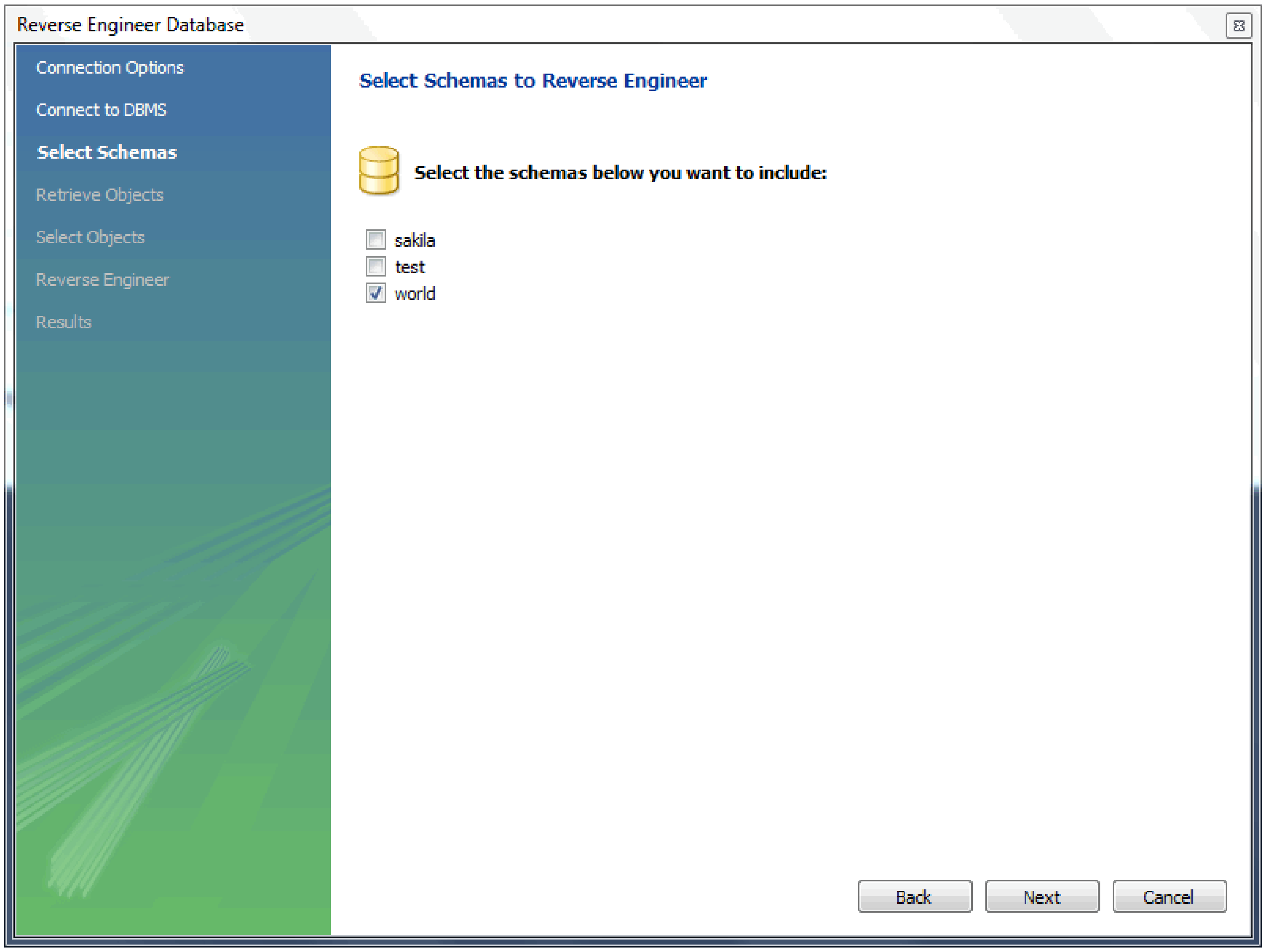
Execution Completed Successfully

Fetch finished.

Review the displayed information to make sure that the connection did not generate errors, then click **Next**.

The next page displays the schemata available on the server. Click the check box or check boxes for any schemata you wish to process.

Figure 9.45 Select Schemas



Select Schemas to Reverse Engineer

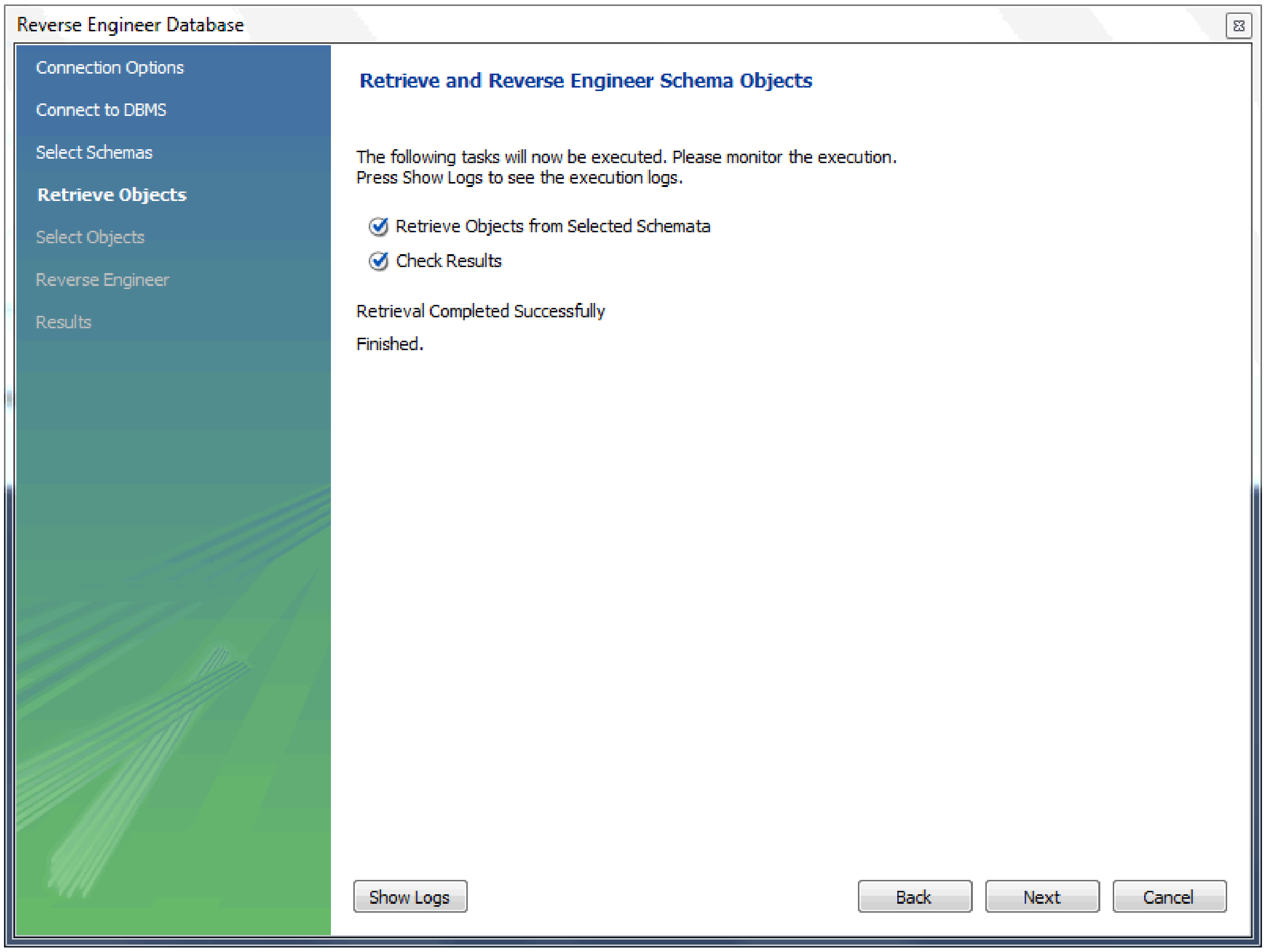
Select the schemas below you want to include:

- [ ] sakila
- [ ] test
- [✓] world

After you have selected the desired schemas, click the **Next** button to continue.

The wizard then displays the tasks it carried out and summarizes the results of the operation.

Figure 9.46 Retrieve Objects



Retrieve and Reverse Engineer Schema Objects

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Retrieve Objects from Selected Schemata

✓ Check Results

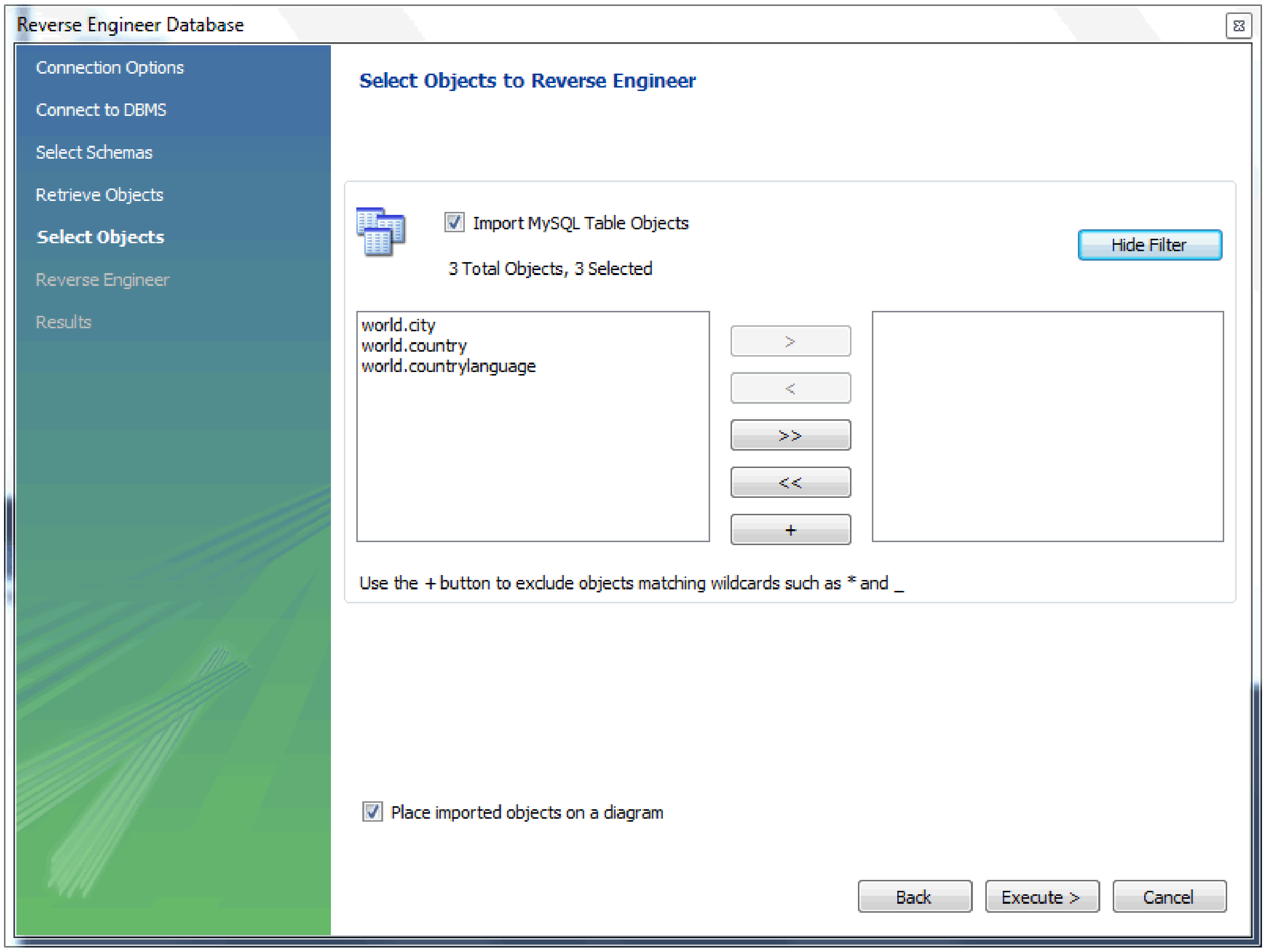
Retrieval Completed Successfully

Finished.

Review the results before clicking **Next** to continue.

The next page is the `Select Objects` page. It has a section for each object type present in the schema (tables, views, routines, and so forth). This page is of special interest if you do not wish to import all the objects from the existing database. It gives you the option of filtering which objects are imported. Each section has a **Show Filter** button. Click this button if you do not want to import all the objects of a specific type. Here is this page with the filter open:

Figure 9.47 Select Objects



☑ Import MySQL Table Objects

3 Total Objects, 3 Selected

world.city
world.country
world.countrylanguage

Use the + button to exclude objects matching wildcards such as * and _

☑ Place imported objects on a diagram

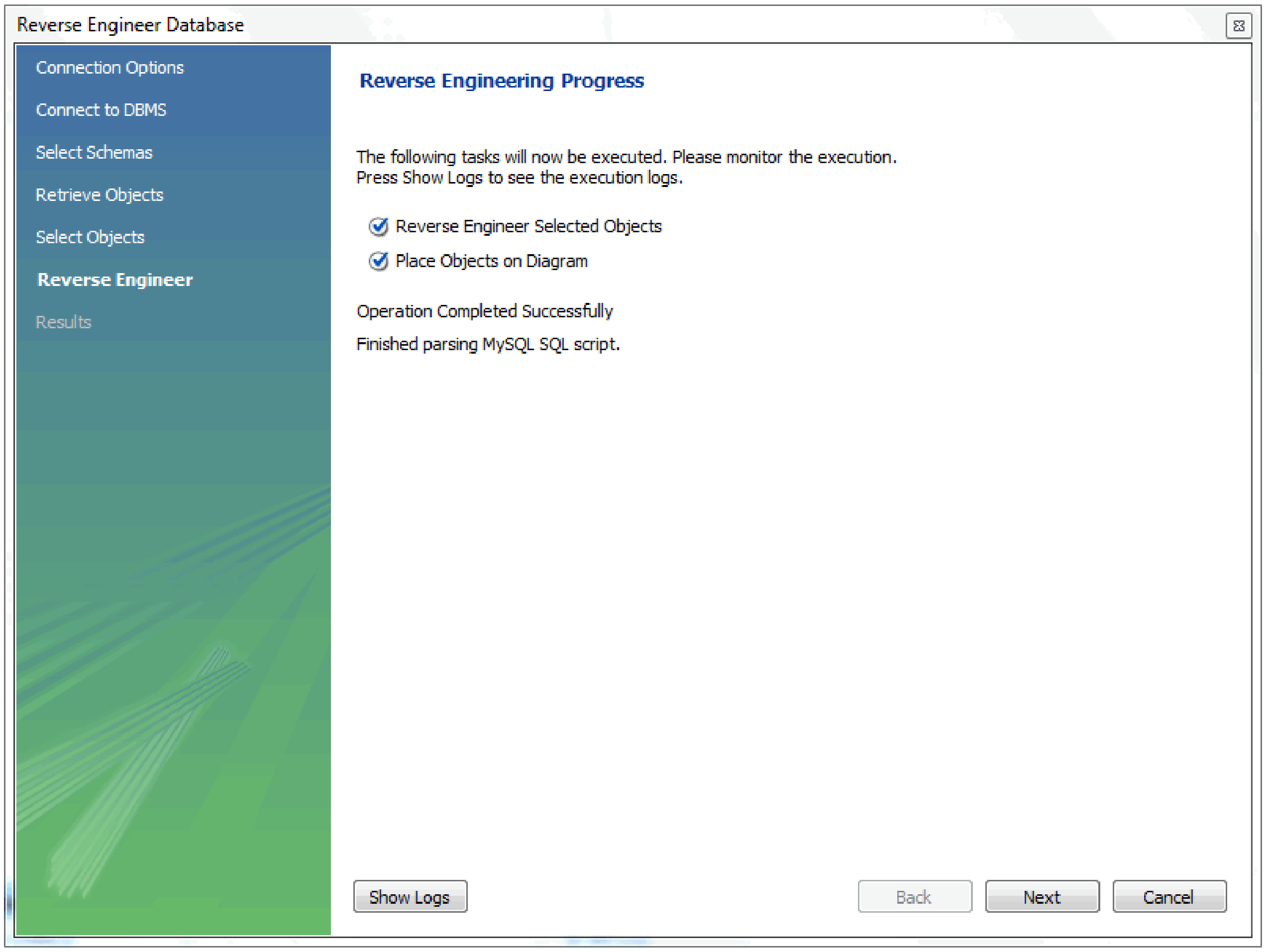
This page enables you to select specific tables for import. Having selected the desired tables, you can optionally hide the filter by clicking the **Hide Filter** button.

The other sections, such as **MySQL Routine Objects**, have similar filters available.

Click **Execute** to continue to the next page.

The wizard then imports objects, displaying the tasks that have been carried out and whether the operation was successful. If errors were generated, you can click the **Show Logs** button to see the nature of the errors.

Figure 9.48 Reverse Engineer Progress



Reverse Engineering Progress

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Reverse Engineer Selected Objects

✓ Place Objects on Diagram

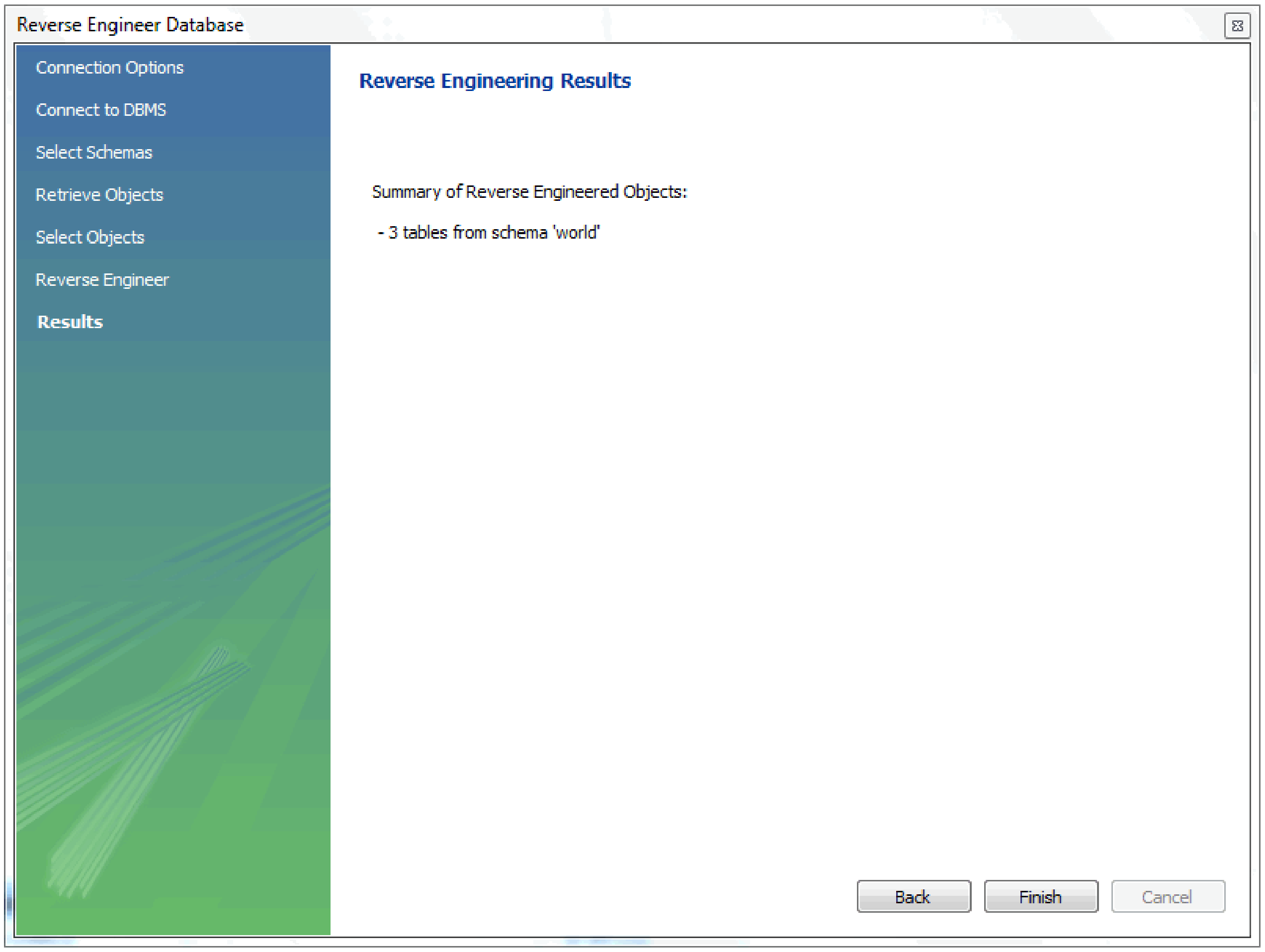
Operation Completed Successfully

Finished parsing MySQL SQL script.

Click **Next** to continue to the next page.

The final page of the wizard provides a summary of the reverse engineered objects.

Figure 9.49 Results



Reverse Engineering Results

Summary of Reverse Engineered Objects:

- 3 tables from schema 'world'

Click **Finish** to exit the wizard.

Before exiting MySQL Workbench be sure to save the schema. Choose the **File**, **Save** menu item to save the reverse-engineered database as a MySQL Workbench file with the extension `mwb`.

Errors During Reverse Engineering

During reverse engineering, the application checks for tables and views that duplicate existing names and disallows duplicate names if necessary. If you attempt to import an object that duplicates the name of an existing object you will be notified with an error message. To see any errors that have occurred during reverse engineering, you can click the button **Show Logs**. This will create a panel containing a list of messages, including any error messages than may have been generated. Click the **Hide Logs** button to close the panel.

If you wish to import an object with the same name as an existing object, rename the existing object before reverse engineering.

If you import objects from more than one schema, there will be a tab in the `Physical Schemata` section of the `MySQL Model` page for each schema imported.

You cannot reverse engineer a live database that has the same name as an existing schema. If you wish to do this, first rename the existing schema.

9.5 Schema Synchronization and Comparison

Database change management is a complex process that involves maintaining different versions of database schemas and manually modifying existing databases. To help with this administration task, MySQL Workbench includes schema synchronization and comparison utilities. You can compare two live databases, two models, or models with live databases, to visually see the differences and optionally perform a synchronization routine.

9.5.1 Database Synchronization

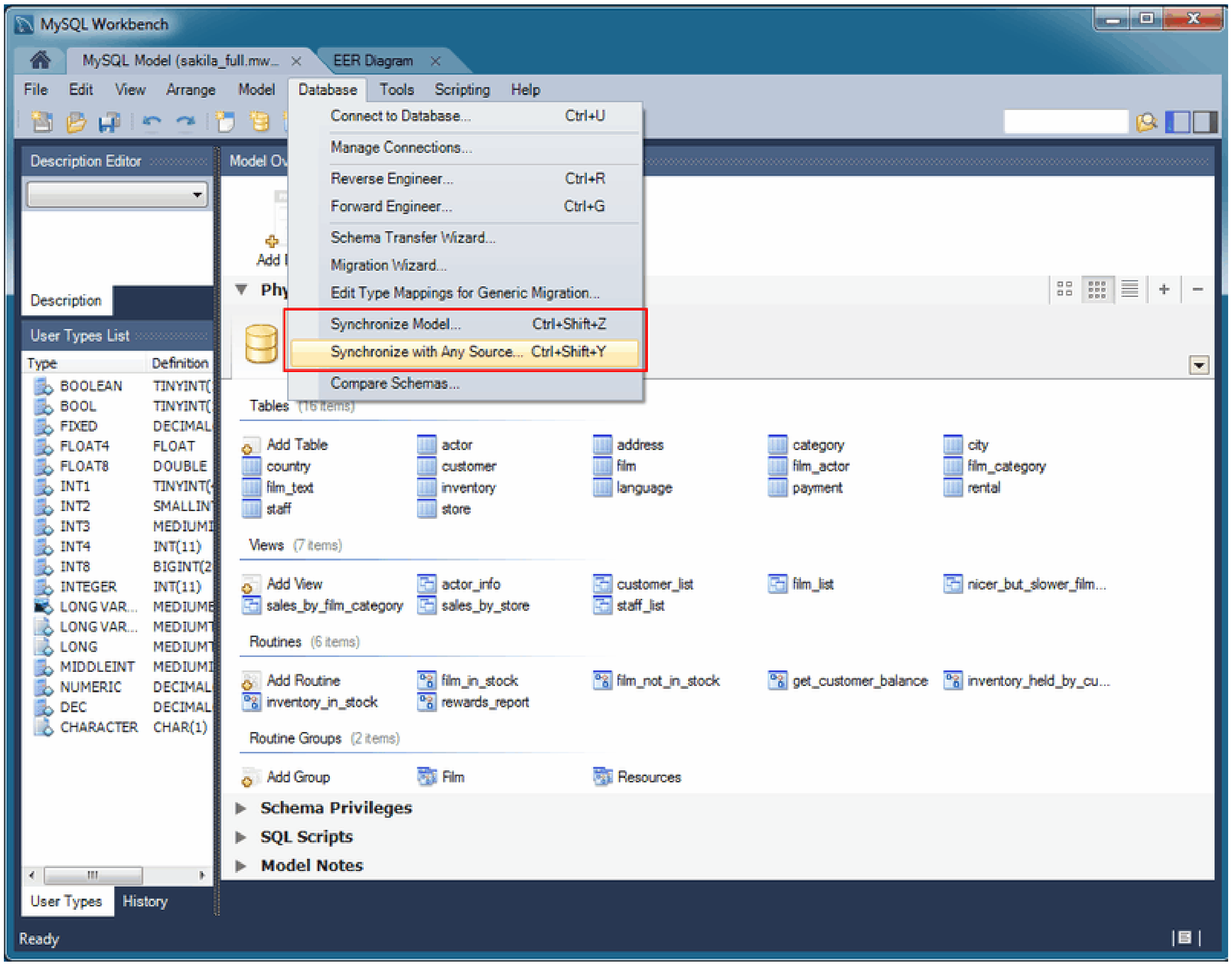
Synchronize data between models, databases, and SQL files. These three types can be the target (destination), source, or both. You can also select/deselect individual objects and modify their direction during the synchronization. For example, synchronize tables from a model to your database, other tables from your database to your model, and skip a few tables, all during the same synchronization process.

Note

Be aware that backward incompatible MySQL syntax changes are introduced over time, so for this reason it is important to set the **Default Target MySQL Version** modeling preference according to your needs. For example, exporting results from a MySQL 5.7 target might yield invalid syntax when executed against MySQL 5.1. See also Section 3.2.4, “Modeling Preferences”.

To start, select **Synchronize With Any Source** from the **Database** navigation menu. Alternatively, select **Synchronize Model** to open the same wizard that defaults to a model. A Model or EER diagram must be selected for these synchronization options to be present under the **Database** navigation menu.

Figure 9.50 Start the Synchronization Wizard


MySQL Workbench
MySQL Model (sakila_full.mw...
EER Diagram
File Edit View Arrange Model Database Tools Scripting Help
Connect to Database... Ctrl+U
Manage Connections...
Reverse Engineer... Ctrl+R
Forward Engineer... Ctrl+G
Schema Transfer Wizard...
Migration Wizard...
Edit Type Mappings for Generic Migration...
Synchronize Model... Ctrl+Shift+Z
Synchronize with Any Source... Ctrl+Shift+Y
Compare Schemas...
Description Editor
Description
User Types List
Tables (16 items)
Views (7 items)
Routines (6 items)
Routine Groups (2 items)
Schema Privileges
SQL Scripts
Model Notes
User Types
History
Ready

Caution

Because MySQL databases correspond to directories within the data directory, you must consider case sensitivity for database, table, and trigger names, which follow the case sensitivity rules of the underlying file system for your operating system. Synchronizing models with objects that differ in case may lead to MySQL Workbench producing a `DROP` statement for that object, before recreating it as lowercase. For more information, see Identifier Case Sensitivity

Workarounds include using a consistent convention, where the most portable code uses lower case database and table names. Or a temporary workaround is to delete the `DROP SCHEMA IF EXISTS` line from the generated query.

MySQL Workbench enables control over objects to synchronize, and the direction of synchronization for each object. Synchronization options include:

- Specify all or specific tables and objects to synchronize
- Synchronize both the model and live database, or only update one or the other (unidirectional or bidirectional)
- Optionally update from or to an SQL script file

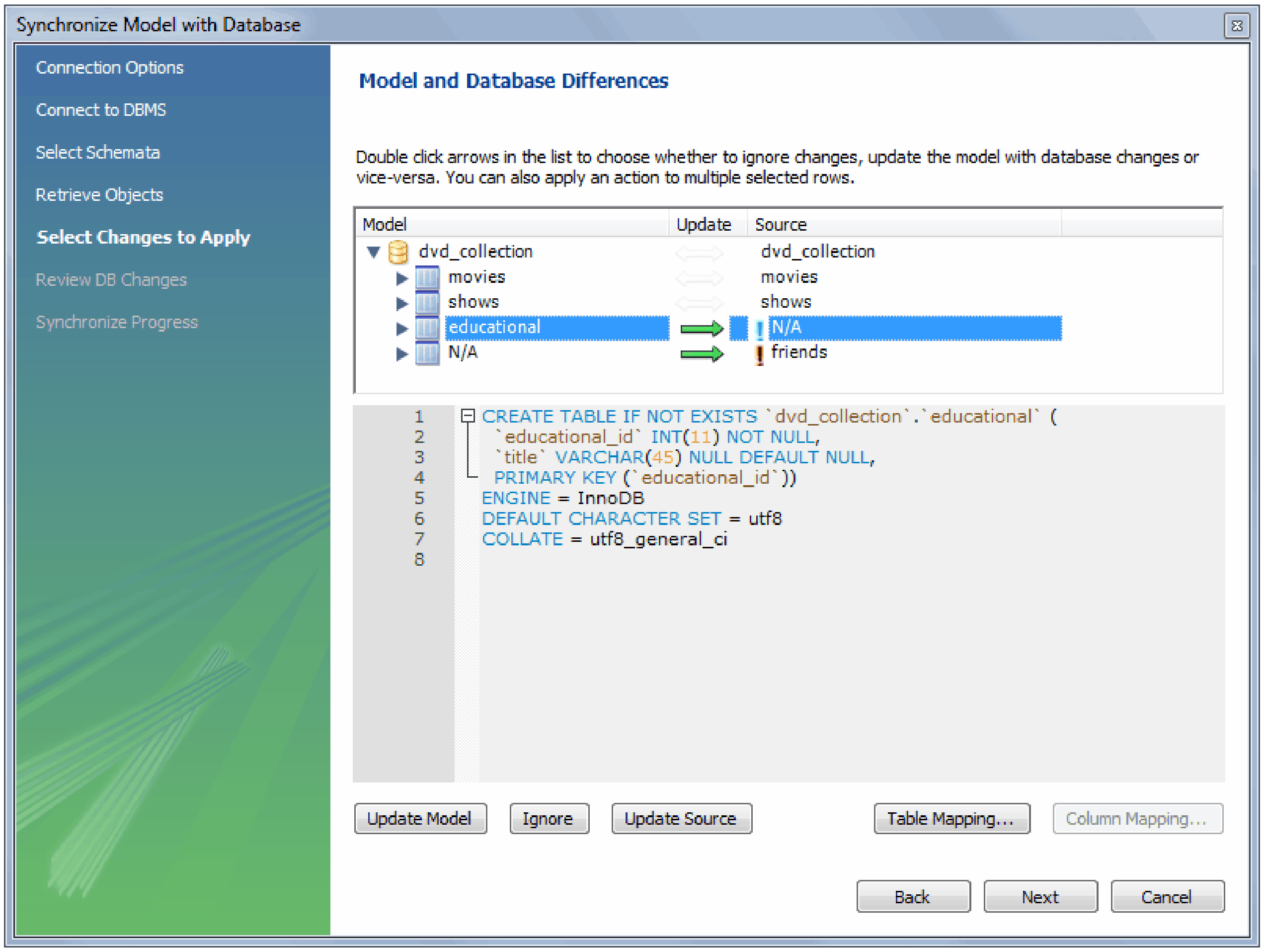
- Instead of executing the synchronization, you may generate an **ALTER Script File** to later perform the appropriate updates
- Fine-tune how the synchronization will be performed by choosing the direction of each individual object, or configuring particular objects to be ignored

There are two similar database synchronization wizards available from the **Database** menu. The simpler **Synchronize Model** wizard, and the more flexible **Synchronize with Any Source** wizard. The descriptions below apply to both, unless explicitly told otherwise.

Synchronize Model (with database)

To start the wizard, open a model and select **Database**, **Synchronize Model** from the main menu. Follow the steps until you reach the **Select Changes to Apply** step:

Figure 9.51 Model and Database Differences



Double click arrows in the list to choose whether to ignore changes, update the model with database changes or vice-versa. You can also apply an action to multiple selected rows.

Model	Update	Source
dvd_collection		dvd_collection
movies		movies
shows		shows
educational	⇒	N/A
N/A	⇒	friends

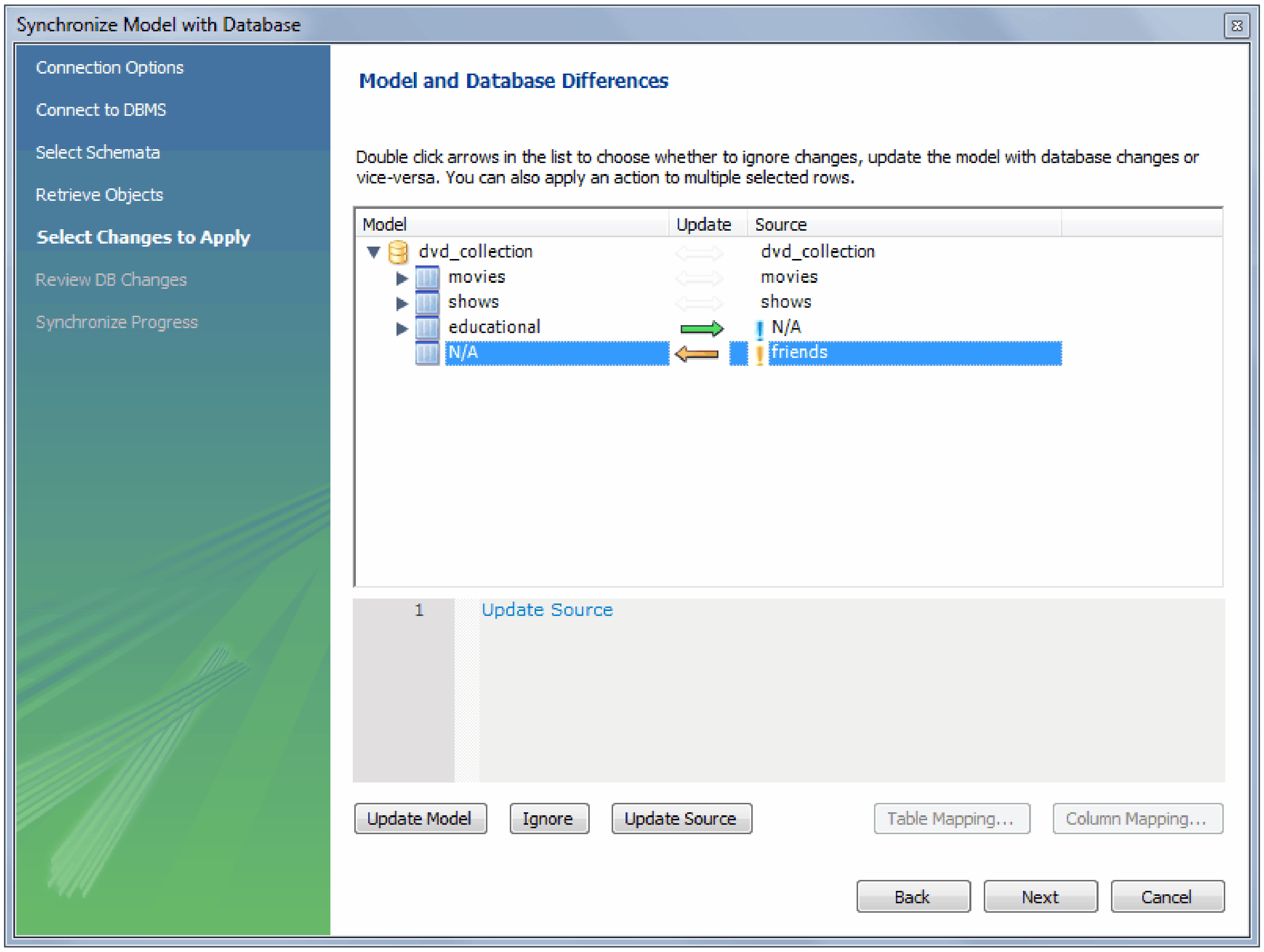
```
CREATE TABLE IF NOT EXISTS `dvd_collection`.`educational` (
  `educational_id` INT(11) NOT NULL,
  `title` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`educational_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci
```

In the preceding example, the live database and model both have `movies shows` tables. In the MySQL Workbench, an additional table, `educational`, has been created in the model, but it lacks an equivalent in the live database. Further, `friends` exists in the live database, but it is not in the model. By default, the actions will synchronize the database with the model, so in this example the `educational` table will be added to the source, and the `friends` table will be removed from the source.

As described in the GUI, double-clicking the arrows will alternate between the **Update Model**, **Ignore**, and **Update Source** actions. You may also select a row and click one of the three action buttons. Also note that clicking on a row will reveal the associated SQL statement, as shown in the screenshot above.

The next example shows how the direction of synchronization can be changed.

Figure 9.52 Controlling Synchronization Direction



Model and Database Differences

Double click arrows in the list to choose whether to ignore changes, update the model with database changes or vice-versa. You can also apply an action to multiple selected rows.

Model	Update	Source
dvd_collection		dvd_collection
movies		movies
shows		shows
educational	→	N/A
N/A	←	friends

```
1    Update Source
```

In this case, the synchronization direction has been changed so that rather than the default action of `friends` being dropped from the live database, it will be incorporated into the MySQL Workbench model. As before, `educational` table will be added to the live (source) database.

The three actions available actions are:

- **Update Model**: Causes the selected changes to be applied to the model, from the live database.
- **Ignore**: Causes the changes to be ignored. No synchronization will take place for those changes. This is designated with a double arrow that is crossed out.
- **Update Source**: Causes the changes to be applied only to the live database.

Clicking **Table Mapping** offers additional mapping options:

Figure 9.53 Table Mapping

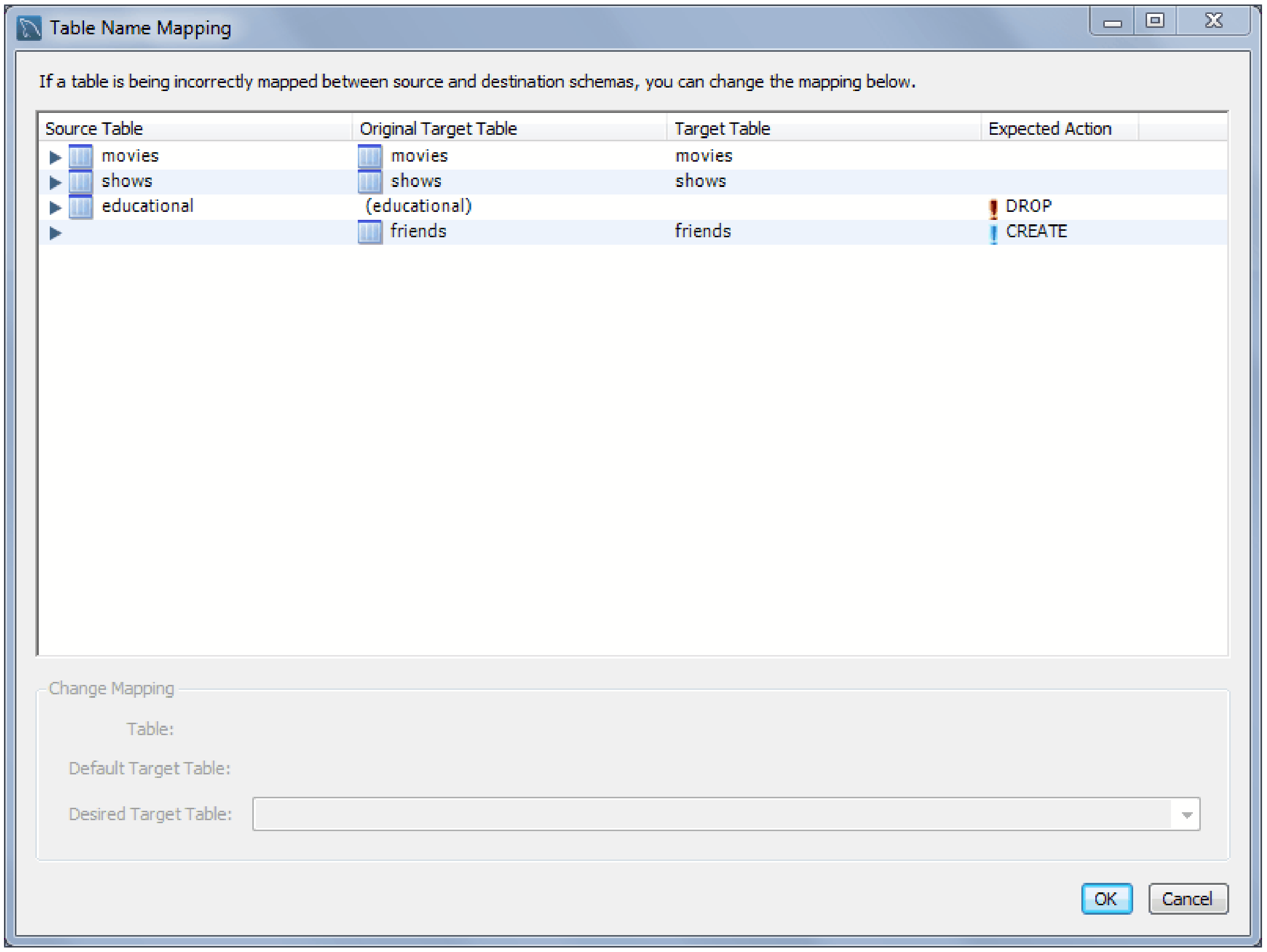


Table Name Mapping

If a table is being incorrectly mapped between source and destination schemas, you can change the mapping below.

Source Table	Original Target Table	Target Table	Expected Action
movies	movies	movies	
shows	shows	shows	
educational	(educational)		DROP
	friends	friends	CREATE

Change Mapping

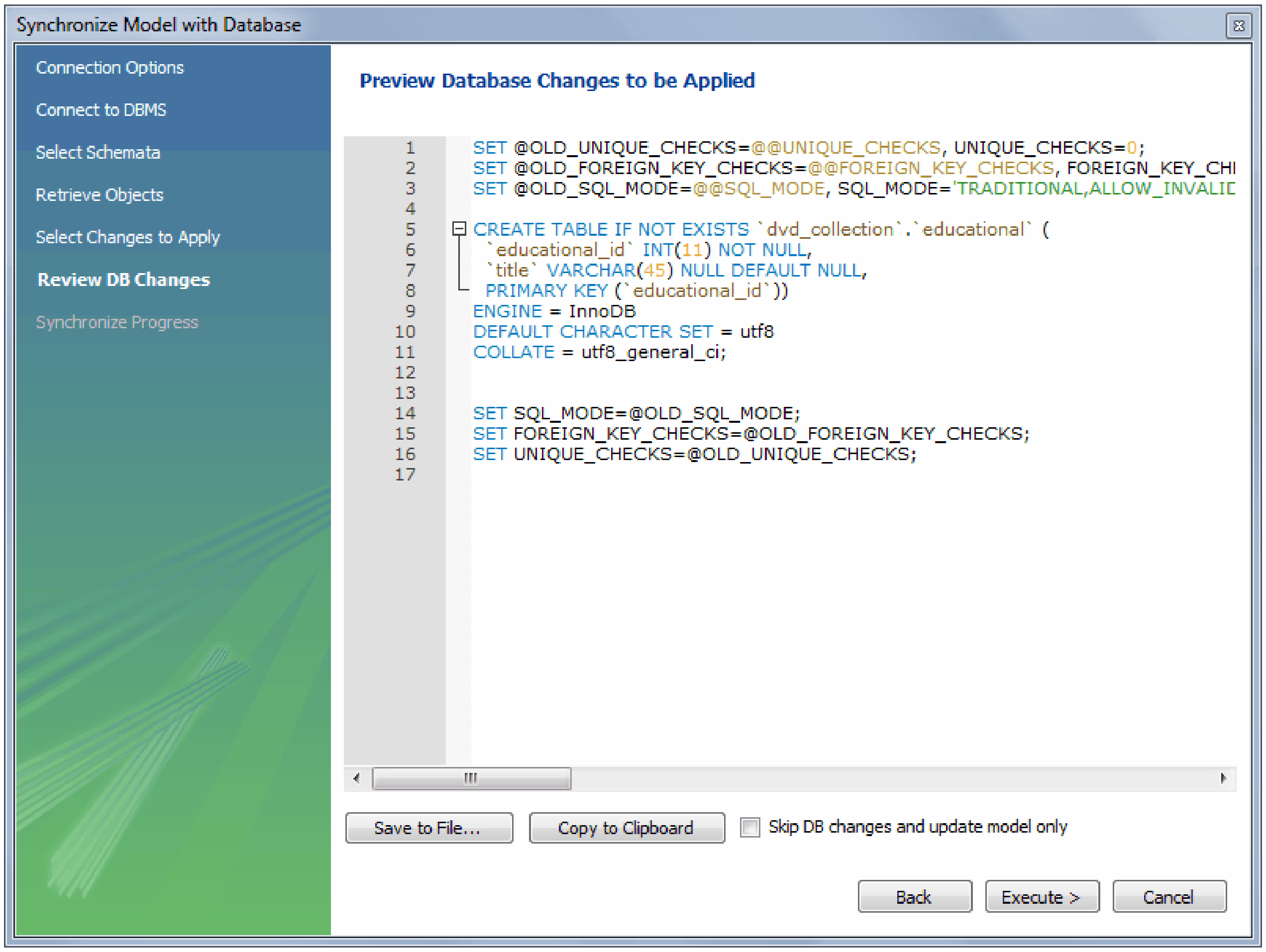
Table:

Default Target Table:

Desired Target Table:

Pressing **Next** will reveal the SQL statement to perform the configured model and live database (source) synchronization:

Figure 9.54 Previewing The Synchronization SQL Statement



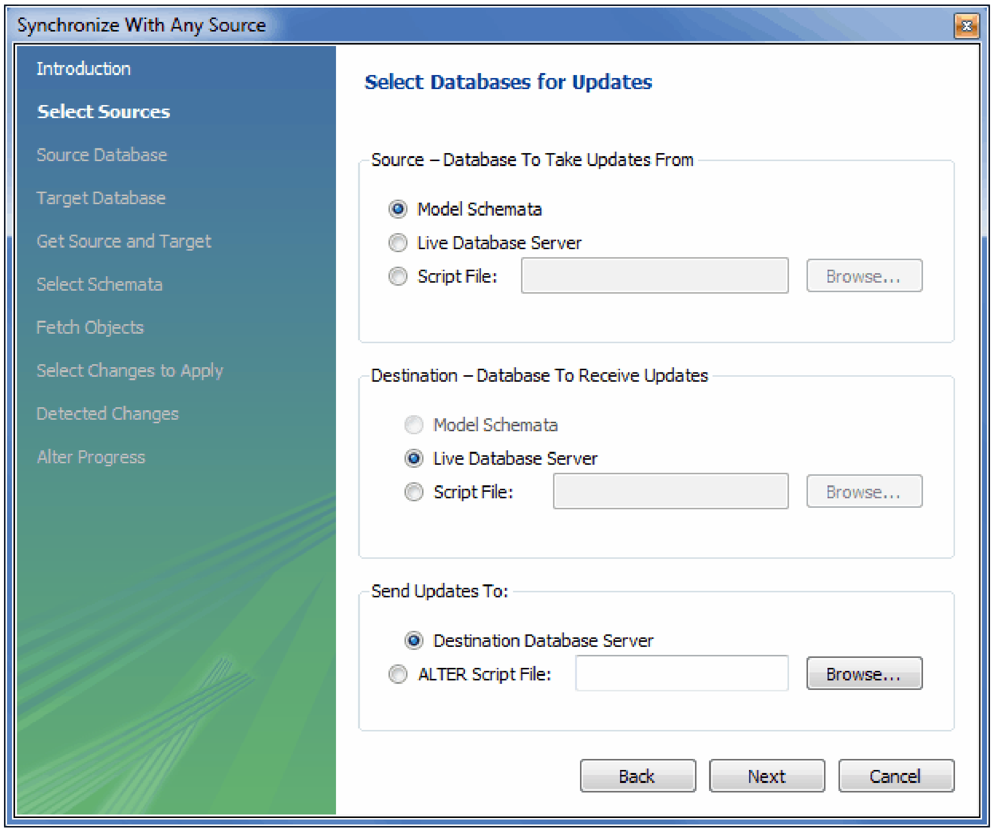
```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHI
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALIC

CREATE TABLE IF NOT EXISTS `dvd_collection`.`educational` (
  `educational_id` INT(11) NOT NULL,
  `title` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`educational_id`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_general_ci;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

You may now save the SQL statement to a file or the clipboard, or execute the SQL statement. If you choose to execute the change in MySQL Workbench, then you may optionally choose to skip "DB changes" so that only your model is altered.

Synchronize With Any Source

To start the wizard, open a model and select **Database**, **Synchronize With Any Source** from the main menu. The steps are similar to the Synchronize Model wizard, but with additional options to create and/or use SQL script files. See the **Select Sources** page:

Figure 9.55 Synchronize With Any Source: Select Sources



Select Databases for Updates

Source – Database To Take Updates From
- [✓ Model Schemata]
- Live Database Server
- Script File:

Destination – Database To Receive Updates
- Model Schemata
- [✓ Live Database Server]
- Script File:

Send Updates To:
- [✓ Destination Database Server]
- ALTER Script File:

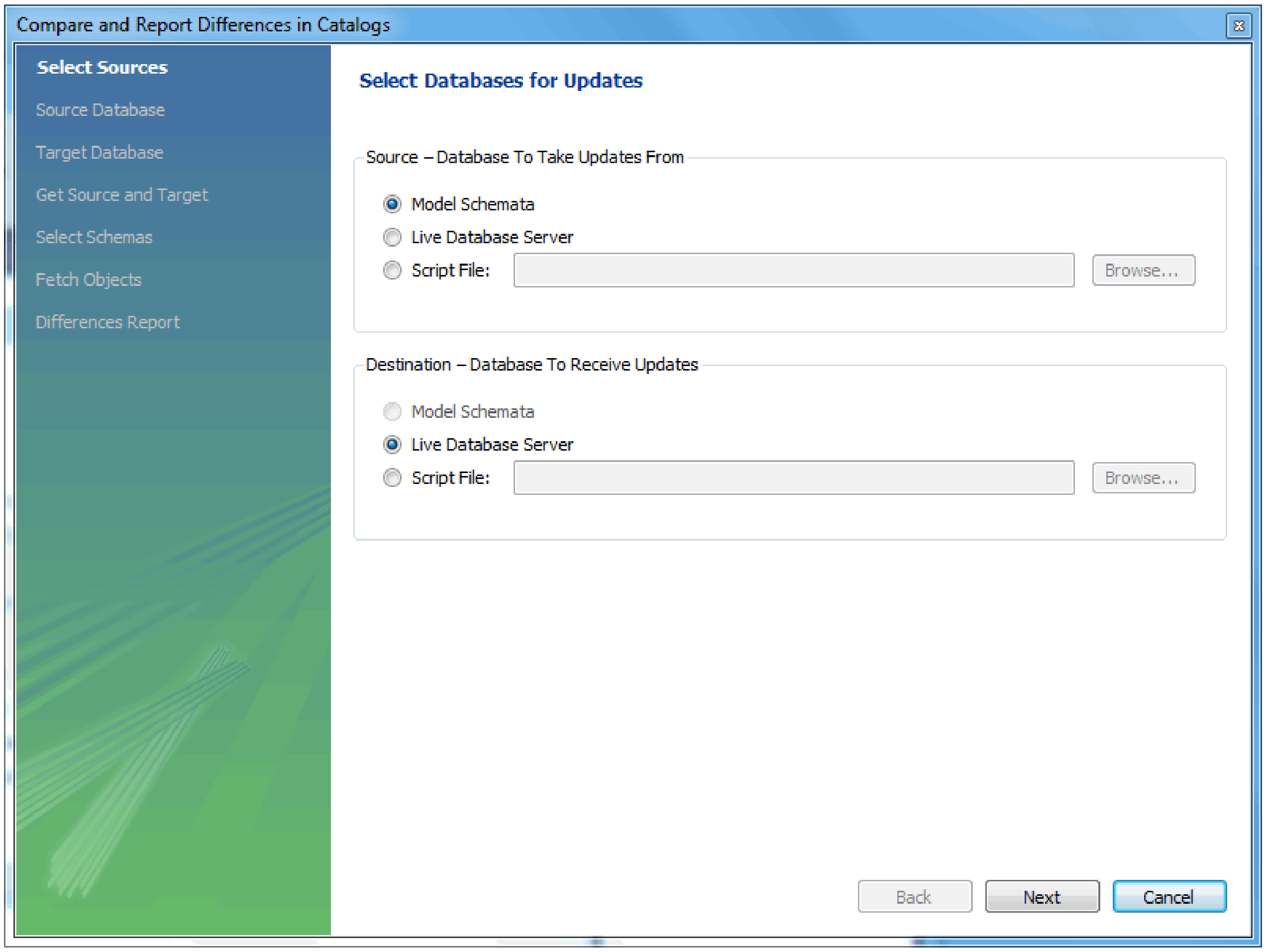
Notice how the source and destination types can be altered. The steps that follow depend on these source and destination types, and the **Synchronize Model** describes the basic functionality of this wizard.

9.5.2 Compare and Report Differences in Catalogs

This facility enables you to create a report detailing the differences between your MySQL Workbench model, and a live database or script. Choose **Database**, **Compare Schemas** from the main menu to run the **Compare and Report Differences in Catalogs** wizard.

The first step in the wizard is to specify which catalogs to compare. For example, you may wish to compare your live database against your current MySQL Workbench model.

Figure 9.56 Catalog Sources Selection



Select Databases for Updates

Source – Database To Take Updates From

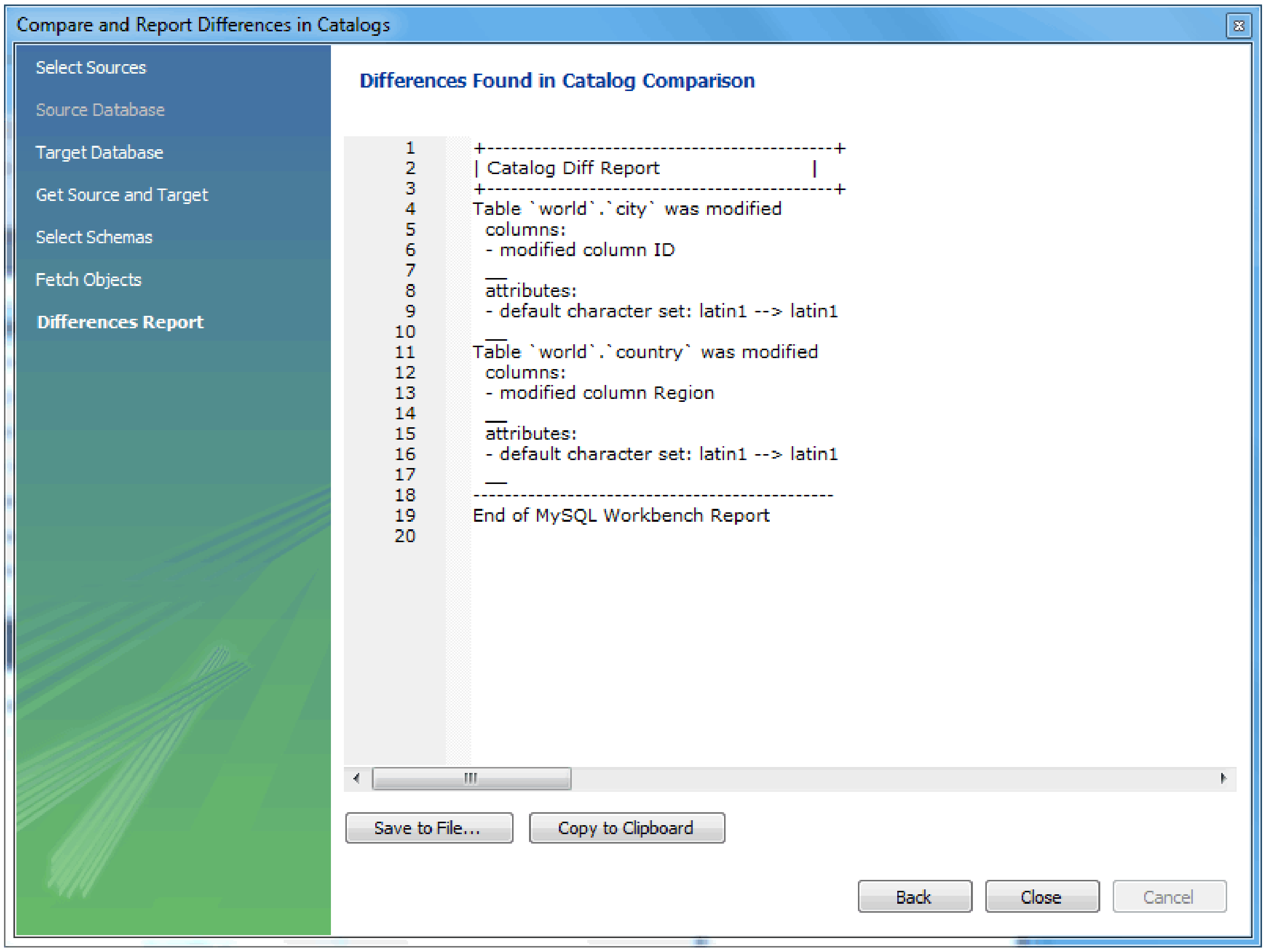
- [✓ Model Schemata]
- Live Database Server
- Script File:

Destination – Database To Receive Updates

- Model Schemata
- [✓ Live Database Server]
- Script File:

You then proceed through the wizard, providing connection information if accessing a live database. The wizard then produces a catalog diff report showing the differences between the compared catalogs.

Figure 9.57 Catalog Differences Report



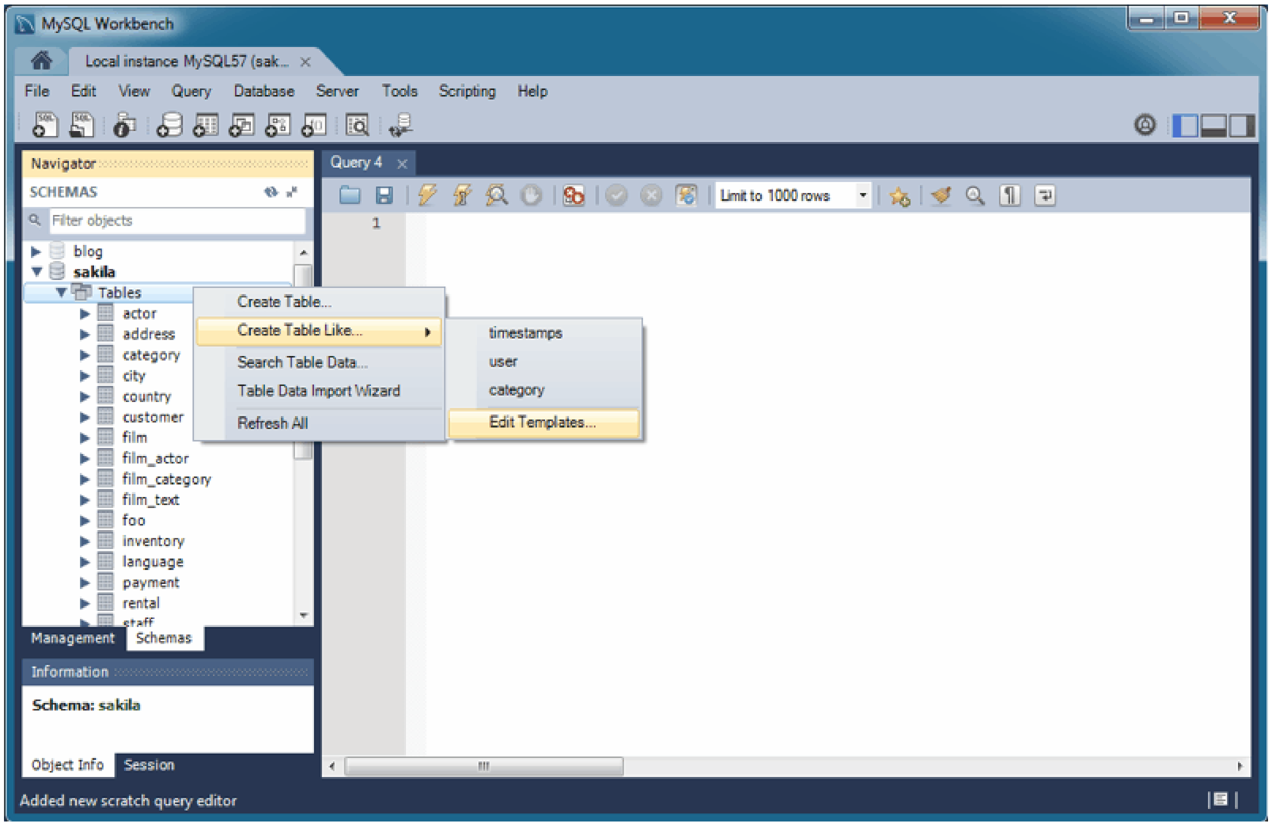
```
+--------------------------------------------+
| Catalog Diff Report                        |
+--------------------------------------------+
Table `world`.`city` was modified
  columns:
  - modified column ID
  __
  attributes:
  - default character set: latin1 --> latin1
  __
Table `world`.`country` was modified
  columns:
  - modified column Region
  __
  attributes:
  - default character set: latin1 --> latin1
  __
--------------------------------------------
End of MySQL Workbench Report
```

9.6 Table Templates

Define table templates with commonly used columns and settings to create new tables from either a live connection or while creating an EER model.

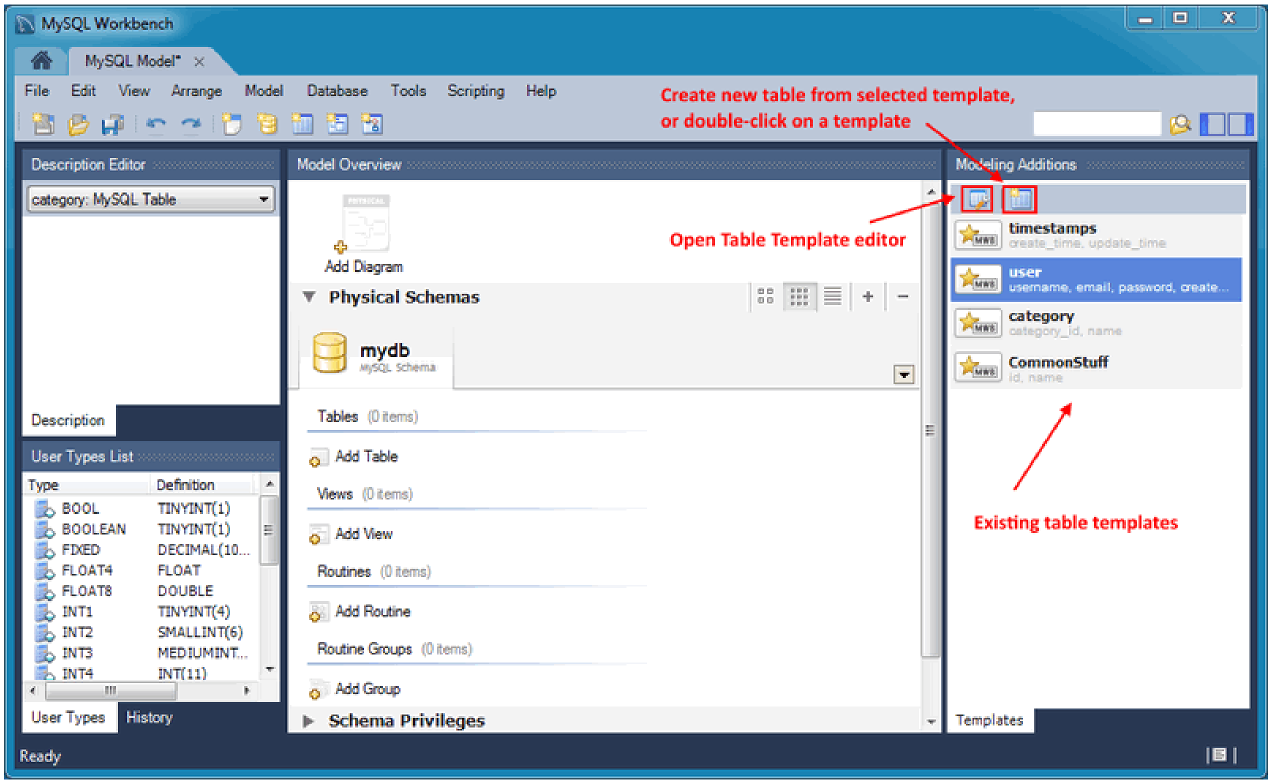
From the SQL editor, select **Create Table Like** from the **Tables** context menu.

Figure 9.58 New Table Template: SQL Editor



Or while modeling, click the "Open the table template editor" icon on the right sidebar under **Modeling Additions**.

Figure 9.59 New Table Template: Modeling



After opening the **Table Templates** manager, make the desired adjustments and click **Apply** to commit the changes.

Figure 9.60 Table Templates Manager

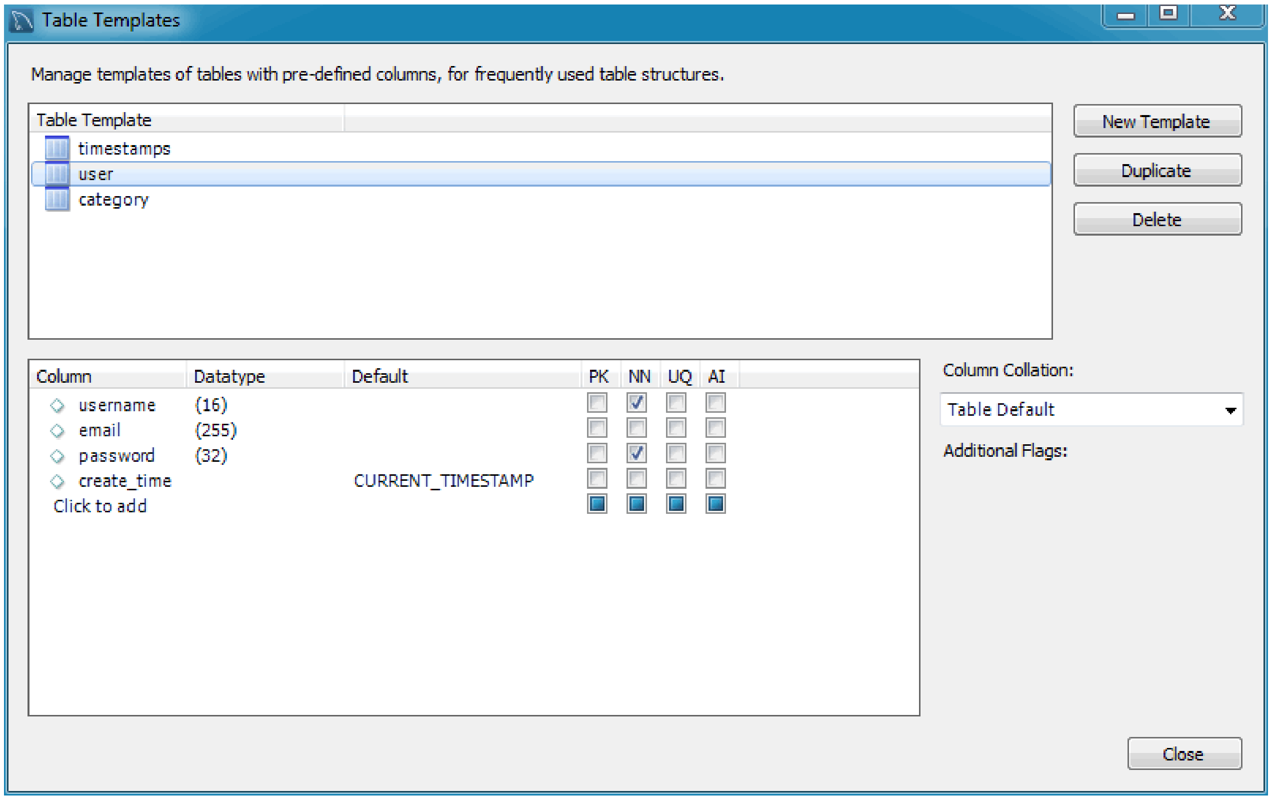


Table Template
timestamps
user
category

Column	Datatype	Default	PK	NN	UQ	AI
username	(16)		☐	✓	☐	☐
email	(255)		☐	☐	☐	☐
password	(32)		☐	✓	☐	☐
create_time		CURRENT_TIMESTAMP	☐	☐	☐	☐
Click to add			■	■	■	■

To open an existing template from the SQL editor, hover over the **Create Table Like** context menu and select the desired table template. For modeling, double-click on a template in the right modeling sidebar.

9.7 Customizing DBDoc Model Reporting Templates

This section provides an overview of creating and modifying DBDoc Model Reporting templates, as used by MySQL Workbench.

The MySQL Workbench DBDoc Model Reporting system is based on the Google Template System. This discussion does not attempt to explain the Google Template System in detail. For a useful overview of how the Google Template System works, see the Google document, How To Use the Google Template System.

The templates employed by the DBDoc Model Reporting system are text files that contain *Markers*. These text files are processed by the template system built into MySQL Workbench, and the markers replaced by actual data. The output files are then generated. It is these output files, typically HTML or text, that are then viewed by the user.

Markers can be of six types:

- Template Include
- Comment
- Set delimiter
- Pragma

- Variable
- Section start and Section end

The last two are the most commonly used in MySQL Workbench templates and these important markers are briefly described in the following sections.

- **Variables**

  Variable usage in the templates is straightforward. Variables denoted by markers in the template file are replaced by their corresponding data prior to the generated output file. The mapping between variables and their corresponding data is stored by MySQL Workbench in a *data dictionary*. In the data dictionary, the variable name is the *key* and the variable's corresponding data is the *value*. MySQL Workbench builds the data dictionaries and fills it with the data contained in the processed model.

  By way of example, the following code snippet shows part of a template file:

  ```
  Total number of Schemata: {{SCHEMA_COUNT}}
  ```

  In the generated output file, the variable `{{SCHEMA_COUNT}}` is replaced by the number of schemata in the model:

  ```
  Total number of Schemata: 2
  ```

  A variable can appear multiple times in the template file.

- **Sections**

  Sections are used to perform iteration in the templates. When MySQL Workbench exchanges the variables in a section for data, it does so iteratively, using all data in the data dictionary in which the variable is defined. MySQL Workbench builds the data dictionaries according to the model currently being processed.

  Consider the following code snippet:

  ```
  {{#SCHEMATA}}
  Schema: {{SCHEMA_NAME}}
  {{/SCHEMATA}}
  ```

  In the preceding snippet, the section start and end are indicated by the `{{#SCHEMATA}}` and `{{/SCHEMATA}}` markers. When MySQL Workbench processes the template, it notes the section and iterates it until the variable data for `{{SCHEMA_NAME}}` in the corresponding data dictionary is exhausted. For example, if the model being processed contains two schemata, the output for the section might resemble the following:

  ```
  Schema: Airlines
  Schema: Airports
  ```

Data Dictionaries

It is important to understand the relationship between sections and data dictionaries in more detail. In a data dictionary the *key* for a variable is the variable name, a marker. The variable *value* is the variable's data. The entry for a section in a data dictionary is different. For a section entry in a data dictionary, the key is the section name, the marker. However, the value associated with the key is a list of data dictionaries. In MySQL Workbench each section is usually associated with a data dictionary. You can think of a section as *activating* its associated dictionary (or dictionaries).

When a template is processed, data dictionaries are loaded in a hierarchical pattern, forming a tree of data dictionaries. This is illustrated by the following table.

Table 9.1 Data Dictionaries Tree

Data Dictionary	**Loads Data Dictionary**
MAIN	SCHEMATA
SCHEMATA	TABLES, COLUMNS (Detailed is true), FOREIGN_KEYS (Detailed is true), INDICES (Detailed is true)
TABLES	REL_LISTING, INDICES_LISTING, COLUMNS_LISTING, TABLE_COMMENT_LISTING, DDL_LISTING
COLUMNS_LISTING	COLUMNS (Detailed is false)
REL_LISTING	REL (Detailed is false)
INDICES_LISTING	INDICES (Detailed is false)

The root of the tree is the *main* dictionary. Additional dictionaries are loaded from the root to form the dictionary tree.

Note

If a template has no sections, any variables used in the template are looked up in the main dictionary. If a variable is not found in the main dictionary (which can be thought of as associated with the default, or main, section), no data is generated in the output file for that marker.

Evaluation of variables

The tree structure of the data dictionaries is important with respect to variable evaluation. As variables are defined in data dictionaries, their associated values have meaning only when that particular data dictionary is active, and that means when the section associated with that data dictionary is active. When a variable lookup occurs, the system checks the data dictionary associated with the current section. If the variable value can be found there, the replacement is made. However, if the variable's value is not found in the current data dictionary, the parent data dictionary is checked for the variable's value, and so on up the tree until the main data dictionary, or root, is reached.

Suppose that we want to display the names of all columns in a model. Consider the following template as an attempt to achieve this:

```
Report
------
Column Name: {{COLUMN_NAME}}
```

This template produces no output, even for a model that contains one or more columns. In this example, the only data dictionary active is the main dictionary. However, `COLUMN_NAME` is stored in the `COLUMNS` data dictionary, which is associated with the `COLUMNS` section.

With this knowledge, the template can be improved as follows:

```
Report
------
{{#COLUMNS}}
Column Name: {{COLUMN_NAME}}
{{/COLUMNS}}
```

This still does not produce output. To see why, see Table 9.1, “Data Dictionaries Tree”. The COLUMNS data dictionary has the parent dictionary COLUMNS_LISTING. COLUMNS_LISTING has the parent TABLES, which has the parent SCHEMATA, whose parent is the main dictionary. Remember that for a dictionary to be involved in variable lookup, its associated section must currently be active.

To achieve the desired output, the template must be something like the following:

```
Report
------

{{#SCHEMATA}}
{{#TABLES}}
{{#COLUMNS_LISTING}}
{{#COLUMNS}}
Column Name: {{COLUMN_NAME}}
{{/COLUMNS}}
{{/COLUMNS_LISTING}}
{{/TABLES}}
{{/SCHEMATA}}
```

The following template is the same, but with explanatory comments added:

```
Report
------

{{! Main dictionary active}}
{{#SCHEMATA}}  {{! SCHEMATA dictionary active}}
{{#TABLES}}  {{! TABLES dictionary active}}
{{#COLUMNS_LISTING}} {{! COLUMNS_LISTING dictionary active}}
{{#COLUMNS}}  {{! COLUMNS dictionary active}}
Column Name: {{COLUMN_NAME}} {{! COLUMN_NAME variable is looked-up,
and found, in COLUMNS data dictionary}}
{{/COLUMNS}}
{{/COLUMNS_LISTING}}
{{/TABLES}}
{{/SCHEMATA}}
```

Imagine now that for each column name displayed you also wanted to display its corresponding schema name, the template would look like this:

```
Report
------

{{#SCHEMATA}}
{{#TABLES}}
{{#COLUMNS_LISTING}}
{{#COLUMNS}}
Schema Name: {{SCHEMA_NAME}} Column Name: {{COLUMN_NAME}}
{{/COLUMNS}}
{{/COLUMNS_LISTING}}
{{/TABLES}}
{{/SCHEMATA}}
```

When variable lookup is performed for SCHEMA_NAME, the COLUMNS dictionary is checked. As the variable is not found there the parent dictionary will be checked, COLUMNS_LISTING, and so on, until the variable is eventually found where it is held, in the SCHEMATA dictionary.

If there are multiple schemata in the model, the outer section is iterated over a matching number of times, and SCHEMA_NAME accordingly has the correct value on each iteration.

It's important to always consider which dictionary must be active (and which parents) for a variable to be evaluated correctly. The following section has a table that helps you identify section requirements.

9.7.1 Supported Template Markers

The following table shows the supported markers. These markers can be used in any template, including custom templates.

Using the table

The table shows which variables are defined in which sections. The variable should be used in its correct section or its value will not be displayed. If a variable `type` is a variable, then the table describes its data dictionary, and a parent dictionary if `type` is a section. Also remember that the data dictionaries used to perform variable lookups form a hierarchical tree, so it is possible to use a variable in a child section that is defined in a parent section.

Table 9.2 Supported Template Markers

Marker text	Type	Data Dictionary or Parent Dictionary	Corresponding data
TITLE	Variable	MAIN	Title of the report
GENERATED	Variable	MAIN	Date and time when the report was generated
STYLE_NAME	Variable	MAIN	The name of the style selected in MySQL Workbench, this is typically used to load the corresponding CSS file, depending on the name of the style selected in MySQL Workbench
SCHEMA_COUNT	Variable	MAIN	The number of schemata in the model
PROJECT_TITLE	Variable	MAIN	Project title as set for the model in **Document Properties**
PROJECT_NAME	Variable	MAIN	Project name as set for the model in **Document Properties**
PROJECT_AUTHOR	Variable	MAIN	Project author as set for the model in **Document Properties**
PROJECT_VERSION	Variable	MAIN	Project version as set for the model in **Document Properties**
PROJECT_DESCRIPTION	Variable	MAIN	Project description as set for the model in **Document Properties**
PROJECT_CREATED	Variable	MAIN	Automatically set for the model project, but as displayed in **Document Properties**
PROJECT_CHANGED	Variable	MAIN	Automatically set for the model project, but as displayed in **Document Properties**

Marker text	**Type**	**Data Dictionary or Parent Dictionary**	**Corresponding data**
TOTAL_TABLE_COUNT	Variable	MAIN	The number of tables in all schemata in the model
TOTAL_COLUMN_COUNT	Variable	MAIN	The number of columns in all tables in all schemata in the model
TOTAL_INDEX_COUNT	Variable	MAIN	The number of indexes in the model
TOTAL_FK_COUNT	Variable	MAIN	The number of foreign keys in the model
SCHEMATA	Section	MAIN	Used to mark the start and end of a SCHEMATA section; the SCHEMATA data dictionary becomes active in this section
SCHEMA_NAME	Variable	SCHEMATA	The schema name
SCHEMA_ID	Variable	SCHEMATA	The schema ID
TABLE_COUNT	Variable	SCHEMATA	The number of tables in the current schema
COLUMN_COUNT	Variable	SCHEMATA	The number of columns in the current schema
INDICES_COUNT	Variable	SCHEMATA	The number of indexes in the current schema
FOREIGN_KEYS_COUNT	Variable	SCHEMATA	The number of foreign keys in the current schema
TABLES	Section	SCHEMATA	Marks the start and end of a TABLES section; the TABLES data dictionary becomes active in this section
TABLE_NAME	Variable	TABLES	The table name
TABLE_ID	Variable	TABLES	The table ID
COLUMNS_LISTING	Section	TABLES	Marks the start and end of a COLUMNS_LISTING section; the COLUMNS_LISTING data dictionary becomes active in this section
COLUMNS	Section	COLUMNS_LISTING	Marks the start and end of a COLUMNS section; the COLUMNS data dictionary becomes active in this section
COLUMN_KEY	Variable	COLUMNS	Whether the column is a primary key
COLUMN_NAME	Variable	COLUMNS	The column name
COLUMN_DATATYPE	Variable	COLUMNS	The column data type
COLUMN_NOTNULL	Variable	COLUMNS	Whether the column permits `NULL` values

Marker text	**Type**	**Data Dictionary or Parent Dictionary**	**Corresponding data**
COLUMN_DEFAULTVALUE	Variable	COLUMNS	The column default value
COLUMN_COMMENT	Variable	COLUMNS	The column comment
COLUMN_ID	Variable	COLUMNS	The column ID
COLUMN_KEY_PART	Variable	COLUMNS (if detailed)	The column key type
COLUMN_NULLABLE	Variable	COLUMNS (if detailed)	Can the column contain `NULL` values
COLUMN_AUTO_INC	Variable	COLUMNS (if detailed)	Does the column auto-increment
COLUMN_CHARSET	Variable	COLUMNS (if detailed)	The column character set
COLUMN_COLLATION	Variable	COLUMNS (if detailed)	The column collation
COLUMN_IS_USERTYPE	Variable	COLUMNS (if detailed)	Whether the column is a user type
INDICES_LISTING	Section	TABLES	Marks the start and end of an INDICES_LISTING section; the INDICES_LISTING data dictionary becomes active in this section
INDICES	Section	INDICES_LISTING	Marks the start and end of an INDICES section; the INDICES data dictionary becomes active in this section
INDEX_NAME	Variable	INDICES	The index name
INDEX_PRIMARY	Variable	INDICES	Whether this is a primary key
INDEX_UNIQUE	Variable	INDICES	Whether this is a unique index
INDEX_TYPE	Variable	INDICES	The index type; for example, PRIMARY
INDEX_KIND	Variable	INDICES	The index kind
INDEX_COMMENT	Variable	INDICES	The index comment
INDEX_ID	Variable	INDICES	The index ID
INDEX_COLUMNS	Section	INDICES	Marks the start and end of an INDEX_COLUMNS section; the INDEX_COLUMNS data dictionary becomes active in this section
INDEX_COLUMN_NAME	Variable	INDEX_COLUMNS	The index column name
INDEX_COLUMN_ORDER	Variable	INDEX_COLUMNS	The index column order; for example, ascending, descending
INDEX_COLUMN_COMMENT	Variable	INDEX_COLUMNS	The index comment
INDEX_KEY_BLOCK_SIZE	Variable	INDEX_COLUMNS (if detailed)	The index key-block size

Marker text	Type	Data Dictionary or Parent Dictionary	Corresponding data
REL_LISTING	Section	TABLES	Marks the start and end of a REL_LISTING section; the REL_LISTING data dictionary becomes active in this section
REL	Section	REL_LISTING	Marks the start and end of a REL section; the REL data dictionary becomes active in this section
REL_NAME	Variable	REL, FOREIGN_KEYS	The relationship name
REL_TYPE	Variable	REL, FOREIGN_KEYS	The relationship type
REL_PARENTTABLE	Variable	REL, FOREIGN_KEYS	The relationship parent table
REL_CHILDTABLE	Variable	REL, FOREIGN_KEYS	The relationship child table
REL_CARD	Variable	REL, FOREIGN_KEYS	The relationship cardinality
FOREIGN_KEY_ID	Variable	REL	Foreign key ID
FOREIGN_KEYS	Section	SCHEMATA	Marks the start and end of a FOREIGN_KEYS section; the FOREIGN_KEYS data dictionary becomes active in this section
FK_DELETE_RULE	Variable	FOREIGN_KEYS	The foreign key delete rule
FK_UPDATE_RULE	Variable	FOREIGN_KEYS	The foreign key update rule
FK_MANDATORY	Variable	FOREIGN_KEYS	Whether the foreign key is mandatory
TABLE_COMMENT_LISTING	Section	TABLES	Marks the start and end of a TABLE_COMMENT_LISTING section; the TABLE_COMMENT_LISTING data dictionary becomes active in this section
TABLE_COMMENT	Variable	TABLE_COMMENT_LISTING	The table comment
DDL_LISTING	Section	TABLES	Marks the start and end of a DDL_LISTING section; the DDL_LISTING data dictionary becomes active in this section
DDL_SCRIPT	Variable	DDL_LISTING	Display the DDL script of the currently active entity; for example, SCHEMATA, TABLES

9.7.2 Creating a Custom Template

In the simplest case, a template consists of two files: a template file, which has a `.tpl` extension, and a special file `info.xml`. The `info.xml` file has important metadata about the template. A third file is optional, which is the preview image file. This preview file provides a thumbnail image illustrating the appearance of the generated report.

One of the easiest ways to create a custom template is to make a copy of any existing template.

For example, the following procedure describes how to make a custom template based on the `Text Basic` template.

1. Navigate to the folder where the templates are stored. Assuming that MySQL Workbench has been installed into the default location on Windows, this would be `C:\Program Files\MySQL\MySQL Workbench 5.0 SE\modules\data\wb_model_reporting`.

2. Copy the `Text_Basic.tpl` folder. The copy can be given any suitable name; for example, `Custom_Basic.tpl`.

3. Edit the `info.xml` file to reflect your custom template. The unedited file in this case is shown here:

```
<?xml version="1.0"?>
<data>
  <value type="object" struct-name="workbench.model.reporting.TemplateInfo"
  id="{BD6879ED-814C-4CA3-A869-9864F83B88DF}" struct-checksum="0xb46b524d">
    <value type="string" key="description">
      A basic TEXT report listing schemata and objects.
    </value>
    <value type="string" key="name">HTML Basic Frame Report</value>
    <value type="list" content-type="object"
    content-struct-name="workbench.model.reporting.TemplateStyleInfo"
    key="styles">
      <value type="object" struct-name="workbench.model.reporting.TemplateStyleInfo"
      id="{7550655C-CD4B-4EB1-8FAB-AAEE49B2261E}" struct-checksum="0xab08451b">
        <value type="string" key="description">
          Designed to be viewed with a fixed sized font.
        </value>
        <value type="string" key="name">Fixed Size Font</value>
        <value type="string" key="previewImageFileName">
          preview_basic.png
        </value>
        <value type="string" key="styleTagValue">fixed</value>
      </value>
    </value>
    <value type="string" key="mainFileName">report.txt</value>
  </value>
</data>
```

   The file defines wwo objects: the `TemplateInfo` object and the `TemplateStyleInfo` object. These objects contain information about the template that will be displayed in the DBDoc Model Reporting wizard main page.

4. Change the object GUIDs that are used in the file. In this example, there are two that need replacing:

```
id="{BD6879ED-814C-4CA3-A869-9864F83B88DF}"
...
id="{7550655C-CD4B-4EB1-8FAB-AAEE49B2261E}"
```

   Generate two new GUIDS. This is done using a suitable command-line tool, and there are also free online tools that generate GUIDs. MySQL's `UUID()` function also generates GUIDs:

```
mysql> SELECT UUID();
+--------------------------------------+
| UUID()                               |
+--------------------------------------+
| 648f4240-7d7a-11e0-870b-89c43de3bd0a |
+--------------------------------------+
```

Once you have the new GUID values, edit the `info.xml` file accordingly.

5. Edit the textual information for the `TemplateInfo` and `TemplateStyleInfo` objects to reflect the purpose of the custom template.

6. The modified file will now look something like the following:

```
<?xml version="1.0"?>
<data>
  <value type="object" struct-name="workbench.model.reporting.TemplateInfo"
  id="{cac9ba3f-ee2a-49f0-b5f6-32580fab1640}" struct-checksum="0xb46b524d">
    <value type="string"
    key="description">
      Custom basic TEXT report listing schemata and objects.
    </value>
    <value type="string" key="name">Custom Basic text report</value>
    <value type="list" content-type="object"
    content-struct-name="workbench.model.reporting.TemplateStyleInfo" key="styles">
      <value type="object"
      struct-name="workbench.model.reporting.TemplateStyleInfo"
      id="{39e3b767-a832-4016-8753-b4cb93aa2dd6}" struct-checksum="0xab08451b">
        <value type="string" key="description">
          Designed to be viewed with a fixed sized font.
        </value>
        <value type="string" key="name">Fixed Size Font</value>
        <value type="string" key="previewImageFileName">preview_basic.png</value>
        <value type="string" key="styleTagValue">fixed</value>
      </value>
    </value>
    <value type="string" key="mainFileName">custom_report.txt</value>
  </value>
</data>
```

7. Create the new template file. This too may best be achieved, depending on your requirements, by editing an existing template. In this example the template file `report.txt.tpl` is shown here:

```
+--------------------------------------------+
| MySQL Workbench Report                     |
+--------------------------------------------+

Total number of Schemata: {{SCHEMA_COUNT}}
=============================================
{{#SCHEMATA}}
{{SCHEMA_NR}}. Schema: {{SCHEMA_NAME}}
----------------------------------------------
## Tables ({{TABLE_COUNT}}) ##
{{#TABLES}}{{TABLE_NR_FMT}}. Table: {{TABLE_NAME}}
{{#COLUMNS_LISTING}}## Columns ##
Key  Column  Name  Datatype  Not Null  Default  Comment
{{#COLUMNS}}{{COLUMN_KEY}}{{COLUMN_NAME}}{{COLUMN_DATATYPE}} »
{{COLUMN_NOTNULL}}{{COLUMN_DEFAULTVALUE}}{{COLUMN_COMMENT}}
{{/COLUMNS}}{{/COLUMNS_LISTING}}
{{#INDICES_LISTING}}## Indices ##
Index  Name  Columns  Primary  Unique  Type  Kind  Comment
{{#INDICES}}{{INDEX_NAME}}{{#INDICES_COLUMNS}}{{INDEX_COLUMN_NAME}} »
{{INDEX_COLUMN_ORDER}}{{INDEX_COLUMN_COMMENT}}{{/INDICES_COLUMNS}} »
{{INDEX_PRIMARY}}{{INDEX_UNIQUE}}{{INDEX_TYPE}}{{INDEX_KIND}}{{INDEX_COMMENT}}
{{/INDICES}}{{/INDICES_LISTING}}
{{#REL_LISTING}}## Relationships ##
Relationship  Name  Relationship  Type  Parent Table  Child Table Cardinality
{{#REL}}{{REL_NAME}}{{REL_TYPE}}{{REL_PARENTTABLE}}{{REL_CHILDTABLE}}{{REL_CARD}}
{{/REL}}{{/REL_LISTING}}
---------------------------------------------
```

```
{{/TABLES}}
{{/SCHEMATA}}
=============================================
End of MySQL Workbench Report
```

This template shows details for all schemata in the model.

8. The preceding template file can be edited in any way you like, with new markers being added, and existing markers being removed as required. For the custom template example, you might want to create a much simpler template, such as the one following:

```
+--------------------------------------------+
| MySQL Workbench Custom Report               |
+--------------------------------------------+

Total number of Schemata: {{SCHEMA_COUNT}}
=============================================
{{#SCHEMATA}}
Schema Name: {{SCHEMA_NAME}}
----------------------------------------------
## Tables ({{TABLE_COUNT}}) ##

{{#TABLES}}
Table Name: {{TABLE_NAME}}
{{/TABLES}}
{{/SCHEMATA}}

Report Generated On: {{GENERATED}}
=============================================
End of MySQL Workbench Custom Report
```

This simplified report just lists the schemata and the tables in a model. The date and time the report was generated is also displayed as a result of the use of the `{{GENERATED}}` variable.

9. The custom template can then be tested. Start MySQL Workbench, load the model to generate the report for, select the **Model**, **DBDOC - Model Reporting** menu item. Then select the new custom template from the list of available templates, select an output directory, and click **Finish** to generate the report. Finally, navigate to the output directory to view the finished report.

Chapter 10 Database Migration Wizard

Table of Contents

MySQL Workbench provides the ability to migrate ODBC compliant databases to MySQL.

- Convert (migrate) different database types, including MySQL, across servers
- Convert tables and copy data, but will not convert stored procedures, views, or triggers
- Allows customization and editing during the migration process
- Works on Linux, OS X, and Microsoft Windows

This is not an exhaustive list. The following sections discuss these and additional migration capabilities.

Set up may be the most challenging aspect of using the MySQL Workbench Migration Wizard. There is the installation section, which describes setting up ODBC requirements for Linux, OS X, and Microsoft

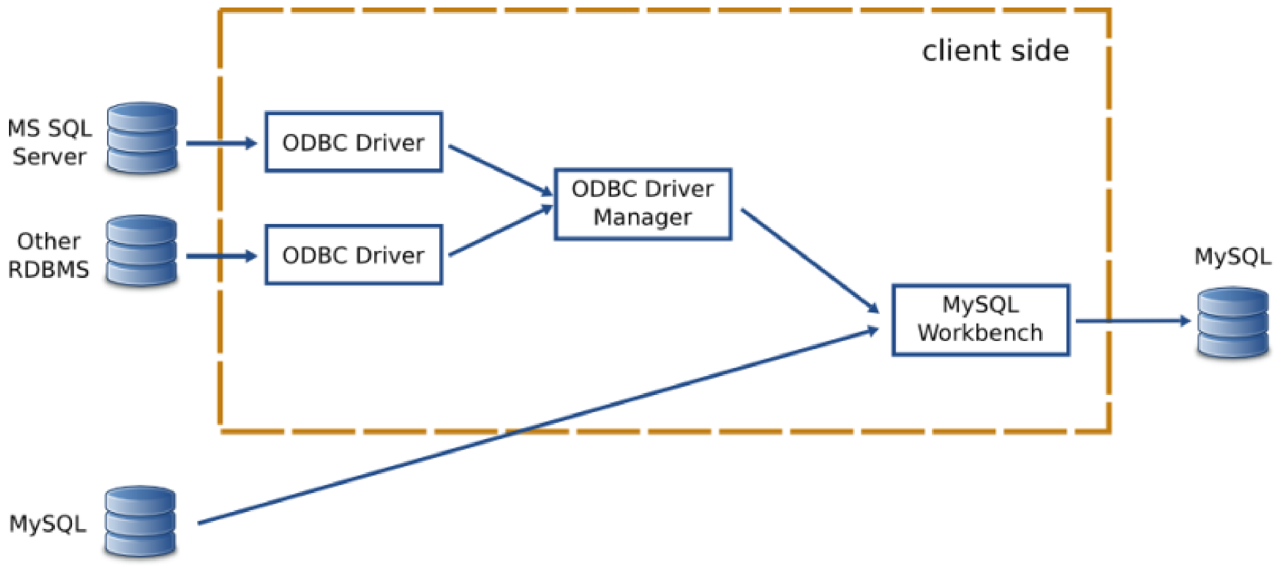
Windows, and the Database Product Specific Notes section that references setup conditions for each RDBMS.

10.1 General installation requirements

The MySQL Workbench Migration Wizard uses ODBC to connect to a source database, except for MySQL. You will need the ODBC driver installed that corresponds to the database you want to migrate from. For example, PostgreSQL can be migrated with the psqlodbc ODBC driver; Microsoft SQL Server can be migrated using the native Microsoft SQL Server driver on Windows or with FreeTDS on Linux and OS X.

The following diagram shows the general components involved in an ODBC connection:

Figure 10.1 MySQL Workbench migration installation diagram



When specifying the source RDBMS, you can either use a data source configured externally, or provide the individual connection parameters to MySQL Workbench. If you already have an ODBC Data Source configured in your system, then you can use that in MySQL Workbench.

Note

The migration process does not support source or target RDBMS connections through SSH.

A workaround is to set up an encrypted tunnel, and then treat the MySQL target as a standard TCP (unencrypted) connection.

10.1.1 ODBC Libraries

Note

This section may be skipped when using a MySQL Workbench binary that is provided by Oracle.

An ODBC Driver Manager library must be present. Both Windows and OS X provide one.

Linux

`iODBC`: MySQL Workbench binaries provided by Oracle already include iODBC and no additional action is required. If you compile it yourself, you must install iODBC or unixODBC. iODBC is recommended. You can use the iODBC library provided by your distribution by installing the libiodbc2 package on Debian based systems, or libiodbc on RPM based systems.

`pyodbc`: is the Python module used by MySQL Workbench to interface with ODBC, and may be used to migrate ODBC compliant databases such as PostgreSQL and DB2. In Windows and OS X, it is included with Workbench. In Linux, binaries provided by Oracle also include pyodbc.

If you're using a self-compiled binary, make sure you have the latest version, and that it is compiled against the ODBC manager library that you chose, whether it is iODBC or unixODBC. As of version 3.0.6, pyodbc will compile against unixODBC by default. If you are compiling against iODBC then you must perform the following steps:

1. For compiling, make sure you have the iODBC headers installed. For Linux, the name depends on your system's package manager but common names are `libiodbc-devel` (RPM based systems) or `libiodbc2-dev` (Debian based systems). For OS X, the headers come with the system and no additional action is required for this step.
2. In the pyodbc source directory, edit the setup.py file and around line 157, replace the following line: `settings['libraries'].append('odbc')` with `settings['libraries'].append('iodbc')`
3. Execute the following command as the root user: `` CFLAGS=`iodbc-config --cflags` LDFLAGS=`iodbc-config --libs` python setup.py install ``

10.1.2 ODBC Drivers

For each RDBMS, you need its corresponding ODBC driver, which must also be installed on the same machine that MySQL Workbench is running on. This driver is usually provided by the RDBMS manufacturer, but in some cases they can also be provided by third party vendors or open source projects.

Operating systems usually provide a graphical interface to help set up ODBC drivers and data sources. Use that to install the driver (i.e., make the ODBC Manager "see" a newly installed ODBC driver). You can also use it to create a data source for a specific database instance, to be connected using a previously configured driver. Typically you need to provide a name for the data source (the DSN), in addition to the database server IP, port, username, and sometimes the database the user has access to.

If MySQL Workbench is able to locate an ODBC manager GUI for your system, the **Open ODBC Administrator** button on the migration wizard's overview page will open it.

- `Linux`: There are a few GUI utilities, some of which are included with unixODBC. Refer to the documentation for your distribution. iODBC provides `iodbcadm-gtk`.
- `OS X`: You can use the ODBC Administrator tool that is separate download from Apple, or an ODBC Management tool from a different vendor. If the tool is installed in the `/Applications/Utilities` folder, you can start it using the **Open ODBC Administrator** button.
- `Microsoft Windows`: You can use the Data Sources (ODBC) tool under Administrative Tools. If present, the **Open ODBC Administrator** button will start it.

ODBC Driver architecture

Since the ODBC driver needs to be installed in the client side, you will need an ODBC driver that supports your clients operating system and architecture. For example, if you are running MySQL Workbench from Linux x64, then you need a Linux x64 ODBC driver for your RDBMS. In OS X, MySQL Workbench is built as a 32-bit application, so you need the 32-bit drivers.

10.2 Migration Overview

The Migration Wizard performs the following steps when migrating a database to MySQL:

1. Connects to the source RDBMS and retrieves a list of available databases/schemas.
2. Reverse engineers selected database/schemas into a internal representation specific to the source RDBMS. This step will also perform the renaming of objects/schemas depending on the type of object name mapping method that is chosen.
3. Automatically migrates the source RDBMS objects into MySQL specific objects.
    a. Target schema objects are created.
    b. Target table objects are created.
        i. Columns for each table are copied.
            A. Data types are mapped to MySQL data types.
            B. Default values are mapped to a MySQL supported default value, if possible.
        ii. Indexes are converted.
        iii. Primary Keys are converted.
        iv. Triggers are copied, and commented out if the source is not MySQL.
    c. Foreign Keys for all tables (of all schemas) are converted.
    d. View objects are copied, and commented out if the source is not MySQL.
    e. Stored Procedure and Function objects are copied, and commented out if the source is not MySQL.
4. Provides an opportunity to review the changes, for editing and correcting errors in the migrated objects.
5. Creates the migrated objects in the target MySQL server. If there are errors, you can return to the previous step and correct them, and retry the target creation.
6. Copy data of the migrated tables from the source RDBMS to MySQL.

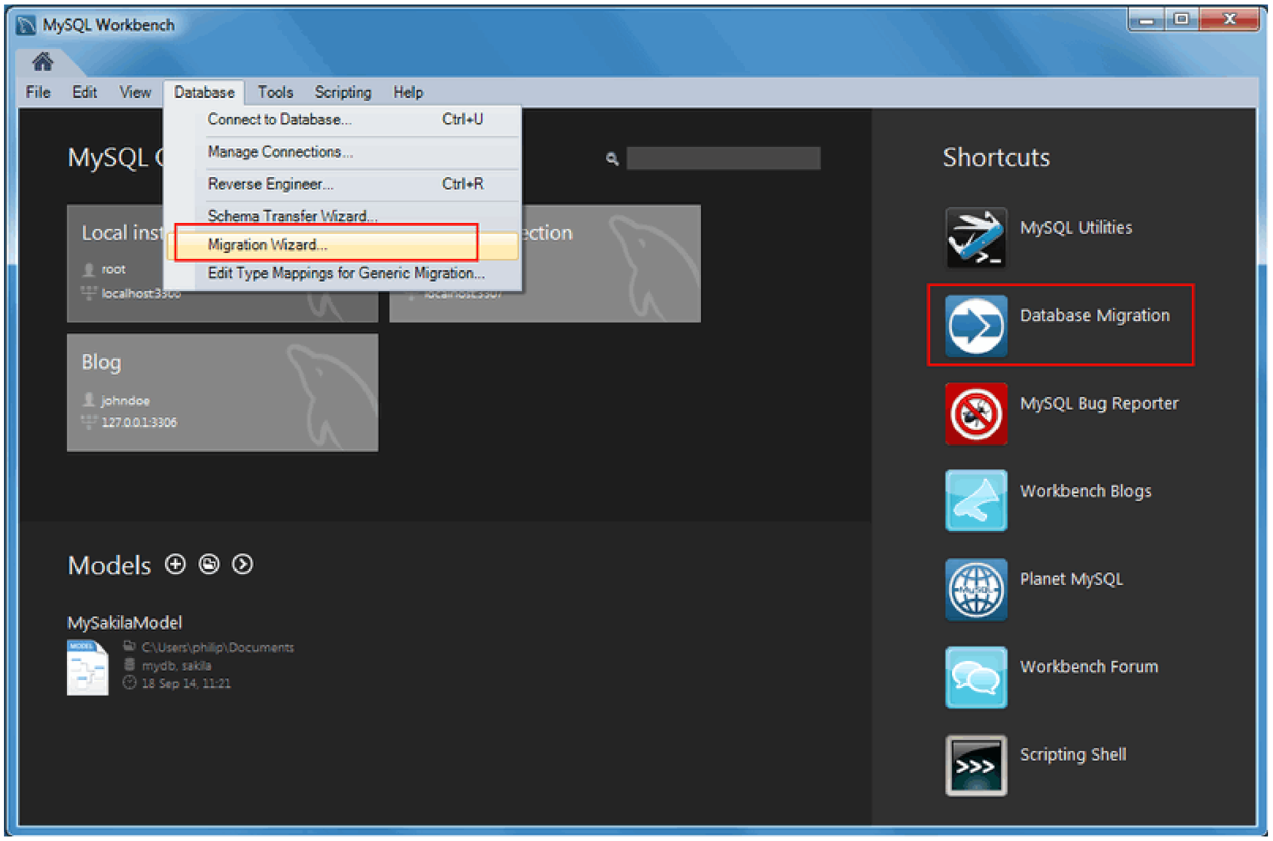
MySQL Workbench provides support for migrating from some specific RDBMS products. The Migration Wizard will provide the best results when migrating from such products. However, in some cases, other unsupported database products can also be migrated by using its Generic database support, as long as you have an ODBC driver for it. In this case, the migration will be less automatic, but should still work nonetheless.

10.2.1 A visual guide to performing a database migration

This example will migrate a Microsoft SQL Server database to MySQL, and include a screenshot for each step.

From MySQL Workbench, choose **Database**, **Migrate** to open the migration wizard.

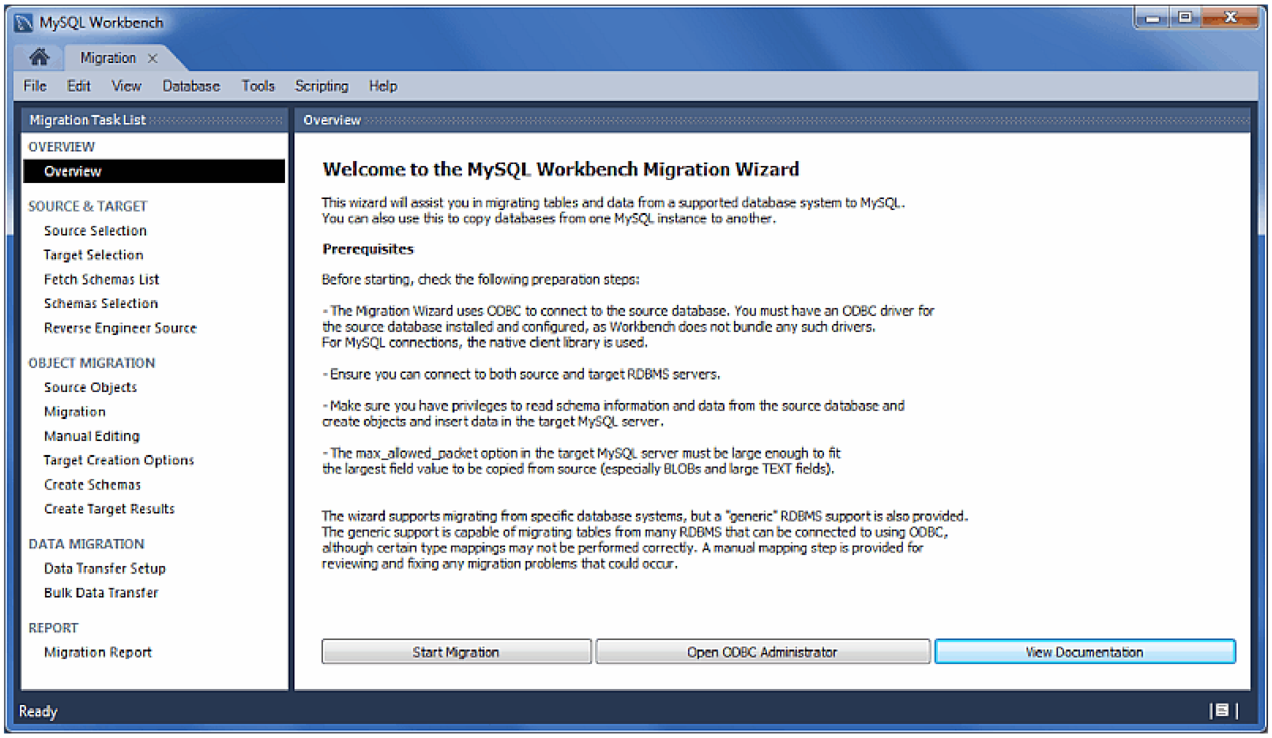
Figure 10.2 MySQL Workbench migration: Start



Read the migration wizard overview:

Overview

Figure 10.3 MySQL Workbench migration: Overview



Welcome to the MySQL Workbench Migration Wizard

This wizard will assist you in migrating tables and data from a supported database system to MySQL.
You can also use this to copy databases from one MySQL instance to another.

Prerequisites

Before starting, check the following preparation steps:

- The Migration Wizard uses ODBC to connect to the source database. You must have an ODBC driver for the source database installed and configured, as Workbench does not bundle any such drivers.
For MySQL connections, the native client library is used.

- Ensure you can connect to both source and target RDBMS servers.

- Make sure you have privileges to read schema information and data from the source database and create objects and insert data in the target MySQL server.

- The max_allowed_packet option in the target MySQL server must be large enough to fit the largest field value to be copied from source (especially BLOBs and large TEXT fields).

The wizard supports migrating from specific database systems, but a "generic" RDBMS support is also provided.
The generic support is capable of migrating tables from many RDBMS that can be connected to using ODBC, although certain type mappings may not be performed correctly. A manual mapping step is provided for reviewing and fixing any migration problems that could occur.

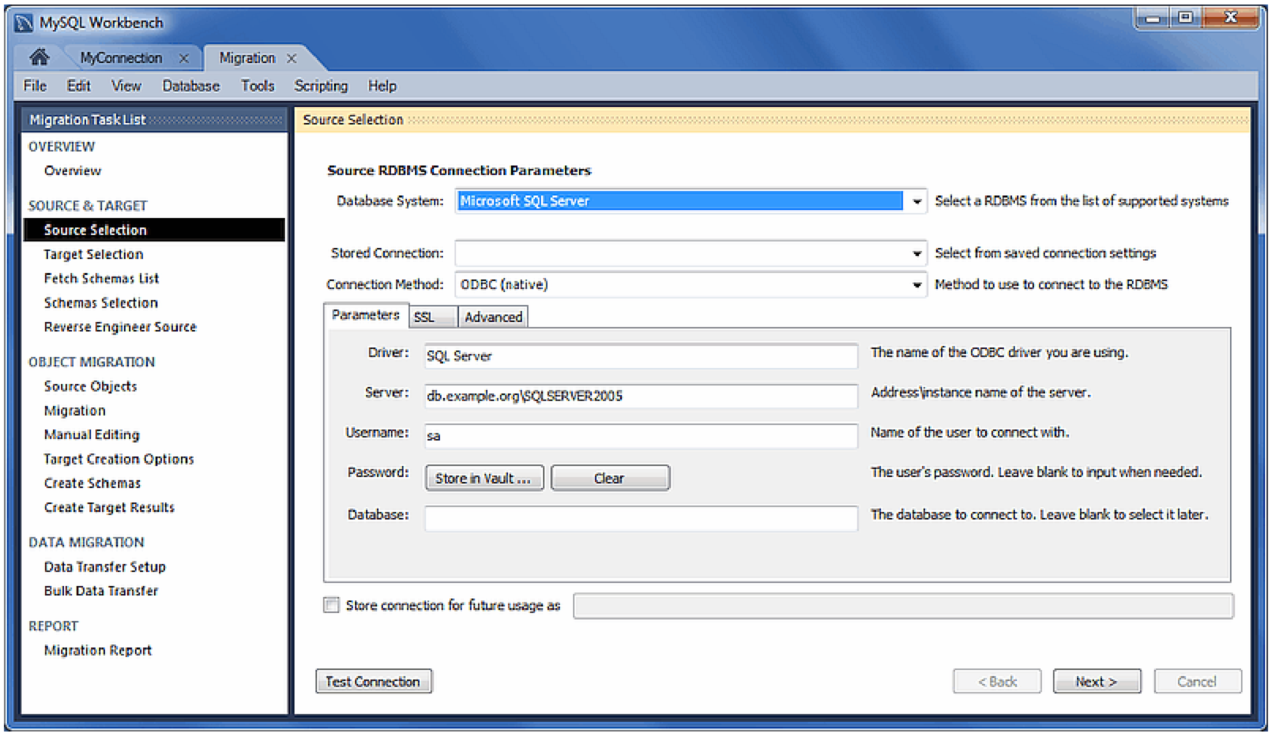
It describes the prerequisites and requirements that should be understood before proceeding further. The **Open ODBC Administrator** option will load `odbcad32.exe`, and is used to confirm that the ODBC Driver for SQL Server is installed, and to make configuration changes if needed.

Click **Start Migration** to continue.

Source Selection

Select the source RDBMS that is migrating to MySQL. Choose the **Database System** that is being migrated, and the other connection parameters will change accordingly.

Figure 10.4 MySQL Workbench migration: Source Selection (Parameters)



Source RDBMS Connection Parameters

Database System: Microsoft SQL Server — Select a RDBMS from the list of supported systems

Stored Connection: — Select from saved connection settings

Connection Method: ODBC (native) — Method to use to connect to the RDBMS

Driver: SQL Server — The name of the ODBC driver you are using.

Server: db.example.org\SQLSERVER2005 — Address\instance name of the server.

Username: sa — Name of the user to connect with.

Password: — The user's password. Leave blank to input when needed.

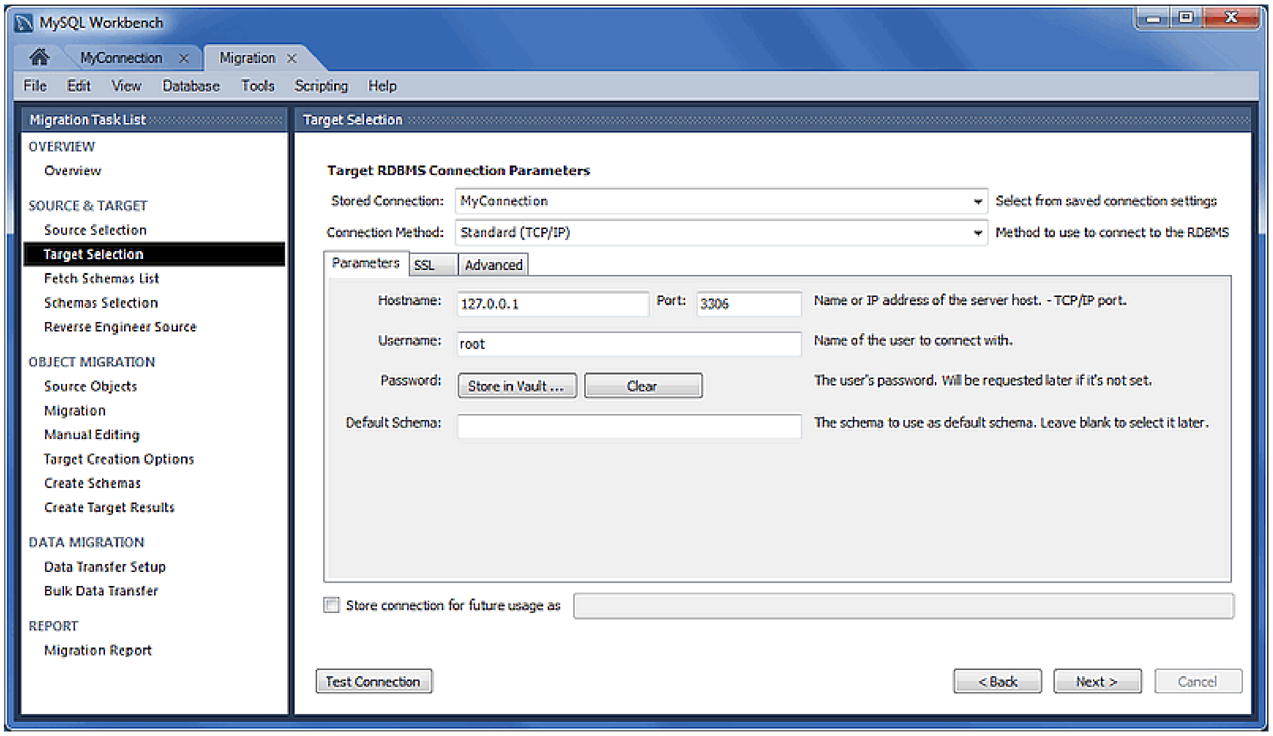
Database: — The database to connect to. Leave blank to select it later.

Store connection for future usage as

Target Selection

The target is the MySQL database that will contain the newly migrated database. The current Workbench MySQL connections will be available here, or you can choose **Manage DB Connections** to create a new connection.

Figure 10.5 MySQL Workbench migration: Target selection



Target RDBMS Connection Parameters

Stored Connection: MyConnection — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host. - TCP/IP port.

Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

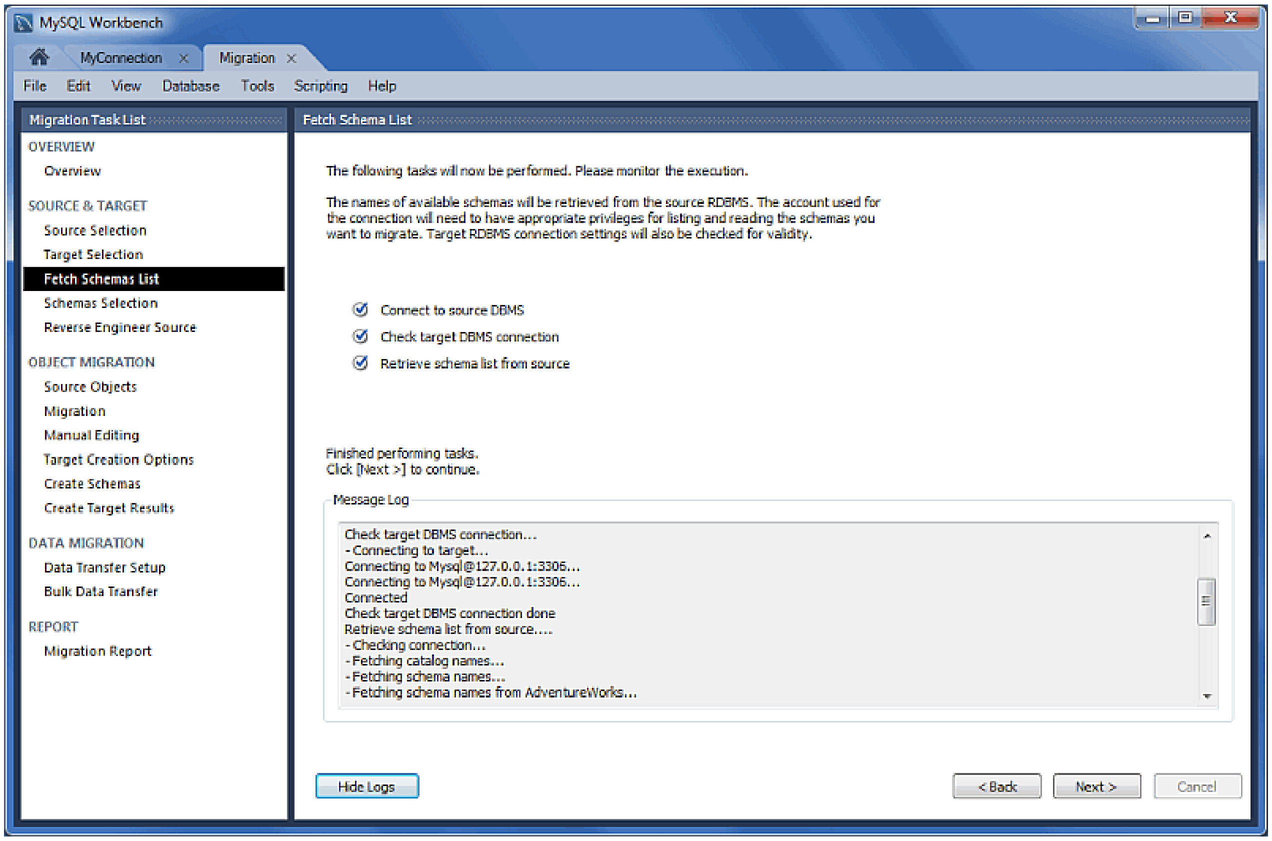
Default Schema: — The schema to use as default schema. Leave blank to select it later.

Store connection for future usage as

Fetch Schemata List

The Schemata list is retrieved from both the source and target RDBMS. This is an automated and informational step that reports connection related errors and/or general log information. Press **Next** to continue.

Figure 10.6 MySQL Workbench migration: Fetch Schemata List



The following tasks will now be performed. Please monitor the execution.

The names of available schemas will be retrieved from the source RDBMS. The account used for the connection will need to have appropriate privileges for listing and reading the schemas you want to migrate. Target RDBMS connection settings will also be checked for validity.

✓ Connect to source DBMS

✓ Check target DBMS connection

✓ Retrieve schema list from source

Finished performing tasks.
Click [Next >] to continue.

```
Check target DBMS connection...
- Connecting to target...
Connecting to Mysql@127.0.0.1:3306...
Connecting to Mysql@127.0.0.1:3306...
Connected
Check target DBMS connection done
Retrieve schema list from source....
- Checking connection...
- Fetching catalog names...
- Fetching schema names...
- Fetching schema names from AdventureWorks...
```

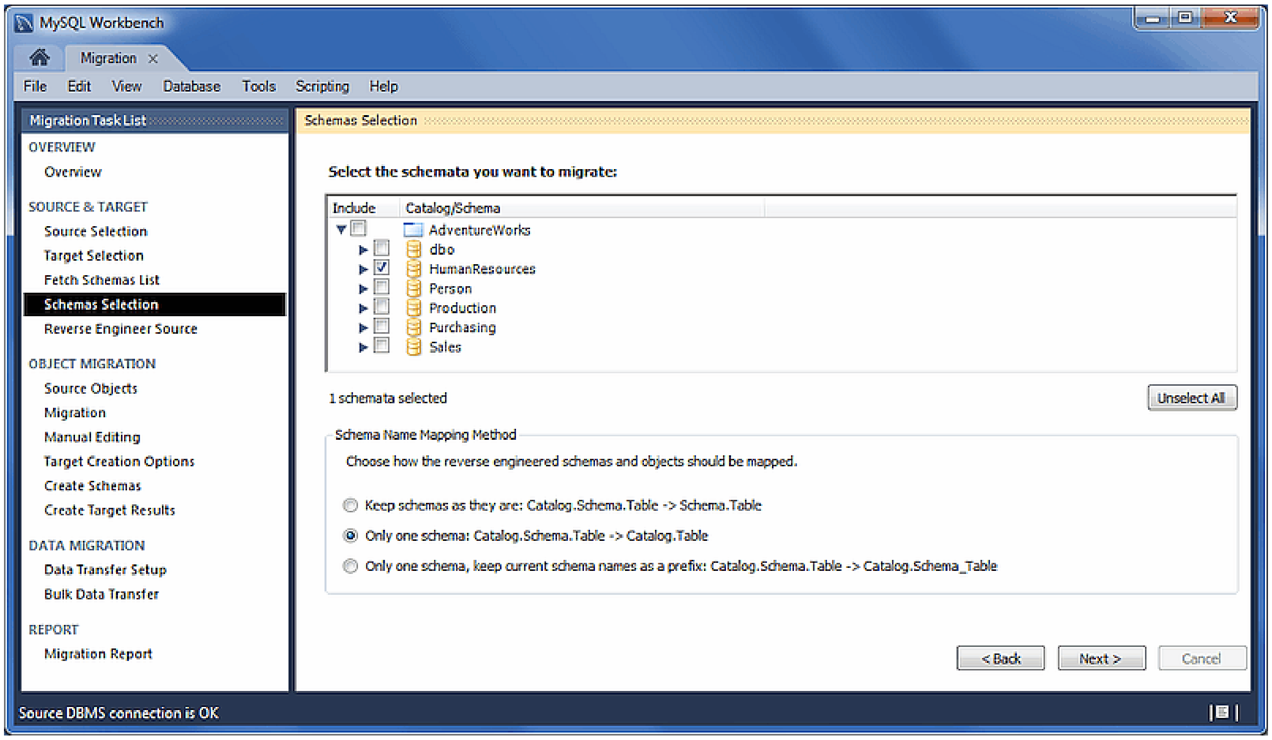
Schemata Selection

Choose the schemata you want to migrate.

"Schema Name Mapping Method" options while migrating Microsoft SQL Server:

- Keep schemata as they are: Catalog.Schema.Table -> Schema.Table: This will create multiple databases, one per schema.
- Only one schema: Catalog.Schema.Table -> Catalog.Table: Merges each schema into a single database.
- Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema_table: Preserves the schema name as a prefix.

Figure 10.7 MySQL Workbench migration: Schemata Selection



Select the schemata you want to migrate:

Include	Catalog/Schema
☐	AdventureWorks
☐	dbo
✓	HumanResources
☐	Person
☐	Production
☐	Purchasing
☐	Sales

1 schemata selected

Schema Name Mapping Method

Choose how the reverse engineered schemas and objects should be mapped.

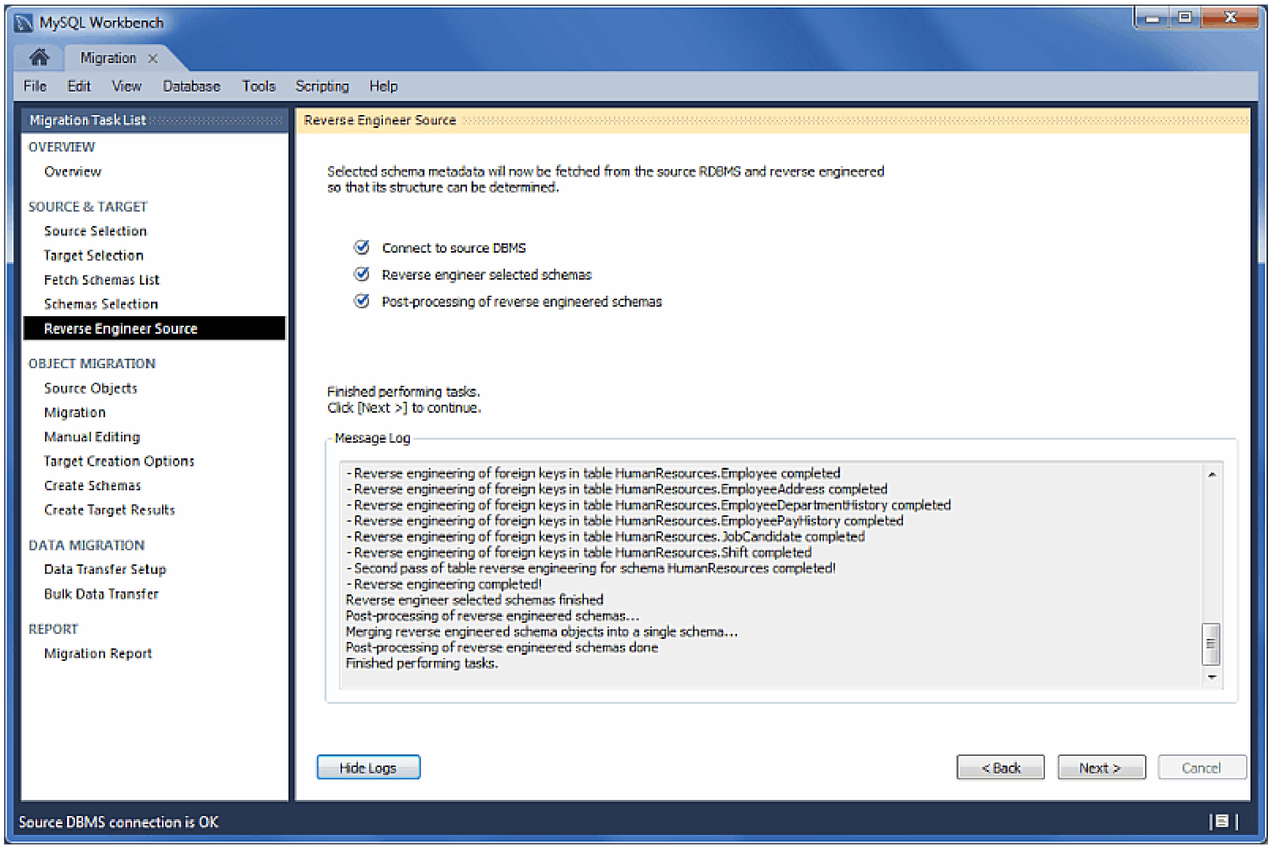
- ○ Keep schemas as they are: Catalog.Schema.Table -> Schema.Table
- ● Only one schema: Catalog.Schema.Table -> Catalog.Table
- ○ Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema_Table

Source DBMS connection is OK

Reverse Engineer Source

The source metadata is fetched from the source RDBMS, and reverse engineered. This is an automated and informational step that reports related errors and/or general log information. View the logs and then press **Next** to continue.

Figure 10.8 MySQL Workbench migration: Reverse Engineer Source



Selected schema metadata will now be fetched from the source RDBMS and reverse engineered so that its structure can be determined.

- ✓ Connect to source DBMS
- ✓ Reverse engineer selected schemas
- ✓ Post-processing of reverse engineered schemas

Finished performing tasks.
Click [Next >] to continue.

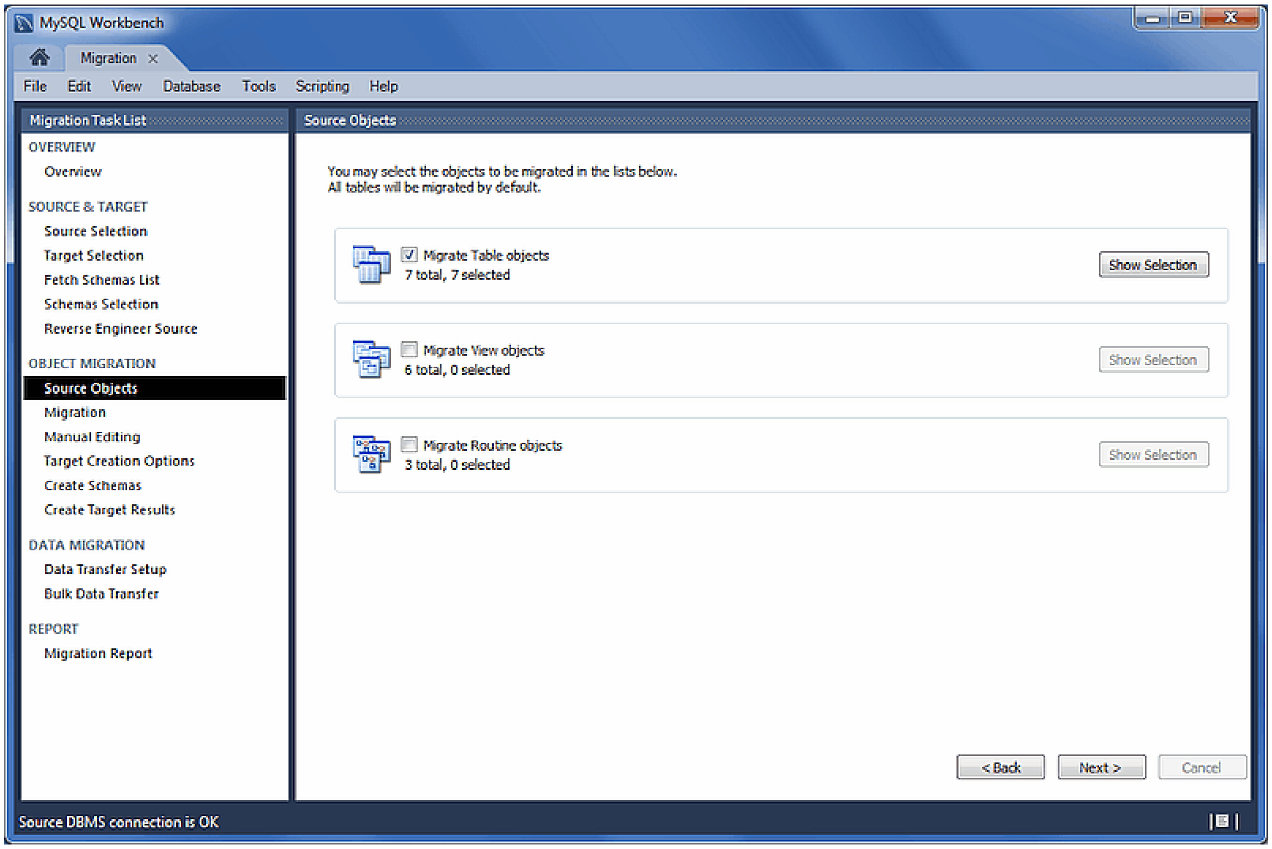
```
- Reverse engineering of foreign keys in table HumanResources.Employee completed
- Reverse engineering of foreign keys in table HumanResources.EmployeeAddress completed
- Reverse engineering of foreign keys in table HumanResources.EmployeeDepartmentHistory completed
- Reverse engineering of foreign keys in table HumanResources.EmployeePayHistory completed
- Reverse engineering of foreign keys in table HumanResources.JobCandidate completed
- Reverse engineering of foreign keys in table HumanResources.Shift completed
- Second pass of table reverse engineering for schema HumanResources completed!
- Reverse engineering completed!
Reverse engineer selected schemas finished
Post-processing of reverse engineered schemas...
Merging reverse engineered schema objects into a single schema...
Post-processing of reverse engineered schemas done
Finished performing tasks.
```

Source DBMS connection is OK

Source Objects

The discovered objects from the **Reverse Engineer Source** stage are revealed and made available. This includes Table, View, and Routine objects, with only the Table objects being selected by default.

Figure 10.9 MySQL Workbench migration: Source Objects



You may select the objects to be migrated in the lists below.
All tables will be migrated by default.

[✓] Migrate Table objects
7 total, 7 selected

[ ] Migrate View objects
6 total, 0 selected

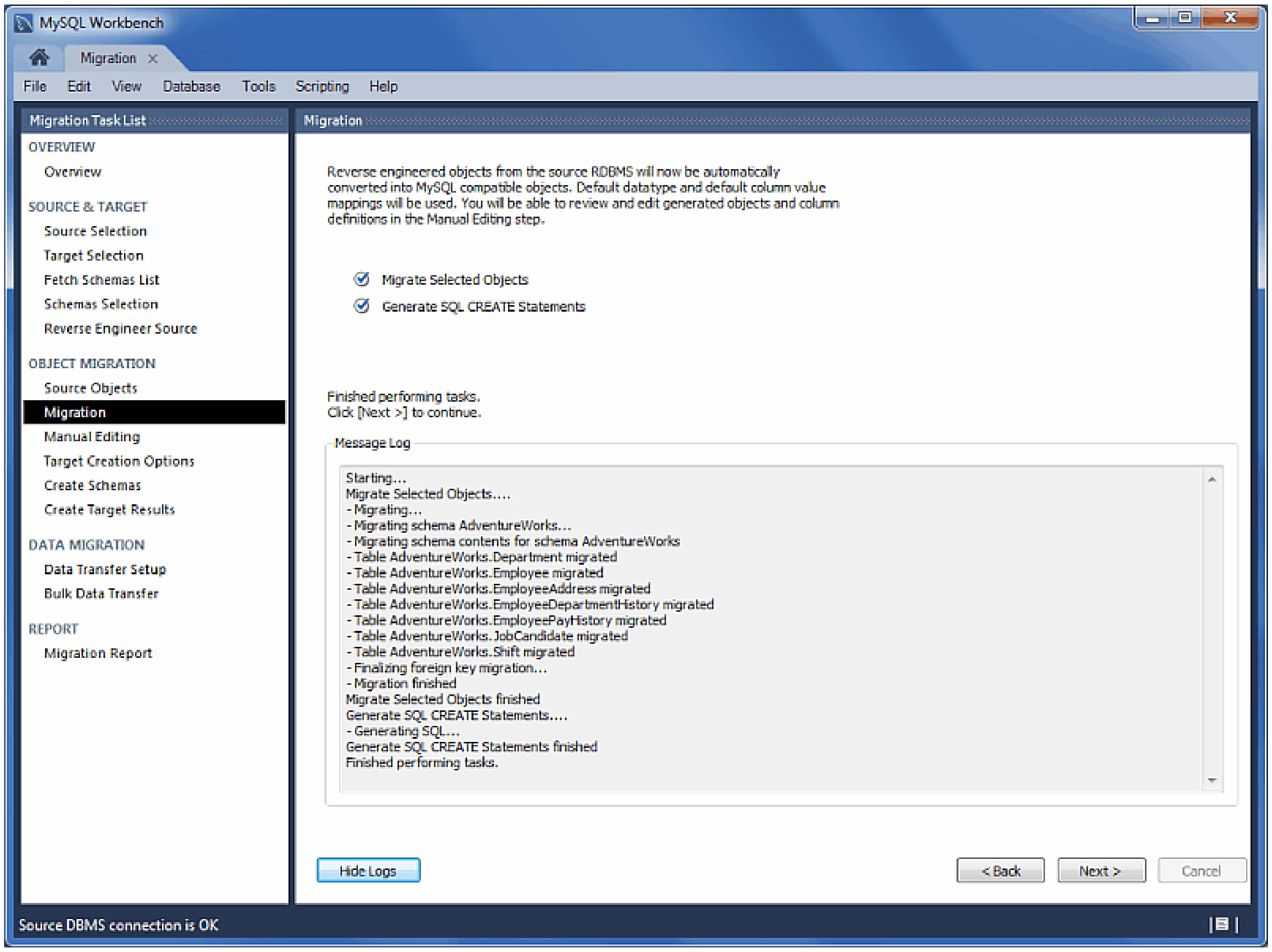
[ ] Migrate Routine objects
3 total, 0 selected

Source DBMS connection is OK

Migration

The migration process now converts the selected objects into MySQL compatible objects. View the logs and then proceed.

Figure 10.10 MySQL Workbench migration: Migration



Reverse engineered objects from the source RDBMS will now be automatically converted into MySQL compatible objects. Default datatype and default column value mappings will be used. You will be able to review and edit generated objects and column definitions in the Manual Editing step.

✓ Migrate Selected Objects
✓ Generate SQL CREATE Statements

Finished performing tasks.
Click [Next >] to continue.

```
Starting...
Migrate Selected Objects....
- Migrating...
- Migrating schema AdventureWorks...
- Migrating schema contents for schema AdventureWorks
- Table AdventureWorks.Department migrated
- Table AdventureWorks.Employee migrated
- Table AdventureWorks.EmployeeAddress migrated
- Table AdventureWorks.EmployeeDepartmentHistory migrated
- Table AdventureWorks.EmployeePayHistory migrated
- Table AdventureWorks.JobCandidate migrated
- Table AdventureWorks.Shift migrated
- Finalizing foreign key migration...
- Migration finished
Migrate Selected Objects finished
Generate SQL CREATE Statements....
- Generating SQL...
Generate SQL CREATE Statements finished
Finished performing tasks.
```

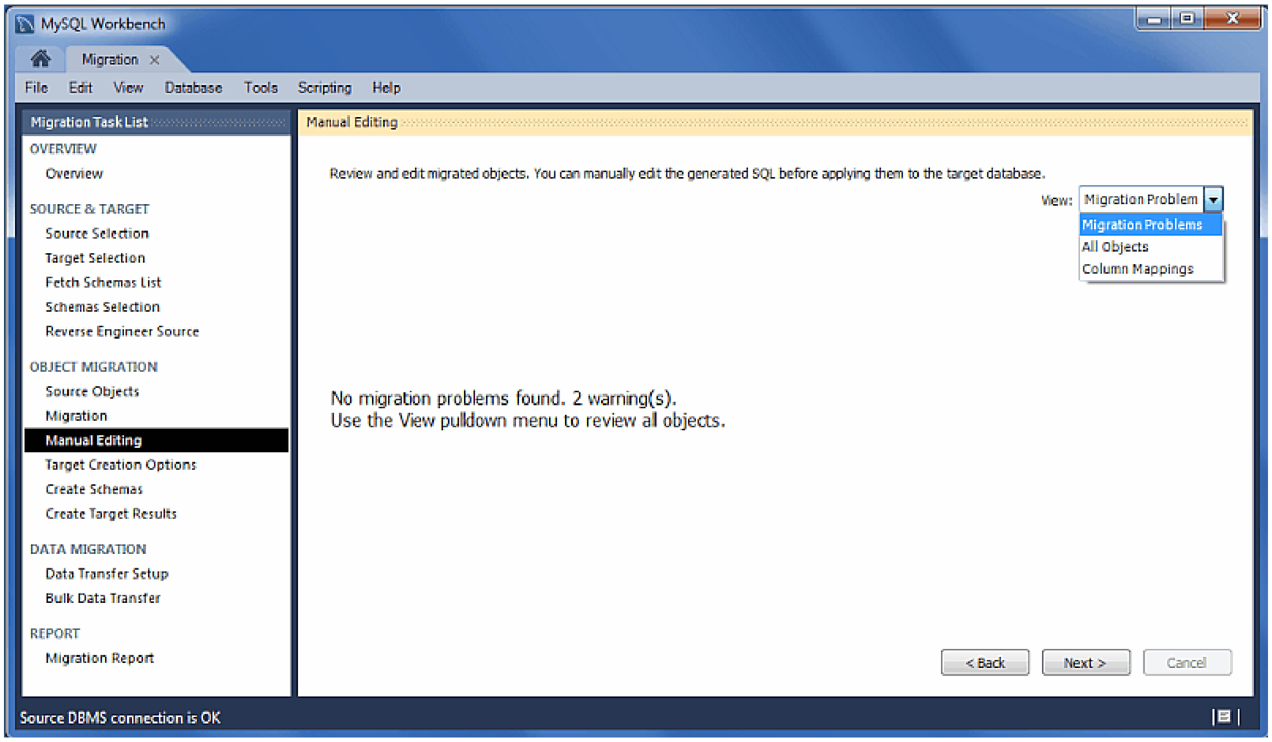
Source DBMS connection is OK

Manual Editing

There are three sections to edit here, which are selected via the **View** select box on the top right. The **Show Code and Messages** button is available with every view, and it will show the generated MySQL code that corresponds to the selected object.

- **Migration Problems**: This will either report problems or display "No mapping problems found." It is an informational screen.
- **All Objects**: An object view that allows you to view and edit the object definitions. Double-click on a row to modify a target objects name.
- **Column Mappings**: Shows all of the table column mappings, and allows you to individually review and fix the mapping for all column types, default values, and other attributes.

Figure 10.11 MySQL Workbench migration: Manual Editing (Migration Problems)

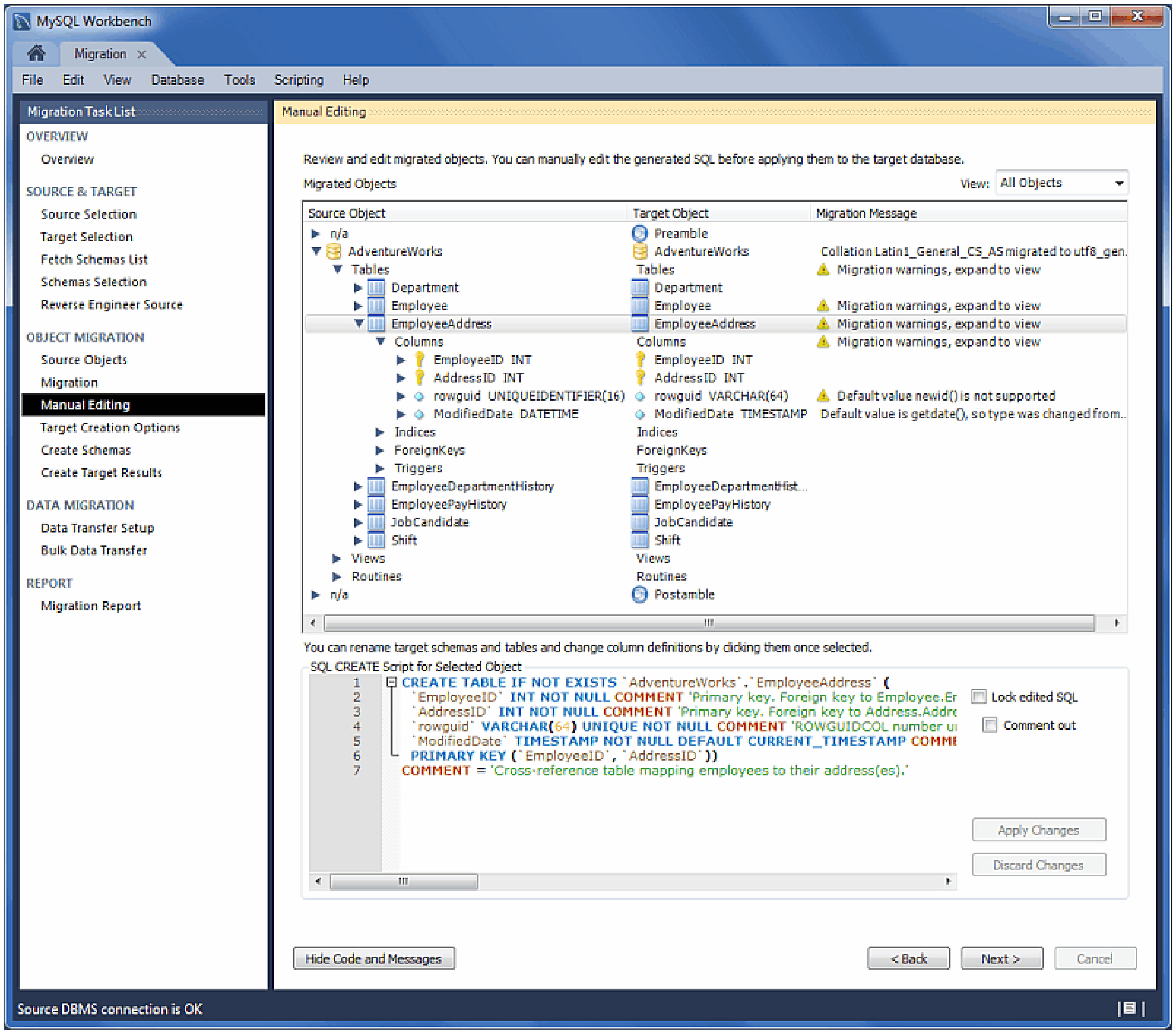


Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

No migration problems found. 2 warning(s).
Use the View pulldown menu to review all objects.

Source DBMS connection is OK

Figure 10.12 MySQL Workbench migration: Manual Editing (All Objects)



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects — View: All Objects

```
Source Object                              Target Object                  Migration Message
▶ n/a                                      Preamble
▼ AdventureWorks                           AdventureWorks                 Collation Latin1_General_CS_AS migrated to utf8_gen.
  ▼ Tables                                 Tables                         Migration warnings, expand to view
    ▶ Department                           Department
    ▶ Employee                             Employee                       Migration warnings, expand to view
    ▼ EmployeeAddress                      EmployeeAddress                Migration warnings, expand to view
      ▼ Columns                            Columns                        Migration warnings, expand to view
        ▶ EmployeeID  INT                  EmployeeID  INT
        ▶ AddressID  INT                   AddressID  INT
        ▶ rowguid  UNIQUEIDENTIFIER(16)    rowguid  VARCHAR(64)           Default value newid() is not supported
        ▶ ModifiedDate  DATETIME           ModifiedDate  TIMESTAMP        Default value is getdate(), so type was changed from..
      ▶ Indices                            Indices
      ▶ ForeignKeys                        ForeignKeys
      ▶ Triggers                           Triggers
    ▶ EmployeeDepartmentHistory            EmployeeDepartmentHist...
    ▶ EmployeePayHistory                   EmployeePayHistory
    ▶ JobCandidate                         JobCandidate
    ▶ Shift                                Shift
  ▶ Views                                  Views
  ▶ Routines                               Routines
▶ n/a                                      Postamble
```

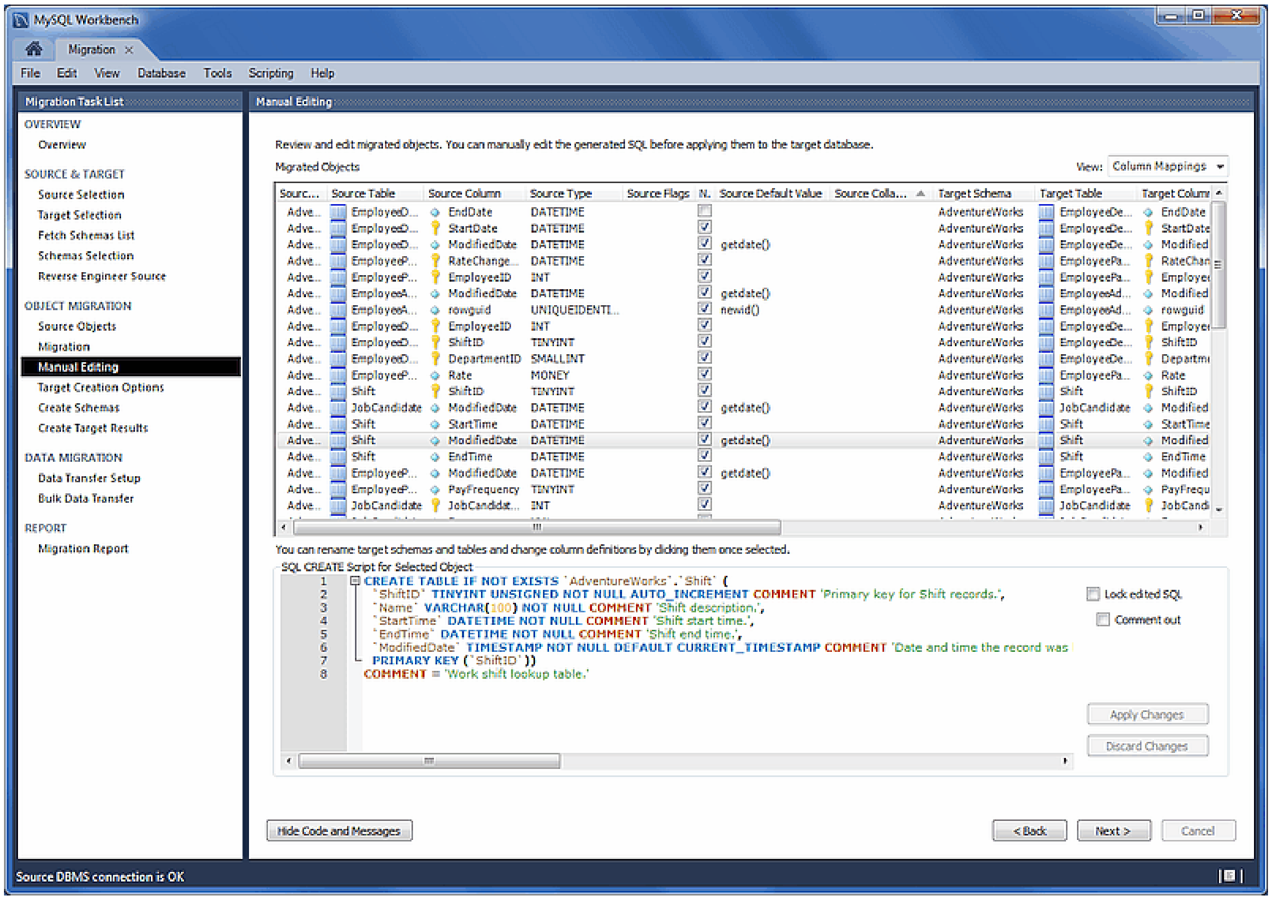
You can rename target schemas and tables and change column definitions by clicking them once selected.

SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `AdventureWorks`.`EmployeeAddress` (
  `EmployeeID` INT NOT NULL COMMENT 'Primary key. Foreign key to Employee.Er
  `AddressID` INT NOT NULL COMMENT 'Primary key. Foreign key to Address.Addr
  `rowguid` VARCHAR(64) UNIQUE NOT NULL COMMENT 'ROWGUIDCOL number ui
  `ModifiedDate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMI
  PRIMARY KEY (`EmployeeID`, `AddressID`))
COMMENT = 'Cross-reference table mapping employees to their address(es).'
```

Source DBMS connection is OK

Figure 10.13 MySQL Workbench migration: Manual Editing (Column Mappings)



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects

Sourc...	Source Table	Source Column	Source Type	Source Flags	N.	Source Default Value	Source Colla...	Target Schema	Target Table	Target Column
Adve...	EmployeeD...	EndDate	DATETIME		☐			AdventureWorks	EmployeeDe...	EndDate
Adve...	EmployeeD...	StartDate	DATETIME		✓			AdventureWorks	EmployeeDe...	StartDate
Adve...	EmployeeD...	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	EmployeeDe...	Modified
Adve...	EmployeeP...	RateChange...	DATETIME		✓			AdventureWorks	EmployeePa...	RateChan
Adve...	EmployeeP...	EmployeeID	INT		✓			AdventureWorks	EmployeePa...	Employe
Adve...	EmployeeA...	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	EmployeeAd...	Modified
Adve...	EmployeeA...	rowguid	UNIQUEIDENTI...		✓	newid()		AdventureWorks	EmployeeAd...	rowguid
Adve...	EmployeeD...	EmployeeID	INT		✓			AdventureWorks	EmployeeDe...	Employe
Adve...	EmployeeD...	ShiftID	TINYINT		✓			AdventureWorks	EmployeeDe...	ShiftID
Adve...	EmployeeD...	DepartmentID	SMALLINT		✓			AdventureWorks	EmployeeDe...	Departm
Adve...	EmployeeP...	Rate	MONEY		✓			AdventureWorks	EmployeePa...	Rate
Adve...	Shift	ShiftID	TINYINT		✓			AdventureWorks	Shift	ShiftID
Adve...	JobCandidate	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	JobCandidate	Modified
Adve...	Shift	StartTime	DATETIME		✓			AdventureWorks	Shift	StartTime
Adve...	Shift	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	Shift	Modified
Adve...	Shift	EndTime	DATETIME		✓			AdventureWorks	Shift	EndTime
Adve...	EmployeeP...	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	EmployeePa...	Modified
Adve...	EmployeeP...	PayFrequency	TINYINT		✓			AdventureWorks	EmployeePa...	PayFrequ
Adve...	JobCandidate	JobCandidat...	INT		✓			AdventureWorks	JobCandidate	JobCandi

You can rename target schemas and tables and change column definitions by clicking them once selected.

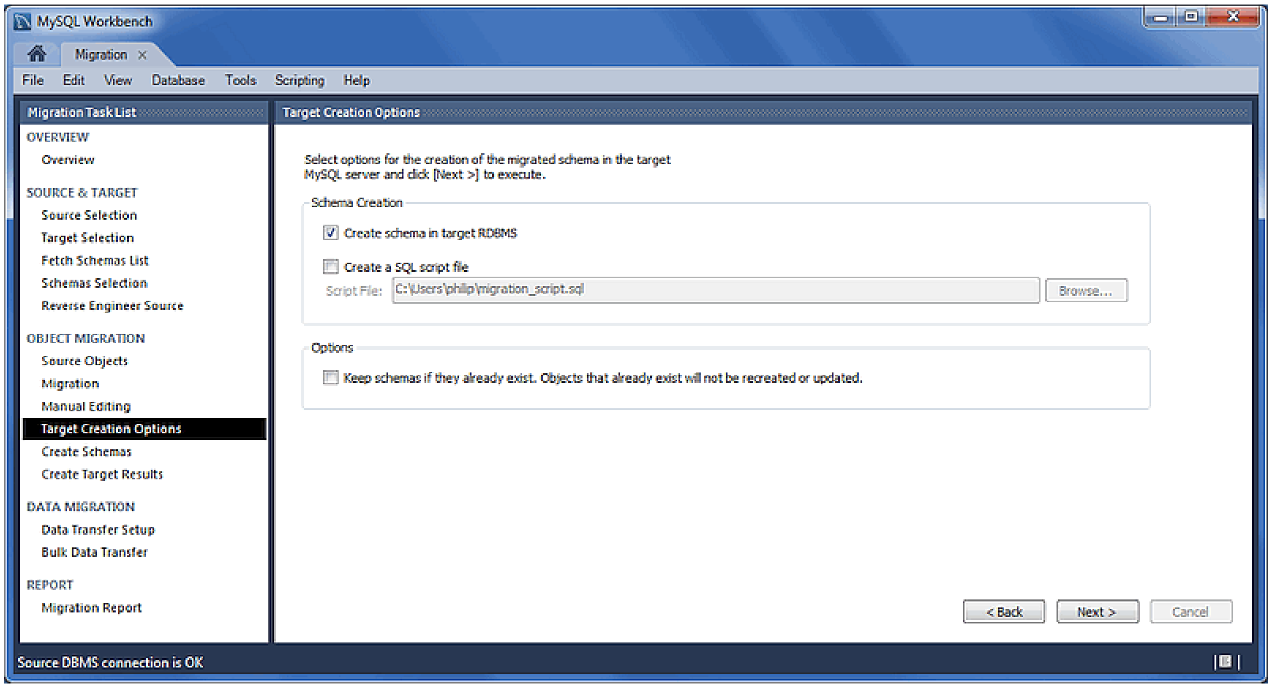
SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `AdventureWorks`.`Shift` (
  `ShiftID` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT 'Primary key for Shift records.',
  `Name` VARCHAR(100) NOT NULL COMMENT 'Shift description.',
  `StartTime` DATETIME NOT NULL COMMENT 'Shift start time.',
  `EndTime` DATETIME NOT NULL COMMENT 'Shift end time.',
  `ModifiedDate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Date and time the record was
  PRIMARY KEY (`ShiftID`))
COMMENT = 'Work shift lookup table.'
```

Target Creation Options

The schema may be created by either adding it to the target RDBMS, creating an SQL script file, or both.

Figure 10.14 MySQL Workbench migration: Target Creation Options



Select options for the creation of the migrated schema in the target MySQL server and click [Next >] to execute.

Schema Creation

- [✓] Create schema in target RDBMS
- [ ] Create a SQL script file

Script File: C:\Users\philip\migration_script.sql

Options

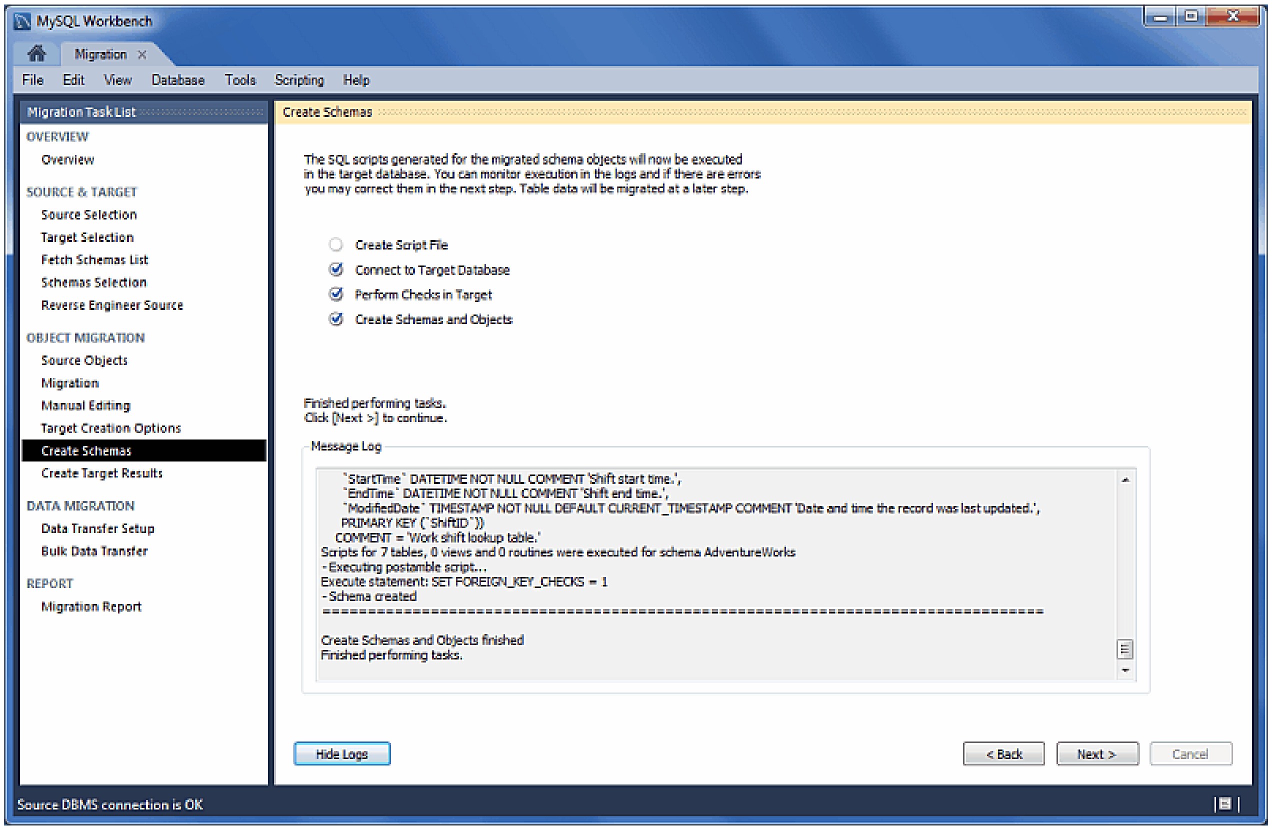
- [ ] Keep schemas if they already exist. Objects that already exist will not be recreated or updated.

Source DBMS connection is OK

Create Schemata

Now the schemata is created. The complete log is also available here.

Figure 10.15 MySQL Workbench migration: Create Schemata



The SQL scripts generated for the migrated schema objects will now be executed in the target database. You can monitor execution in the logs and if there are errors you may correct them in the next step. Table data will be migrated at a later step.

- Create Script File
- ✓ Connect to Target Database
- ✓ Perform Checks in Target
- ✓ Create Schemas and Objects

Finished performing tasks.
Click [Next >] to continue.

```
      `StartTime` DATETIME NOT NULL COMMENT 'Shift start time.',
      `EndTime` DATETIME NOT NULL COMMENT 'Shift end time.',
      `ModifiedDate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Date and time the record was last updated.',
     PRIMARY KEY (`ShiftID`))
   COMMENT = 'Work shift lookup table.'
Scripts for 7 tables, 0 views and 0 routines were executed for schema AdventureWorks
- Executing postamble script...
Execute statement: SET FOREIGN_KEY_CHECKS = 1
- Schema created
================================================================================

Create Schemas and Objects finished
Finished performing tasks.
```

Create Target Results

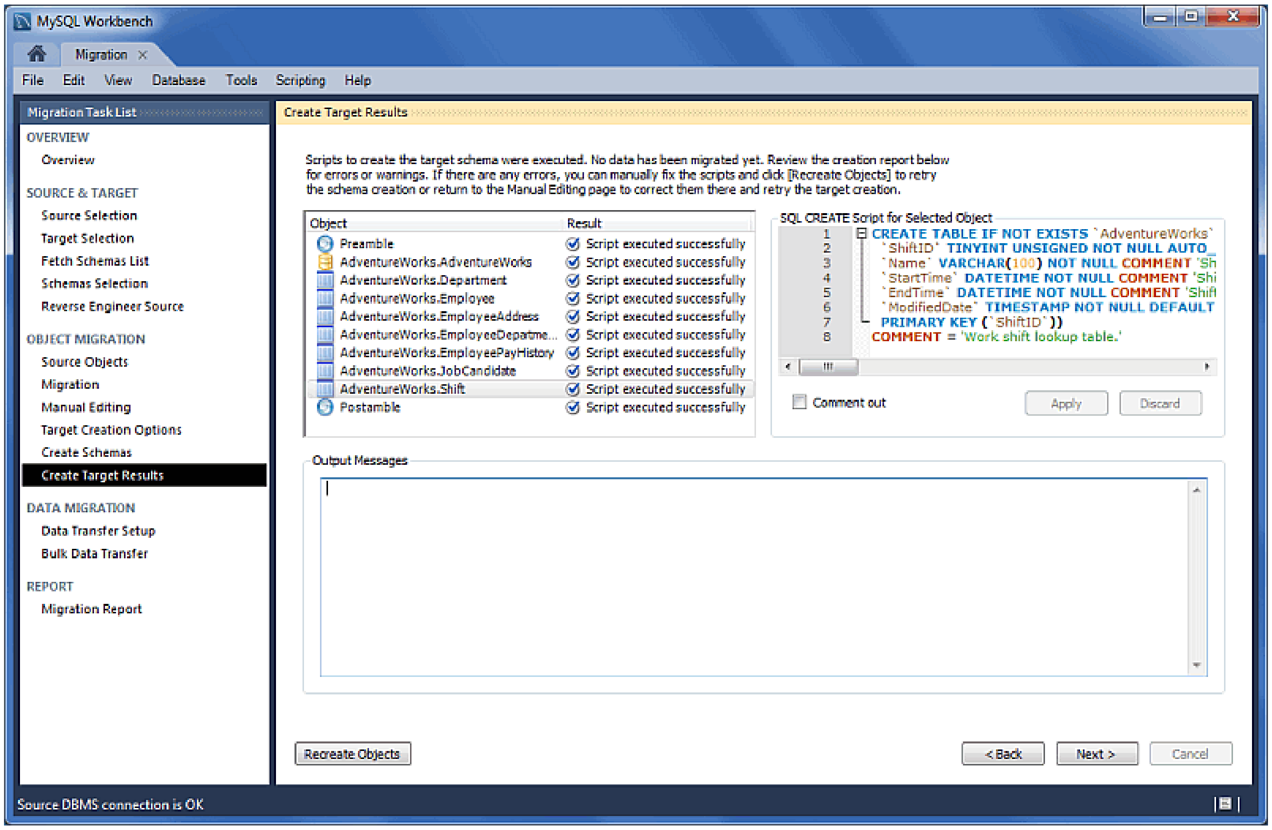
The generated objects are listed here, along with the error messages if any exist.

The migration code may also be viewed and edited here. To make changes, select an object, edit the query code, and press **Apply**. Repeat this process for each object that will be edited. And then, press **Recreate Objects** to save the results.

Note

The **Recreate Objects** operation is required to save any changes here. It will then execute the previous migration step (**Create Schemata**) with the modified code, and then continue the migration process. This also means that the previously saved schema will be dropped.

Figure 10.16 MySQL Workbench migration: Create Target Results



Scripts to create the target schema were executed. No data has been migrated yet. Review the creation report below for errors or warnings. If there are any errors, you can manually fix the scripts and click [Recreate Objects] to retry the schema creation or return to the Manual Editing page to correct them there and retry the target creation.

Object	Result
Preamble	✓ Script executed successfully
AdventureWorks.AdventureWorks	✓ Script executed successfully
AdventureWorks.Department	✓ Script executed successfully
AdventureWorks.Employee	✓ Script executed successfully
AdventureWorks.EmployeeAddress	✓ Script executed successfully
AdventureWorks.EmployeeDepatme...	✓ Script executed successfully
AdventureWorks.EmployeePayHistory	✓ Script executed successfully
AdventureWorks.JobCandidate	✓ Script executed successfully
AdventureWorks.Shift	✓ Script executed successfully
Postamble	✓ Script executed successfully

SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `AdventureWorks`
  `ShiftID` TINYINT UNSIGNED NOT NULL AUTO_
  `Name` VARCHAR(100) NOT NULL COMMENT 'Sh
  `StartTime` DATETIME NOT NULL COMMENT 'Shi
  `EndTime` DATETIME NOT NULL COMMENT 'Shift
  `ModifiedDate` TIMESTAMP NOT NULL DEFAULT
  PRIMARY KEY (`ShiftID`))
COMMENT = 'Work shift lookup table.'
```

Source DBMS connection is OK

Data Transfer Setup

The next step transfers data from the source RDBMS to the target MySQL database. The setup screen includes the following options:

Data Copy:

- *Online copy of table data to target RDBMS*: This (default) will copy the data to the target RDBMS.
- *Create a batch file to copy the data at another time*: The data may also be dumped to a file that can be executed at a later time, or be used as a backup. This script uses a MySQL connection to transfer the data.
- *Create a shell script to use native server dump and load abilities for fast migration*: Unlike the simple batch file that performs a live online copy, this generates a script to be executed on the *source* host to then generate a Zip file containing all of the data and information needed to migrate the data locally on the target host. Copy and extract the generated Zip file on the target host and then execute the import script (on the target host) to import the data into MySQL using a LOAD DATA call.

  This faster method avoids the need to traffic all data through MySQL Workbench, or to have a permanent network connection between the MySQL servers.

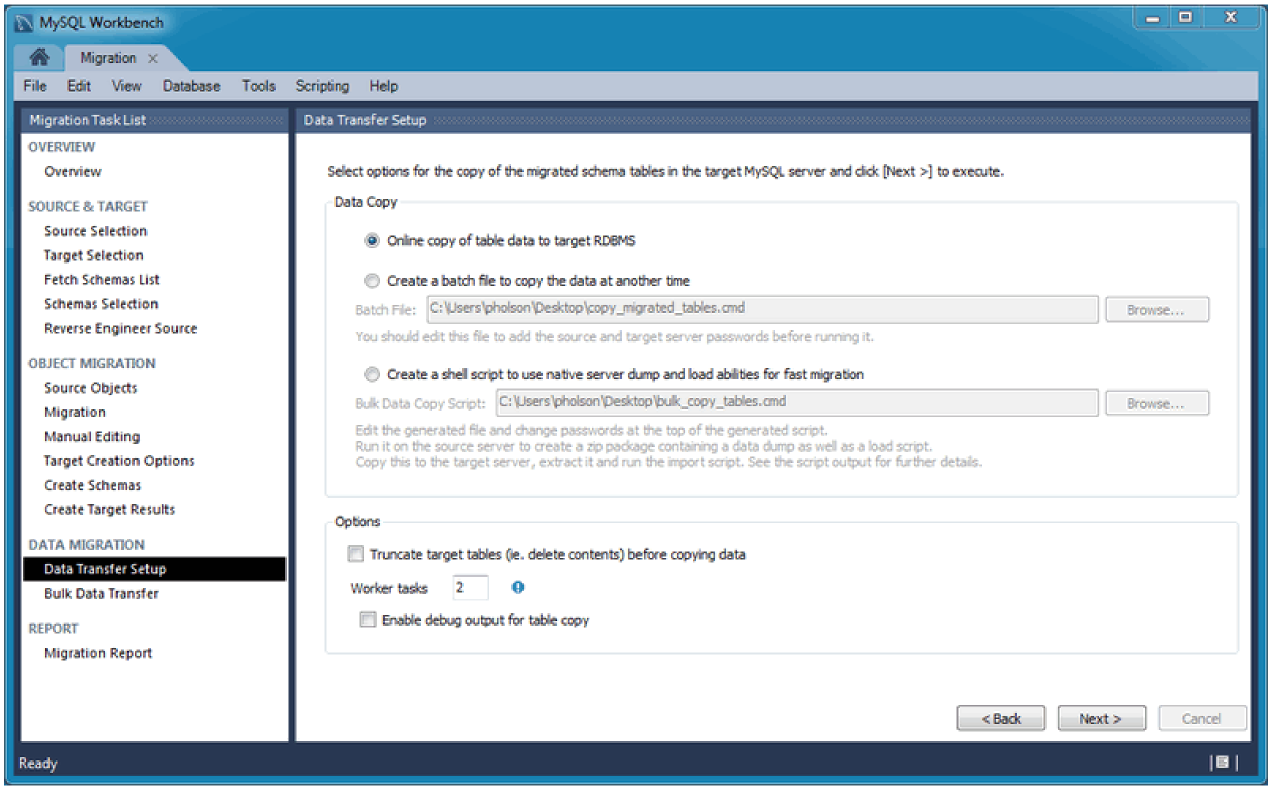
Note

This option was added in MySQL Workbench 6.3.0.

Options:

- Truncate target tables before copying data: In case the target database already exists, this will delete said data.
- Worker tasks: The default value is 2. This is the number of tasks (database connections) used while copying the data.
- Enable debug output for table copy: Shows debugging information.

Figure 10.17 MySQL Workbench migration: Data Transfer Setup



Data Transfer Setup

Select options for the copy of the migrated schema tables in the target MySQL server and click [Next >] to execute.

Data Copy

[✓ Online copy of table data to target RDBMS]

Create a batch file to copy the data at another time

Batch File: C:\Users\pholson\Desktop\copy_migrated_tables.cmd

You should edit this file to add the source and target server passwords before running it.

Create a shell script to use native server dump and load abilities for fast migration

Bulk Data Copy Script: C:\Users\pholson\Desktop\bulk_copy_tables.cmd

Edit the generated file and change passwords at the top of the generated script.
Run it on the source server to create a zip package containing a data dump as well as a load script.
Copy this to the target server, extract it and run the import script. See the script output for further details.

Options

Truncate target tables (ie. delete contents) before copying data

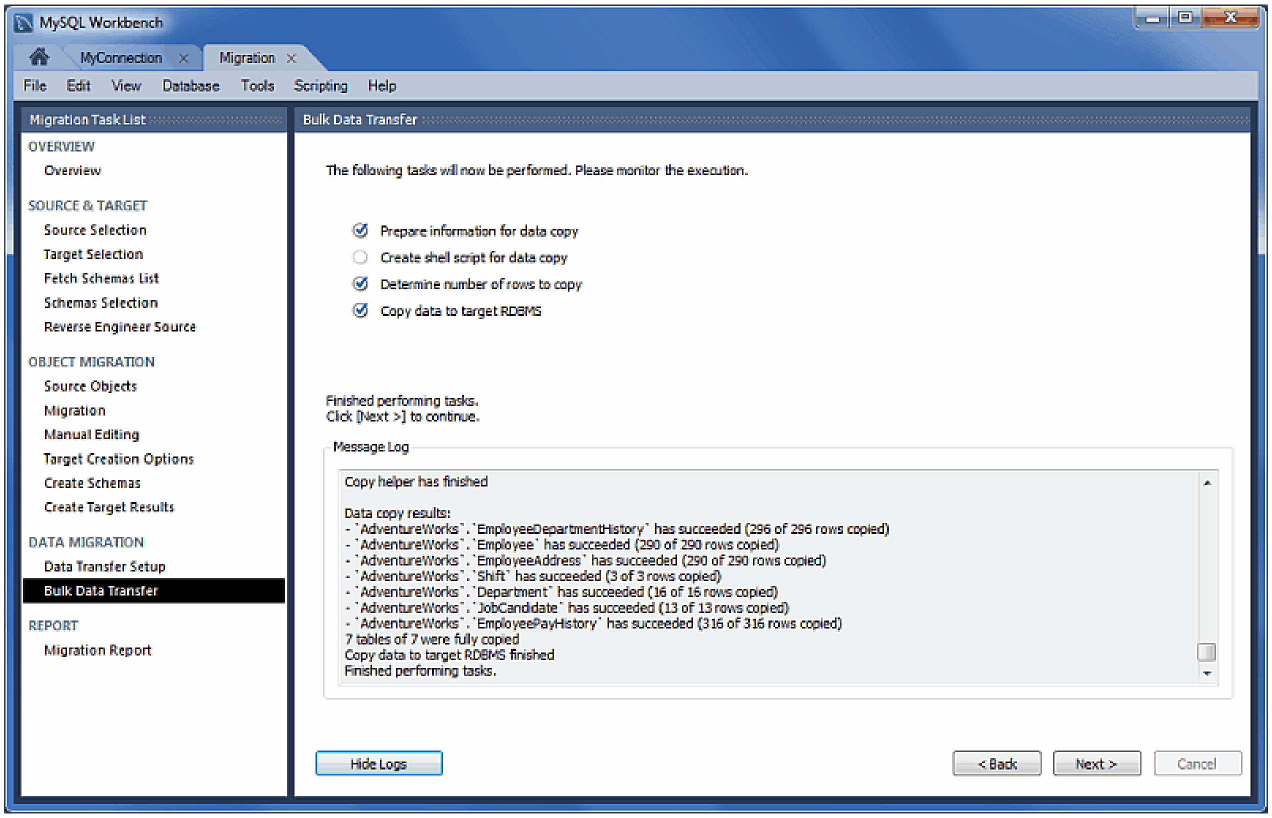
Worker tasks: 2

Enable debug output for table copy

Bulk Data Transfer

Depending on the selected option, this will either transfer the data to the target RDMS (default), generate a simple script for the online data transfer, or generate script to execute on the source host that then generates a Zip file containing both the transfer script and data that will be executed on the target host. Optionally, view the logs to confirm.

Figure 10.18 MySQL Workbench migration: Bulk Data Transfer



The following tasks will now be performed. Please monitor the execution.

- ✓ Prepare information for data copy
- Create shell script for data copy
- ✓ Determine number of rows to copy
- ✓ Copy data to target RDBMS

Finished performing tasks.
Click [Next >] to continue.

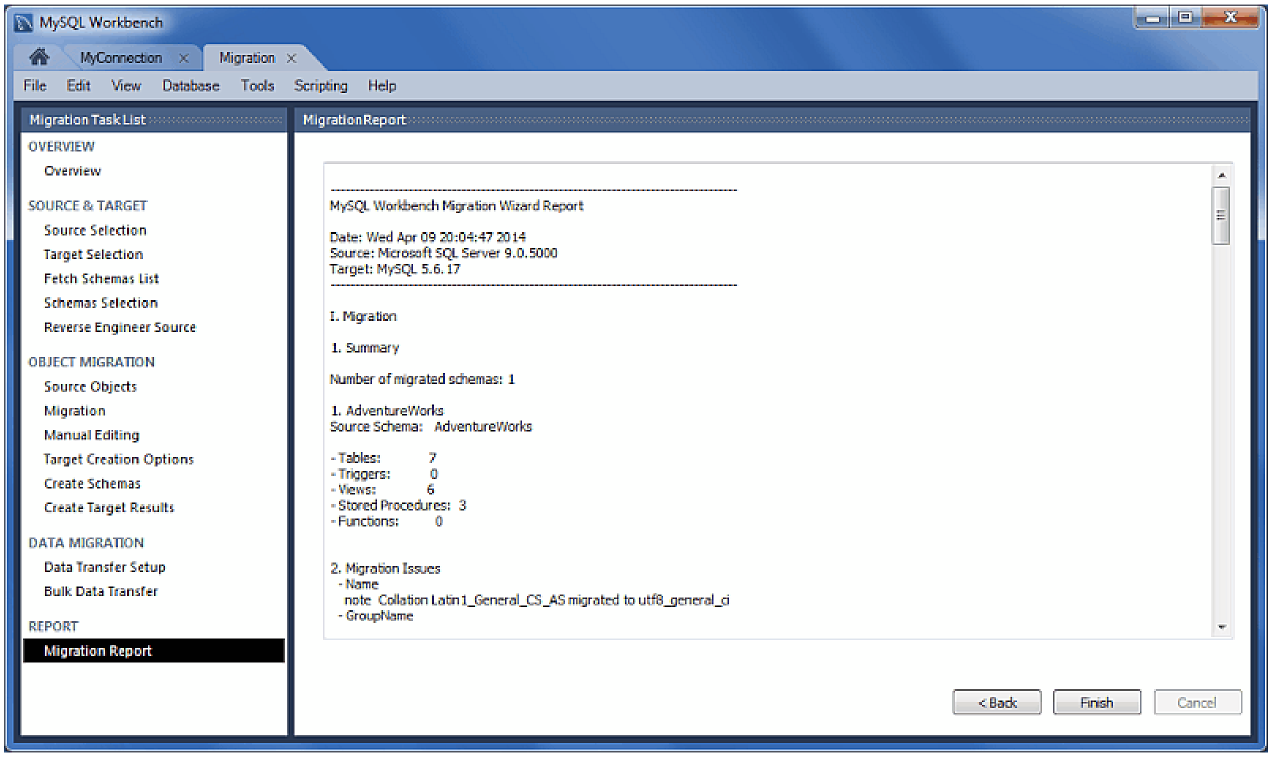
Message Log

```
Copy helper has finished

Data copy results:
- `AdventureWorks`.`EmployeeDepartmentHistory` has succeeded (296 of 296 rows copied)
- `AdventureWorks`.`Employee` has succeeded (290 of 290 rows copied)
- `AdventureWorks`.`EmployeeAddress` has succeeded (290 of 290 rows copied)
- `AdventureWorks`.`Shift` has succeeded (3 of 3 rows copied)
- `AdventureWorks`.`Department` has succeeded (16 of 16 rows copied)
- `AdventureWorks`.`JobCandidate` has succeeded (13 of 13 rows copied)
- `AdventureWorks`.`EmployeePayHistory` has succeeded (316 of 316 rows copied)
7 tables of 7 were fully copied
Copy data to target RDBMS finished
Finished performing tasks.
```

Migration Report

And finally, the migration report is available and summarizes the entire migration process.

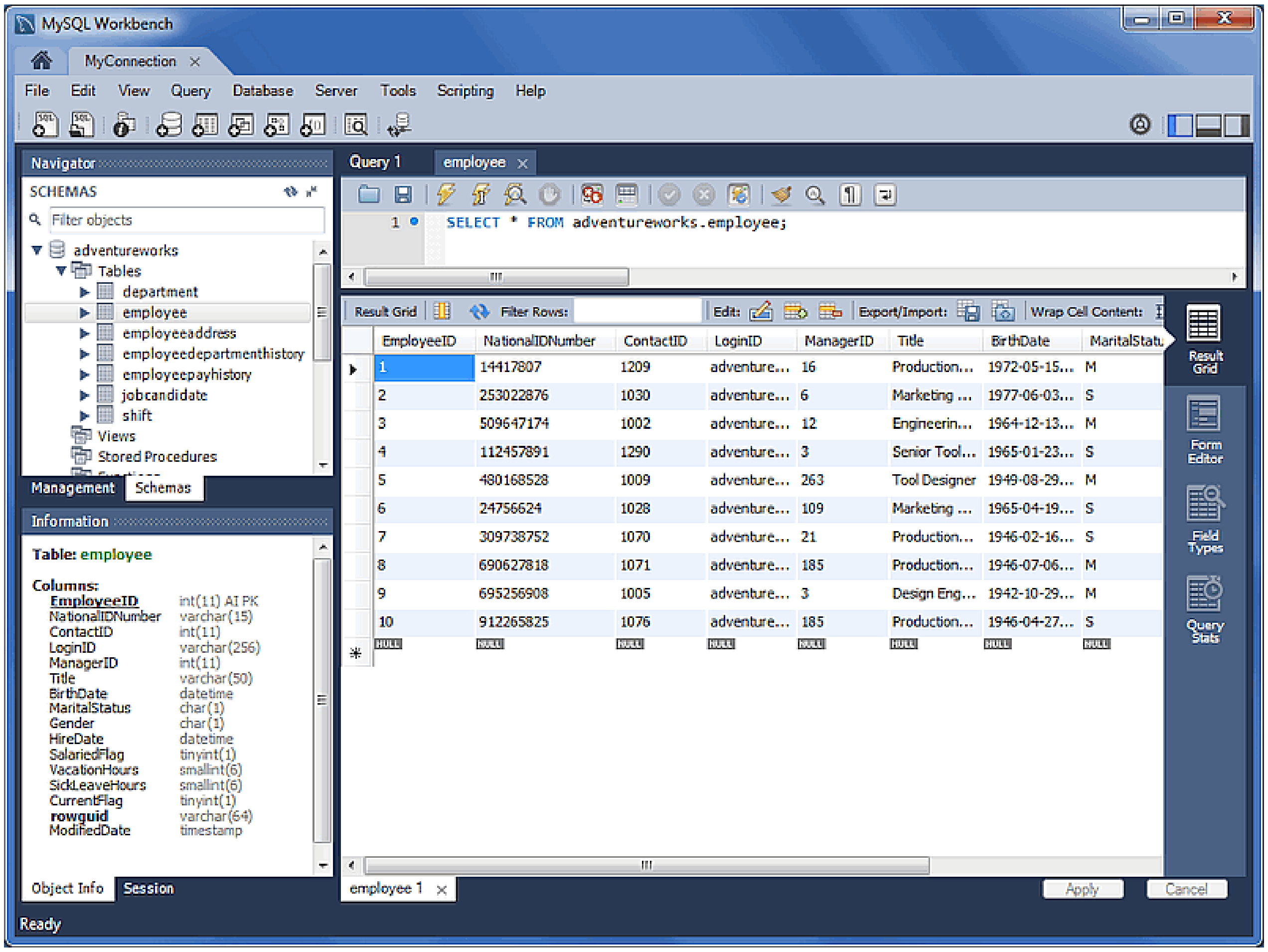
Figure 10.19 MySQL Workbench migration: Migration Report



```
--------------------------------------------------------------------------------
MySQL Workbench Migration Wizard Report

Date: Wed Apr 09 20:04:47 2014
Source: Microsoft SQL Server 9.0.5000
Target: MySQL 5.6.17
--------------------------------------------------------------------------------

I. Migration

1. Summary

Number of migrated schemas: 1

1. AdventureWorks
Source Schema:   AdventureWorks

- Tables:           7
- Triggers:         0
- Views:            6
- Stored Procedures: 3
- Functions:        0


2. Migration Issues
  - Name
    note  Collation Latin1_General_CS_AS migrated to utf8_general_ci
  - GroupName
```

Pressing **Finish** will close the migration window. If you chose the online copy then the database can now be viewed within the MySQL Workbench SQL editor.

Figure 10.20 MySQL Workbench migration: Viewing the migrated database



```
SELECT * FROM adventureworks.employee;
```

	EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID	Title	BirthDate	MaritalStatu
▶	1	14417807	1209	adventure...	16	Production...	1972-05-15...	M
	2	253022876	1030	adventure...	6	Marketing ...	1977-06-03...	S
	3	509647174	1002	adventure...	12	Engineerin...	1964-12-13...	M
	4	112457891	1290	adventure...	3	Senior Tool...	1965-01-23...	S
	5	480168528	1009	adventure...	263	Tool Designer	1949-08-29...	M
	6	24756624	1028	adventure...	109	Marketing ...	1965-04-19...	S
	7	309738752	1070	adventure...	21	Production...	1946-02-16...	S
	8	690627818	1071	adventure...	185	Production...	1946-07-06...	M
	9	695256908	1005	adventure...	3	Design Eng...	1942-10-29...	M
	10	912265825	1076	adventure...	185	Production...	1946-04-27...	S
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: employee

Columns:

EmployeeID	int(11) AI PK
NationalIDNumber	varchar(15)
ContactID	int(11)
LoginID	varchar(256)
ManagerID	int(11)
Title	varchar(50)
BirthDate	datetime
MaritalStatus	char(1)
Gender	char(1)
HireDate	datetime
SalariedFlag	tinyint(1)
VacationHours	smallint(6)
SickLeaveHours	smallint(6)
CurrentFlag	tinyint(1)
rowguid	varchar(64)
ModifiedDate	timestamp

Note

If a MySQL Workbench SQL Editor tab is already opened, then the schema list within the Object Browser must be refreshed in order to view the newly imported schema.

10.2.2 Migrating from supported databases

When a supported RDBMS product is being migrated, the MySQL Workbench Migration Wizard will automatically convert as much information as it can, but you may still be required to manually edit the automatically migrated schema for difficult cases, or when the default mapping is not as desired.

Generally speaking, only table information and its data are automatically converted to MySQL. Code objects such as views, stored procedures, and triggers, are not. But supported RDBMS products will be retrieved and displayed in the wizard. You can then manually convert them, or save them for converting at a later time.

The following RDBMS products and versions are currently tested and supported by the MySQL Workbench Migration Wizard, although other RDBMS products can also be migrated with Section 10.2.3, “Migrating from unsupported (generic) databases”

- Microsoft SQL Server 2000, 2005, 2008, 2012

- Microsoft Access 2007 and greater
- MySQL Server 4.1 and greater as the source, and MySQL Server 5.1 and greater as the target
- PostgreSQL 8.0 and greater
- SQL Anywhere
- SQLite
- Sybase Adaptive Server Enterprise 15.x and greater

10.2.3 Migrating from unsupported (generic) databases

Most ODBC compliant databases may be migrated using the generic database support. In this case, code objects will not be retrieved from the source database; only tables and data.

When using the generic support, column data types are mapped using the following steps:

1. It searches for the first entry in the **Generic Datatype Mapping Table** for the source type name. If the length/scale ranges of the entry matches the source column, it will pick that type. Otherwise, it continues searching.
2. If no matches were found in the generic table, then it tries to directly map the source type to a MySQL type of the same name.
3. If the source type name doesn't match any of the MySQL data types, then it is not converted and an error is logged. From here you can specify the target datatype in the **Manual Object Editing** step of the wizard.

10.3 Conceptual DBMS equivalents

Table 10.1 Conceptual equivalents between supported DBMS products and MySQL

Concept	MS SQL Server	Sybase ASE	PostgreSQL	MySQL	Note
Authentication	Yes	Yes	Yes	Yes	
Auto_Increment	Yes	Yes	Yes	Yes	PostgreSQL uses sequences for Auto_Increment.
Backup	Yes	Yes	Yes	Yes	See MySQL Enterprise Backup
Catalog	Yes	Yes	Yes	N/A	You can map a catalog into a schema and drop the , use the owner as the schema name or merge the owner and object name together. ownerobject
Constraints	Yes	Yes	Yes	Yes	
Data Dictionary				N/A	
Database	Yes	Yes	Yes	Yes	
Database Instance					
Dump	Yes	Yes	Yes	Yes	`mysqldump`
Events	Yes	Yes	Yes	Yes	
Foreign Keys	Yes	Yes	Yes	Yes	

Concept	MS SQL Server	Sybase ASE	PostgreSQL	MySQL	Note
Full Text Search	Yes	Yes	Yes	Yes	In InnoDB as of MySQL Server 5.6, and in all versions of MyISAM
Index	Yes	Yes	Yes	Yes	
Information Schema	Yes	No	Yes	Yes	
Object Names Case Sensitivity	Depends on collation	Depends on collation	Mixed	Mixed	MySQL: sensitivity of database, table, and trigger names OS dependent; other object names are case insensitive. PostgreSQL: as specified in the SQL-99 standard, unquoted object names are treated as case insensitive while quoted object names are case sensitive. Unlike the standard, unquoted object names are converted to lowercase instead of uppercase.
Object Naming Conventions	Yes	Yes	Yes	Yes	
Packages	N/A	N/A	N/A	N/A	
Partitioning	Yes	Yes	Yes	Yes	
Performance Schema	N/A	N/A	Yes	Yes	
Permissions	Yes	Yes	Yes	Yes	
Primary Key	Yes	Yes	Yes	Yes	
Referential Integrity	Yes	Yes	Yes	Yes	Sybase ASE: referential integrity only through triggers.
Replication	Yes	Yes	Yes	Yes	
Role	Yes	Yes	Yes	N/A	Roles are not available in MySQL at the database level.
Schema	Yes	Yes*	Yes	Yes	Equivalent to database in MySQL. Sybase ASE: Schemata corresponds to user names.
Sequences	Yes*	Yes*	Yes	Yes*	Standalone sequence objects are not supported in MySQL. Similar functionality can be obtained with IDENTITY columns in MSSQL and AUTO_INCREMENT columns in MySQL
SQL Modes	Yes		Yes	Yes	SET_ANSI_* in MSSQL
Storage Engines	N/A	N/A	Yes*	Yes	PostgreSQL itself supports and uses only one storage engine (Postgresql). Other companies have added extra storage engines to PostgreSQL.
Stored Procedures	Yes	Yes	Yes	Yes	
Synonyms	N/A	N/A	N/A	N/A	

Concept	MS SQL Server	Sybase ASE	PostgreSQL	MySQL	Note
Table	Yes	Yes	Yes	Yes	
Tablespace	Yes	Yes*	Yes	N/A	MSSQL groups tables in schemata (unless referring to CREATE TABLESPACE). Sybase ASE: tables are grouped in schemata which are more like user names.
Temporary Tables	Yes	Yes	Yes	Yes	
Transactions	Yes	Yes	Yes	Yes	
Triggers	Yes	Yes	Yes	Yes	
UDFs	Yes	Yes	Yes	Yes	
Unicode	Yes	Yes	Yes	Yes	
Unique Key	Yes	Yes	Yes	Yes	
User	Yes	Yes	Yes	Yes	
Views	Yes	Yes	Yes	Yes	

Handling Microsoft SQL Server and MySQL structural differences

A Microsoft SQL Server database is made up of one catalog and one or more schemata. MySQL only supports one schema for each database (or rather, a MySQL database is a schema) so this difference in design must be planned for. The Migration Wizard must know how to handle the migration of schemata for the source (Microsoft SQL Server) database. It can either keep all of the schemata as they are (the Migration Wizard will create one database per schema), or merge them into a single MySQL database. Additional configure options include: either remove the schema names (the Migration Wizard will handle the possible name collisions that may appear along the way), and an option to add the schema name to the database object names as a prefix.

10.4 Microsoft Access Migration

Note

This feature was added in MySQL Workbench 6.2.0.

General Information

Microsoft Windows is required because Microsoft Access ODBC drivers are only available on Windows. As for the destination MySQL server, you can have it in the same local machine or elsewhere in your network.

Preparing a Microsoft Access Database for Migration

Microsoft Access stores relationship/foreign key information in an internal table called **MSysRelationships**. That table is protected against read access even to the Admin user, so if you try to migrate without opening up access to it, then you will get an error like this:

```
[42000] [Microsoft][ODBC Microsoft Access Driver] Record(s) cannot be read; no read permission on 'msysobjects
```

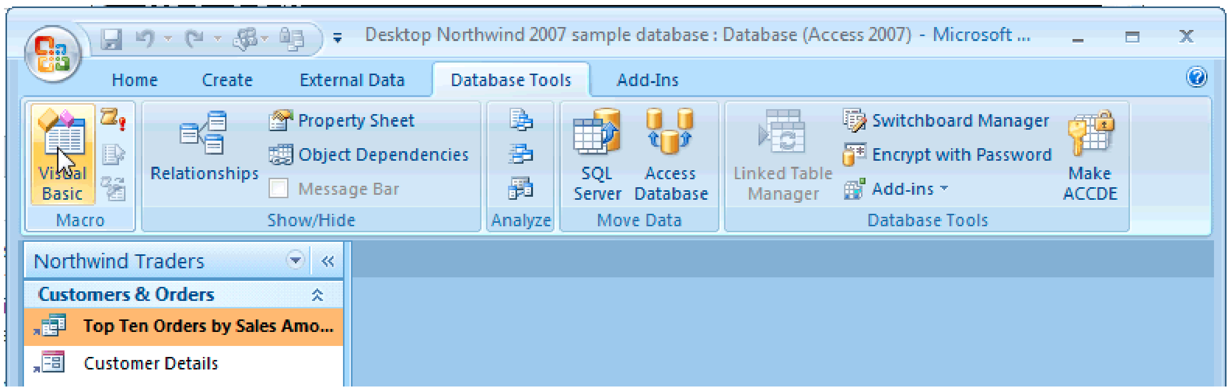
The steps to grant read access to the Admin are explained below.

Note

The screen shots in this documentation use Microsoft Access 2007.

- Open up database in Microsoft Access
- Under the **Database Tools** menu, click the **Visual Basic** macro button to open the Visual Basic (VB) console.

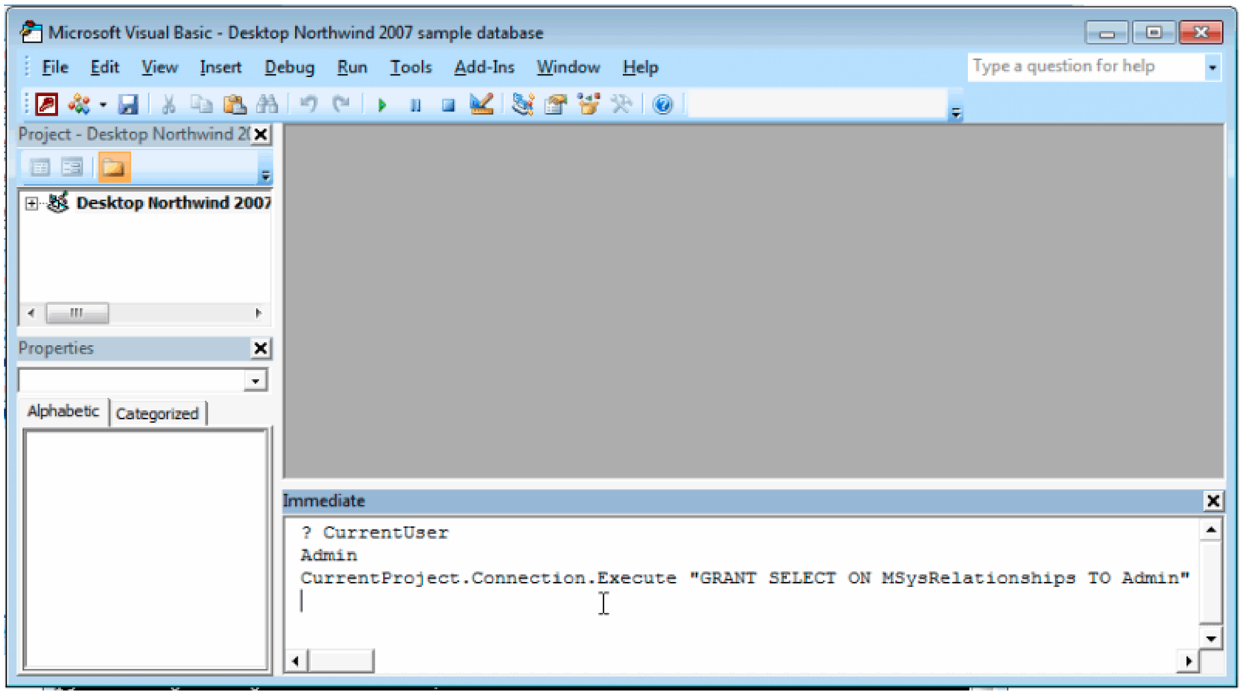
Figure 10.21 Locating the Visual Basic Macro Database Tool



- To confirm that you're logged in as the "Admin" user, locate the **Immediate** panel and type the "? CurrentUser" and press **Enter**. This should output "Admin" under "? CurrentUser" in the panel.
- Also in the **Immediate** panel, type the following command to grant access:

```
CurrentProject.Connection.Execute "GRANT SELECT ON MSysRelationships TO Admin"
```

Figure 10.22 GRANT SELECT ON MSysRelationships TO Admin


Microsoft Visual Basic - Desktop Northwind 2007 sample database
File Edit View Insert Debug Run Tools Add-Ins Window Help
Type a question for help
Project - Desktop Northwind 2(
Desktop Northwind 2007
Properties
Alphabetic
Categorized
Immediate

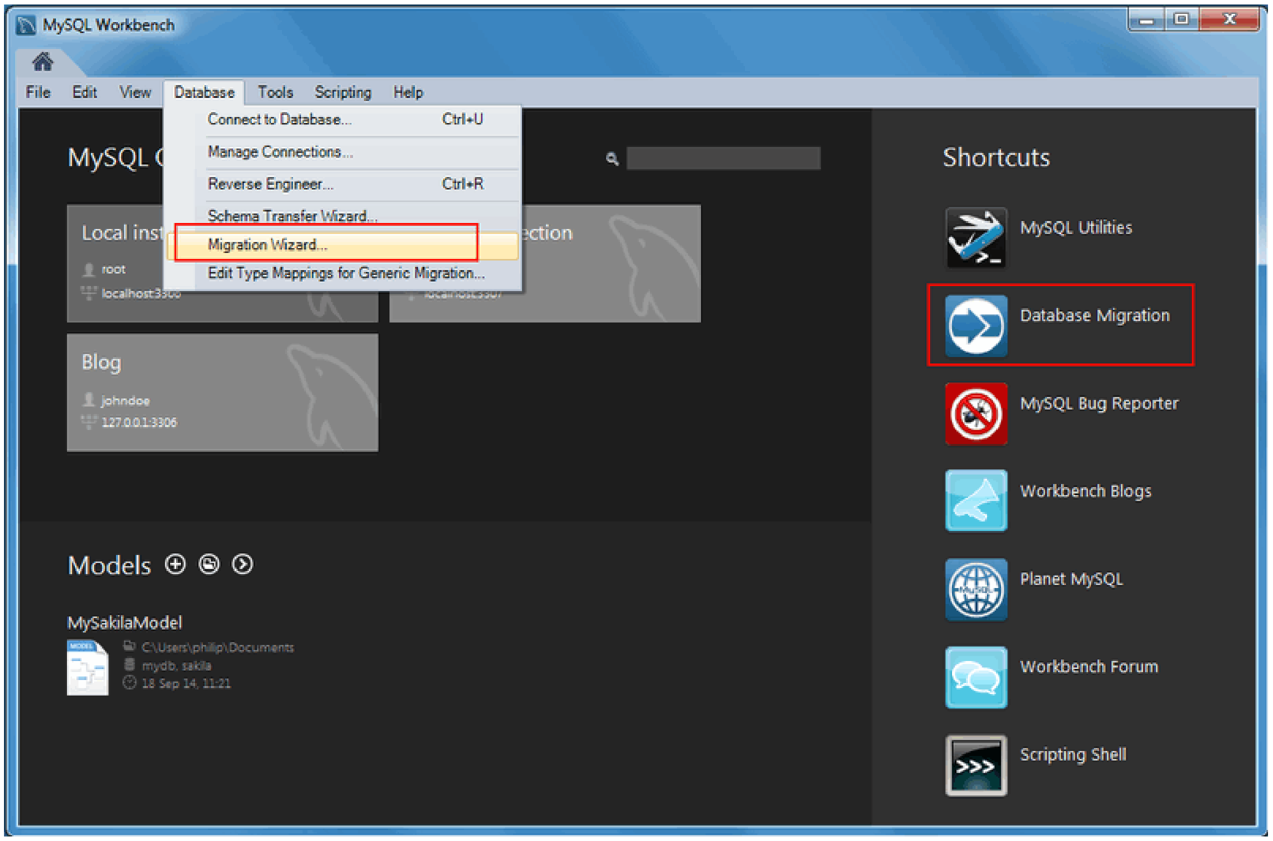
```
? CurrentUser
Admin
CurrentProject.Connection.Execute "GRANT SELECT ON MSysRelationships TO Admin"
```

- Quit the Microsoft Access application

Start the MySQL Workbench Migration Wizard

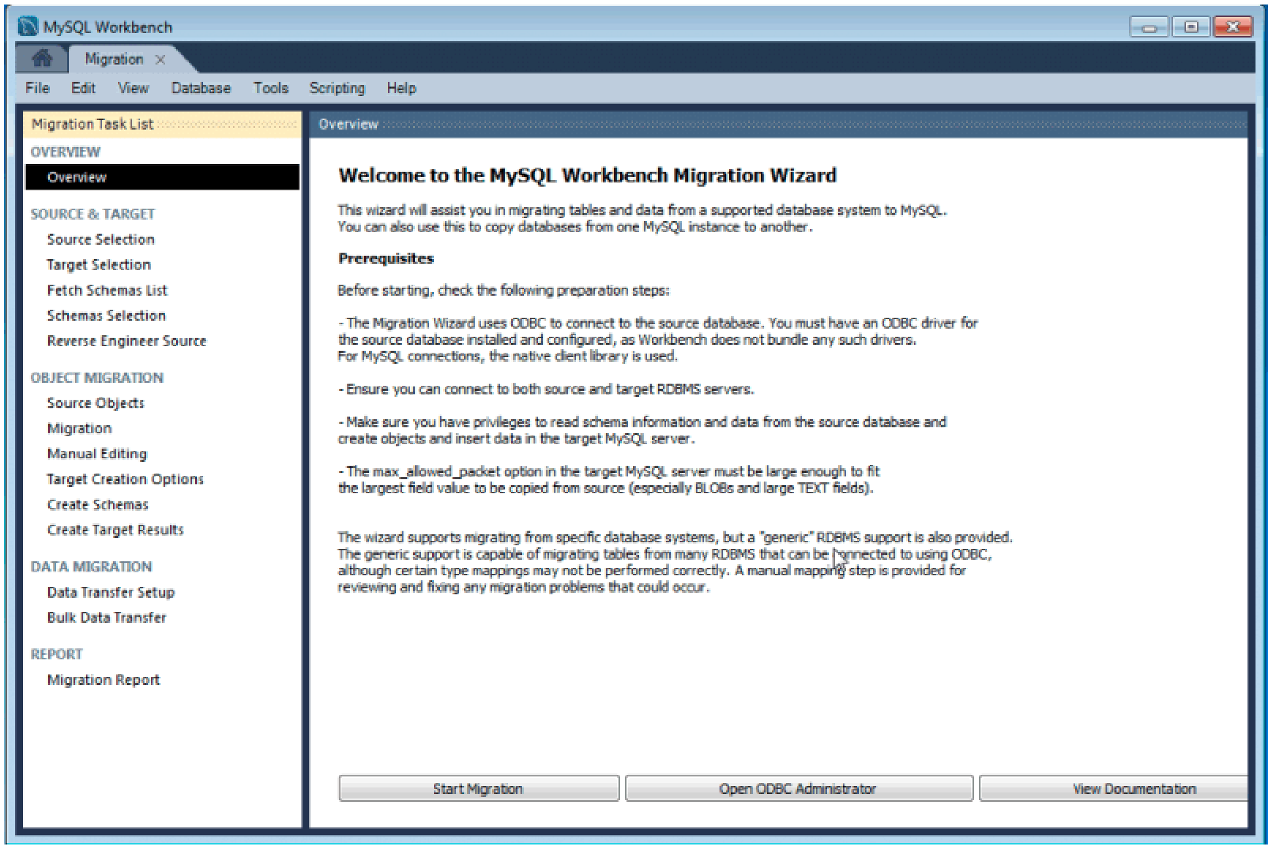
From the main MySQL Workbench screen you can start the Migration Wizard by clicking on the Database Migration launcher in the Workbench Central panel or through **Database**, **Migrate** from the main menu.

Figure 10.23 Start the Migration Wizard



A new tab showing the Overview page of the Migration Wizard should appear.

Figure 10.24 Migration Overview Page



Welcome to the MySQL Workbench Migration Wizard

This wizard will assist you in migrating tables and data from a supported database system to MySQL.
You can also use this to copy databases from one MySQL instance to another.

Prerequisites

Before starting, check the following preparation steps:

- The Migration Wizard uses ODBC to connect to the source database. You must have an ODBC driver for the source database installed and configured, as Workbench does not bundle any such drivers.
For MySQL connections, the native client library is used.

- Ensure you can connect to both source and target RDBMS servers.

- Make sure you have privileges to read schema information and data from the source database and create objects and insert data in the target MySQL server.

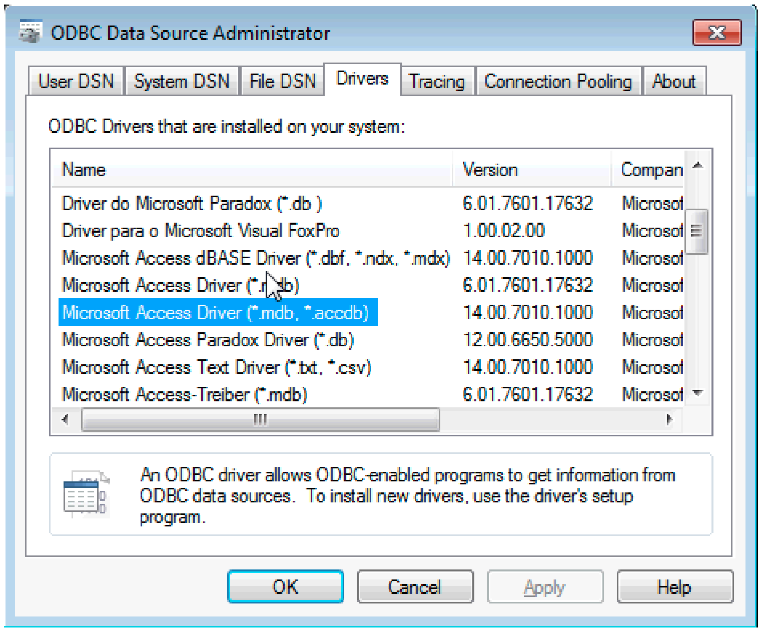
- The max_allowed_packet option in the target MySQL server must be large enough to fit the largest field value to be copied from source (especially BLOBs and large TEXT fields).

The wizard supports migrating from specific database systems, but a "generic" RDBMS support is also provided.
The generic support is capable of migrating tables from many RDBMS that can be connected to using ODBC, although certain type mappings may not be performed correctly. A manual mapping step is provided for reviewing and fixing any migration problems that could occur.

Setting Up ODBC Drivers

To check if you have the ODBC driver installed, click **Open ODBC Administrator** from the MySQL Workbench migration overview page to open the system ODBC tool. Then, select the **Drivers** tab.

Figure 10.25 Checking the ODBC Drivers for Access Support



ODBC Drivers that are installed on your system:

Name	Version	Compan
Driver do Microsoft Paradox (*.db)	6.01.7601.17632	Microsof
Driver para o Microsoft Visual FoxPro	1.00.02.00	Microsof
Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)	14.00.7010.1000	Microsof
Microsoft Access Driver (*.mdb)	6.01.7601.17632	Microsof
Microsoft Access Driver (*.mdb, *.accdb)	14.00.7010.1000	Microsof
Microsoft Access Paradox Driver (*.db)	12.00.6650.5000	Microsof
Microsoft Access Text Driver (*.txt, *.csv)	14.00.7010.1000	Microsof
Microsoft Access-Treiber (*.mdb)	6.01.7601.17632	Microsof

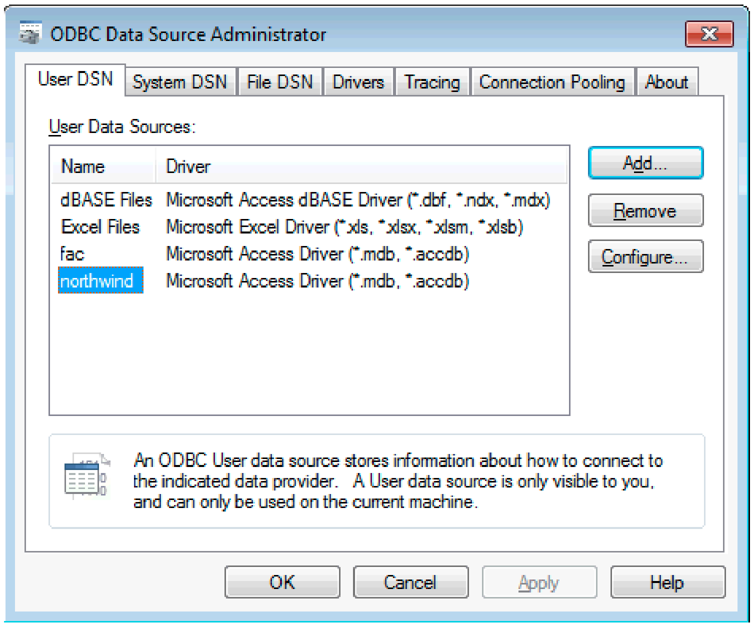
An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

Important

MySQL Workbench has both 32-bit and 64-bit executables. The ODBC drivers you use must be of the same architecture as the MySQL Workbench binaries you are using. Because Office 2007 and older was 32-bit only and even Office 2010 installs as 32-bit by default, you may need to install the 32-bit version of MySQL Workbench to migrate from Access, even if you have a 64-bit machine. If during migration you get an ODBC error about "architecture mismatch between the Driver and Application", you installed the wrong version of MySQL Workbench.

In the **User DSN** tab, click on **Add...** to create a DSN for your database file. For this example, we created one for the northwind sample database.

Figure 10.26 Adding a New DSN



User Data Sources:

Name	Driver
dBASE Files	Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)
Excel Files	Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)
fac	Microsoft Access Driver (*.mdb, *.accdb)
northwind	Microsoft Access Driver (*.mdb, *.accdb)

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.

Setting Up Source Parameters

Click on the **Start Migration** from the Overview page to advance to the **Source Selection** page. Here you need to provide the information about the Access database you are migrating from, the ODBC driver to use, and the parameters for the Access connection.

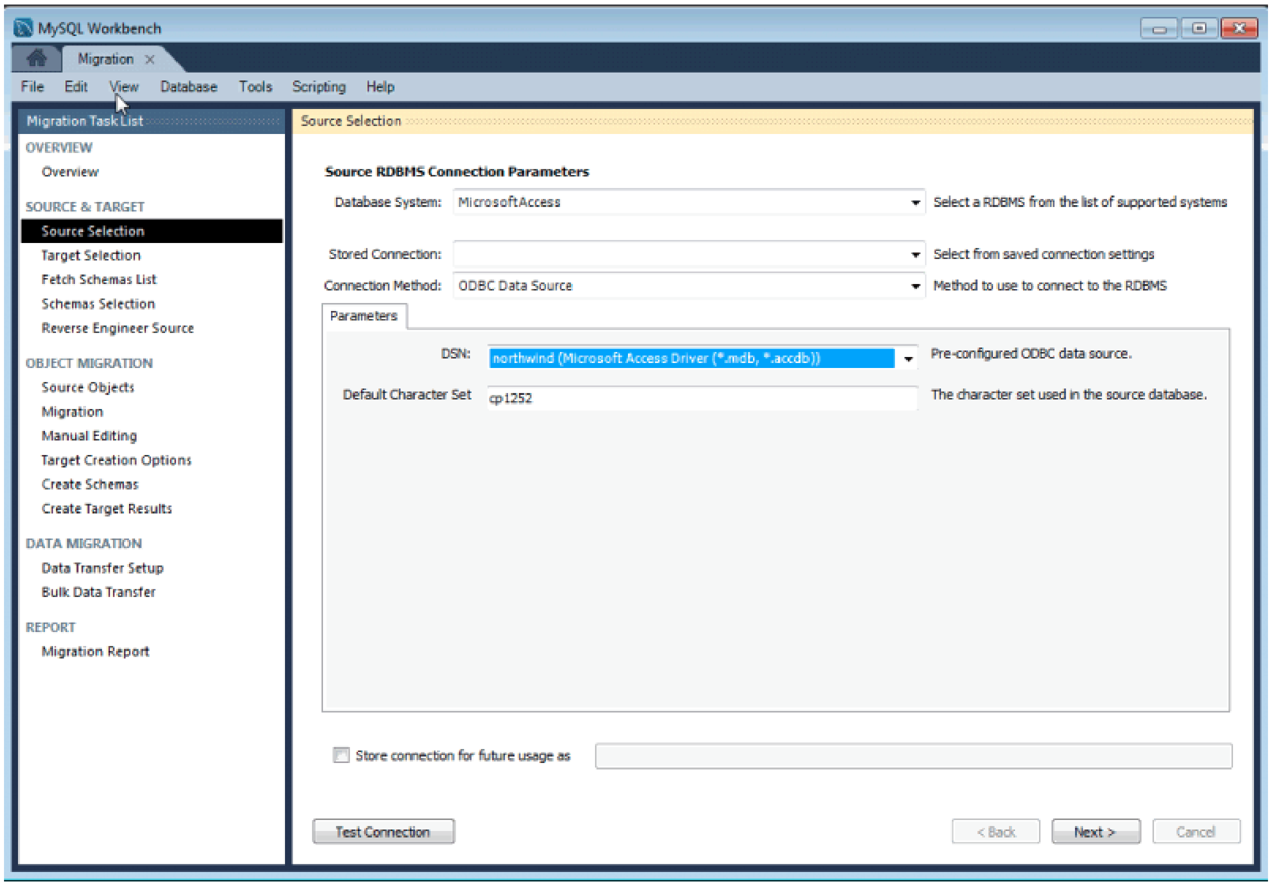
Open the **Database System** combo box for a list of supported RDBMSes, and select **Microsoft Access** from the list. There is another combo box below it named **Stored Connection**. It lists saved connection settings for that RDBMS. You can save connections by marking the checkbox at the bottom of the page, along with a name for the saved connection.

The next combo box selects the **Connection Method**. This time we are going to select *ODBC Data Source* from the list. This allows you to select pre-existing DSNs that you have configured in your system.

The **DSN** dropdown will have all DSNs you have defined in your system. Pick the one you created for the Access database being migrated from the list.

In the **Default Character Set** field you can select the character set of your database. If your Access version uses western/latin characters, you can leave the default *cp1252*. However, if you use a localized version of Access, such as Japanese, you must enter the correct character set used by your edition of Microsoft Office, otherwise the data will be copied incorrectly.

Figure 10.27 Access Source Selection



Source RDBMS Connection Parameters

Database System: MicrosoftAccess — Select a RDBMS from the list of supported systems

Stored Connection: — Select from saved connection settings

Connection Method: ODBC Data Source — Method to use to connect to the RDBMS

DSN: northwind (Microsoft Access Driver (*.mdb, *.accdb)) — Pre-configured ODBC data source.

Default Character Set: cp1252 — The character set used in the source database.

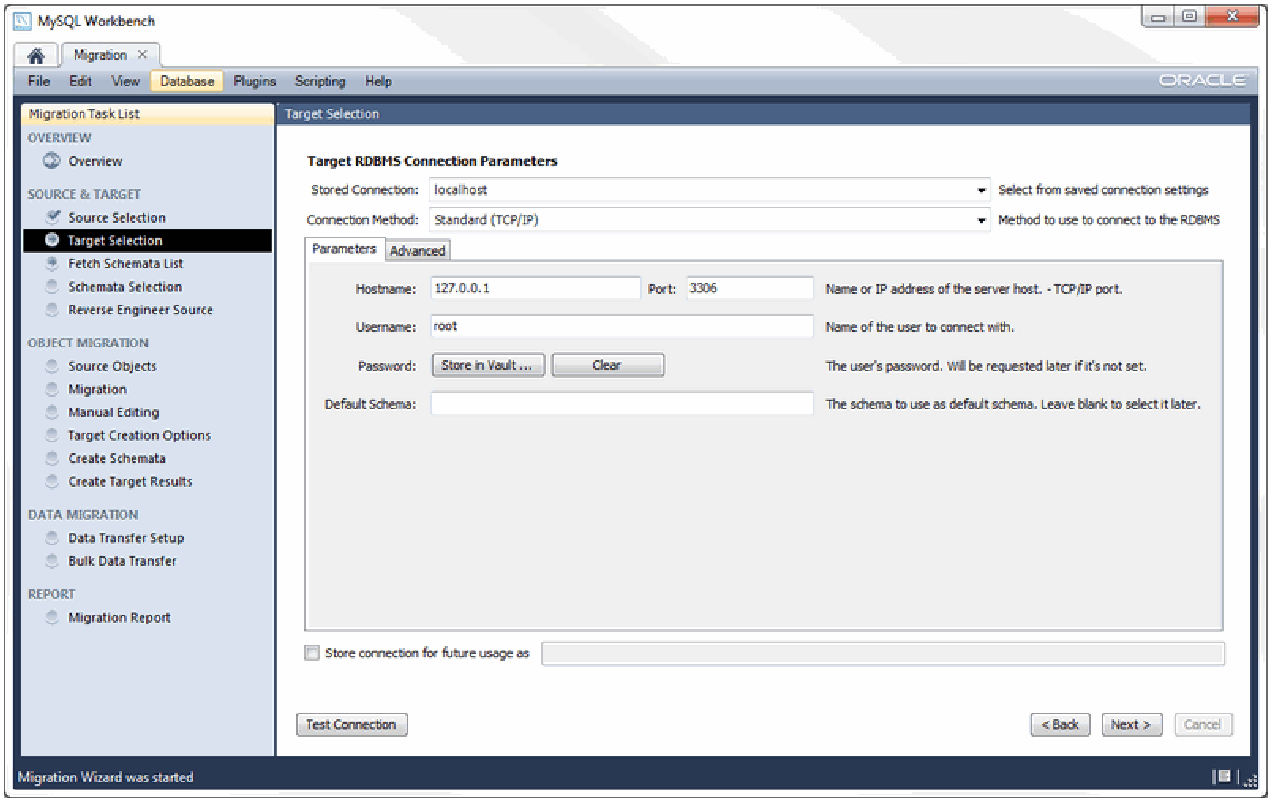
Store connection for future usage as

Lastly, click **Test Connection** to check whether an ODBC connection can be established. If you entered the correct parameters then you should see a message reporting a successful connection attempt.

Setting Up Target Parameters

Next, set up the target (MySQL) database parameters by defining the parameters that connect to your MySQL Server instance. When finished, click **Test Connection** to verify the connection definition.

Figure 10.28 Target Database Selection



Target RDBMS Connection Parameters

Stored Connection: localhost — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host. - TCP/IP port.

Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

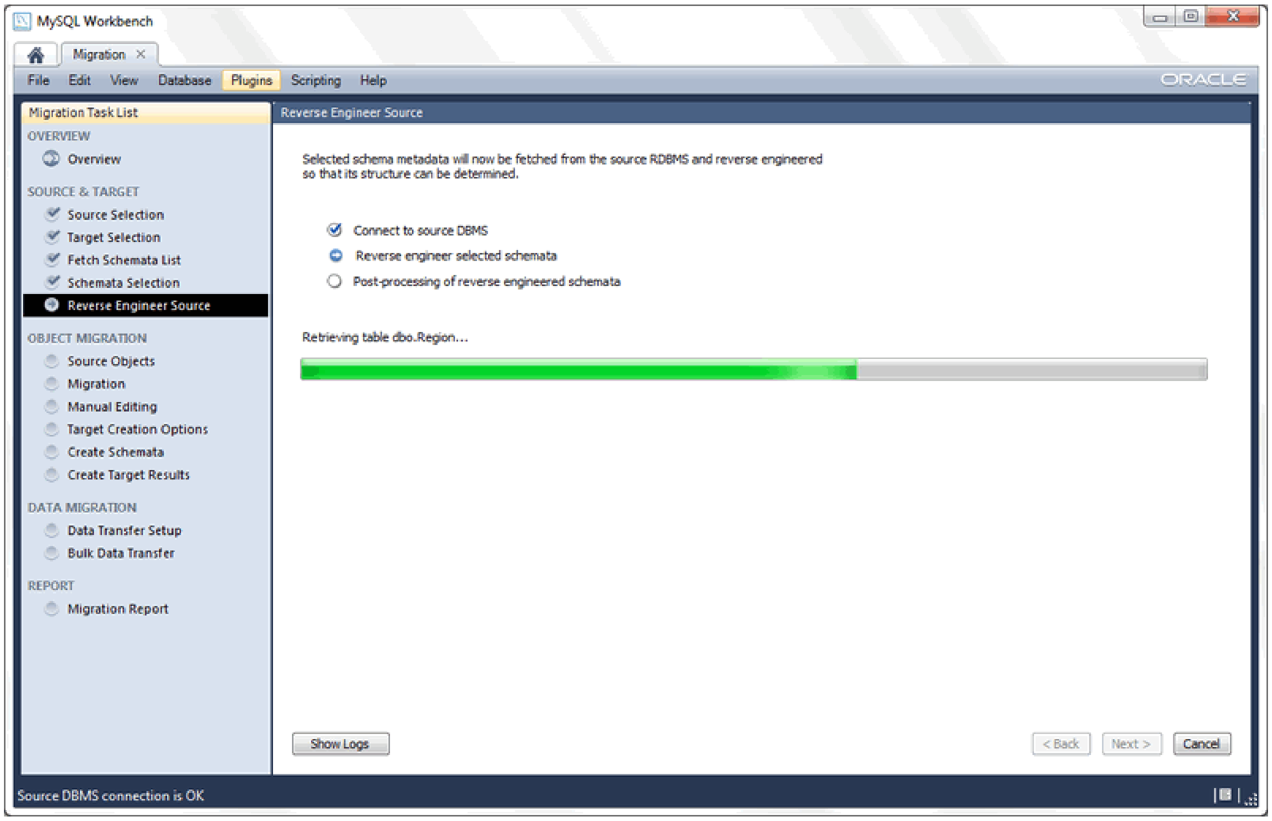
Store connection for future usage as

Migration Wizard was started

Select the objects to migrate

Next, you should see the reverse engineering of the selected database objects progress. At this point, the migration wizard is retrieving relevant information about the involved database objects (such as table names, table columns, primary and foreign keys, indices, triggers, views, and more). You will be presented a page showing the progress as shown below.

Figure 10.29 Reverse Engineer Source



Reverse Engineer Source

Selected schema metadata will now be fetched from the source RDBMS and reverse engineered so that its structure can be determined.

Connect to source DBMS

Reverse engineer selected schemata

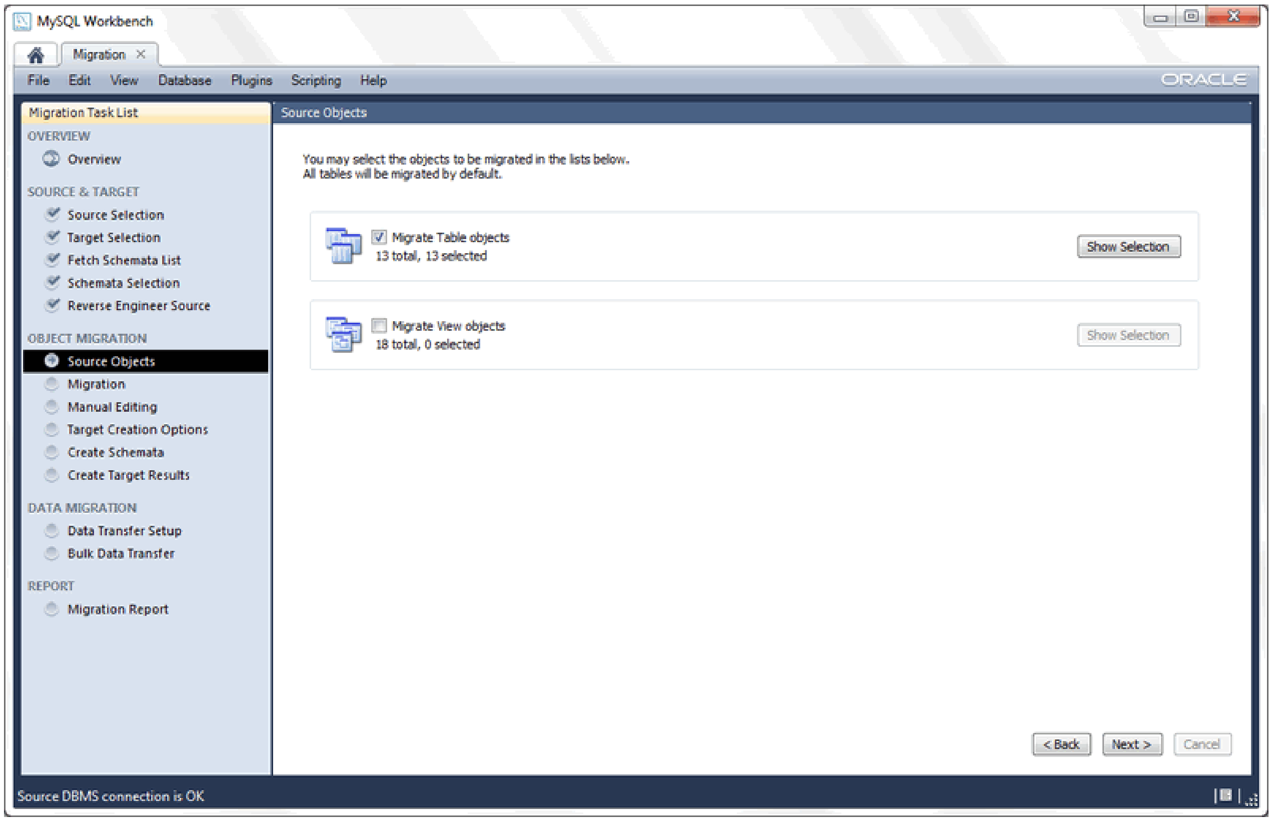
Post-processing of reverse engineered schemata

Retrieving table dbo.Region...

Source DBMS connection is OK

Wait for it to finish and verify that everything went well. Next, the **Source Objects** displays a list with the objects that were retrieved and are available for migration. It will look similar to:

Figure 10.30 Source Objects



You may select the objects to be migrated in the lists below.
All tables will be migrated by default.

[✓] Migrate Table objects
13 total, 13 selected

[ ] Migrate View objects
18 total, 0 selected

Source DBMS connection is OK

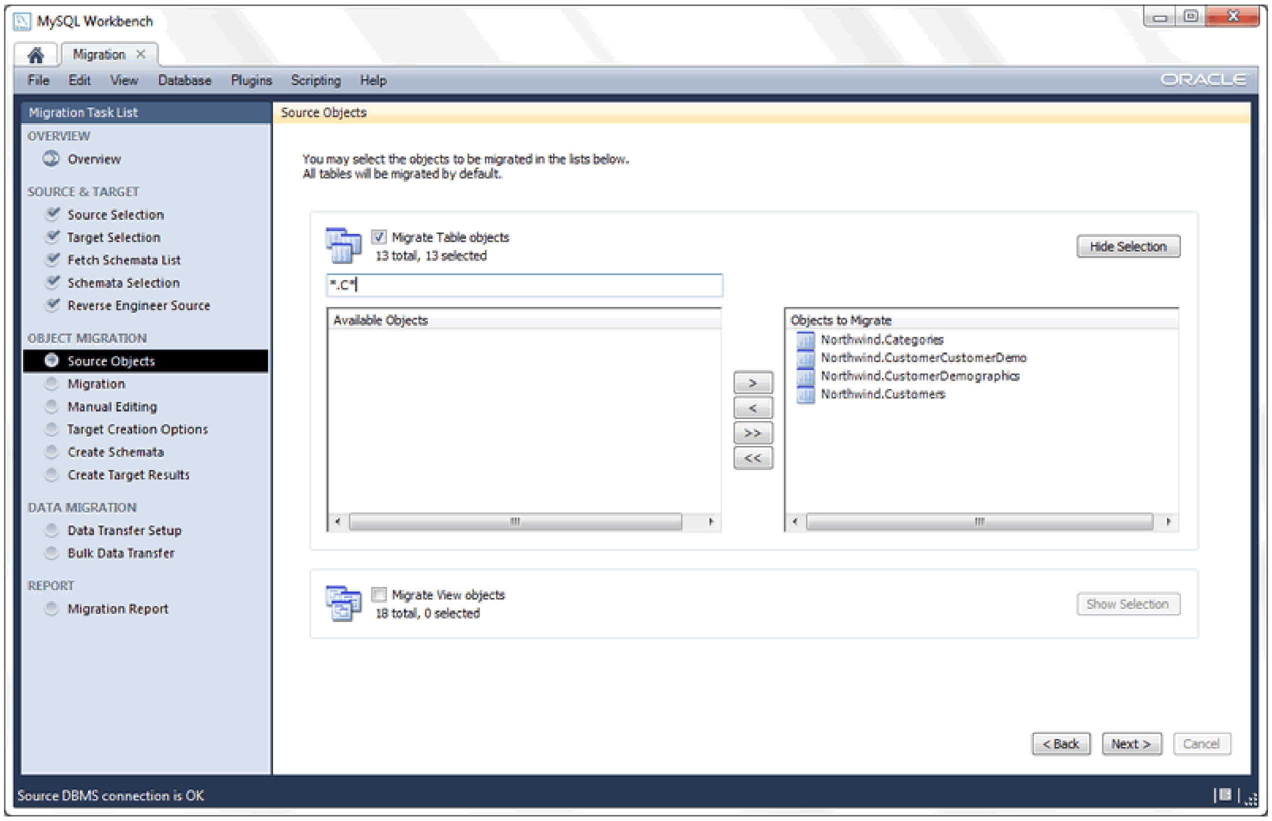
In the above example, the migration wizard discovered table and view objects for our source database. Only the table objects are selected by default for migration.

Note

You can also select the view objects but you must also provide their corresponding MySQL equivalent code later (no automatic migration is available for them) so our example will leave the views unchecked. The same applies for stored procedures, functions and triggers.

Click **Show Selection** to configure exactly which objects you want to migrate, as seen below:

Figure 10.31 Source Objects Selection



You may select the objects to be migrated in the lists below.
All tables will be migrated by default.

Migrate Table objects [✓]
13 total, 13 selected

.C

Available Objects

Objects to Migrate
Northwind.Categories
Northwind.CustomerCustomerDemo
Northwind.CustomerDemographics
Northwind.Customers

Migrate View objects
18 total, 0 selected

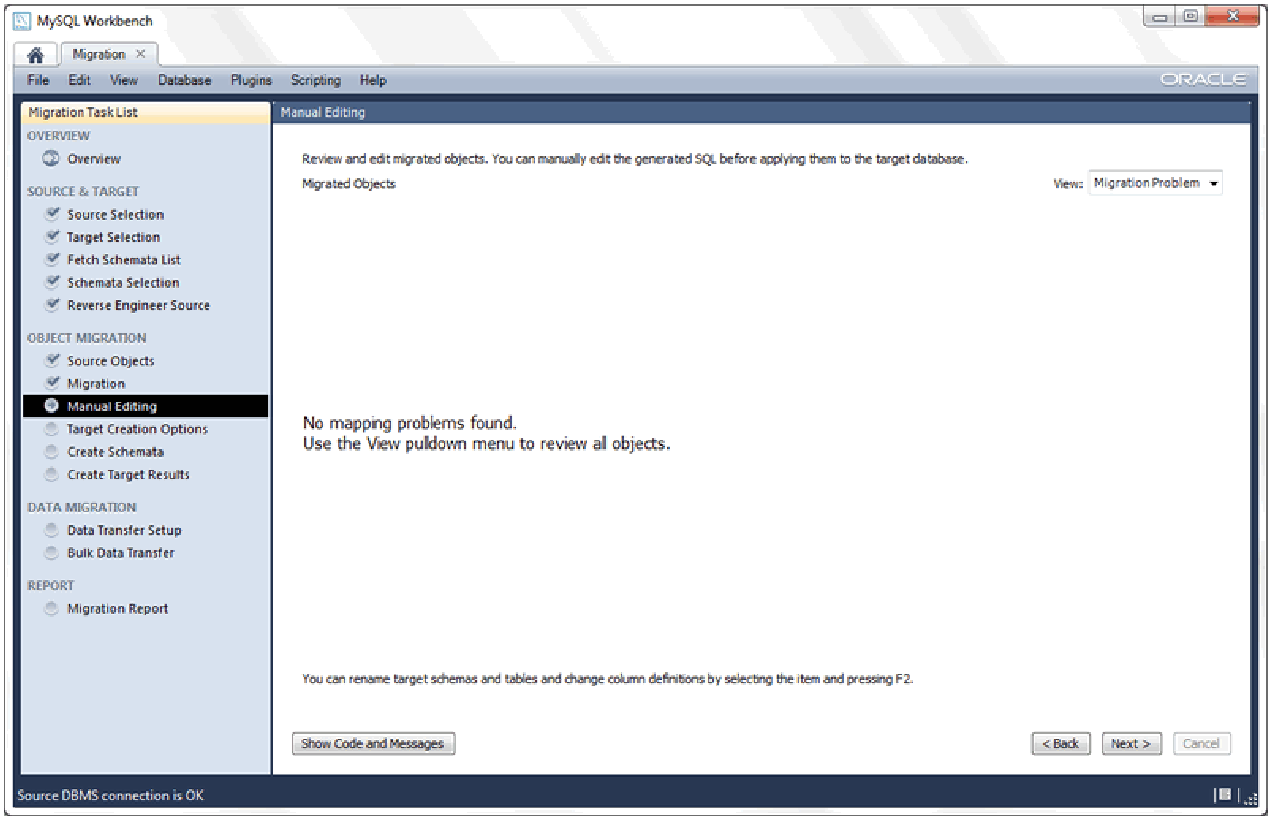
Source DBMS connection is OK

The objects on the right will be migrated. The filter box can filter the list (wildcards are allowed, as demonstrated above). By using the arrow buttons you can filter out the objects that you do not want to migrate. Before continuing, clear the filter text box to check the full list of the selected objects. Our example migrates all of the table objects so all of them are in the **Objects to Migrate** list, and the **Migrate Table Objects** checkbox is checked.

Review the proposed migration

At this point, the migration wizard converts the selected objects into their equivalent objects into the target MySQL server, and it also generates the MySQL code needed to create them. You might have to wait before the **Manual Editing** page is ready, but here is the initial page:

Figure 10.32 Manual Editing: Initial Page



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects

View: Migration Problem

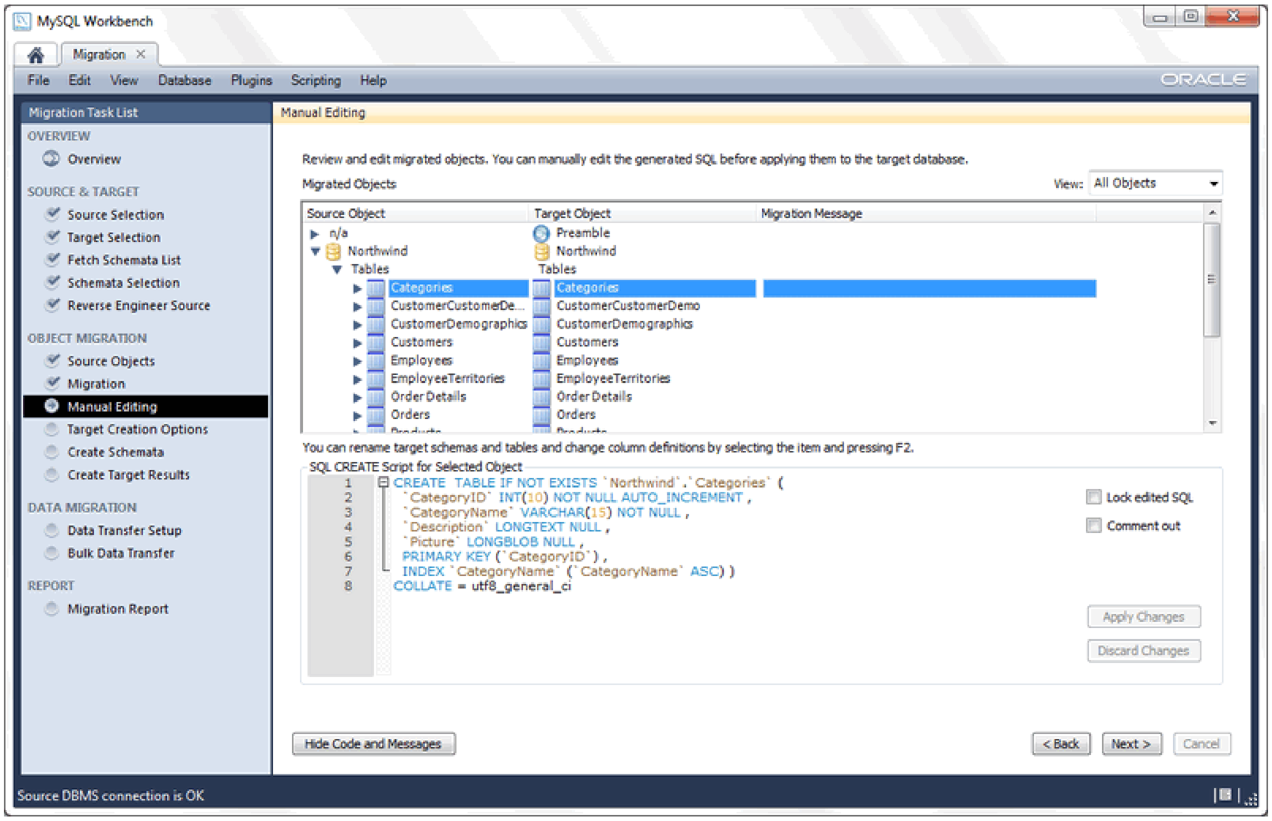
No mapping problems found.
Use the View pulldown menu to review all objects.

You can rename target schemas and tables and change column definitions by selecting the item and pressing F2.

Source DBMS connection is OK

The **View** combo box changes the way the migrated database objects are shown. Click **Show Code** to view and edit the generated MySQL code that corresponds to the selected object. Additionally, you can double-click on a row in the object tree to edit the object name, or double-click the database row to change its name.

Figure 10.33 Manual Editing: All Objects



Manual Editing

Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects

View: All Objects

Source Object	Target Object	Migration Message
n/a	Preamble	
Northwind	Northwind	
Tables	Tables	
Categories	Categories	
CustomerCustomerDe...	CustomerCustomerDemo	
CustomerDemographics	CustomerDemographics	
Customers	Customers	
Employees	Employees	
EmployeeTerritories	EmployeeTerritories	
Order Details	Order Details	
Orders	Orders	

You can rename target schemas and tables and change column definitions by selecting the item and pressing F2.

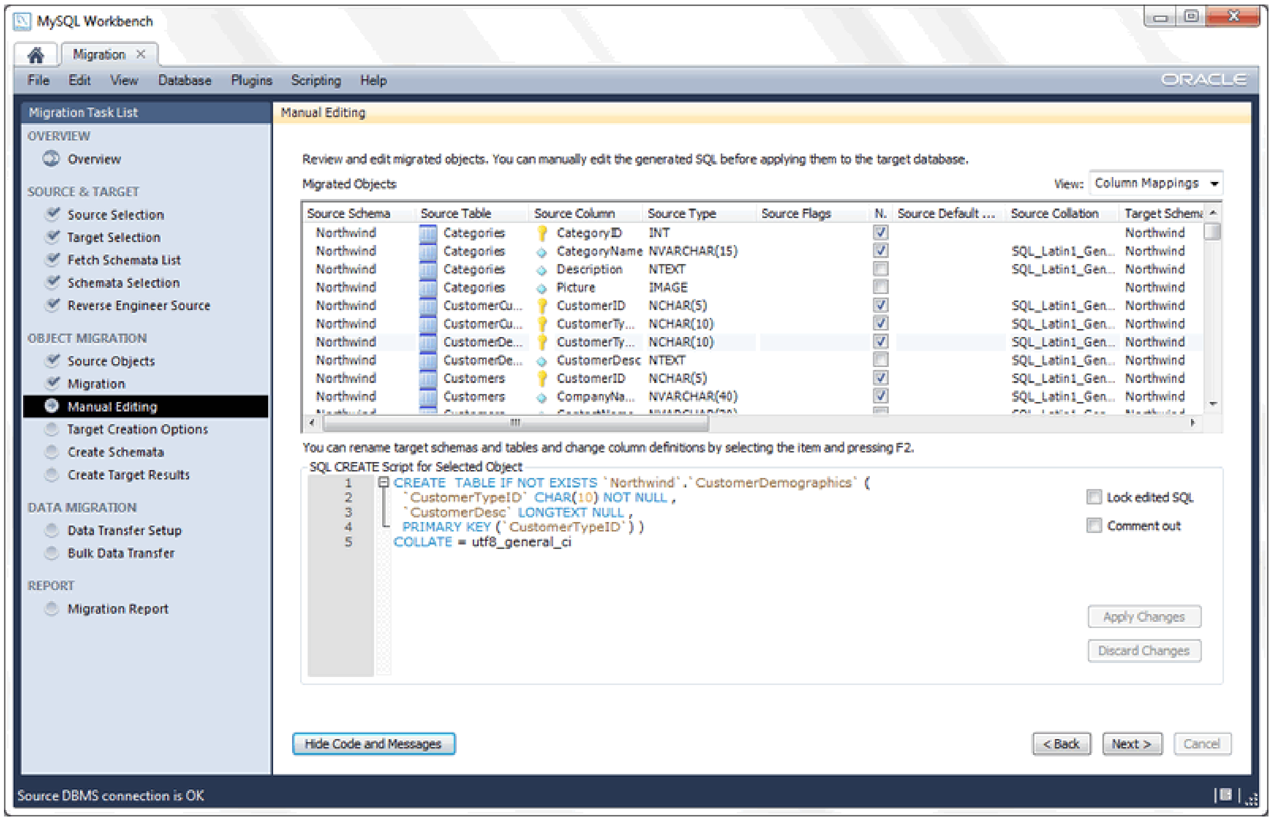
SQL CREATE Script for Selected Object

```
CREATE  TABLE IF NOT EXISTS `Northwind`.`Categories` (
  `CategoryID` INT(10) NOT NULL AUTO_INCREMENT ,
  `CategoryName` VARCHAR(15) NOT NULL ,
  `Description` LONGTEXT NULL ,
  `Picture` LONGBLOB NULL ,
  PRIMARY KEY (`CategoryID`) ,
  INDEX `CategoryName` (`CategoryName` ASC) )
COLLATE = utf8_general_ci
```

Source DBMS connection is OK

The **View** combo box also has a **Column Mappings** option. It shows the table columns and allows you to review and fix the mapping of column types, default values, and other attributes.

Figure 10.34 Manual Editing: Column Mappings



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects

View: Column Mappings

Source Schema	Source Table	Source Column	Source Type	Source Flags	N.	Source Default ...	Source Collation	Target Schema
Northwind	Categories	CategoryID	INT		✓			Northwind
Northwind	Categories	CategoryName	NVARCHAR(15)		✓		SQL_Latin1_Gen...	Northwind
Northwind	Categories	Description	NTEXT		☐		SQL_Latin1_Gen...	Northwind
Northwind	Categories	Picture	IMAGE		☐			Northwind
Northwind	CustomerCu...	CustomerID	NCHAR(5)		✓		SQL_Latin1_Gen...	Northwind
Northwind	CustomerCu...	CustomerTy...	NCHAR(10)		✓		SQL_Latin1_Gen...	Northwind
Northwind	CustomerDe...	CustomerTy...	NCHAR(10)		✓		SQL_Latin1_Gen...	Northwind
Northwind	CustomerDe...	CustomerDesc	NTEXT		☐		SQL_Latin1_Gen...	Northwind
Northwind	Customers	CustomerID	NCHAR(5)		✓		SQL_Latin1_Gen...	Northwind
Northwind	Customers	CompanyNa...	NVARCHAR(40)		✓		SQL_Latin1_Gen...	Northwind

You can rename target schemas and tables and change column definitions by selecting the item and pressing F2.

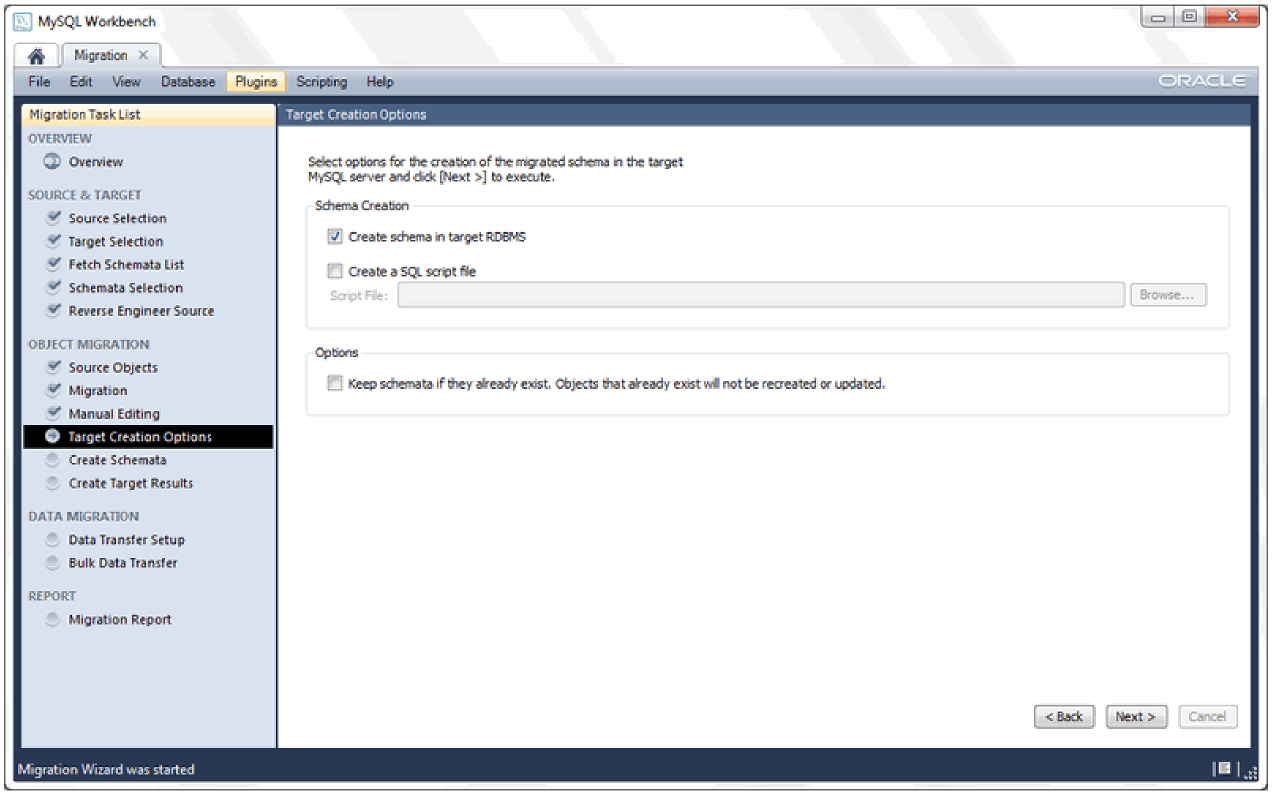
SQL CREATE Script for Selected Object

```
CREATE  TABLE IF NOT EXISTS `Northwind`.`CustomerDemographics` (
  `CustomerTypeID` CHAR(10) NOT NULL ,
  `CustomerDesc` LONGTEXT NULL ,
  PRIMARY KEY (`CustomerTypeID`) )
COLLATE = utf8_general_ci
```

Create the database objects

Next is the **Target Creation Options** page:

Figure 10.35 Target Creation Options



Select options for the creation of the migrated schema in the target MySQL server and click [Next >] to execute.

Schema Creation

- [✓] Create schema in target RDBMS
- [ ] Create a SQL script file

Script File:

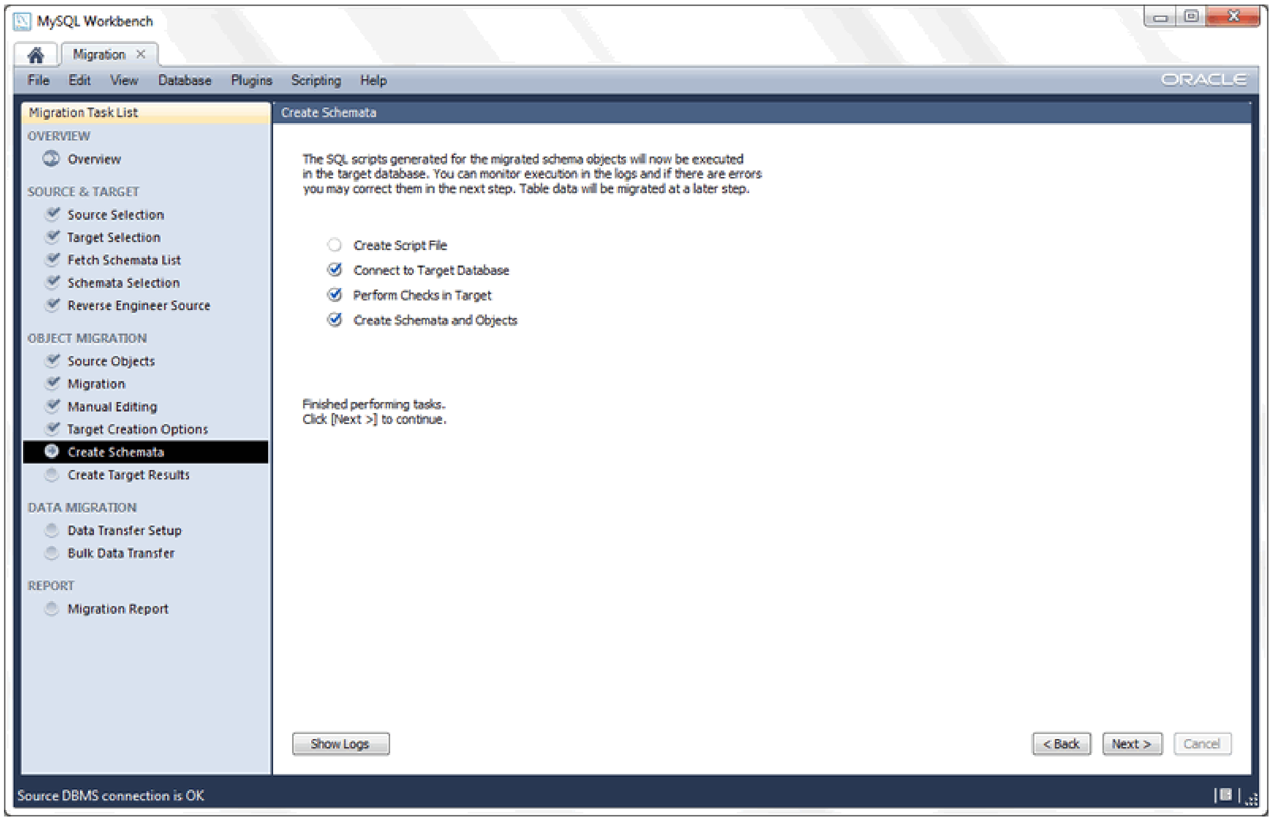
Options

- [ ] Keep schemata if they already exist. Objects that already exist will not be recreated or updated.

Migration Wizard was started

Here are options for executing the generated code in the target RDBMS (your MySQL instance from the second step), or you can dump it to an SQL script file. Leave it as shown above and move to the next page. The migrated SQL code will be executed on the target MySQL server. You can view its progress in the **Create Schemata** page:

Figure 10.36 Create Schemata



The SQL scripts generated for the migrated schema objects will now be executed in the target database. You can monitor execution in the logs and if there are errors you may correct them in the next step. Table data will be migrated at a later step.

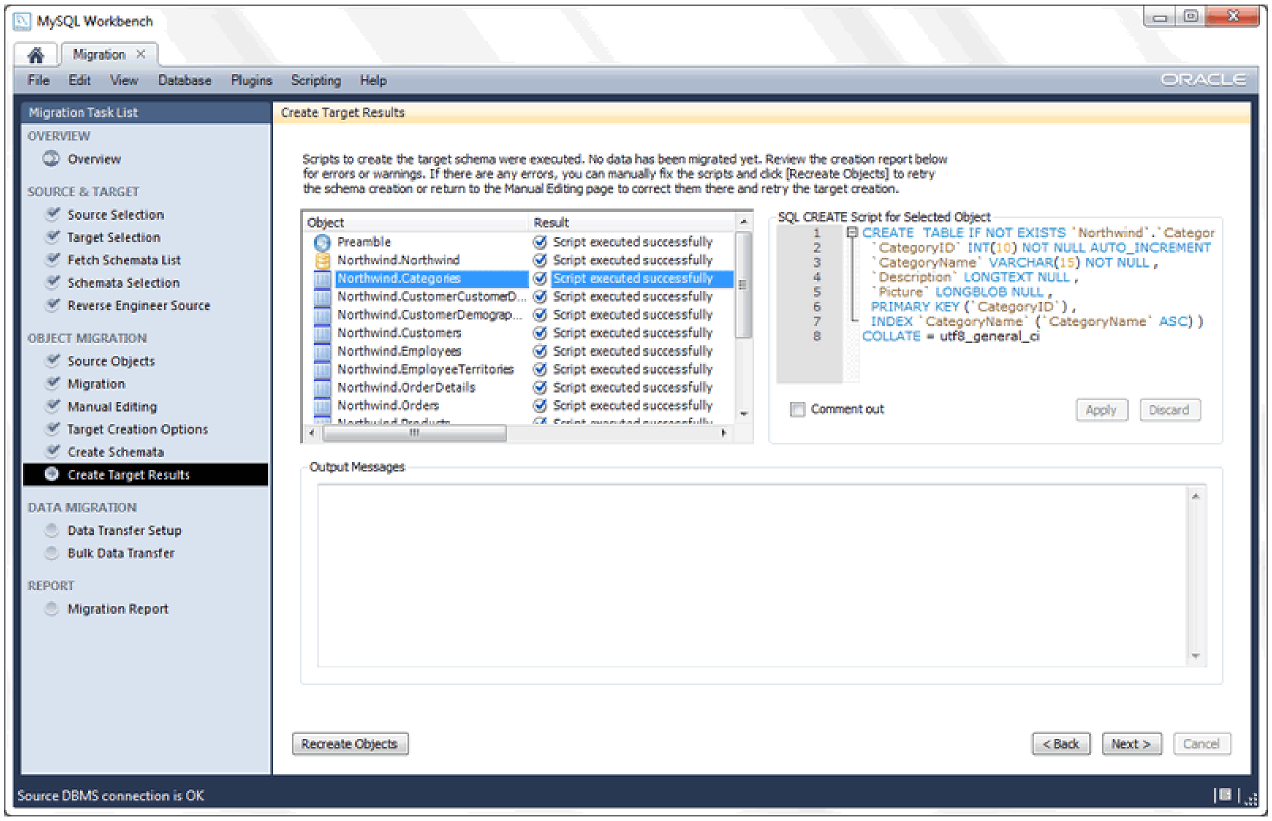
Create Script File
Connect to Target Database
Perform Checks in Target
Create Schemata and Objects

Finished performing tasks.
Click [Next >] to continue.

Source DBMS connection is OK

Once the creation of the schemata and objects finishes, you can move to the **Create Target Results** page. It presents a list of created objects and includes any generated errors while they were created. It will look similar to:

Figure 10.37 Create Target Results



Create Target Results

Scripts to create the target schema were executed. No data has been migrated yet. Review the creation report below for errors or warnings. If there are any errors, you can manually fix the scripts and click [Recreate Objects] to retry the schema creation or return to the Manual Editing page to correct them there and retry the target creation.

Object	Result
Preamble	✓ Script executed successfully
Northwind.Northwind	✓ Script executed successfully
Northwind.Categories	✓ Script executed successfully
Northwind.CustomerCustomerD...	✓ Script executed successfully
Northwind.CustomerDemograp...	✓ Script executed successfully
Northwind.Customers	✓ Script executed successfully
Northwind.Employees	✓ Script executed successfully
Northwind.EmployeeTerritories	✓ Script executed successfully
Northwind.OrderDetails	✓ Script executed successfully
Northwind.Orders	✓ Script executed successfully

SQL CREATE Script for Selected Object

```
CREATE  TABLE IF NOT EXISTS `Northwind`.`Categor
  `CategoryID` INT(10) NOT NULL AUTO_INCREMENT
  `CategoryName` VARCHAR(15) NOT NULL ,
  `Description` LONGTEXT NULL ,
  `Picture` LONGBLOB NULL ,
  PRIMARY KEY (`CategoryID`) ,
  INDEX `CategoryName` (`CategoryName` ASC) )
COLLATE = utf8_general_ci
```

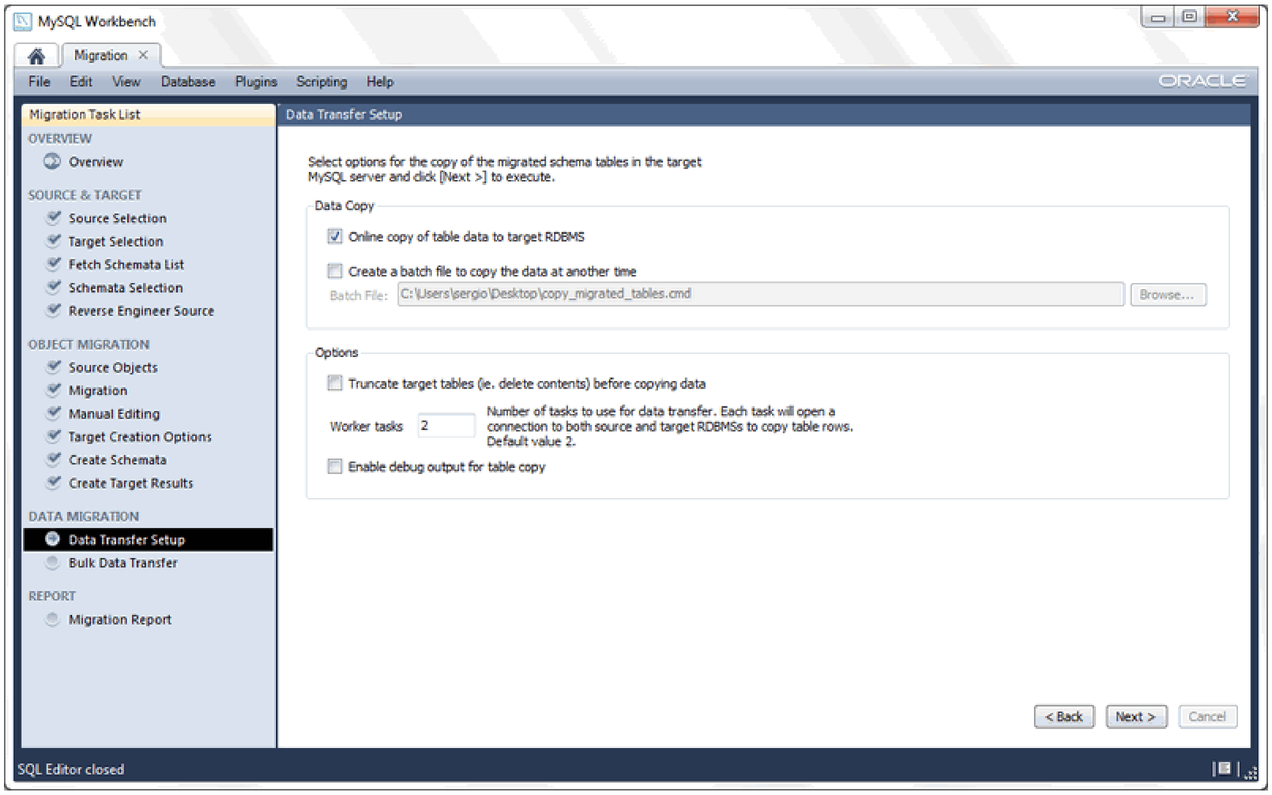
Source DBMS connection is OK

You can edit the migration code using the code box to the right, and save your changes by clicking **Apply**. If edits were made, you still need to recreate the objects with the modified code in order to perform the changes. This is done by clicking **Recreate Objects**. In this tutorial we are not changing anything, so leave the code as it is, and continue on to the **Data Transfer Setup** page.

Transfer the data to the MySQL database

The next step transfers data from the source Access database into your newly created target MySQL database. The Data Transfer Setup page allows you to configure this process.

Figure 10.38 Data Transfer Setup



Data Transfer Setup

Select options for the copy of the migrated schema tables in the target MySQL server and click [Next >] to execute.

Data Copy

[✓] Online copy of table data to target RDBMS

[ ] Create a batch file to copy the data at another time

Batch File: C:\Users\sergio\Desktop\copy_migrated_tables.cmd Browse...

Options

[ ] Truncate target tables (ie. delete contents) before copying data

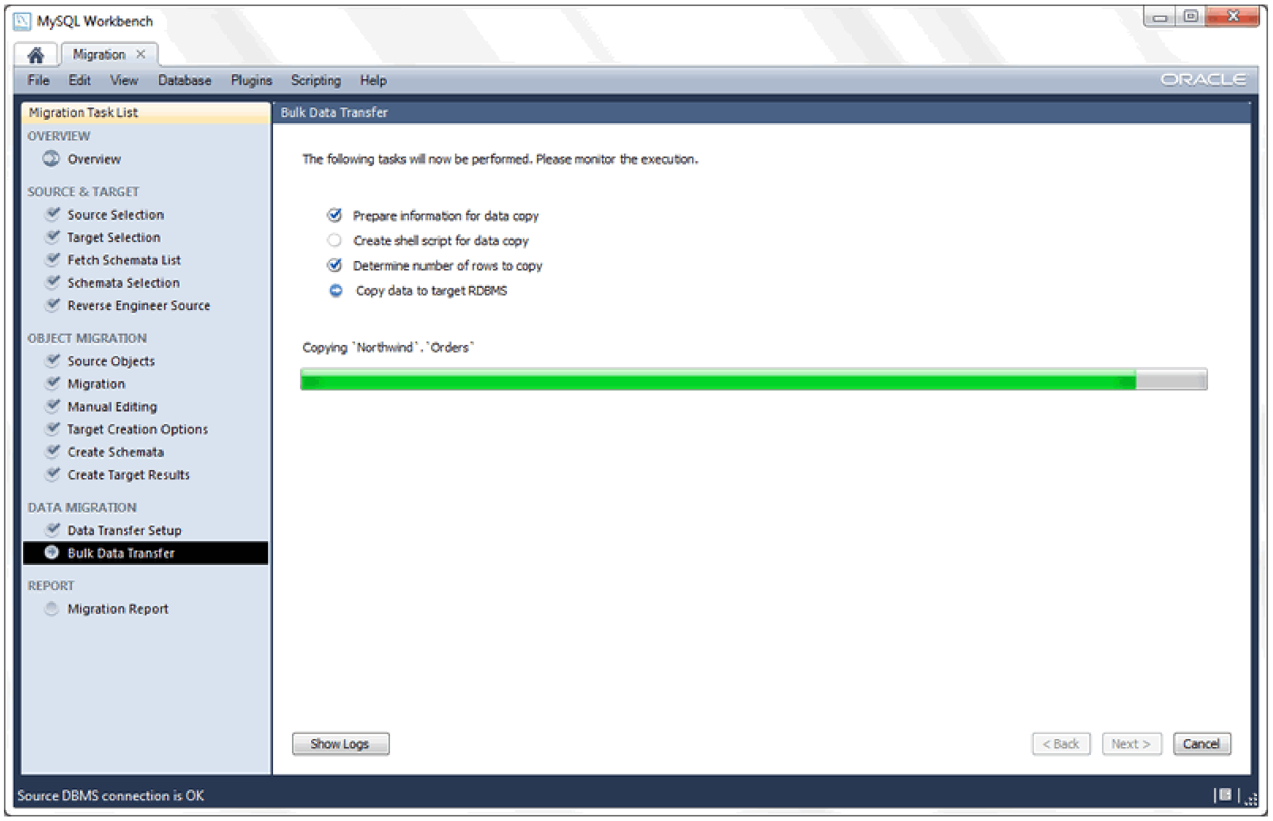
Worker tasks 2 — Number of tasks to use for data transfer. Each task will open a connection to both source and target RDBMSs to copy table rows. Default value 2.

[ ] Enable debug output for table copy

There are two sets of options here. The first allows you to perform a live transference and/or to dump the data into a batch file that you can execute later. The other set of options allows you to alter this process.

This tutorial uses the default values for the options in this page as shown in the above screenshot. Next, the data is transferred. At this point the corresponding progress page will look familiar:

Figure 10.39 Bulk Data Transfer



The following tasks will now be performed. Please monitor the execution.

- ✓ Prepare information for data copy
- ○ Create shell script for data copy
- ✓ Determine number of rows to copy
- ● Copy data to target RDBMS

Copying `Northwind`.`Orders`

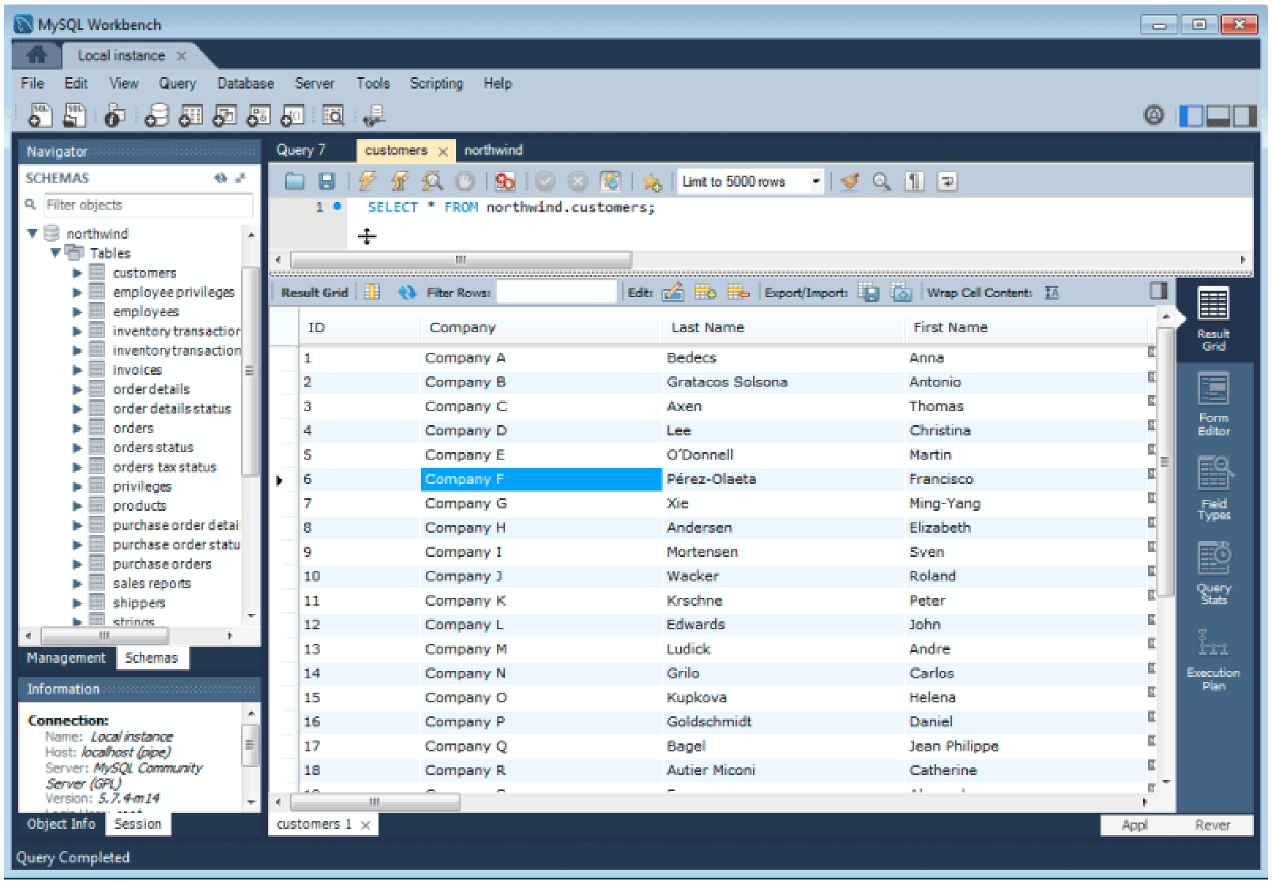
Source DBMS connection is OK

Once it finishes, move to the next page. You will be presented a report page summarizing the whole process. Now, review and click **Finish** to close the wizard.

Verification

Now that the Northwind database was successfully migrated, next we will view the results. Open an SQL Editor that is associated with your MySQL Server instance, and then query the Northwind database. You can try something like "SELECT * FROM Northwind.customers":

Figure 10.40 Verify Your Results



```
SELECT * FROM northwind.customers;
```

ID	Company	Last Name	First Name
1	Company A	Bedecs	Anna
2	Company B	Gratacos Solsona	Antonio
3	Company C	Axen	Thomas
4	Company D	Lee	Christina
5	Company E	O'Donnell	Martin
6	Company F	Pérez-Olaeta	Francisco
7	Company G	Xie	Ming-Yang
8	Company H	Andersen	Elizabeth
9	Company I	Mortensen	Sven
10	Company J	Wacker	Roland
11	Company K	Krschne	Peter
12	Company L	Edwards	John
13	Company M	Ludick	Andre
14	Company N	Grilo	Carlos
15	Company O	Kupkova	Helena
16	Company P	Goldschmidt	Daniel
17	Company Q	Bagel	Jean Philippe
18	Company R	Autier Miconi	Catherine

Connection:
Name: Local instance
Host: localhost (pipe)
Server: MySQL Community Server (GPL)
Version: 5.7.4-m14

Query Completed

10.5 Microsoft SQL Server migration

The MySQL Workbench Migration Wizard is tested against the following Microsoft SQL Server versions: 2000, 2005, 2008, and 2012.

10.5.1 Preparations

To be able to migrate from Microsoft SQL Server, ensure the following:

- The source SQL Server instance is running, and accepts TCP connections.
- You know the IP and port of the source SQL server instance. If you will be migrating using a Microsoft ODBC driver for SQL Server (the default in Windows), you will need to know the host and the name of the SQL Server instance.
- Make sure that the SQL Server is reachable from where you will be running MySQL Workbench. More specifically, check the firewall settings.
- Make sure that the user account has proper privileges to the database that will be migrated.

10.5.2 Drivers

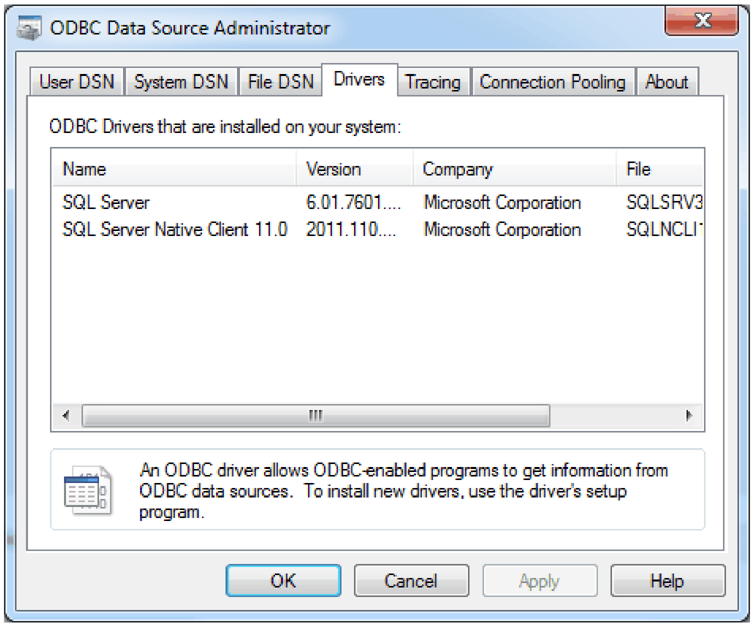
Microsoft Windows does not require additional drivers to be installed and configured, but Linux (and OS X) do. The following sections include specific instructions for each type of system.

10.5.2.1 Windows

Microsoft Windows XP and newer includes at least one ODBC driver for Microsoft SQL Server, so additional actions are likely not required on your system. Multiple SQL Server driver options exist, as described below.

You can check your ODBC driver information by starting the Windows ODBC Data Source Administrator that is linked from the MySQL Workbench migration wizard's home page. Alternatively, open a Windows terminal and execute `odbcad32.exe`. Open the **Drivers** tab to see something similar to:

Figure 10.41 Windows ODBC Data Source Administrator: SQL Server Drivers



ODBC Drivers that are installed on your system:

Name	Version	Company	File
SQL Server	6.01.7601....	Microsoft Corporation	SQLSRV3
SQL Server Native Client 11.0	2011.110....	Microsoft Corporation	SQLNCLI

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

Common ODBC drivers available on Windows are:

- **SQL Driver**: preinstalled on Windows, but is limited to the functionality provided by SQL Server 2000. It functions okay if your database does not use features and data types introduced after SQL Server 2000, so it should be enough for you if your database does not make use of the new features and data types introduced after this SQL Server version.

- **SQL Server Native Client XX.X**: if you have an SQL Server instance on the same machine as MySQL Workbench, then you will also have this additional driver. This comes with SQL Server and fully supports the companion SQL Server version. If this is not on your system then you can download and install this it from Microsoft. For example, download the Microsoft SQL Server 2014 Feature Pack to install the Native Client that supports SQL Server 2014 and earlier.

Note

XX.X represents the major version number for SQL Server, so an actual name might be "SQL Server Native Client 11.0".

Decide which driver you want to use, and remember its name as shown in the ODBC Data Source Administrator. This specific name is used in MySQL Workbench to connect your SQL Server instance.

Jump to the documentation titled Section 10.5.3, “Connection Setup”.

10.5.2.2 Linux

Setting up drivers on Linux.

FreeTDS

FreeTDS version 0.92 or greater is required. Many distributions ship older versions of FreeTDS, so it may need to be installed separately. Additionally, the FreeTDS version provided by distributions may also be compiled for the wrong ODBC library (usually to unixODBC instead of iODBC, which MySQL Workbench uses). Because of that, you will probably need to build this library yourself.

A script is provided to compile FreeTDS using the options required for MySQL Workbench. You can find it at `/usr/share/mysql-workbench/extras/build_freetds.sh` on Linux or `MySQLWorkbench.app/Contents/SharedSupport/build_freetds.sh` on OS X. To use it, follow these steps:

Using FreeTDS with iODBC

When compiling FreeTDS for use with iODBC (the default with the official binaries), it must be compiled with the `--enable-odbc-wide` command line. Failing to do so will result in crashes and other unpredictable errors. The provided `build_freetds.sh` script does this for you.

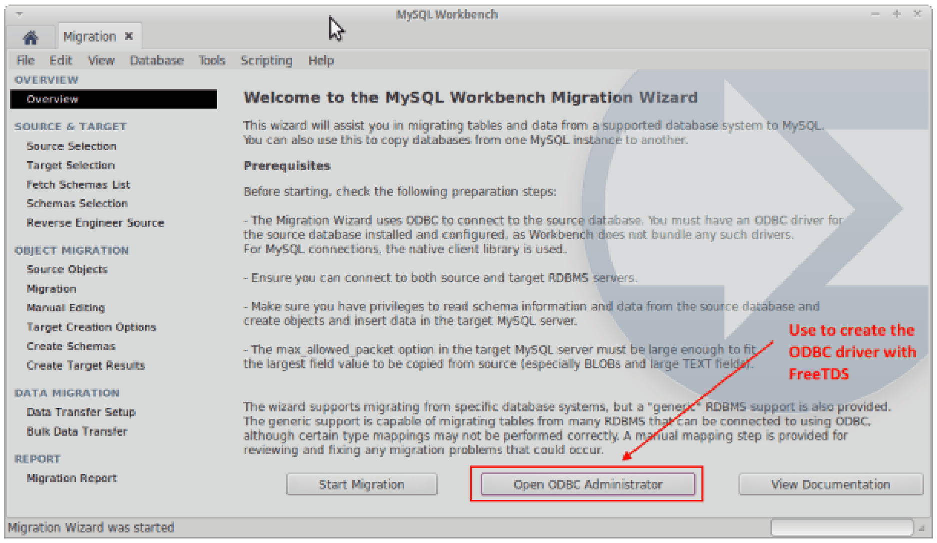
1. For compiling, make sure you have the iODBC headers installed. For Linux, the name depends on your system's package manager but common names are `libiodbc-devel` (RPM based systems) or `libiodbc2-dev` (Debian based systems). For OS X, the headers come with the system and no additional action is required for this step.

   Note

   If you are using Oracle Enterprise Linux, RedHat, CentOS, and similar, you must have the EPEL repository set up in yum for it to find the `libiodbc-devel` package. For additional information about this step, see Installing Oracle Enterprise Linux and similar.

2. `mkdir ~/freetds` to create a directory - within the users home directory.
3. Copy the `build_freetds.sh` script to `~/freetds`
4. Get the latest FreeTDS sources from ftp://ftp.freetds.org/pub/freetds/ and place the `.tar.gz` source file into the `~/freetds` directory. Make sure to get FreeTDS version 0.92 or newer.
5. `cd ~/freetds`
6. Execute `build_freetds.sh`
7. After compilation is done, install it using `make install` from the path given by the script.
8. Install the driver using ODBC Administrator so that the ODBC subsystem recognizes it. Open ODBC Administrator from the migration tab in MySQL Workbench.

Figure 10.42 Open the ODBC Administrator



Welcome to the MySQL Workbench Migration Wizard

This wizard will assist you in migrating tables and data from a supported database system to MySQL.
You can also use this to copy databases from one MySQL instance to another.

Prerequisites

Before starting, check the following preparation steps:

- The Migration Wizard uses ODBC to connect to the source database. You must have an ODBC driver for the source database installed and configured, as Workbench does not bundle any such drivers.
For MySQL connections, the native client library is used.

- Ensure you can connect to both source and target RDBMS servers.

- Make sure you have privileges to read schema information and data from the source database and create objects and insert data in the target MySQL server.

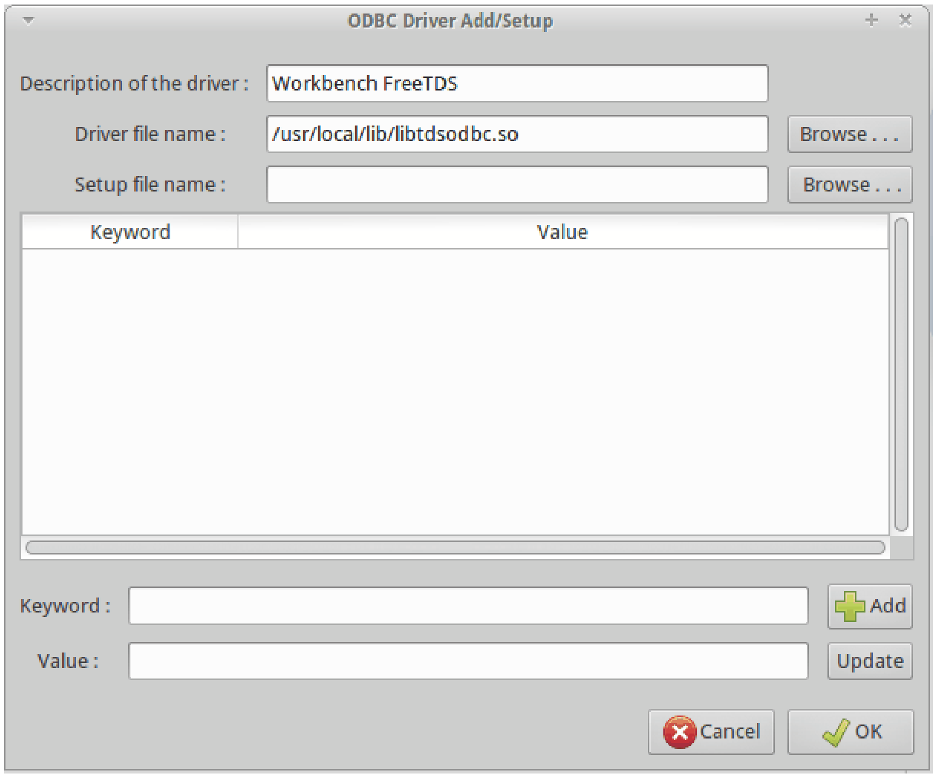
- The max_allowed_packet option in the target MySQL server must be large enough to fit the largest field value to be copied from source (especially BLOBs and large TEXT fields).

The wizard supports migrating from specific database systems, but a "generic" RDBMS support is also provided. The generic support is capable of migrating tables from many RDBMS that can be connected to using ODBC, although certain type mappings may not be performed correctly. A manual mapping step is provided for reviewing and fixing any migration problems that could occur.

Migration Wizard was started

The name of the driver file is `libtdsodbc.so` and it is located in `/usr/lib` or `/usr/local/lib`. For example, under the **ODBC Drivers** tab click **Add Driver** and fill out the description (name) and path to the driver file. Remember the name you define here as it will be needed later on. Save the driver.

Figure 10.43 ODBC Driver Add/Setup



Description of the driver :	Workbench FreeTDS
Driver file name :	/usr/local/lib/libtdsodbc.so
Setup file name :	

Note

Only the driver file name is required, while the setup file name can remain undefined.

9. Close the ODBC Administrator and click **Start Migration**. For information about making a Microsoft SQL Server connection using the MySQL Workbench migration wizard, see Section 10.5.3.2, “Linux”.

10.5.2.3 OS X

See the FreeTDS setup notes for Linux, Section 10.5.2.2, “Linux”.

10.5.3 Connection Setup

This section focuses on creating a connection to the source Microsoft SQL Server, because creating a MySQL connection is a standard operation.

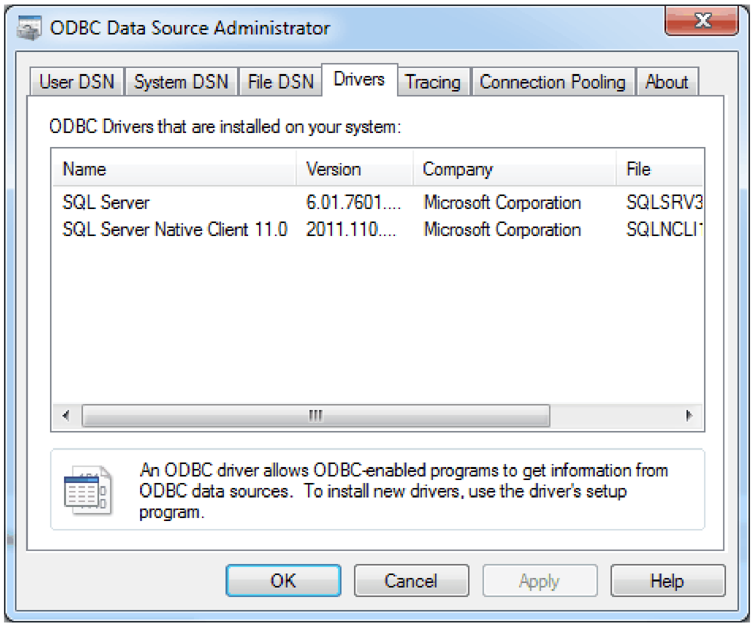
Note

Prerequisite: that you already installed and configured the required Microsoft SQL Server driver on the system running MySQL Workbench.

10.5.3.1 Windows

Select **Microsoft SQL Server** as the database system and fill out the remaining options as described below.

Figure 10.44 SQL Server Connection Parameters Example on Windows



ODBC Drivers that are installed on your system:

Name	Version	Company	File
SQL Server	6.01.7601....	Microsoft Corporation	SQLSRV3
SQL Server Native Client 11.0	2011.110....	Microsoft Corporation	SQLNCLI

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

- **Database System**: Microsoft SQL Server
- **Connection Method**: choose **ODBC (native)** to use the native ODBC driver that is provided by Microsoft. Alternatives include "ODBC data sources" and "ODBC FreeTDS". FreeTDS is a popular open source driver for SQL Server and Sybase.

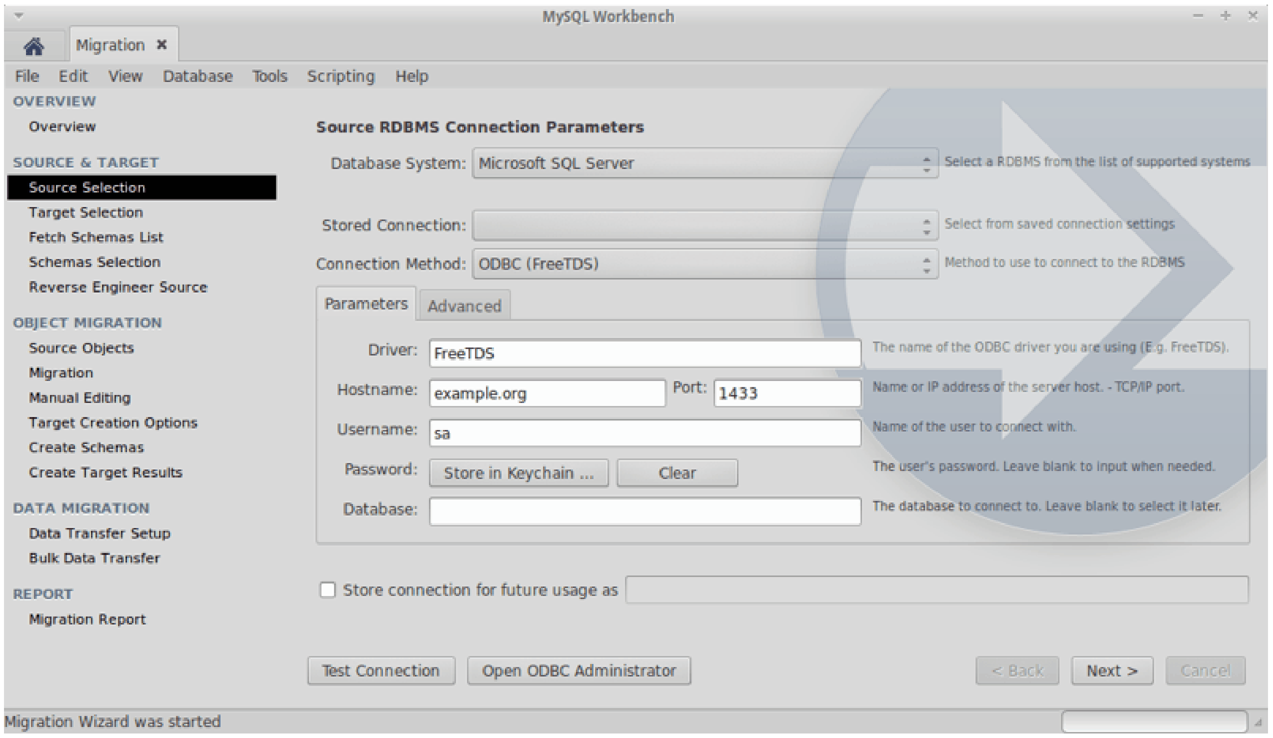
- **Driver**: use the SQL Server driver name, as described in the documentation titled Section 10.5.2.1, “Windows”. Typically this will be "SQL Server" or a versioned client, such as "SQL Server Native Client 11.0".
- **Server**: the address and optionally instance name of the SQL server, such as "example.com" or "example.org\SQLEXPRESS".
- **Username**: the user name on the SQL Server, with "sa" being a commonly used name.
- **Password**: optionally enter a password to save locally, or leave it blank to enter the password when the SQL Server connection is made later on in the process.
- **Database**: optionally enter a database name. Leave it blank to select a database name after the MySQL Workbench wizard fetches the available databases.
- **Store connection for future**: optionally store the connection details locally for future use by checking this box and entering a name for the connection.
- **Advanced**: optionally enter additional options.

Click **Test Connection** to confirm that the parameters are correct before moving on.

10.5.3.2 Linux

Select **Microsoft SQL Server** as the source database system and fill out the remaining options as described below.

Figure 10.45 SQL Server Connection Parameters Example on Linux



Source RDBMS Connection Parameters

Database System: Microsoft SQL Server — Select a RDBMS from the list of supported systems

Stored Connection: — Select from saved connection settings

Connection Method: ODBC (FreeTDS) — Method to use to connect to the RDBMS

Driver: FreeTDS — The name of the ODBC driver you are using (E.g. FreeTDS).

Hostname: example.org Port: 1433 — Name or IP address of the server host. - TCP/IP port.

Username: sa — Name of the user to connect with.

Password: — The user's password. Leave blank to input when needed.

Database: — The database to connect to. Leave blank to select it later.

☐ Store connection for future usage as

Migration Wizard was started

- **Database System**: Microsoft SQL Server
- **Connection Method**: choose **ODBC (FreeTDS)** to use the local FreeTDS that was installed in an earlier step. For additional information about how to install a FreeTDS driver on Linux that will work with the MySQL Workbench migration wizard, see Section 10.5.2.2, “Linux”.

Alternatively, choose **ODBC Data Source (FreeTDS)** if you defined a DSN when creating the SQL Server driver. The available pre-configured DSN options will be available to choose from.

- **Driver**: The name of the driver that you created with the ODBC Administrator, as described in the documentation titled Section 10.5.2.2, “Linux”.

  An example name might be "Workbench FreeTDS", or "FreeTDS", but it is the name you defined in an earlier step, so it may or may not be "FreeTDS". Use the ODBC Administrator to find the correct driver name, as otherwise the connection will fail.

- **Hostname**: the address and instance name of the SQL server, such as "example.org".
- **Port**: the port number. Port number 1433 is commonly used for MySQL server.
- **Username**: the user name on the SQL server, with "sa" being a commonly used name.
- **Password**: optionally enter a password to save locally, or leave it blank to enter the password when the SQL Server connection is made later on in the process.
- **Database**: optionally enter a database name. Leave it blank to select a database name after the MySQL Workbench wizard fetches the available databases.
- **Store connection for future**: optionally store the connection details locally for future use by checking this box and entering a name for the connection.
- **Advanced**: Deselect the **Driver sends Unicode data as UTF-8** option to use UCS-2.

Note

If your MSSQL server connection succeeded but the data import failed, it could be because this setting was enabled.

Click **Test Connection** to confirm that the parameters are correct before moving on.

10.5.3.3 OS X

Connection parameters are similar to Linux, see Section 10.5.3.2, “Linux”.

10.5.4 Microsoft SQL Server Type Mapping

Table 10.2 Type mapping

Source Type	MySQL Type	Comment
INT	INT	
TINYINT	TINYINT	UNSIGNED flag set in MySQL
SMALLINT	SMALLINT	
BIGINT	BIGINT	
BIT	TINYINT(1)	
FLOAT	FLOAT	Precision value is used for storage size in both
REAL	FLOAT	
NUMERIC	DECIMAL	
DECIMAL	DECIMAL	
MONEY	DECIMAL	

Source Type	**MySQL Type**	**Comment**
SMALLMONEY	DECIMAL	
CHAR	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have CHAR columns with a length up to 255 characters. Anything larger is migrated as LONGTEXT
NCHAR	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, character set of strings depend on the column character set instead of the datatype.
VARCHAR	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types.
NVARCHAR	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, character set of strings depend on the column character set instead of the datatype.
DATE	DATE	
DATETIME	DATETIME	
DATETIME2	DATETIME	Date range in MySQL is '1000-01-01 00:00:00.000000' to '9999-12-31 23:59:59.999999'. Note: fractional second values are only stored as of MySQL Server 5.6.4
SMALLDATETIME	DATETIME	
DATETIMEOFFSET	DATETIME	
TIME	TIME	
TIMESTAMP	TIMESTAMP	
ROWVERSION	TIMESTAMP	
BINARY	BINARY/MEDIUMBLOB/ LONGBLOB	Depending on its length
VARBINARY	VARBINARY/ MEDIUMBLOB/ LONGBLOB	Depending on its length
TEXT	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length
NTEXT	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length

Source Type	MySQL Type	Comment
IMAGE	TINYBLOB/ MEDIUMBLOB/ LONGBLOB	Depending on its length
SQL_VARIANT	not migrated	There is not specific support for this datatype.
TABLE	not migrated	There is not specific support for this datatype.
HIERARCHYID	not migrated	There is not specific support for this datatype.
UNIQUEIDENTIFIER	VARCHAR(64)	A unique flag set in MySQL. There is not specific support for inserting unique identifier values.
SYSNAME	VARCHAR(160)	
XML	TEXT	

10.6 PostgreSQL migration

10.6.1 Preparations

Before proceeding, you will need the following:

- Follow the installation guide for installing iODBC on your system. For more information, see Section 10.1, “General installation requirements”.
- Access to a running PostgreSQL instance with privileges to the database you want to migrate, otherwise known as the "source database." The Migration Wizard officially supports PostgreSQL 8.0 and above, although older versions may work.
- Access to a running MySQL Server instance with privileges to the database you want to migrate. The Migration Wizard officially supports MySQL 5.0 and above.

10.6.2 Drivers

10.6.2.1 Microsoft Windows

Download and install the MSI package for psqlODBC. Choose the newest file from http://www.postgresql.org/ftp/odbc/versions/msi/, which will be at the bottom of the downloads page. This will install psqlODBC on your system and allow you to migrate from Postgresql to MySQL using MySQL Workbench.

10.6.2.2 Linux

After installing iODBC, proceed to install the PostgreSQL ODBC drivers.

Download the psqlODBC source tarball file from http://www.postgresql.org/ftp/odbc/versions/src/. Use the latest version available for download, which will be at the bottom of the downloads page. The file will look similar to `psqlodbc-09.03.0400.tar.gz`. Extract this tarball to a temporary location, open a terminal, and cd into that directory. The installation process is:

```
shell> cd the/src/directory
shell> ./configure --with-iodbc --enable-pthreads
shell> make
shell> sudo make install
```

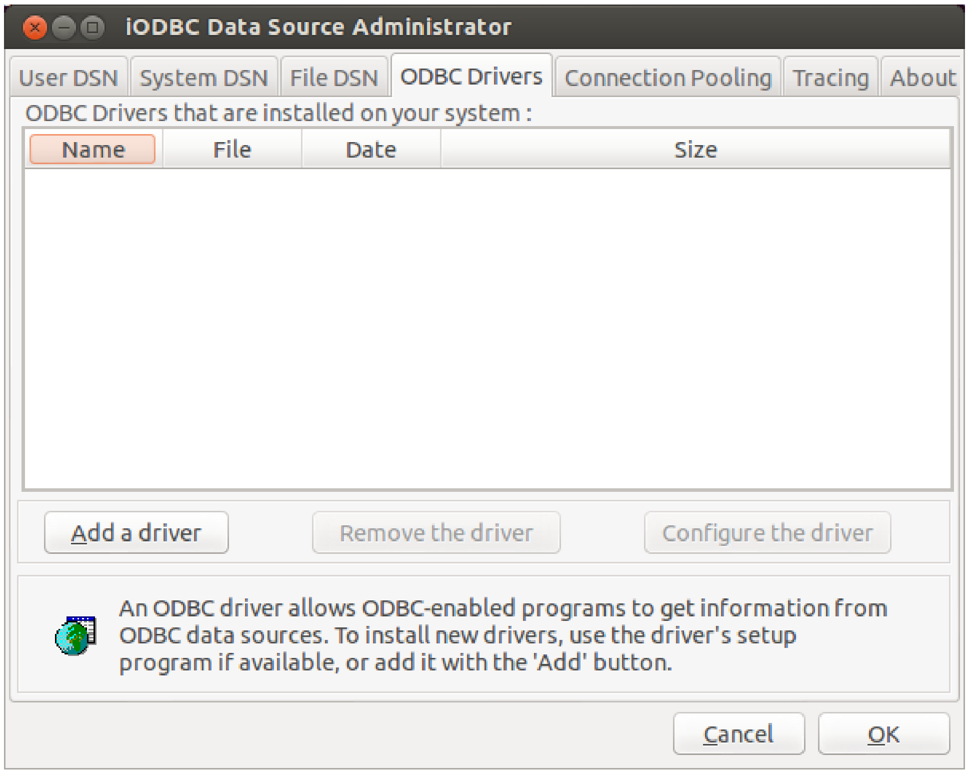
Verify the installation by confirming that the file `psqlodbcw.so` is in the `/usr/local/lib` directory.

Next, you must register your new ODBC Driver.

Open the iODBC Data Source Administrator application by either executing `iodbcadm-gtk` in the command-line, or by launching it from the **Overview** page of the MySQL Workbench Migration Wizard by clicking the **Open ODBC Administrator** button.

Go to the **ODBC Drivers** tab in the iODBC Data Source Administrator. It should look similar to:

Figure 10.46 The iODBC Data Source Administrator



ODBC Drivers that are installed on your system :

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program if available, or add it with the 'Add' button.

Click **Add a driver** then fill out the form with the following values:

- **Description of the driver**: psqlODBC
- **Driver file name**: `/usr/local/lib/psqlodbcw.so`
- **Setup file name**: No value is needed here

And lastly, clicking **OK** will complete the `psqlODBC` driver registration.

10.6.2.3 OS X

To compile `psqlODBC` on OS X, you will need to have Xcode and its "Command Line Tools" component installed on your system, as this includes the required `gcc` compiler. Xcode is free, and available from the AppStore. And after installing Xcode, open it and go to **Preferences**, **Downloads**, **Components**, and then install the "Command Line Tools" component.

Download the psqlODBC source tarball file from http://www.postgresql.org/ftp/odbc/versions/src/. Use the latest version available for download, which will be at the bottom of the downloads page. The file will look similar to `psqlodbc-09.03.0400.tar.gz`. Extract this tarball to a temporary location, open a terminal, and cd into that directory. The installation process is:

```
shell> cd the/src/directory
shell> ./configure --with-iodbc --enable-pthreads
shell> CFLAGS="-arch i386 -arch x86_64" make
shell> sudo make install
```

10.6.3 Connection Setup

After loading the Migration Wizard, click on the **Start Migration** button in the **Overview** page to begin the migration process. You will first connect to the source PostgreSQL database. Here you will provide the information about the PostgreSQL RDBMS that you are migrating from, the ODBC driver that will be used for the migration, and all of the parameters required for the connection. The name of the ODBC driver is the one you set up when you registered your psqlODBC driver with the driver manager.

Opening the **Database System** dropdown will reveal each RDBMS that is supported on your system. Select PostgreSQL from the list. Below that is the **Stored Connection** dropdown, which is optional. Stored connections will be listed here, which are connections saved after defining a connection with the **Store connection for future use as** checkbox enabled.

The three **Connection Method** options are:

- `ODBC (manually entered parameters)`: Each parameter, like a username, is defined separately
- `ODBC Data Source`: For pre-configured data sources (DSN) -- you can optionally create a DSN using the ODBC Administrator
- `ODBC (direct connection string)`: A full ODBC connection string

Note

The psqlODBC driver does not allow a connection without specifying a database name.

The migration process is similar to other databases. See Section 10.6.4, “PostgreSQL Type Mapping” for information on how the migration wizard migrates types from PostgreSQL to MySQL, and Section 10.2.1, “A visual guide to performing a database migration” for a general migration guide.

10.6.4 PostgreSQL Type Mapping

Table 10.3 Type mapping

Source Type	MySQL Type	Comment
INT	INT	
SMALLINT	SMALLINT	
BIGINT	BIGINT	
SERIAL	INT	Sets AUTO_INCREMENT in its table definition.
SMALLSERIAL	SMALLINT	Sets AUTO_INCREMENT in its table definition.
BIGSERIAL	BIGINT	Sets AUTO_INCREMENT in its table definition.
BIT	BIT	
BOOLEAN	TINYINT(1)	
REAL	FLOAT	
DOUBLE PRECISION	DOUBLE	
NUMERIC	DECIMAL	
DECIMAL	DECIMAL	
MONEY	DECIMAL(19,2)	

Source Type	**MySQL Type**	**Comment**
CHAR	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have CHAR columns with a length up to 255 characters. Anything larger is migrated as LONGTEXT
NATIONAL CHARACTER	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, character set of strings depend on the column character set instead of the datatype.
VARCHAR	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types.
NATIONAL CHARACTER VARYING	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length. MySQL Server 5.5 and above can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, character set of strings depend on the column character set instead of the datatype.
DATE	DATE	
TIME	TIME	
TIMESTAMP	DATETIME	
INTERVAL	TIME	
BYTEA	LONGBLOB	
TEXT	LONGTEXT	
CIDR	VARCHAR(43)	
INET	VARCHAR(43)	
MACADDR	VARCHAR(17)	
UUID	VARCHAR(36)	
XML	LONGTEXT	
JSON	LONGTEXT	
TSVECTOR	LONGTEXT	
TSQUERY	LONGTEXT	
ARRAY	LONGTEXT	
POINT	POINT	
LINE	LINESTRING	Although LINE length is infinite, and LINESTRING is finite in MySQL, it is approximated
LSEG	LINESTRING	A LSEG is like a LINESTRING with only two points
BOX	POLYGON	A BOX is a POLYGON with five points and right angles
PATH	LINESTRING	
POLYGON	POLYGON	

Source Type	MySQL Type	Comment
CIRCLE	POLYGON	A POLYGON is used to approximate a CIRCLE
TXID_SNAPSHOT	VARCHAR	

10.7 MySQL migration

Perform MySQL server version upgrades to move off older MySQL versions to the latest version. The standard migration wizard can migrate MySQL to MySQL, and a simpler **Schema Transfer Wizard** can also be used.

MySQL Schema Transfer Wizard

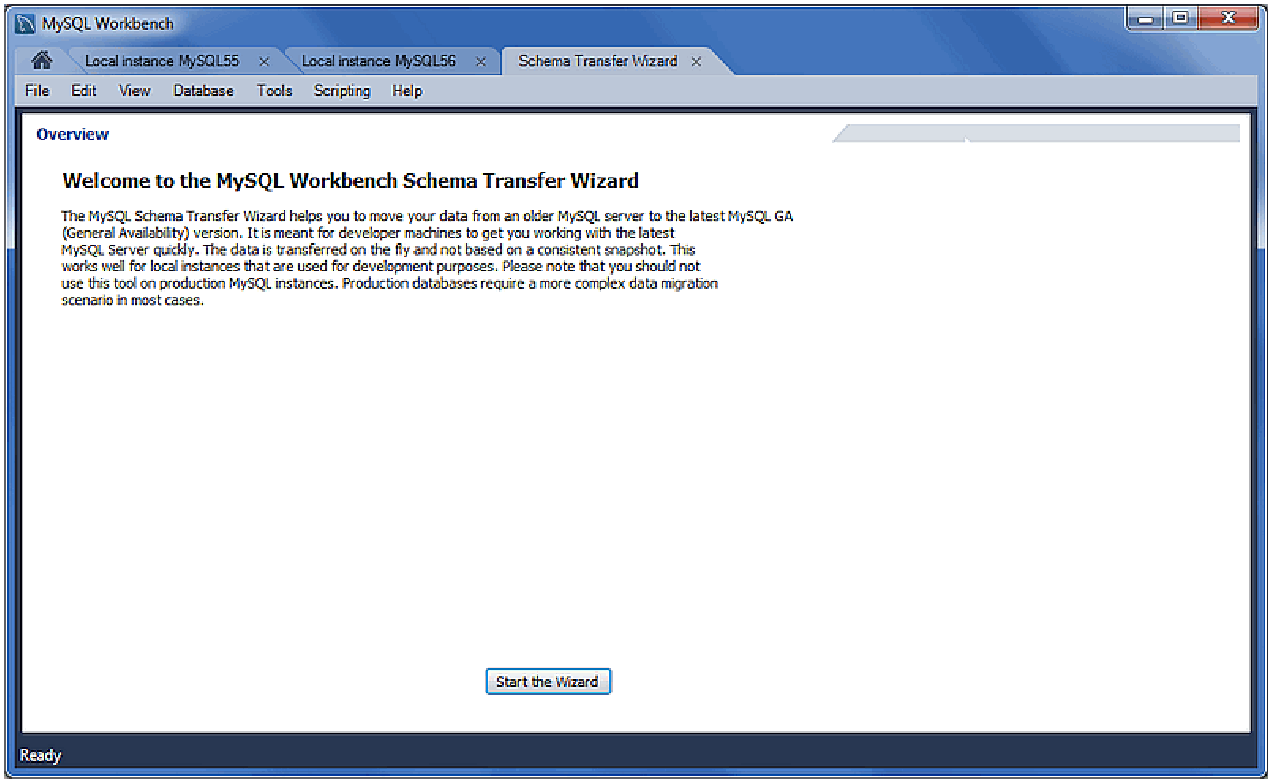
The MySQL Schema Transfer wizard helps you move your data from an older MySQL server version to a different (typically later) MySQL version. This migration tool is meant for developer hosts as it is simpler than the standard migration wizard, because it only migrates MySQL to MySQL. The data is transferred and not based on a consistent snapshot, so it works best on local MySQL instances.

Note

You should not use this wizard on production MySQL instances.

To open the wizard, select **Database**, **Schema Transfer Wizard** from the main menu.

Figure 10.47 MySQL Schema Transfer Wizard: Overview

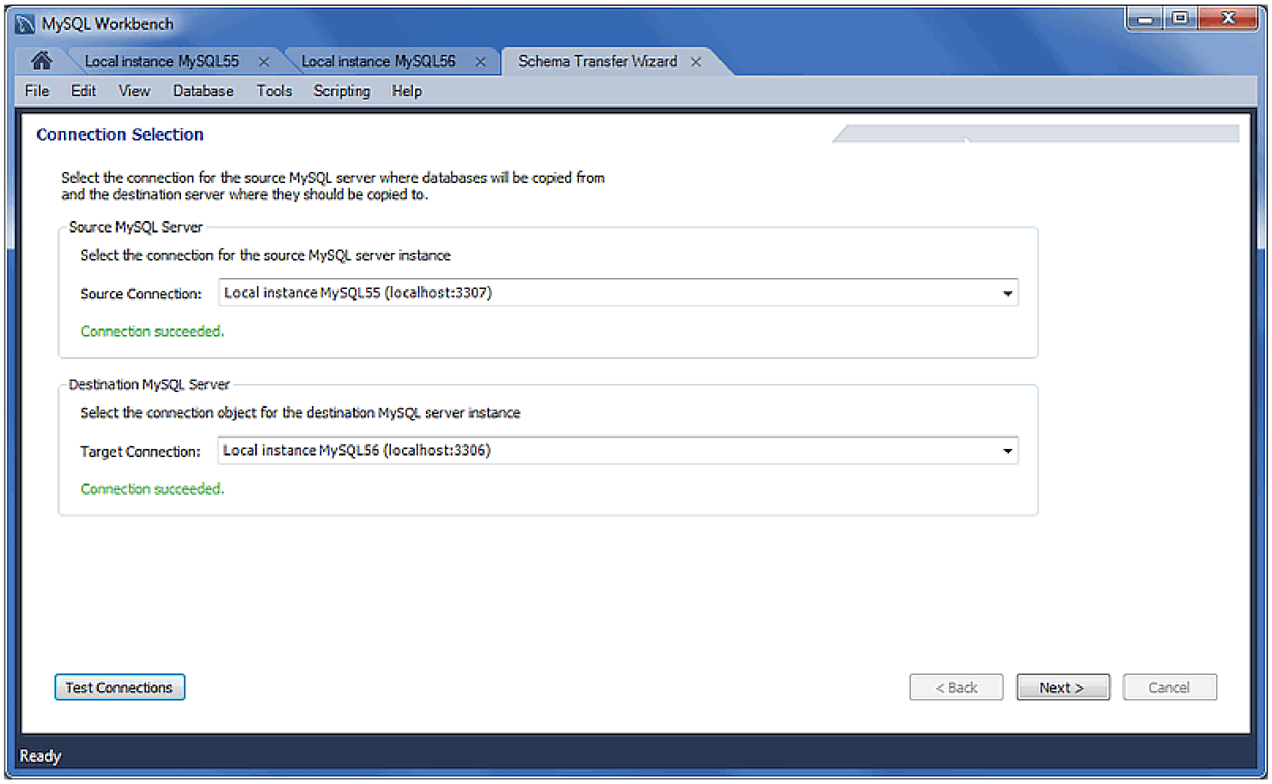


Welcome to the MySQL Workbench Schema Transfer Wizard

The MySQL Schema Transfer Wizard helps you to move your data from an older MySQL server to the latest MySQL GA (General Availability) version. It is meant for developer machines to get you working with the latest MySQL Server quickly. The data is transferred on the fly and not based on a consistent snapshot. This works well for local instances that are used for development purposes. Please note that you should not use this tool on production MySQL instances. Production databases require a more complex data migration scenario in most cases.

Read the overview text and click **Start the Wizard** to begin.

Figure 10.48 MySQL Schema Transfer Wizard: Connection Selection



Connection Selection

Select the connection for the source MySQL server where databases will be copied from and the destination server where they should be copied to.

Source MySQL Server

Select the connection for the source MySQL server instance

Source Connection: Local instance MySQL55 (localhost:3307)

Connection succeeded.

Destination MySQL Server

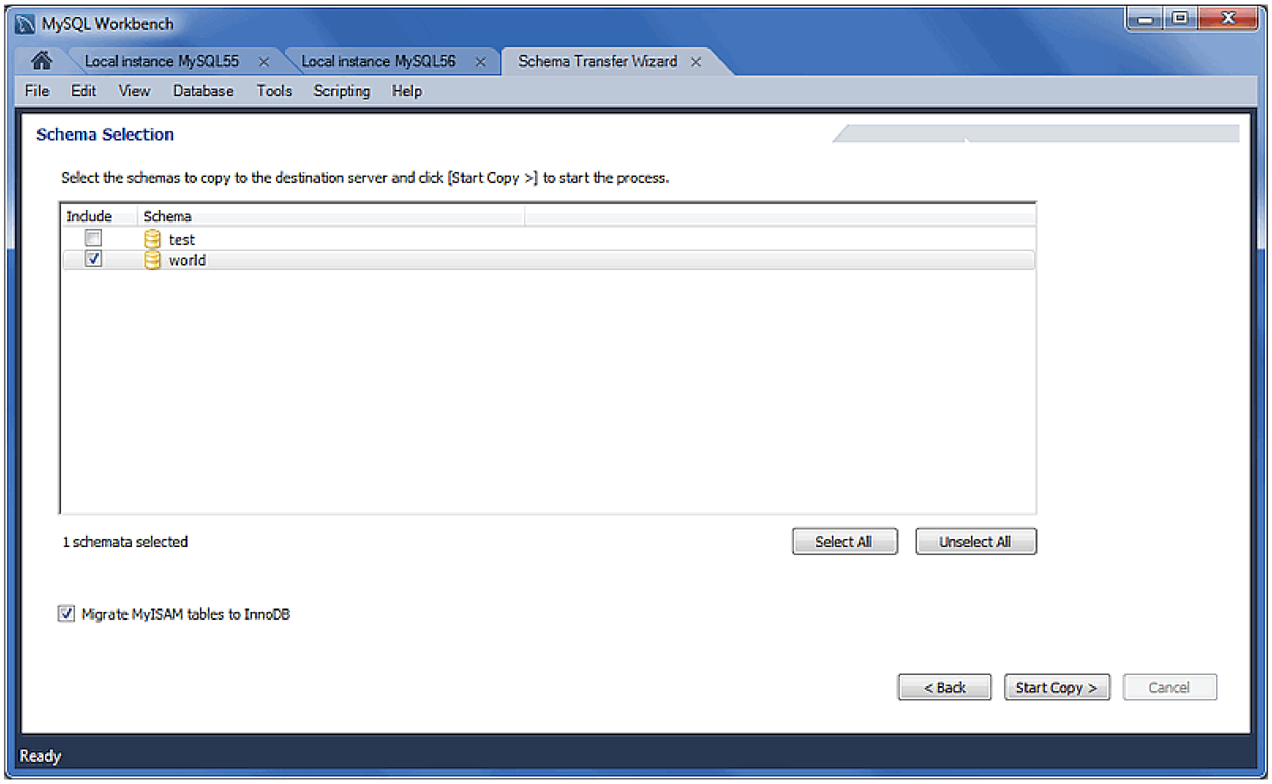
Select the connection object for the destination MySQL server instance

Target Connection: Local instance MySQL56 (localhost:3306)

Connection succeeded.

Choose your target and source MySQL connections. After choosing and testing your MySQL connections, click **Next** to continue.

Figure 10.49 MySQL Schema Transfer Wizard: Schema Selection



Schema Selection

Select the schemas to copy to the destination server and click [Start Copy >] to start the process.

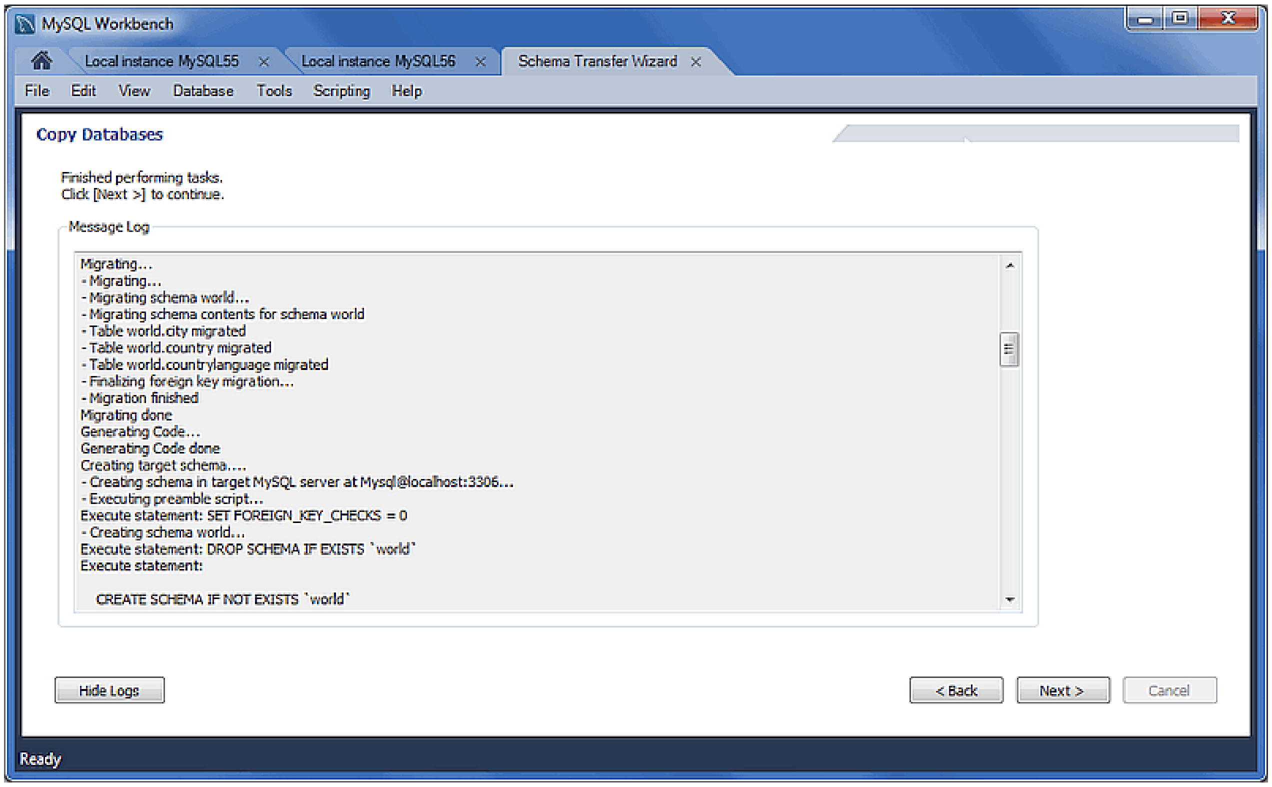
Include	Schema
☐	test
☑	world

1 schemata selected

☑ Migrate MyISAM tables to InnoDB

Choose the schemas to migrate, and click **Start Copy** to begin copying the selected schemas from the source to target MySQL server.

Figure 10.50 MySQL Schema Transfer Wizard: Copy Databases

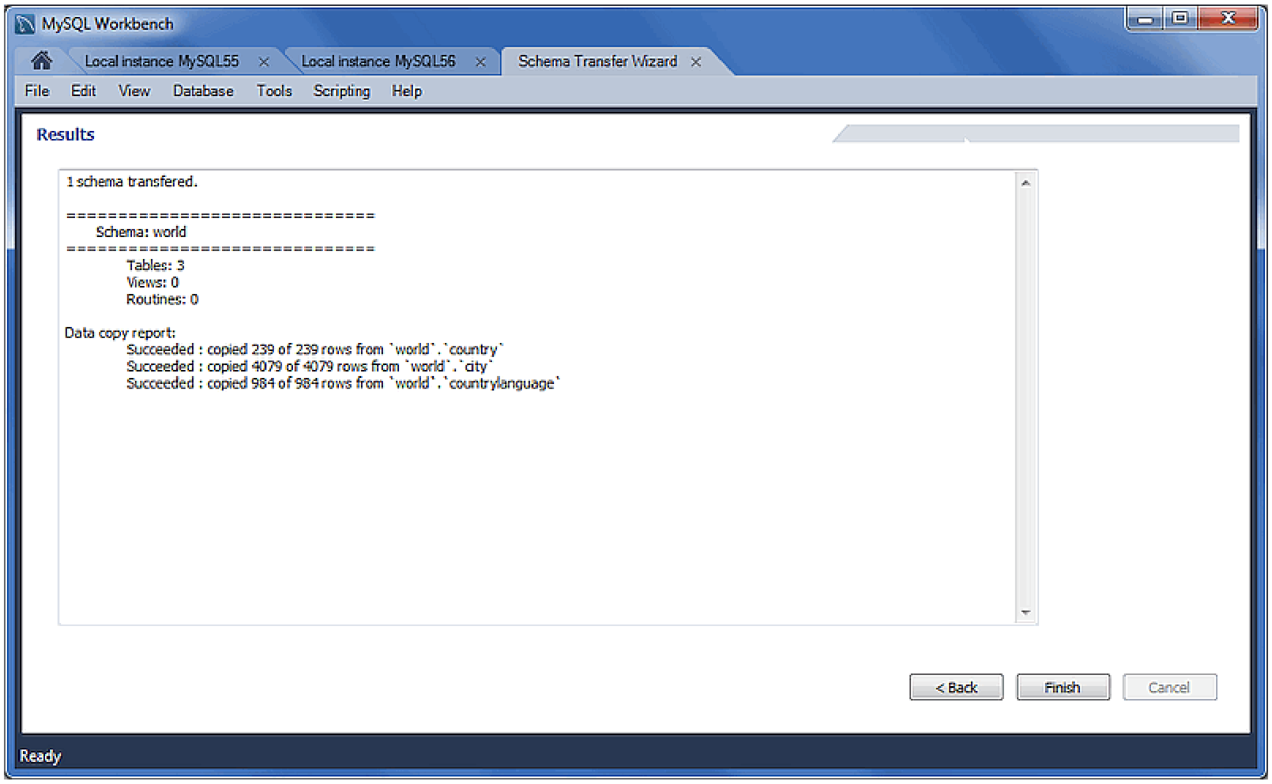


Finished performing tasks.
Click [Next >] to continue.

```
Migrating...
- Migrating...
- Migrating schema world...
- Migrating schema contents for schema world
- Table world.city migrated
- Table world.country migrated
- Table world.countrylanguage migrated
- Finalizing foreign key migration...
- Migration finished
Migrating done
Generating Code...
Generating Code done
Creating target schema....
- Creating schema in target MySQL server at Mysql@localhost:3306...
- Executing preamble script...
Execute statement: SET FOREIGN_KEY_CHECKS = 0
- Creating schema world...
Execute statement: DROP SCHEMA IF EXISTS `world`
Execute statement:

  CREATE SCHEMA IF NOT EXISTS `world`
```

Review the **Message Log** to confirm that the migration completed with success. Click **Next** to view a summary of the results.

Figure 10.51 MySQL Schema Transfer Wizard: Results



```
1 schema transfered.

==============================
    Schema: world
==============================
        Tables: 3
        Views: 0
        Routines: 0

Data copy report:
        Succeeded : copied 239 of 239 rows from `world`.`country`
        Succeeded : copied 4079 of 4079 rows from `world`.`city`
        Succeeded : copied 984 of 984 rows from `world`.`countrylanguage`
```

Click **Finish** to close the wizard.

10.8 Using the MySQL Workbench Migration Wizard

For a visual walk-through of the migration wizard, see Section 10.2.1, “A visual guide to performing a database migration”.

10.8.1 Connecting to the databases

A connection is made to the source and target database servers.

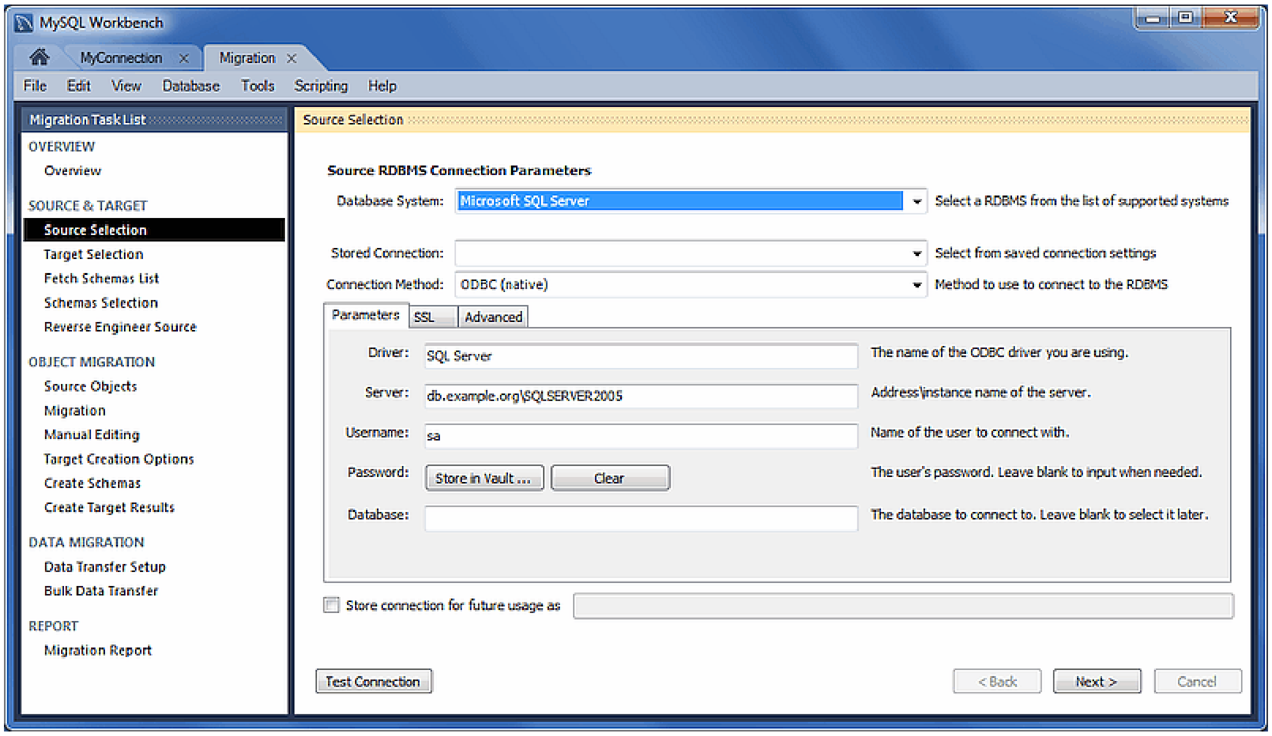
Source Connection

Select the source RDBMS that is migrating to MySQL. Choose the **Database System** that is being migrated, and the other connection parameters will change accordingly.

Note

This connection definition may be saved using the `Store connection for future use as` option, and there is also the **Test Connection** option.

Figure 10.52 MySQL Workbench migration: Source Selection (Parameters)



Source RDBMS Connection Parameters

Database System: Microsoft SQL Server — Select a RDBMS from the list of supported systems

Stored Connection: — Select from saved connection settings

Connection Method: ODBC (native) — Method to use to connect to the RDBMS

Driver: SQL Server — The name of the ODBC driver you are using.

Server: db.example.org\SQLSERVER2005 — Address\instance name of the server.

Username: sa — Name of the user to connect with.

Password: — The user's password. Leave blank to input when needed.

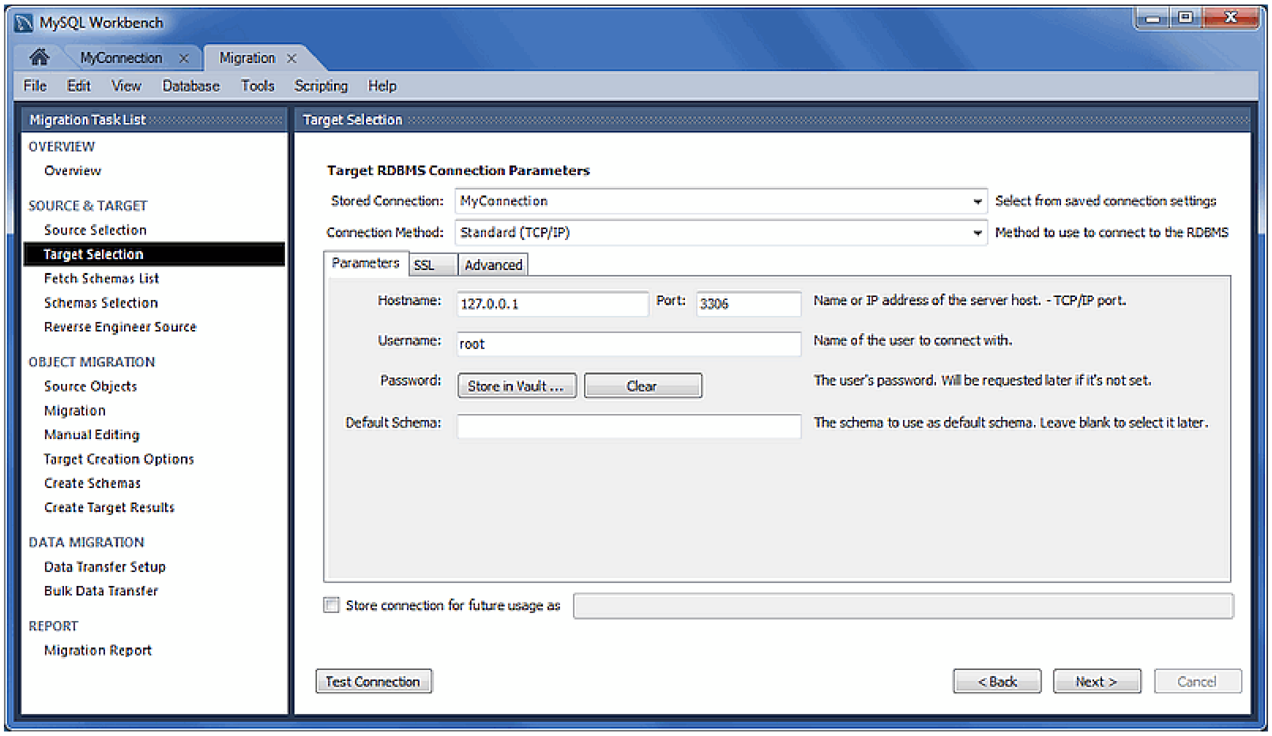
Database: — The database to connect to. Leave blank to select it later.

Store connection for future usage as

Target Selection

The target is the MySQL database that will contain the migrated data. Choose an existing MySQL Workbench connection or select **Manage DB Connections** to create a new MySQL connection.

Figure 10.53 MySQL Workbench migration: Target selection



Target RDBMS Connection Parameters

Stored Connection: MyConnection — Select from saved connection settings

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host. - TCP/IP port.

Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

Default Schema: — The schema to use as default schema. Leave blank to select it later.

Store connection for future usage as

10.8.2 Schemata Retrieval and Selection

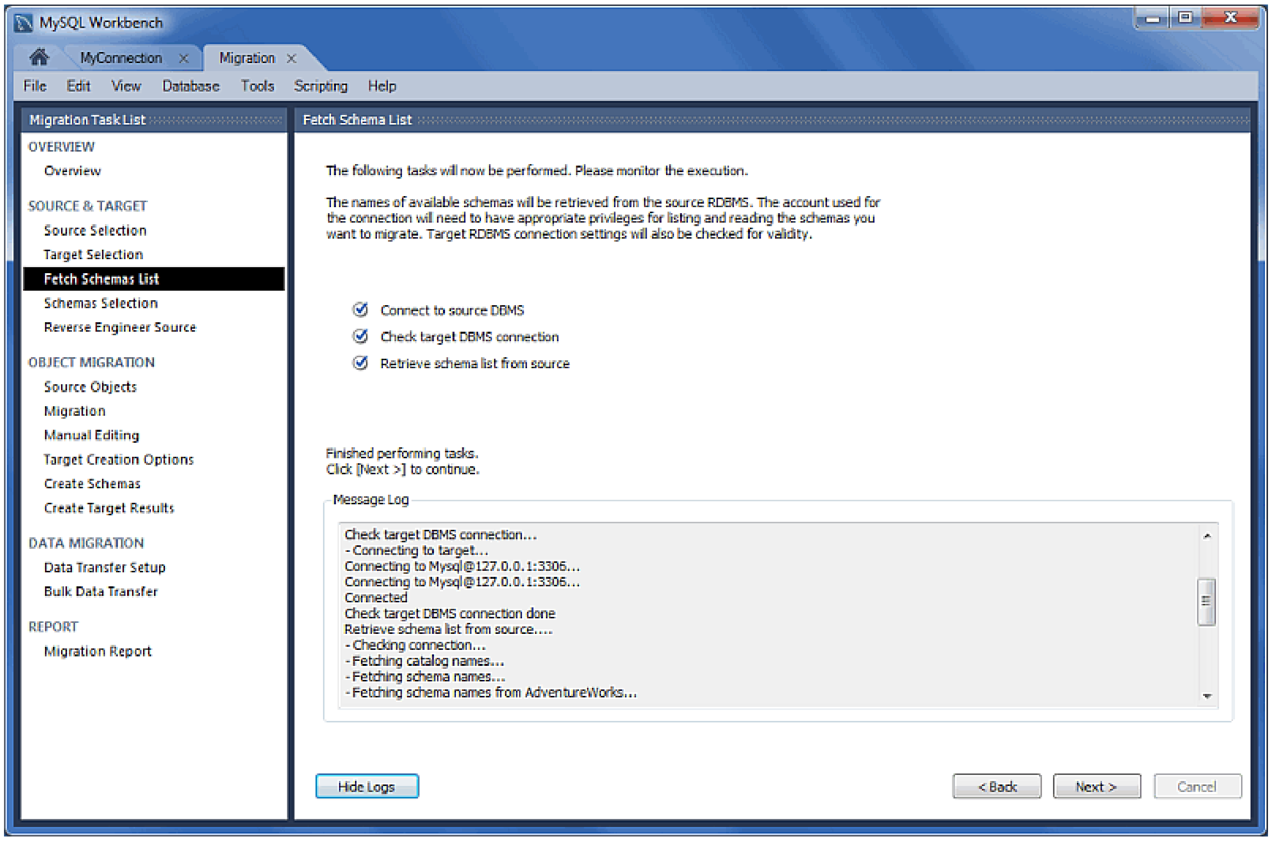
Retrieve a list of available databases and choose the specific databases (and tables) that you want to migrate to MySQL.

Fetch Schemas List

The Schemata list is retrieved from both the source and target RDBMS. The account used for the connection will need to have appropriate privileges for listing and reading the schemas you want to migrate. Target RDBMS connection settings will also be validated. This is an automated and informational step that reports connection related errors and/or general log information.

The steps that are performed include: connects to the source DBMS, checks the connection, and retrieves the schema list from the source.

Figure 10.54 MySQL Workbench migration: Fetch Schemas List



The following tasks will now be performed. Please monitor the execution.

The names of available schemas will be retrieved from the source RDBMS. The account used for the connection will need to have appropriate privileges for listing and reading the schemas you want to migrate. Target RDBMS connection settings will also be checked for validity.

✓ Connect to source DBMS
✓ Check target DBMS connection
✓ Retrieve schema list from source

Finished performing tasks.
Click [Next >] to continue.

```
Check target DBMS connection...
- Connecting to target...
Connecting to Mysql@127.0.0.1:3306...
Connecting to Mysql@127.0.0.1:3306...
Connected
Check target DBMS connection done
Retrieve schema list from source....
- Checking connection...
- Fetching catalog names...
- Fetching schema names...
- Fetching schema names from AdventureWorks...
```

Schemata Selection

Choose the databases you want to migrate over to MySQL.

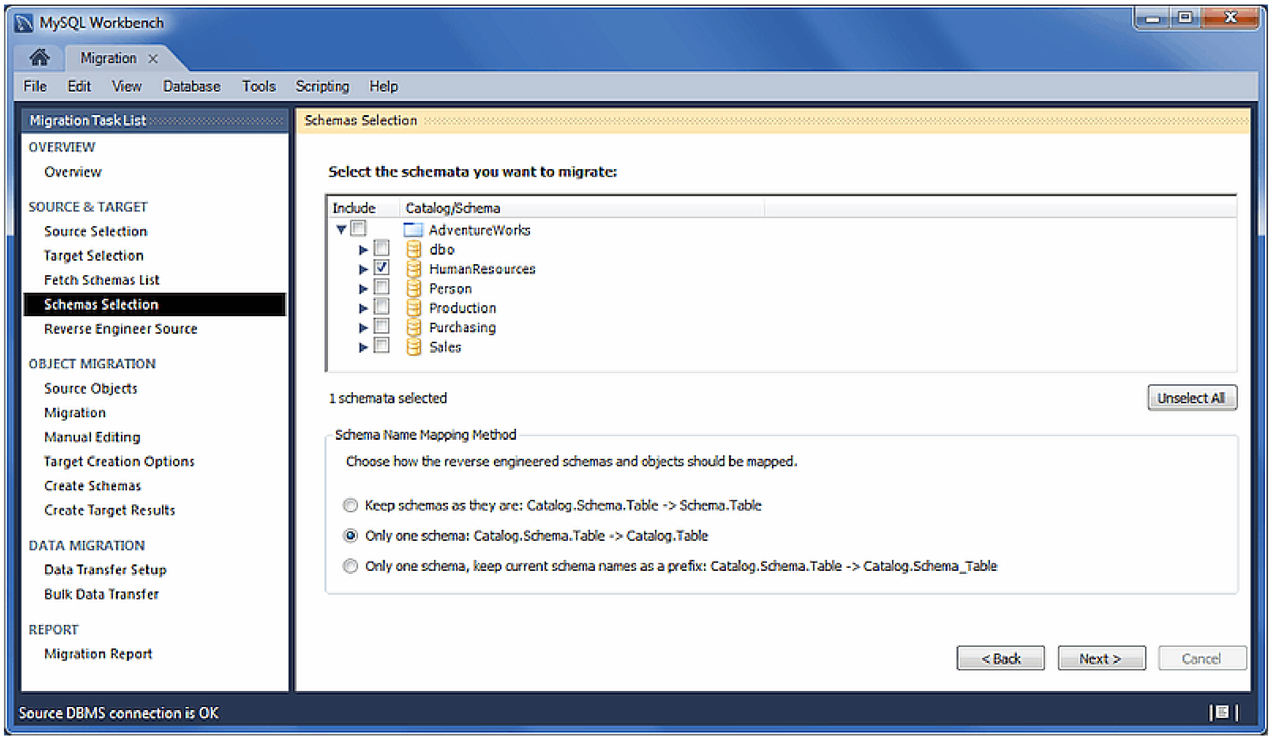
This **Schema Name Mapping Method** options while migrating from Microsoft SQL Server:

Note

This example uses Microsoft SQL Server as the source RDMS. Although the options will be different for other database systems, the concept remains the same.

- Keep schemata as they are: Catalog.Schema.Table -> Schema.Table: This will create multiple databases, one per schema.
- Only one schema: Catalog.Schema.Table -> Catalog.Table: Merges each schema into a single database.
- Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema_table: Preserves the schema name as a prefix.

Figure 10.55 MySQL Workbench migration: Schemata Selection



Select the schemata you want to migrate:

Include	Catalog/Schema
☐	AdventureWorks
☐	dbo
✓	HumanResources
☐	Person
☐	Production
☐	Purchasing
☐	Sales

1 schemata selected

Schema Name Mapping Method

Choose how the reverse engineered schemas and objects should be mapped.

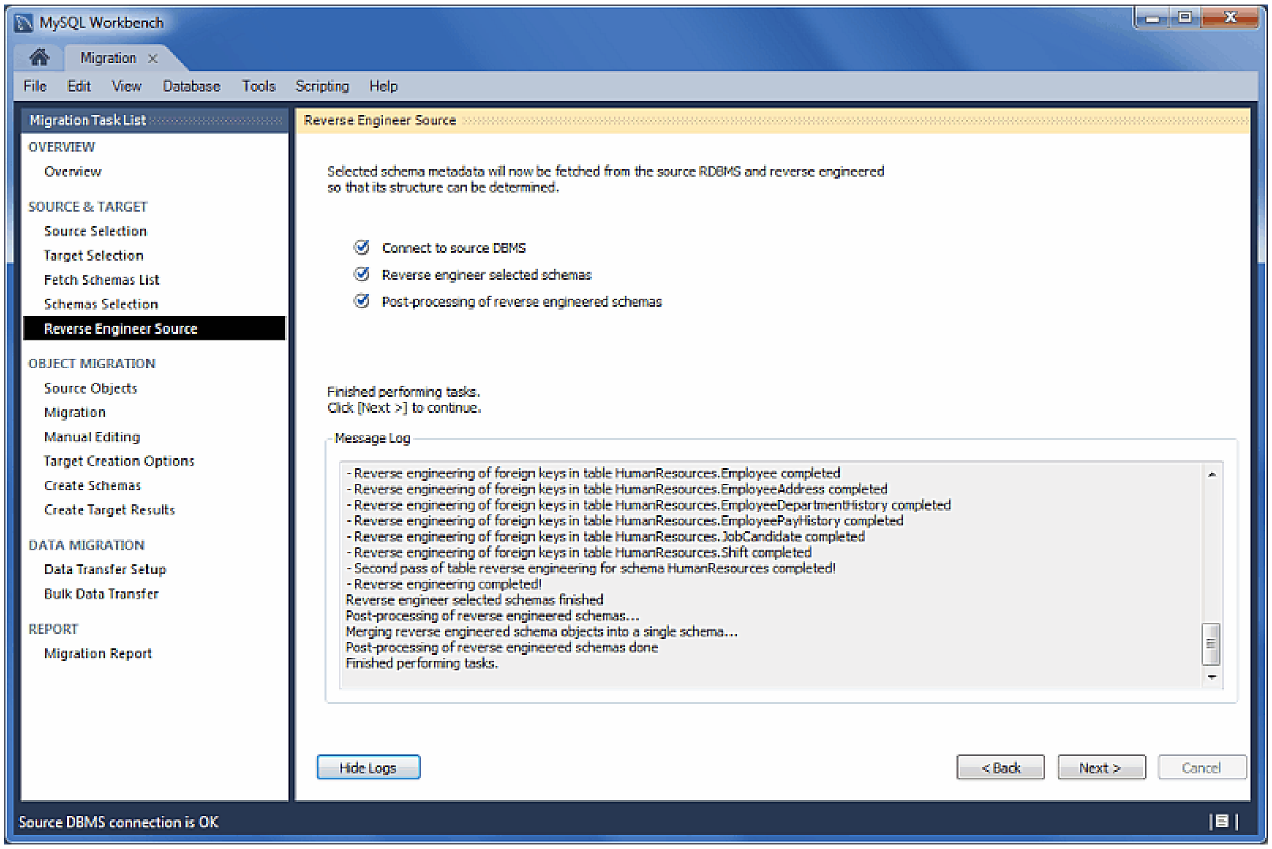
- ○ Keep schemas as they are: Catalog.Schema.Table -> Schema.Table
- ● Only one schema: Catalog.Schema.Table -> Catalog.Table
- ○ Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema_Table

Source DBMS connection is OK

10.8.3 Reverse Engineering

The source metadata is fetched from the source RDBMS, and reverse engineered. This is an automated and informational step that reports related errors and/or general log information. View the logs and then press **Next** to continue.

Figure 10.56 MySQL Workbench migration: Reverse Engineer Source



Selected schema metadata will now be fetched from the source RDBMS and reverse engineered so that its structure can be determined.

- ✓ Connect to source DBMS
- ✓ Reverse engineer selected schemas
- ✓ Post-processing of reverse engineered schemas

Finished performing tasks.
Click [Next >] to continue.

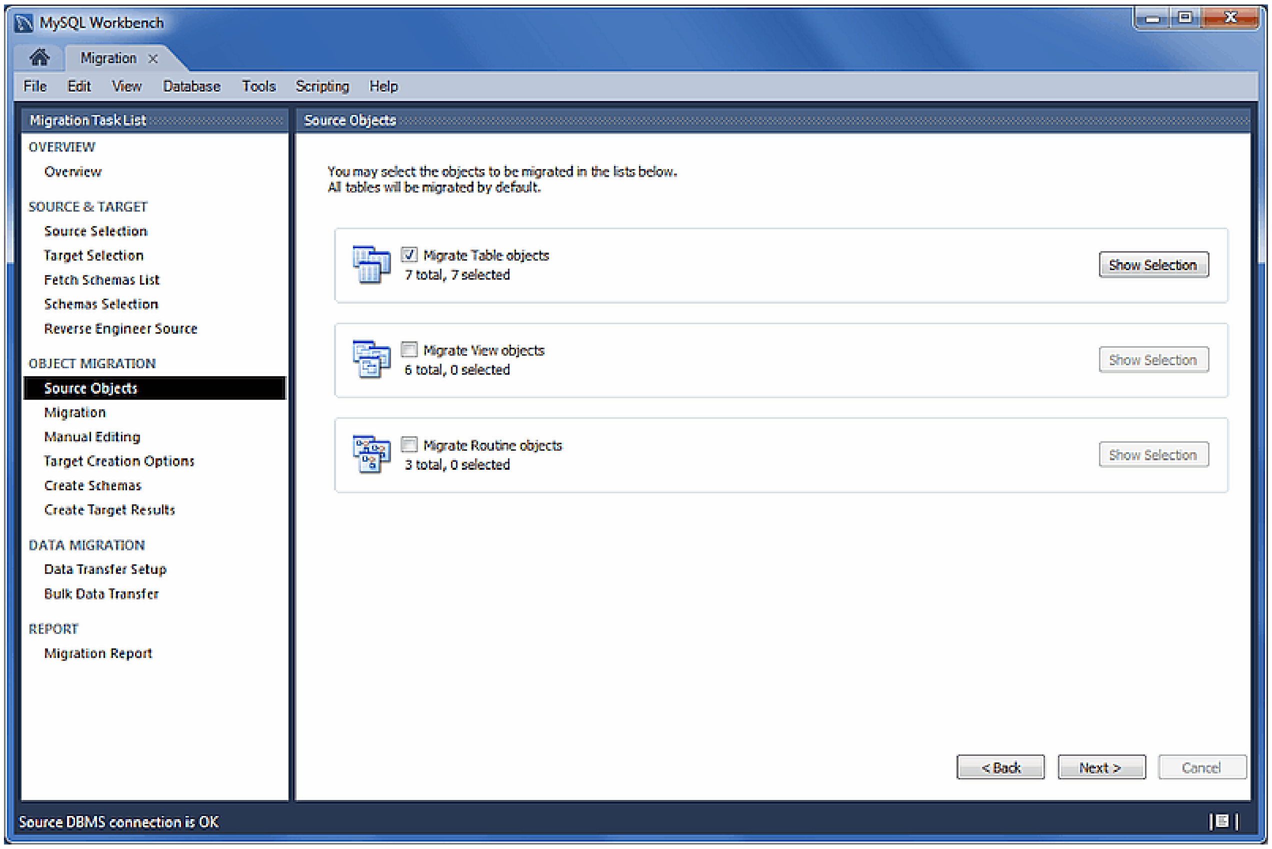
```
- Reverse engineering of foreign keys in table HumanResources.Employee completed
- Reverse engineering of foreign keys in table HumanResources.EmployeeAddress completed
- Reverse engineering of foreign keys in table HumanResources.EmployeeDepartmentHistory completed
- Reverse engineering of foreign keys in table HumanResources.EmployeePayHistory completed
- Reverse engineering of foreign keys in table HumanResources.JobCandidate completed
- Reverse engineering of foreign keys in table HumanResources.Shift completed
- Second pass of table reverse engineering for schema HumanResources completed!
- Reverse engineering completed!
Reverse engineer selected schemas finished
Post-processing of reverse engineered schemas...
Merging reverse engineered schema objects into a single schema...
Post-processing of reverse engineered schemas done
Finished performing tasks.
```

Source DBMS connection is OK

10.8.4 Object Selection

Objects discovered by the **Reverse Engineer Source** stage are made available here. This includes Table, View, and Routine objects, with only the Table objects being selected by default. Use the **Show Selection** button in order to disable individual table objects from being migrated.

Figure 10.57 MySQL Workbench migration: Source Objects



You may select the objects to be migrated in the lists below.
All tables will be migrated by default.

[✓] Migrate Table objects
7 total, 7 selected

[ ] Migrate View objects
6 total, 0 selected

[ ] Migrate Routine objects
3 total, 0 selected

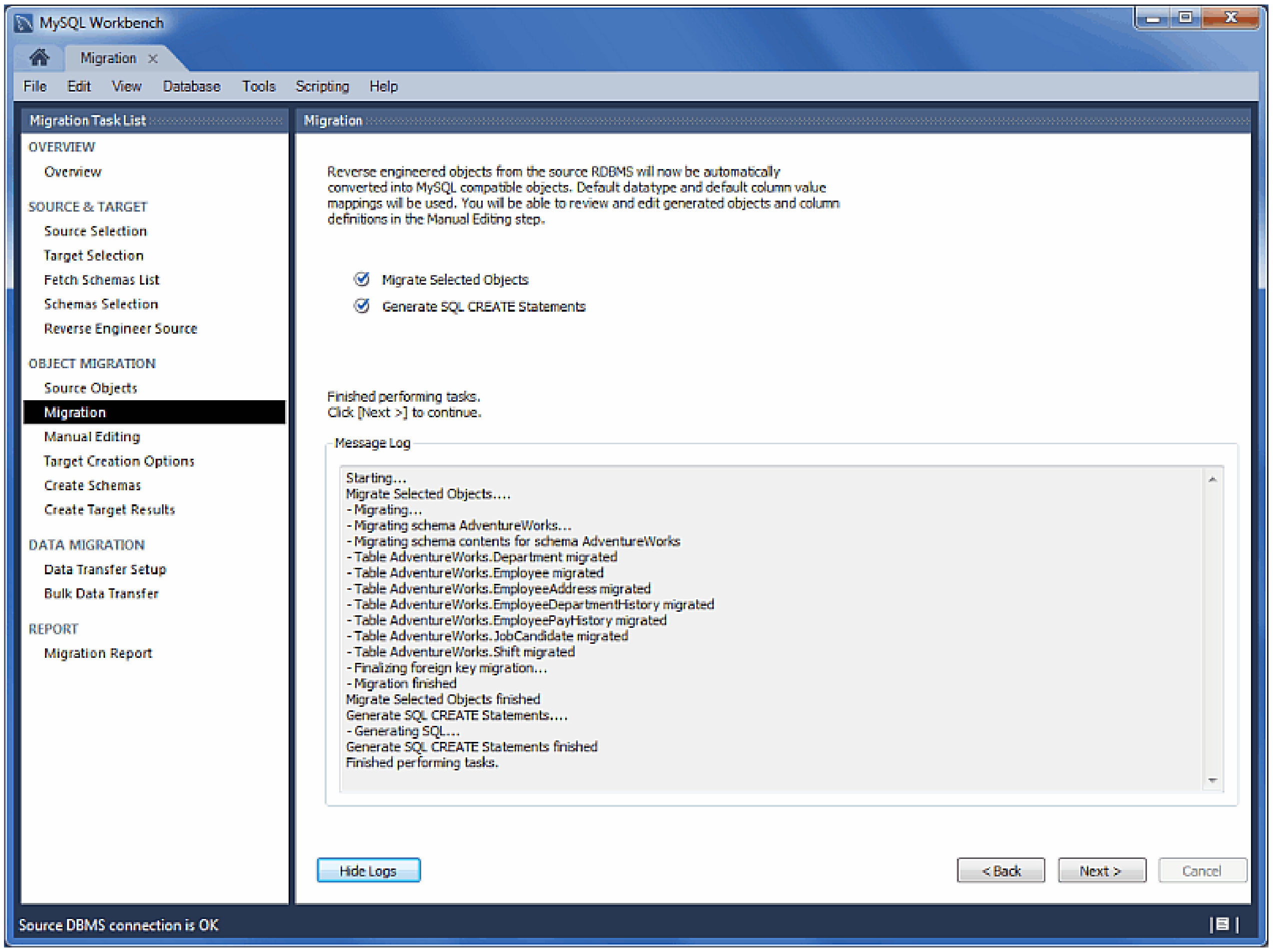
Source DBMS connection is OK

10.8.5 Migration

Reverse engineered objects from the source RDBMS are automatically converted to MySQL compatible objects. Default data type and default column value mappings are used, and the generated objects and column definitions may be reviewed and edited in the next step.

The steps performed include Migrating the selected objects, and generating the SQL CREATE statements.

Figure 10.58 MySQL Workbench migration: Migration



Reverse engineered objects from the source RDBMS will now be automatically converted into MySQL compatible objects. Default datatype and default column value mappings will be used. You will be able to review and edit generated objects and column definitions in the Manual Editing step.

☑ Migrate Selected Objects

☑ Generate SQL CREATE Statements

Finished performing tasks.
Click [Next >] to continue.

Message Log

```
Starting...
Migrate Selected Objects....
- Migrating...
- Migrating schema AdventureWorks...
- Migrating schema contents for schema AdventureWorks
- Table AdventureWorks.Department migrated
- Table AdventureWorks.Employee migrated
- Table AdventureWorks.EmployeeAddress migrated
- Table AdventureWorks.EmployeeDepartmentHistory migrated
- Table AdventureWorks.EmployeePayHistory migrated
- Table AdventureWorks.JobCandidate migrated
- Table AdventureWorks.Shift migrated
- Finalizing foreign key migration...
- Migration finished
Migrate Selected Objects finished
Generate SQL CREATE Statements....
- Generating SQL...
Generate SQL CREATE Statements finished
Finished performing tasks.
```

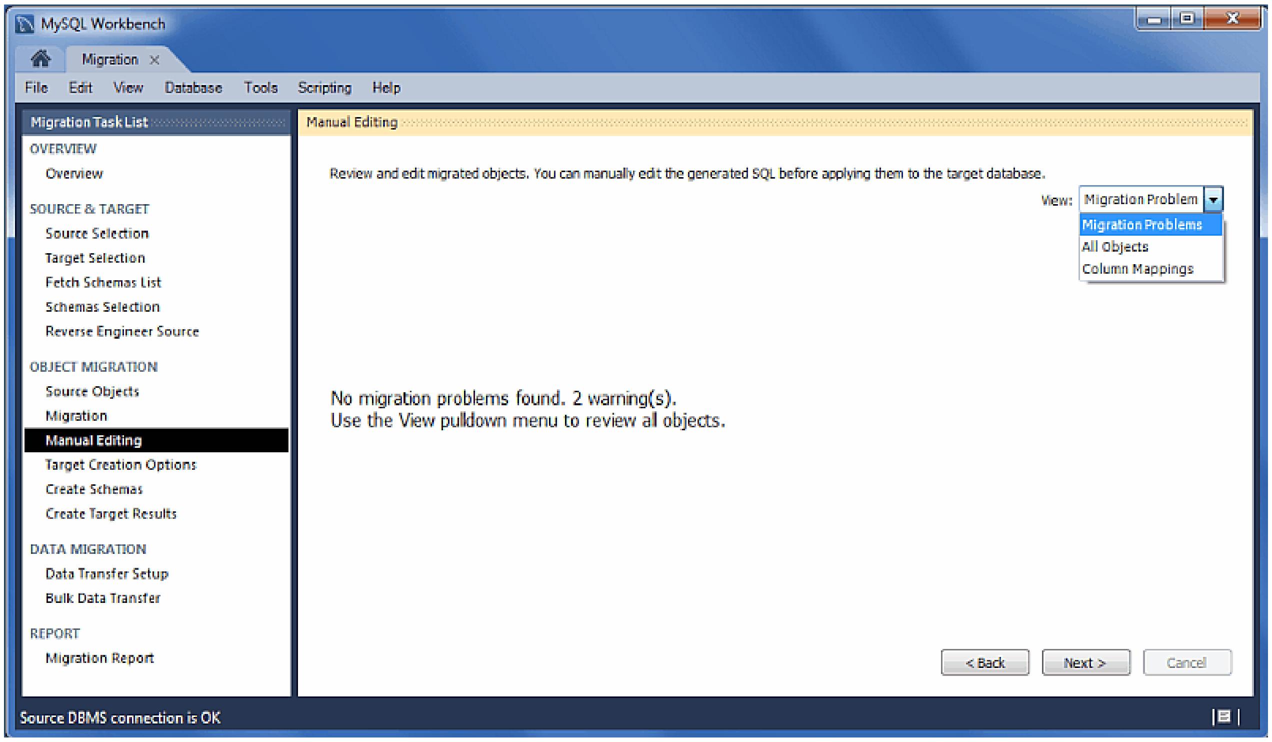
Source DBMS connection is OK

10.8.6 Manual Editing

Use the **View** select box to choose the section to edit. The **Show Code and Messages** button is available with on every page, and it shows the generated MySQL code that corresponds to the selected object.

- **Migration Problems**: This either reports problems or displays "No mapping problems found." It is an informational screen.
- **All Objects**: An object view that allows you to view and edit the object definitions. Double-click on a row to modify a target objects name.
- **Column Mappings**: Shows all of the table column mappings, and allows you to individually review and fix the mapping for all column types, default values, and other attributes.

Figure 10.59 MySQL Workbench migration: Manual Editing (Migration Problems)

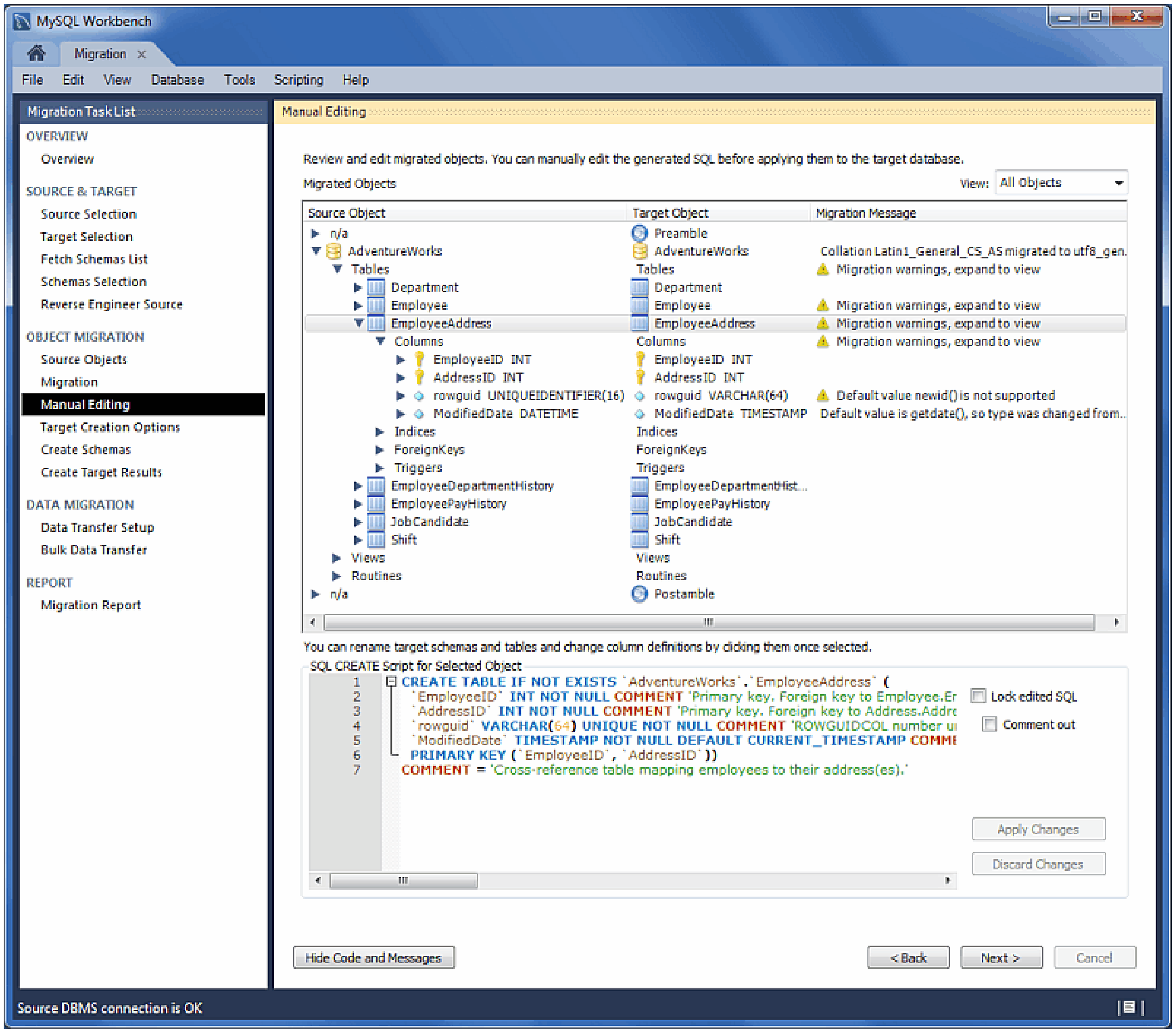


Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

No migration problems found. 2 warning(s).
Use the View pulldown menu to review all objects.

Source DBMS connection is OK

Figure 10.60 MySQL Workbench migration: Manual Editing (All Objects)



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects — View: All Objects

Source Object	Target Object	Migration Message
n/a	Preamble	
AdventureWorks	AdventureWorks	Collation Latin1_General_CS_AS migrated to utf8_gen.
Tables	Tables	Migration warnings, expand to view
Department	Department	
Employee	Employee	Migration warnings, expand to view
EmployeeAddress	EmployeeAddress	Migration warnings, expand to view
Columns	Columns	Migration warnings, expand to view
EmployeeID INT	EmployeeID INT	
AddressID INT	AddressID INT	
rowguid UNIQUEIDENTIFIER(16)	rowguid VARCHAR(64)	Default value newid() is not supported
ModifiedDate DATETIME	ModifiedDate TIMESTAMP	Default value is getdate(), so type was changed from..
Indices	Indices	
ForeignKeys	ForeignKeys	
Triggers	Triggers	
EmployeeDepartmentHistory	EmployeeDepartmentHist...	
EmployeePayHistory	EmployeePayHistory	
JobCandidate	JobCandidate	
Shift	Shift	
Views	Views	
Routines	Routines	
n/a	Postamble	

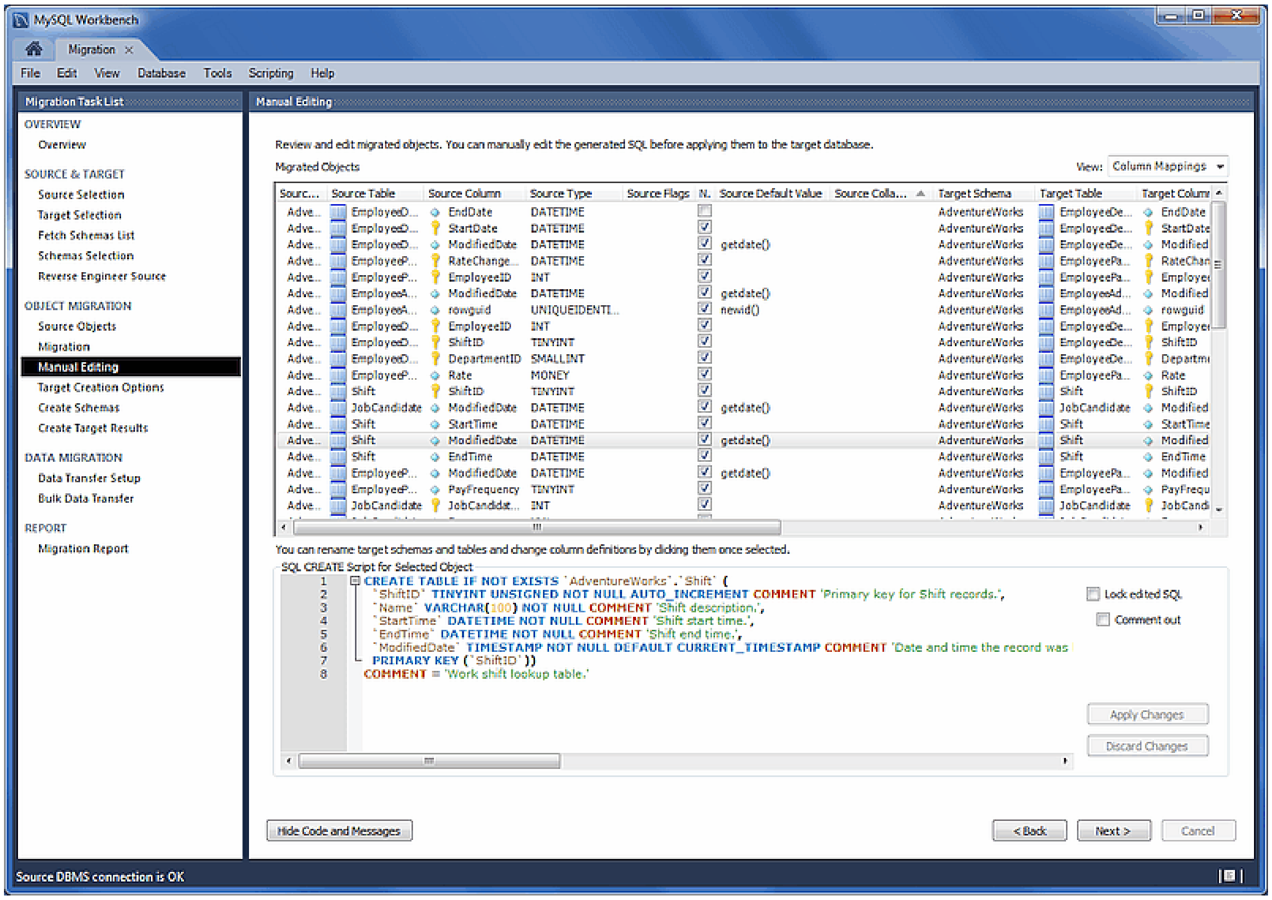
You can rename target schemas and tables and change column definitions by clicking them once selected.

SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `AdventureWorks`.`EmployeeAddress` (
  `EmployeeID` INT NOT NULL COMMENT 'Primary key. Foreign key to Employee.Er
  `AddressID` INT NOT NULL COMMENT 'Primary key. Foreign key to Address.Addre
  `rowguid` VARCHAR(64) UNIQUE NOT NULL COMMENT 'ROWGUIDCOL number ui
  `ModifiedDate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMME
  PRIMARY KEY (`EmployeeID`, `AddressID`))
COMMENT = 'Cross-reference table mapping employees to their address(es).'
```

Source DBMS connection is OK

Figure 10.61 MySQL Workbench migration: Manual Editing (Column Mappings)



Review and edit migrated objects. You can manually edit the generated SQL before applying them to the target database.

Migrated Objects

Sourc...	Source Table	Source Column	Source Type	Source Flags	N.	Source Default Value	Source Colla...	Target Schema	Target Table	Target Column
Adve...	EmployeeD...	EndDate	DATETIME		☐			AdventureWorks	EmployeeDe...	EndDate
Adve...	EmployeeD...	StartDate	DATETIME		✓			AdventureWorks	EmployeeDe...	StartDate
Adve...	EmployeeD...	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	EmployeeDe...	Modified
Adve...	EmployeeP...	RateChange...	DATETIME		✓			AdventureWorks	EmployeePa...	RateChan
Adve...	EmployeeP...	EmployeeID	INT		✓			AdventureWorks	EmployeePa...	Employe
Adve...	EmployeeA...	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	EmployeeAd...	Modified
Adve...	EmployeeA...	rowguid	UNIQUEIDENTI...		✓	newid()		AdventureWorks	EmployeeAd...	rowguid
Adve...	EmployeeD...	EmployeeID	INT		✓			AdventureWorks	EmployeeDe...	Employe
Adve...	EmployeeD...	ShiftID	TINYINT		✓			AdventureWorks	EmployeeDe...	ShiftID
Adve...	EmployeeD...	DepartmentID	SMALLINT		✓			AdventureWorks	EmployeeDe...	Departm
Adve...	EmployeeP...	Rate	MONEY		✓			AdventureWorks	EmployeePa...	Rate
Adve...	Shift	ShiftID	TINYINT		✓			AdventureWorks	Shift	ShiftID
Adve...	JobCandidate	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	JobCandidate	Modified
Adve...	Shift	StartTime	DATETIME		✓			AdventureWorks	Shift	StartTime
Adve...	Shift	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	Shift	Modified
Adve...	Shift	EndTime	DATETIME		✓			AdventureWorks	Shift	EndTime
Adve...	EmployeeP...	ModifiedDate	DATETIME		✓	getdate()		AdventureWorks	EmployeePa...	Modified
Adve...	EmployeeP...	PayFrequency	TINYINT		✓			AdventureWorks	EmployeePa...	PayFrequ
Adve...	JobCandidate	JobCandidat...	INT		✓			AdventureWorks	JobCandidate	JobCandi

You can rename target schemas and tables and change column definitions by clicking them once selected.

SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `AdventureWorks`.`Shift` (
  `ShiftID` TINYINT UNSIGNED NOT NULL AUTO_INCREMENT COMMENT 'Primary key for Shift records.',
  `Name` VARCHAR(100) NOT NULL COMMENT 'Shift description.',
  `StartTime` DATETIME NOT NULL COMMENT 'Shift start time.',
  `EndTime` DATETIME NOT NULL COMMENT 'Shift end time.',
  `ModifiedDate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Date and time the record was
  PRIMARY KEY (`ShiftID`))
COMMENT = 'Work shift lookup table.'
```

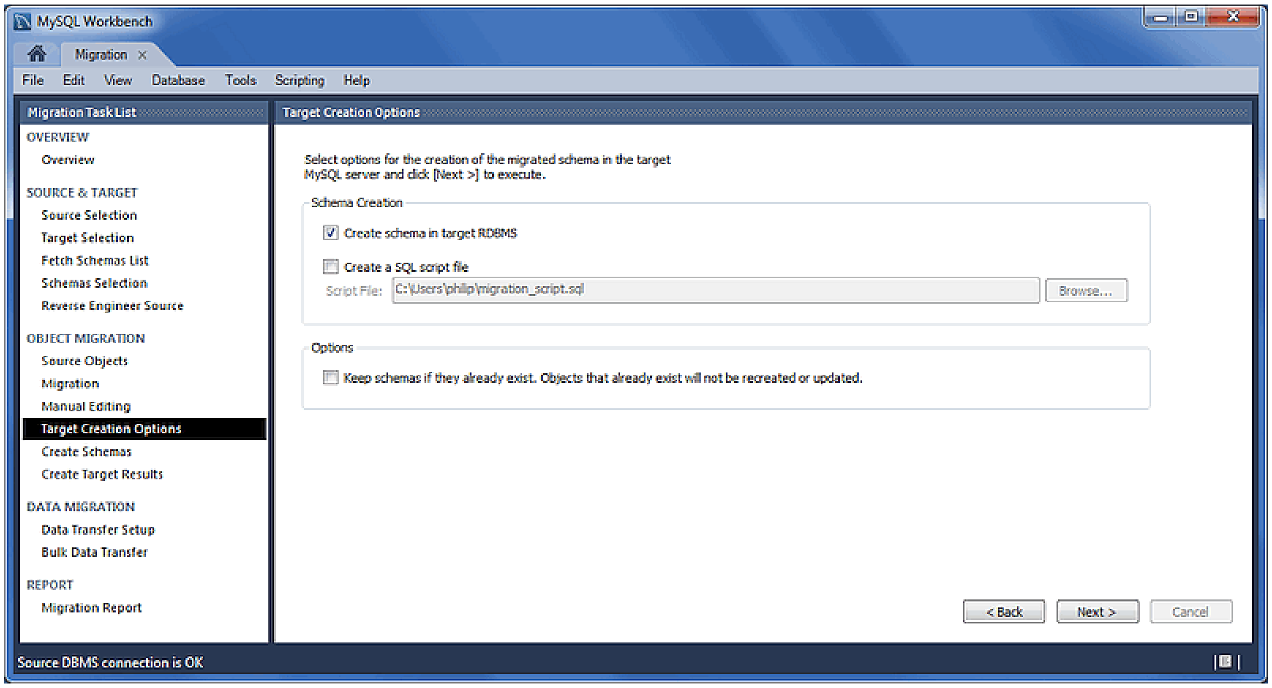
Source DBMS connection is OK

10.8.7 Target Creation Options

Defines addition settings for the target schema. Configuration options include:

- `Create schema in target RDBMS`:
- `Create an SQL script file`:
- An option to keep the schemata if they already exist. Objects that already exist will not be recreated or update.

Figure 10.62 MySQL Workbench migration: Target Creation Options



Select options for the creation of the migrated schema in the target MySQL server and click [Next >] to execute.

Schema Creation

[✓] Create schema in target RDBMS

[ ] Create a SQL script file

Script File: C:\Users\philip\migration_script.sql

Options

[ ] Keep schemas if they already exist. Objects that already exist will not be recreated or updated.

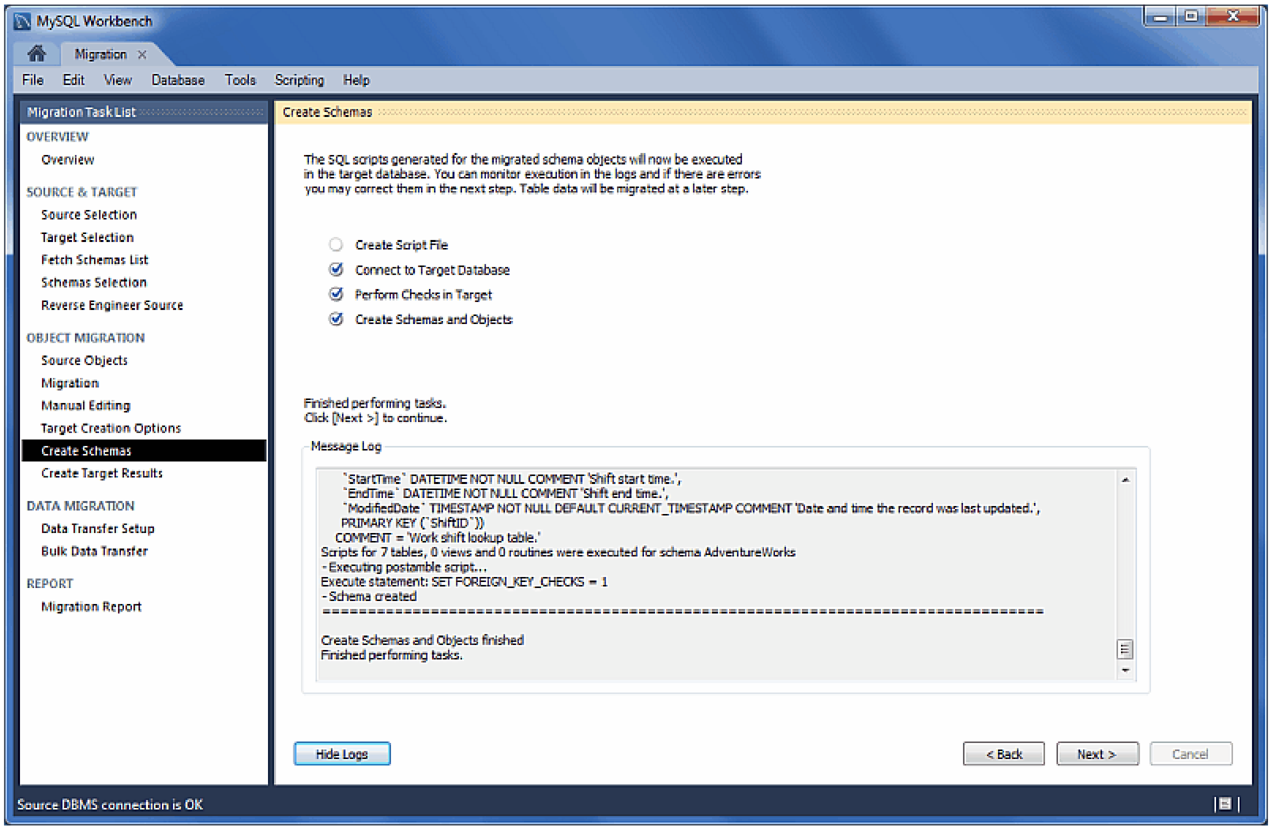
Source DBMS connection is OK

10.8.8 Schema Creation

The SQL scripts generated for the migrated schema objects will now be executed in the target database. You can monitor execution in the logs, if errors exist then they will be fixed in the next step. Table data will be migrated in a later step as well.

This is an automated step, where the actions include: Create Script File, Connect to Target Database, and Create Schemata and Objects.

Figure 10.63 MySQL Workbench migration: Create Schemata



The SQL scripts generated for the migrated schema objects will now be executed in the target database. You can monitor execution in the logs and if there are errors you may correct them in the next step. Table data will be migrated at a later step.

- Create Script File
- ✓ Connect to Target Database
- ✓ Perform Checks in Target
- ✓ Create Schemas and Objects

Finished performing tasks.
Click [Next >] to continue.

Message Log

```
    `StartTime` DATETIME NOT NULL COMMENT 'Shift start time.',
    `EndTime` DATETIME NOT NULL COMMENT 'Shift end time.',
    `ModifiedDate` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Date and time the record was last updated.',
    PRIMARY KEY (`ShiftID`))
  COMMENT = 'Work shift lookup table.'
Scripts for 7 tables, 0 views and 0 routines were executed for schema AdventureWorks
- Executing postamble script...
Execute statement: SET FOREIGN_KEY_CHECKS = 1
- Schema created
================================================================================

Create Schemas and Objects finished
Finished performing tasks.
```

10.8.9 Create Target Results

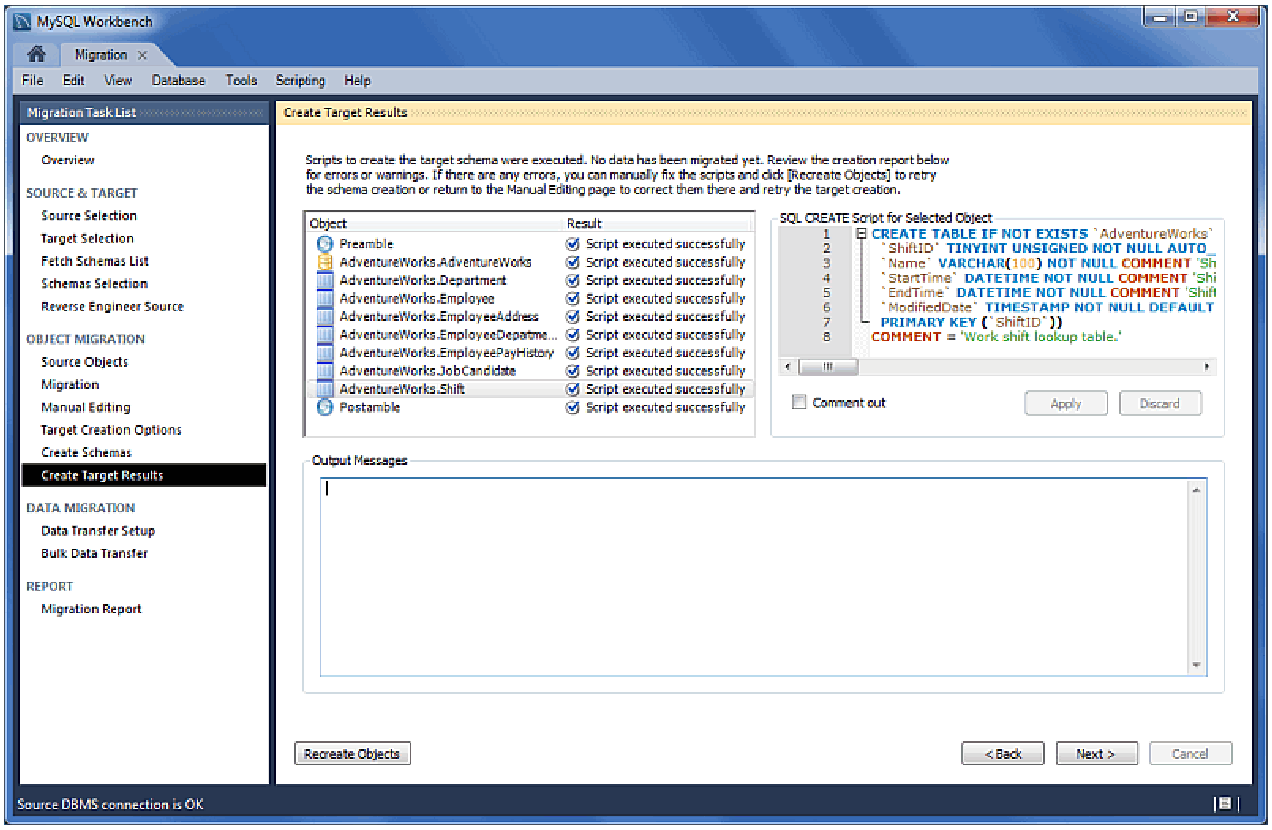
The generated objects are listed here, along with the error messages if any exist.

The migration code may also be viewed and edited here. To make changes, select an object, edit the query code, and press **Apply**. Repeat this process for each object that will be edited. And then, press **Recreate Objects** to save the results.

Note

The **Recreate Objects** operation is required to save any changes here. It will then execute the previous migration step (**Create Schemata**) with the modified code, and then continue the migration process. This also means that the previously saved schema will be dropped.

Figure 10.64 MySQL Workbench migration: Create Target Results



Scripts to create the target schema were executed. No data has been migrated yet. Review the creation report below for errors or warnings. If there are any errors, you can manually fix the scripts and click [Recreate Objects] to retry the schema creation or return to the Manual Editing page to correct them there and retry the target creation.

Object	Result
Preamble	✓ Script executed successfully
AdventureWorks.AdventureWorks	✓ Script executed successfully
AdventureWorks.Department	✓ Script executed successfully
AdventureWorks.Employee	✓ Script executed successfully
AdventureWorks.EmployeeAddress	✓ Script executed successfully
AdventureWorks.EmployeeDepartme...	✓ Script executed successfully
AdventureWorks.EmployeePayHistory	✓ Script executed successfully
AdventureWorks.JobCandidate	✓ Script executed successfully
AdventureWorks.Shift	✓ Script executed successfully
Postamble	✓ Script executed successfully

SQL CREATE Script for Selected Object

```
CREATE TABLE IF NOT EXISTS `AdventureWorks`
  `ShiftID` TINYINT UNSIGNED NOT NULL AUTO_
  `Name` VARCHAR(100) NOT NULL COMMENT 'Sh
  `StartTime` DATETIME NOT NULL COMMENT 'Shi
  `EndTime` DATETIME NOT NULL COMMENT 'Shif
  `ModifiedDate` TIMESTAMP NOT NULL DEFAULT
  PRIMARY KEY (`ShiftID`))
COMMENT = 'Work shift lookup table.'
```

Source DBMS connection is OK

10.8.10 Data Transfer and Migration Setup

Transfers data from the source RDBMS to the target MySQL database. The setup screen includes the following options:

Data Copy:

- *Online copy of table data to target RDBMS*: This (default) will copy the data to the target RDBMS.
- *Create a batch file to copy the data at another time*: The data may also be dumped to a file that can be executed at a later time, or be used as a backup. This script uses a MySQL connection to transfer the data.
- *Create a shell script to use native server dump and load abilities for fast migration*: Unlike the simple batch file that performs a live online copy, this generates a script to be executed on the *source* host to then generate a Zip file containing all of the data and information needed to migrate the data locally on the target host. Copy and extract the generated Zip file on the target host and then execute the import script (on the target host) to import the data into MySQL using a LOAD DATA call.

  This faster method avoids the need to traffic all data through MySQL Workbench, or to have a permanent network connection between the MySQL servers.

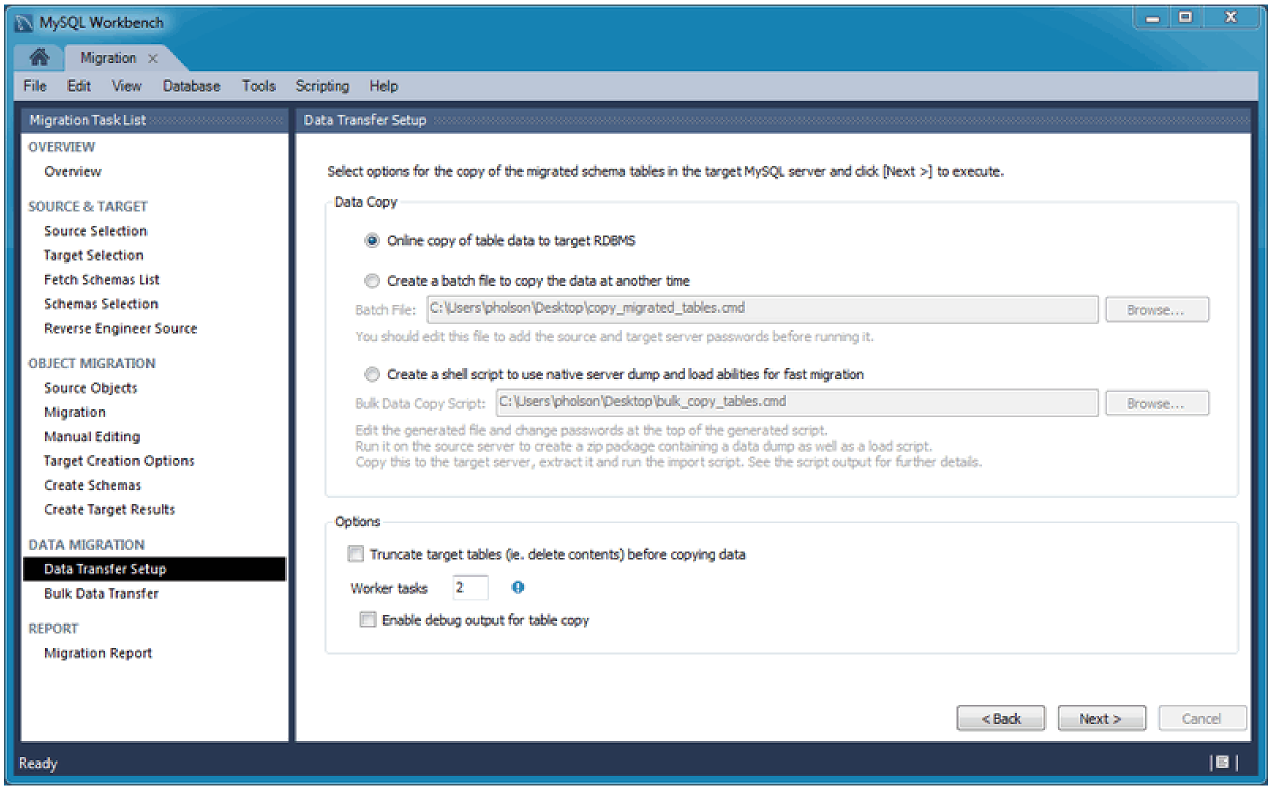
Note

This option was added in MySQL Workbench 6.3.0.

Options:

- Truncate target tables before copying data: In case the target database already exists, this will delete said data.
- Worker tasks: The default value is 2. This is the number of tasks (database connections) used while copying the data.
- Enable debug output for table copy: Shows debugging information.

Figure 10.65 MySQL Workbench migration: Data Transfer Setup



Select options for the copy of the migrated schema tables in the target MySQL server and click [Next >] to execute.

Data Copy

[✓ Online copy of table data to target RDBMS]

Create a batch file to copy the data at another time

Batch File: C:\Users\pholson\Desktop\copy_migrated_tables.cmd

You should edit this file to add the source and target server passwords before running it.

Create a shell script to use native server dump and load abilities for fast migration

Bulk Data Copy Script: C:\Users\pholson\Desktop\bulk_copy_tables.cmd

Edit the generated file and change passwords at the top of the generated script.
Run it on the source server to create a zip package containing a data dump as well as a load script.
Copy this to the target server, extract it and run the import script. See the script output for further details.

Options

Truncate target tables (ie. delete contents) before copying data

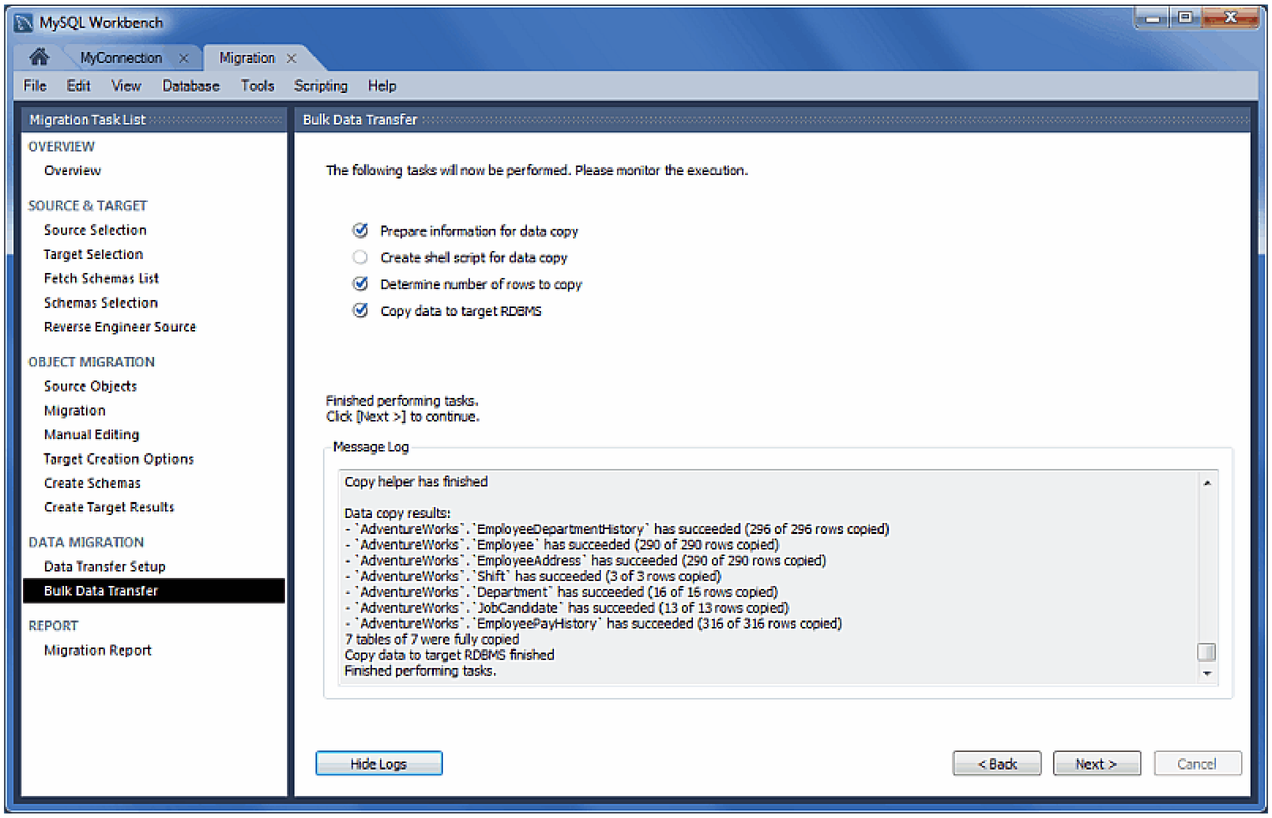
Worker tasks 2

Enable debug output for table copy

10.8.11 Bulk Data Transfer

Depending on the selected option, this will either transfer the data to the target RDMS (default), generate a simple script for the online data transfer, or generate script to execute on the source host that then generates a Zip file containing both the transfer script and data that will be executed on the target host. Optionally, view the logs to confirm.

Figure 10.66 MySQL Workbench migration: Bulk Data Transfer



The following tasks will now be performed. Please monitor the execution.

- ✓ Prepare information for data copy
- Create shell script for data copy
- ✓ Determine number of rows to copy
- ✓ Copy data to target RDBMS

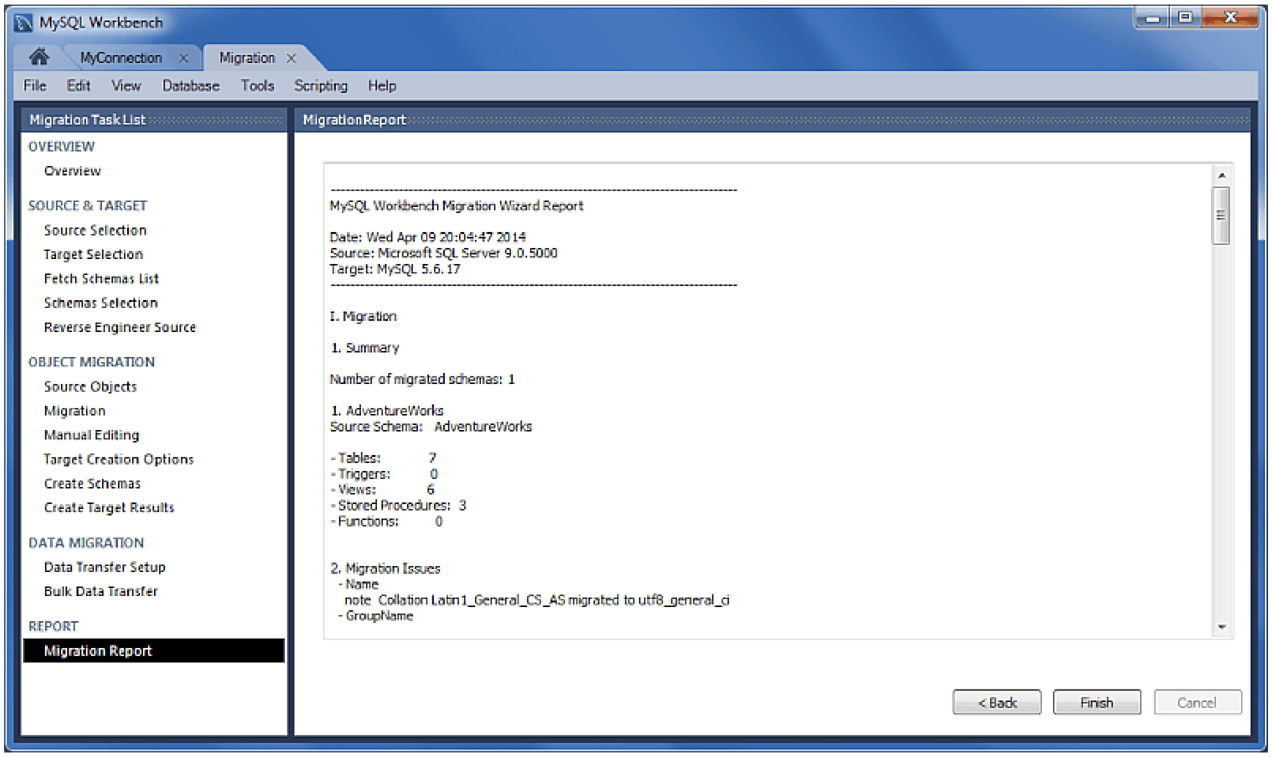
Finished performing tasks.
Click [Next >] to continue.

```
Copy helper has finished

Data copy results:
- `AdventureWorks`.`EmployeeDepartmentHistory` has succeeded (296 of 296 rows copied)
- `AdventureWorks`.`Employee` has succeeded (290 of 290 rows copied)
- `AdventureWorks`.`EmployeeAddress` has succeeded (290 of 290 rows copied)
- `AdventureWorks`.`Shift` has succeeded (3 of 3 rows copied)
- `AdventureWorks`.`Department` has succeeded (16 of 16 rows copied)
- `AdventureWorks`.`JobCandidate` has succeeded (13 of 13 rows copied)
- `AdventureWorks`.`EmployeePayHistory` has succeeded (316 of 316 rows copied)
7 tables of 7 were fully copied
Copy data to target RDBMS finished
Finished performing tasks.
```

10.8.12 Migration Report

Displays the final report that summarizes the migration process.

Figure 10.67 MySQL Workbench migration: Migration Report



```
--------------------------------------------------------------------------------
MySQL Workbench Migration Wizard Report

Date: Wed Apr 09 20:04:47 2014
Source: Microsoft SQL Server 9.0.5000
Target: MySQL 5.6.17
--------------------------------------------------------------------------------

I. Migration

1. Summary

Number of migrated schemas: 1

1. AdventureWorks
Source Schema:   AdventureWorks

- Tables:             7
- Triggers:           0
- Views:              6
- Stored Procedures:  3
- Functions:          0


2. Migration Issues
  - Name
    note  Collation Latin1_General_CS_AS migrated to utf8_general_ci
  - GroupName
```

10.9 MySQL Workbench Migration Wizard FAQ

Frequently Asked Questions with answers.

10.9.1. While using the Postgresql psqlodbc driver, I see the following error: `('08001', '[08001] Already connected. (202) (SQLDriverConnect)')`

This means that PostgreSQL is not configured to accept connections from the source IP.

Appendix A MySQL Workbench Frequently Asked Questions

FAQ Categories

- **Basic Usage**
- **MySQL Workbench Functionality**
- **Data Management**
- **General**

Basic Usage

A.1. What is a MySQL connection? Why might I need to create more than one?

A MySQL connection links (connects) Workbench to a MySQL server. Most actions performed within Workbench are then performed against the connected MySQL server. Each MySQL connection contains its own set of definitions, so you might define multiple MySQL connections in Workbench. For example, the connections might connect to different MySQL servers, or the same MySQL server with different user names, or enable SSL for one, or you might set up a connection to a remote MySQL server (on your web host?) using the SSH options, and so on.

As for multiple connections to the same local MySQL server, you might have one connection using "root" with another using a less privileged user. Depending on how you set up the users, they may (or may not) both have rights to see and use the same databases (information). For example, you might use Workbench to configure and use the less-privileged user that you use for your web application.

So to summarize, connections simply connect to the MySQL server. If two connections use the same exact information then the results in Workbench will be identical. However, that is not a common use case. For additional information about MySQL connections in MySQL Workbench, see Chapter 5, *MySQL Connections*.

A.2. How do I create a MySQL database (schema) in MySQL Workbench?

- Open a MySQL connection to open the SQL editor.
- On the left pane there is an Object Browser that contains two tabs titled **Management** and **Schemas**. Choose the schemas tab (default).
- Right-click anywhere in the **Schemas** pane and choose **Create Schema** from the context-menu.
- Follow the schema creation wizard by naming your new schema, and click **Apply** to create your new schema.

Other options include clicking the "Create Schema" icon on the main navigation bar, or executing a "CREATE SCHEMA your_db_name" query in the SQL editor.

A.3. Is there an easy way to select all data from a table, and then see the results?

From the schema navigator, hover over the table and click the ▦ icon. This executes a "SELECT * FROM schema.table" query and loads the results into the result grid. From there you can view or edit the data.

Alternatively, right-click on a table and select **Select Rows - Limit 1000** form the context menu.

Workbench Functionality

A.1. How do I use the SSL Certificate wizard to enable SSL for both my MySQL server and MySQL client?

Execute the wizard to generate the SSL certificates, and then modify your MySQL server's configuration file (`my.cnf` or `my.ini`) accordingly. You can copy-n-paste entries for the SSL options from the generated `sample-my.cnf` sample file. Next, confirm that the SSL **CA File**, **CERT File**, and **Key File** values are properly set under the **SSL** tab for your MySQL connection. Set **Use SSL** to either *Require* (recommended) or *If available*, and then execute **Test Connection**. This should report that SSL is enabled.

Failed SSL connections are logged in the MySQL Workbench log file. For additional information about the log file's location, see Section 3.3, “MySQL Workbench Settings and Log Files”.

For additional information, see Section 5.3.4, “SSL Wizard (Certificates)”.

A.2. How do I copy my saved MySQL connections in Workbench to a different computer?

From the main navigation menu, choose **Tools**, **Configuration**, and then **Backup Connections** to create a Zip file with your configured MySQL connections. Next, load this file into your new Workbench instance by using the related **Restore Connections** option.

A.3. How can I view my MySQL Workbench query history?

In bottom pane, change **Action Output** to **History** and then choose the appropriate date.

The SQL statement history is stored as plain text on your system under your user's MySQL Workbench configuration path in the `sql_history` directory. These files are organized per date (such as 2014-01-15) and contain your MySQL Workbench SQL statement history for all MySQL connections.

A.4. Can I preserve a results tab rather than have it refresh every time I execute a statement?

Yes, you can pin the results tab to force it to remain and be unaffected by UPDATE and other statements. Do that by right-clicking the result tab and choose "Pin Tab" from the context-menu, or left-click the little pin icon to toggle it. Now, execute your other queries and then refresh the pinned tab (there is a "refresh" icon in the result grid's menu).

A.5. How does the embedded web browser functionality work? For example, clicking **Workbench Forum** on the Home screen opens the forum in its own embedded MySQL Workbench tab.

The Webkit system library is used on OS X, Internet Explorer is used on Windows, and Linux opens the default browser externally rather than an embedded browser. Pressing **Modifier** + **Arrow** moves the browser history forward and back.

Additionally, for information about creating your own Home screen links, see Section 3.4, “Tutorial: Add a Custom Link to the Home Page”.

A.6. How does MySQL Workbench increase import performance?

When a model is exported (**Database**, **Forward Engineer...**), some MySQL server variables are temporarily set to enable faster SQL import by the server. The statements added at the start of the code are:

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';
```

These statements function as follows:

- `SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;`: Determines whether `InnoDB` performs duplicate key checks. Import is much faster for large data sets if this check is not performed. For additional information, see `unique_checks`.
- `SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;`: Determines whether the server should check that a referenced table exists when defining a foreign key. Due to potential circular references, this check must be turned off for the duration of the import, to permit defining foreign keys. For additional information, see `foreign_key_checks`.
- `SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';`: Sets SQL_MODE to `TRADITIONAL`, causing the server to operate in a more restrictive mode, and `ALLOW_INVALID_DATES`, causing dates to not be fully validated.

These server variables are then reset at the end of the script using the following statements:

```
SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Workbench Behavior

A.1. Why do my query results sometimes say **Read Only** but other times I can edit data in the results grid?

Data in the query results grid is only editable when the query results includes a primary key. For example, "SELECT type FROM food" will be read-only if "type" is not a primary key, but "SELECT id, type FROM food" will be editable when "id" is a primary key. Typically, "SELECT *" syntax is used in Workbench which often includes query results with a primary key.

For additional information, hover over the "Read Only" icon to reveal a tooltip that explains why your result set is in read-only mode.

A.2. I'm attempting to execute a DELETE query but the query fails with an "Error Code: 1175" error. How do I proceed?

By default, Workbench is configured to not execute DELETE or UPDATE queries that do not include a WHERE clause on a KEY column. To alter this behavior, open your Workbench **Preferences**, select the **SQL Editor** section, and disable the following preference:

[ ] "Safe Updates". Forbid UPDATEs and DELETEs with no key in WHERE clause or no LIMIT clause.

Changing this preference requires you to reconnect to your MySQL server before it can take affect.

A.3. My MySQL server connection is timing out with an error like "Error Code: 2013. Lost connection to MySQL server during query". Can I adjust the timeout?

Yes, go to **Preferences**, **SQL Editor**, and adjust the **DBMS connection read time out** option that defaults to 600 seconds. This sets the maximum amount of time (in seconds) that a query can take before MySQL Workbench disconnects from the MySQL server.

A.4. What do the column flag acronyms (PK, NN, UQ, BIN, UN, ZF, AI, G) in the MySQL Workbench Table Editor mean?

Checking these boxes will alter the table column by assigning the checked constraints to the designated columns.

Hover over an acronym to view a description, and see the Section 8.1.11.2, "The Columns Tab" and MySQL CREATE TABLE documentation for additional details.

Data Management

A.1. How do I import comma-separated values (CSV) data into MySQL using Workbench?

Importing CSV data into a new or existing *table*: the **Table Data Import** wizard imports configurable CSV data into a new or existing table. This option was added in MySQL Workbench 6.3.

Importing CSV data into a *result set*: the **Import records from external file** wizard imports CSV data directly into a result set's view.

Alternatively, the **Data Import** wizard imports your saved MySQL files into your MySQL server. For additional information, see Section 6.5, "Data Export and Import".

If you are importing Excel files, then consider using the official MySQL for Excel Add-on for Excel.

A.2. How do I export MySQL data to a plain text file with a format such as CSV, JSON, or XML?

The results view panel in Workbench has an "Export recordset to an external file" option that exports your result set to a wide variety of formats. For additional information, see Export a Result Set.

Note

This is different than the **Data Export** wizard that exports your MySQL data to standard MySQL formats. For additional information about that, see Section 6.5, "Data Export and Import".

If you are exporting to Excel, then consider using the official MySQL for Excel Add-on for Excel.

A.3. How to export (save) a MySQL database to a text file?

Open a MySQL connection, and select **Server** from the main navigation menu and choose **Data Export** to open the data export wizard. Alternatively, choose **Data Export** from the left Management pane for the desired MySQL selection.

Here you can choose which databases to export, whether or not to include the data, dump to a single file or multiple files (one per table), and more. For additional details, see Section 6.5, “Data Export and Import”.

General

A.1. I'm forced to use MySQL Workbench 5.2.x, is its documentation available?

Although the 5.2.x branch is no longer maintained, its documentation is archived at http://dev.mysql.com/doc/index-archive.html.

Appendix B Keyboard Shortcuts

The following tables list keyboard shortcuts for MySQL Workbench commands. **Modifier** in the tables stands for the platform-specific modifier key. This is **Command** on OS X, **Control** on other platforms. On OS X, the **Alt** key is **Option**.

There are keyboard shortcut tables for the File, Edit, View, Arrange, Model, Query, Database, Scripting, Help, and EER Diagram Mode menus.

File Menu

Table B.1 File menu keyboard shortcuts

Function	Keyboard Shortcut	Context
New Model	Modifier+N	All
Open Model	Modifier+O	All
Open SQL Script	Modifier+Shift+O	SQL Editor
Close Tab	Modifier+W, Modifier+F4 on Windows	All
Save Model	Modifier+S	Model
Save Script	Modifier+S	SQL Editor
Save Model As	Modifier+Shift+S	Model
Save Script As	Modifier+Shift+S	SQL Editor
Forward Engineer SQL CREATE Script	Modifier+Shift+G	Model
Forward Engineer SQL ALTER Script	Modifier+Alt+Y	Model
Synchronize With SQL CREATE Script	Modifier+Shift+Y	Model
Print	Modifier+P	EER Diagram mode only
Exit	Modifier+Q	All

Edit Menu

Table B.2 Edit menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Undo	Modifier+Z	Model, EER Diagram
Redo	Modifier+Y, Modifier+Shift+Z (OS X)	Model, EER Diagram
Cut	Modifier+X	All
Copy	Modifier+C	All
Paste	Modifier+V	All
Delete	Modifier+Delete, Command+BackSpace (OS X)	All
Edit Selected	Modifier+E	Model, EER Diagram
Edit Selected in New Window	Modifier+Shift+E	Model, EER Diagram
Select All	Modifier+A	EER Diagram

Function	Keyboard Shortcut	Context
Find	Modifier+F	All
Find Advanced	Modifier+Alt+F	All
Find Next	F3	All
Find Previous	Shift+F3	All
Search and Replace	Modifier+Shift+F	All
Comment/Uncomment lines of SQL	Modifier+/	SQL Editor
Auto-Complete SQL	Modifier+Space	SQL Editor

View Menu

Table B.3 View menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Output Window	Modifier+F2, Modifier+Option+2 (OS X)	All
Set Marker n	Modifier+Shift+n (n is integer 1..9)	EER Diagram
Go to Marker n	Modifier+n (n is integer 1..9)	EER Diagram

Arrange Menu

Table B.4 Arrange menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Bring to Front	Modifier+Shift+F	EER Diagram
Send to Back	Modifier+Shift+B	EER Diagram

Model Menu

Table B.5 Model menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Add Diagram	Modifier+T	Model, EER Diagram
Validate All	Modifier+Alt+V	Model, EER Diagram
Validate All (MySQL)	Modifier+Alt+B	Model, EER Diagram
Model Options	Command+Alt+, (Shortcut available only on OS X)	Model, EER Diagram

Query Menu

Table B.6 Query menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Execute statement	Modifier+Return	SQL Editor
Execute statements	Modifier+Shift+Return	SQL Editor
New Tab	Modifier+T	SQL Editor

Database Menu

Table B.7 Database menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Query Database	Modifier+U	All
Reverse Engineer	Modifier+R	Model, EER Diagram
Forward Engineer	Modifier+G	Model, EER Diagram
Synchronize Model	Modifier+Y	Model, EER Diagram

Scripting Menu

Table B.8 Scripting menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Scripting Shell	Modifier+F3, Modifier+Option+3 (on OS X)	All
Run Workbench Script File	Modifier+Shift+R	All

Help Menu

Table B.9 Help menu keyboard shortcuts

Function	Keyboard Shortcut	Context
Help Index	F1, Command+Option+question (on OS X)	All

EER Diagram Mode

In the EER Diagram view, a number of other keyboard shortcuts are available.

Table B.10 EER diagram mode keyboard shortcuts

Function	Keyboard Shortcut
Selection tool	Escape
Hand tool	H
Delete tool	D
Layer tool	L
Note tool	N
Image tool	I
Table tool	T
View tool	V
Routine Group tool	G
Non-Identifying Relationship 1:1	1
Non-Identifying Relationship 1:n	2
Identifying Relationship 1:1	3
Identifying Relationship 1:n	4
Identifying Relationship n:m	5
Relationship Using Existing Columns	6

Table B.11 Keyboard shortcut changes

MySQL Workbench version	The Change
5.2.45	The "Modifier+/" shortcut was added to comment/uncomment SQL in the SQL editor
5.2.45	On Microsoft Windows, the "Modifier+W" shortcut was changed to "Control+F4" -- this shortcut closes MySQL Workbench tabs

Appendix C Extending Workbench

Table of Contents

MySQL Workbench provides an extension and scripting system that enables the developer to extend MySQL Workbench capabilities. While the core of MySQL Workbench is developed using C++, it is possible to harness this core functionality using the Python scripting language. MySQL Workbench also provides access to a cross-platform GUI library, MForms, which enables the creation of extensions that feature a graphical user interface.

The extension system enables the following capabilities:

- Automate common tasks
- Extend the Workbench user-interface
- Create Tools/Plugins (code which can be invoked from the Workbench menu system)
- Manipulate schemata
- Create custom Workbench features

C.1 GRT and Workbench Data Organization

The GRT, or Generic RunTime, is the internal system used by Workbench to hold model document data. It is also the mechanism by which Workbench can interact with Modules and Plugins. Workbench model data, such as diagrams, schemata, and tables, is stored in a hierarchy of objects that can be accessed by any plugin. The information is represented using standard data types: integers, doubles, strings, dicts, lists, and objects.

The GRT can be accessed using the Python scripting language. Awareness is required of how the GRT data types map into Python. For example, the GRT integer, double, and string data types are seen as corresponding Python data types. Lists and dicts are kept in their internal representation, but can generally be treated as Python lists and dicts, and accessed in the usual way. Objects contain data fields and methods, but the GRT recognizes only objects from a pre-registered class hierarchy.

It is possible to fully examine the classes contained within the GRT using the Workbench Scripting Shell. Dots in class names are changed to underscores in their Python counterparts. For example, `db.mysql.Table` becomes `db_mysql_Table` in Python.

The Application Objects Tree (GRT Tree)

As mentioned previously, Workbench document data is stored in an object hierarchy. This hierarchy is known as the GRT Tree. The GRT Tree can be accessed and modified using Python or C++. Be careful when modifying the GRT Tree as mistakes can lead to document corruption. Backups should be made before manipulating the tree. Read-only access to the tree is the safest approach, and is sufficient in most cases.

The main nodes in the Application Object Tree

Table C.1 The main nodes in the Application Object Tree

Node	Description
wb.registry	Application data such as plugin registry, list of editors, and options.
wb.customData	A generic dictionary for data you can use to store your own data. This dictionary is saved and reloaded with Workbench and is global (not document specific).
wb.options	Contains some default options that are used by Workbench.
wb.rdbmsMgmt	Internal registry of supported RDBMS modules, known data types.
wb.doc	The currently loaded model document.
wb.doc.physicalModels[0]	The currently loaded model object, containing the database catalog and diagrams.
wb.doc.physicalModels[0].catalog	The database catalog for the model. Contains the list of schemata.
wb.doc.physicalModels[0]catalog.schemata	List of schemata in the model. Individual schema can be accessed as a list: schemata[0], schemata[1] ...
wb.doc.physicalModels[0].catalog.schemata[0].tables (.views, .routines, ...)	Lists of tables, views, routines in the schema.
wb.doc.physicalModels[0].diagrams	List of EER diagrams in the model.
wb.doc.physicalModels[0].diagrams[0].figures (.layers, .connections, ...)	List of figures, layers, connections (relationships) in the diagram.

C.2 Modules

In the GRT Modules are libraries containing a list of functions that are exported for use by code in other modules, scripts, or Workbench itself. Modules can be written in C++ or Python, but the data types used for arguments and the return value must be GRT types.

GRT modules are similar to Python modules, but are imported from the built-in `grt` module, instead of directly from an external file. The list of modules loaded into the `grt` module is obtained from `grt.modules`. Modules can be imported in Python using statements such as `from grt.modules import WbModel`.

To export functions as a module from Python code, you must carry out the following steps:

1. The source file must be located in the user modules folder. This path is displayed in the Workbench Scripting Shell with the label **Looking for user plugins in...**. It is also possible to install the file using the main menu item **Scripting**, **Install Plugin/Module File**.

Default module file locations:

Table C.2 Default User Module File Location

Operating System	File Path
Windows	%AppData%\MySQL\Workbench\modules
OS X	~username/Library/Application Support/MySQL/Workbench/modules
Linux	~username/.mysql/workbench/modules

2. The source file name must have the extension `_grt.py`; for example, `my_module_grt.py`.

3. Some module metadata must be defined. This can be done using the `DefineModule` function from the wb module:

```
from wb import *
ModuleInfo = DefineModule(name='MyModule', author='Your Name', version='1.0')
```

4. Functions to be exported require their signature to be declared. This is achieved using the export decorator in the previously created ModuleInfo object:

```
@ModuleInfo.export(grt.INT, grt.STRING)
def checkString(s):
   ...
```

For the `export` statement, the return type is listed first, followed by the input parameter types, specified as GRT typenames. The following typenames can be used:

- `grt.INT`: An integer value. Also used for boolean values.
- `grt.DOUBLE`: A floating-point numeric value.
- `grt.STRING`: UTF-8 or ASCII string data.
- `grt.DICT`: A key/value dictionary item. Keys must be strings.
- `grt.LIST`: A list of other values. It is possible to specify the type of the contents as a tuple in the form `(grt.LIST, <type-or-class>)`. For example, (grt.LIST, grt.STRING) for a list of strings. For a list of table objects, the following would be specified: `(grt.LIST, grt.classes.db_table)`.
- `grt.OBJECT`: An instance of a GRT object or a GRT class object, from `grt.classes`.

Note

These types are defined in the `grt` module, which must be imported before they are available for use.

The following code snippet illustrates declaring a module that exports a single function:

```
from wb import *
import grt

ModuleInfo = DefineModule(name='MyModule', author="your name", version='1.0')

@ModuleInfo.export(grt.DOUBLE, grt.STRING, (grt.LIST, grt.DOUBLE))
```

```
def printListSum(message, doubleList):
   sum = 0
   for d in doubleList:
      sum = sum + d
   print message, sum
   return sum
```

C.3 Plugins / Tools

Plugins are special Modules that are exposed to the user through the Workbench GUI. This is typically done using the main menu, or the context-sensitive menu. Much of the MySQL Workbench functionality is implemented using plugins; for example, table, view, and routine editors are native C++ plugins, as are the forward and reverse engineering wizards. The Administrator facility in MySQL Workbench is implemented entirely as a plugin in Python.

A plugin can be a simple function that performs some action on an input, and ends without further interaction with the user. Examples of this include auto-arranging a diagram, or making batch changes to objects. To create a simple plugin, the function must be located in a module and declared as a plugin using the `plugin` decorator of the `ModuleInfo` object.

Plugins can have an indefinite runtime, such as when they are driven by the user through a graphical user interface. This is the case for the object editors and wizards within MySQL Workbench. Although the wizard type of plugin must be declared in the usual way, only the entry point of the plugin will need to be executed in the plugin function, as most of the additional functionality will be invoked as a result of the user interacting with the GUI.

Note

Reloading a plugin requires MySQL Workbench to be restarted.

Imported plugin files (and their compiled counterparts) are stored here:

Table C.3 User Plugin File Location

Operating System	File Path
Windows	%AppData%\MySQL\Workbench\modules
OS X	~username/Library/Application Support/MySQL/Workbench/modules
Linux	~username/.mysql/workbench/modules

Declare a plugin using this syntax:

```
@ModuleInfo.plugin(plugin_name, caption, [input], [groups], [pluginMenu])
```

These parameters are defined as follows:

- **plugin_name**: A unique name for the plugin. It may contain only alphanumeric characters, dots, and underscores.
- **caption**: A caption to use for the plugin in menus.
- **input**: An optional list of input arguments.
- **groups**: Optional list of groups the plugin belongs to. Recognized values are:
  - `Overview/Utility`: The **Context** menu in the Model Overview.

    - `Model/Utility`: The menu for diagram objects.
    - `Menu/<category>`: The **Plugins** menu in the main menu.

- **pluginMenu**: Optional name of a submenu in the Plugins menu where the plugin should appear. For example, **Catalog**, **Objects**, **Utilities**. This is equivalent to adding a `Menu/<category>` in the groups list.

C.4 Adding a GUI to a Plugin Using MForms

MySQL Workbench is implemented with a C++ core back-end, and a native front-end for each supported platform. Currently the front-end is implemented with Windows Forms on Microsoft Windows, GTK+ on Linux, and Cocoa on OS X. This approach permits the application to have a native look and feel, while reducing the amount of work required to maintain the project. However, the GUI functionality required by MySQL Workbench can be met by a subset of graphical operations. These are implemented in a cross-platform GUI library, MForms. This further reduces the development effort because plugin developers can use MForms rather than writing front-end specific code for each supported platform. This also helps consistency of operation across all platforms. MForms is coded in C++, but provides a Python interface. To use it, the Python code must import the `mforms` module.

MForms Containers

Given the problems of using an absolute coordinate system across different platforms, MForms employs containers that perform automatic layout. The basic containers that MForms provides include:

- **Form**: A top-level window which can contain a single control, usually another container. The window will be sized automatically to fit its contents, but can also be sized statically.
- **Box**: This container can be filled with one or more controls in a vertical or horizontal layout. Each child control can be set to use either the minimum of required space, or fill the box in the direction of the layout. In the direction perpendicular to the layout, for example vertical in a horizontal layout, the smallest possible size that can accommodate all child controls will be employed. So, in this example, the smallest height possible to accommodate the controls would be used.
- **Table**: This container can organize one or more controls in a grid. The number of rows and columns in the table, and the location of controls within the grid, can be set by the developer.
- **ScrollView**: This container can contain a single child control, and adds scrollbars if the contents do not fit the available space.

C.5 The Workbench Scripting Shell

The Workbench Scripting Shell provides a means for entering and executing Python scripts. Through the use of the scripting shell, MySQL Workbench can support new behavior and data sources using code written in Python. The shell can also be used to explore the current Workbench GRT (Generic RunTime) facilities.

The scripting shell is not only useful for expanding MySQL Workbench. You can use a script file from the scripting shell command line to perform repetitive tasks programmatically.

Note

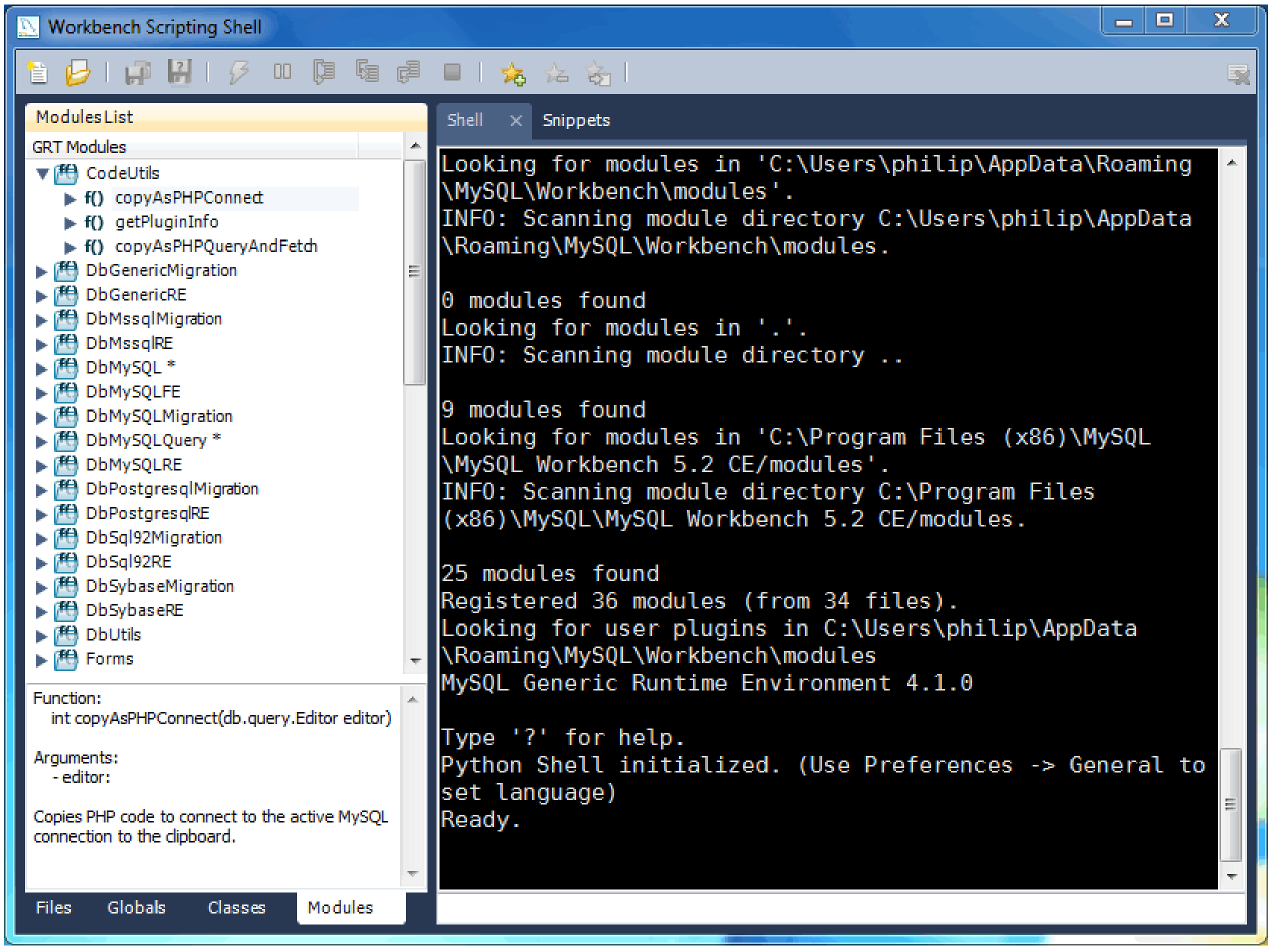
MySQL also has a product named MySQL Utilities, which is different than Workbench Scripting Shell.

C.5.1 Exploring the Workbench Scripting Shell

To open the Workbench Scripting Shell, select **Scripting**, **Scripting Shell** from the main menu. You can also open the Workbench Scripting Shell using the **Control** + **F3** key combination on Windows and Linux, **Command** + **F3** on OS X, or by clicking the shell button above the EER diagram navigator. The Workbench Scripting Shell will then open in a new dialog.

The following screenshot shows the Workbench Scripting Shell dialog.

Figure C.1 The Workbench Scripting Shell



```
Function:
   int copyAsPHPConnect(db.query.Editor editor)

Arguments:
   - editor:

Copies PHP code to connect to the active MySQL
connection to the clipboard.
```

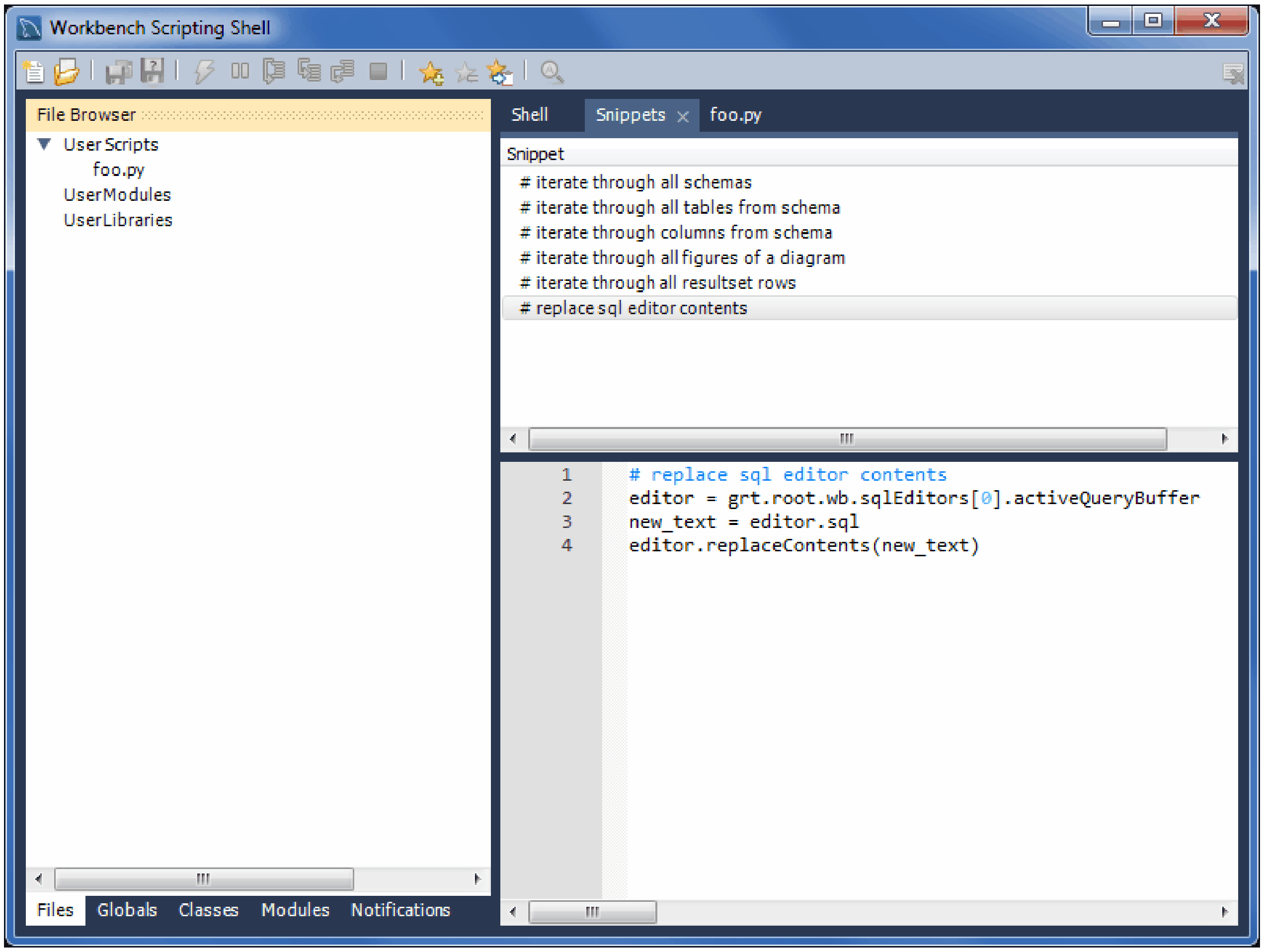
```
Looking for modules in 'C:\Users\philip\AppData\Roaming
\MySQL\Workbench\modules'.
INFO: Scanning module directory C:\Users\philip\AppData
\Roaming\MySQL\Workbench\modules.

0 modules found
Looking for modules in '.'.
INFO: Scanning module directory ..

9 modules found
Looking for modules in 'C:\Program Files (x86)\MySQL
\MySQL Workbench 5.2 CE/modules'.
INFO: Scanning module directory C:\Program Files
(x86)\MySQL\MySQL Workbench 5.2 CE/modules.

25 modules found
Registered 36 modules (from 34 files).
Looking for user plugins in C:\Users\philip\AppData
\Roaming\MySQL\Workbench\modules
MySQL Generic Runtime Environment 4.1.0

Type '?' for help.
Python Shell initialized. (Use Preferences -> General to
set language)
Ready.
```

C.5.2 The Shell Window

The Workbench Scripting Shell is primarily used for running Python scripts, or directly typing commands in Python. However, you can also use it to access the Workbench Scripting Shell Scripting Library functions and global functions and objects. To see the available commands, type "?". You can also cut and paste text to and from the shell window.

The **Snippets** tab is a scratch pad for saving code snippets. This makes it easy to reuse and execute code in MySQL Workbench.

Figure C.2 The Workbench Scripting Shell: Snippets


Workbench Scripting Shell
File Browser
User Scripts
foo.py
UserModules
UserLibraries
Files
Globals
Classes
Modules
Notifications
Shell
Snippets
foo.py
Snippet
# iterate through all schemas
# iterate through all tables from schema
# iterate through columns from schema
# iterate through all figures of a diagram
# iterate through all resultset rows
# replace sql editor contents

```
# replace sql editor contents
editor = grt.root.wb.sqlEditors[0].activeQueryBuffer
new_text = editor.sql
editor.replaceContents(new_text)
```

Opened script file tabs are to the right of the **Snippets** tab. Script tabs are labeled with the script's filename, or `Unnamed` for snippets without a name. You can cut-and-paste to and from the tabs, or right-click on a snippet to open a context menu with options to **Execute Snippet**, **Send to Script Editor**, or **Copy To Clipboard**.

While individual commands can be entered into the shell, it is also possible to run a longer script, stored in an external file, using the main menu item **Scripting**, **Run Workbench Script File**. When scripts are run outside of the shell, to see the output use the main menu item **View**, **Output**.

It is also possible to run script files directly from the shell. For details on running script files, type `? run` at the Workbench Scripting Shell prompt. The following message is displayed:

```
Help Topics
-----------
grt        General information about the Workbench runtime
scripting  Practical information when working on scripts and modules for Workbench
wbdata     Summary about Workbench model data organization
modules    Information about Workbench module usage
plugins    Information about writing Plugins and Modules for Workbench
Type '? [topic]' to get help on the topic.

Custom Python Modules
---------------------
```

```
   grt         Module to work with Workbench runtime (grt) objects
   grt.root    The root object in the internal Workbench object hierarchy
   grt.modules Location where Workbench modules are available
   grt.classes List of classes known to the GRT system
   mforms      A Module to access the cross-platform UI toolkit used in some Workbench features
   wb          Utility module for creating Workbench plugins

Type 'help(module/object/function)' to get information about a module, object or function.
Type 'dir(object)'                  to get a quick list of methods an object has.

For an introductory tutorial on the Python language,    visit http://docs.python.org/tutorial/
For general Python and library reference documentation, visit http://python.org/doc/
```

Within the Workbench Scripting Shell, there are five tabs on the top of the left side panel: **Files**, **Globals**, **Classes**, and **Modules**, and **Notifications**.

Note

An exception is thrown while attempting to use `input()` or read from `stdin`.

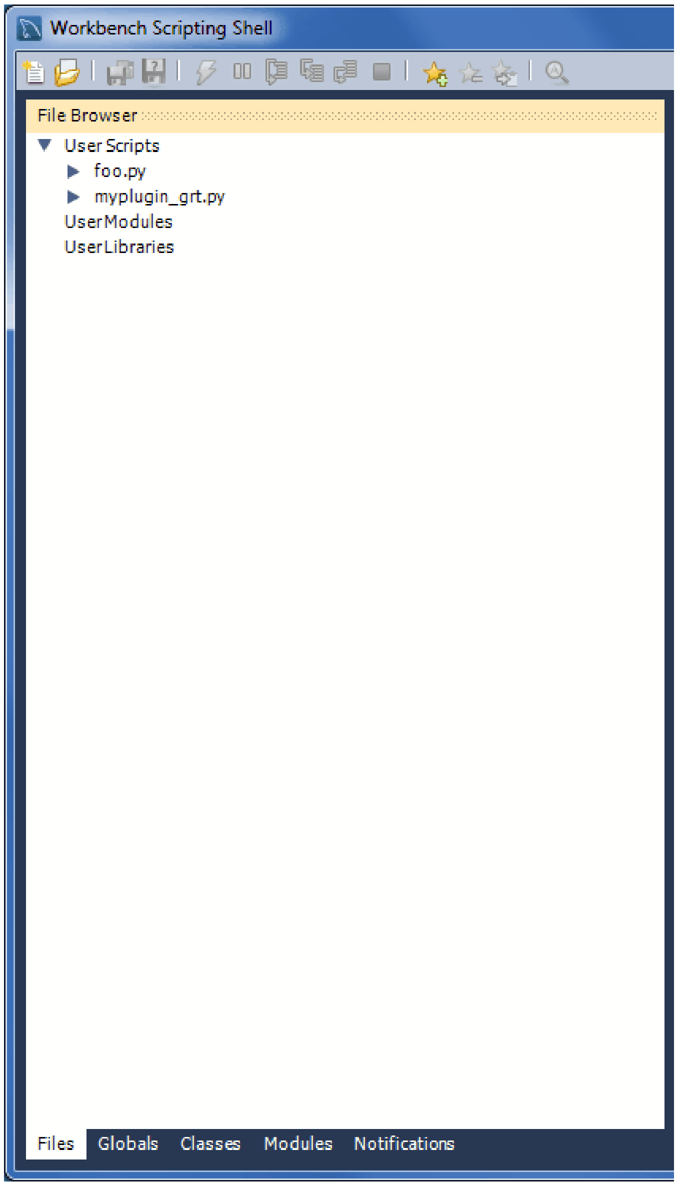
C.5.3 The Files, Globals, Classes, Modules, and Notifications Tabs

The Workbench Scripting Shell features the **Files**, **Globals**, **Classes**, **Modules**, and **Notifications** tabs, in addition to the main **Shell** tab.

The Files Tab

Lists folders and files for user-defined (custom) script files. The categories are **User Scripts**, **User Modules**, and **User Libraries**.

Figure C.3 The Workbench Scripting Shell tab: Files



By default, scripts are stored in the `scripts/` folder of your MySQL Workbench configuration folder. These default locations are:

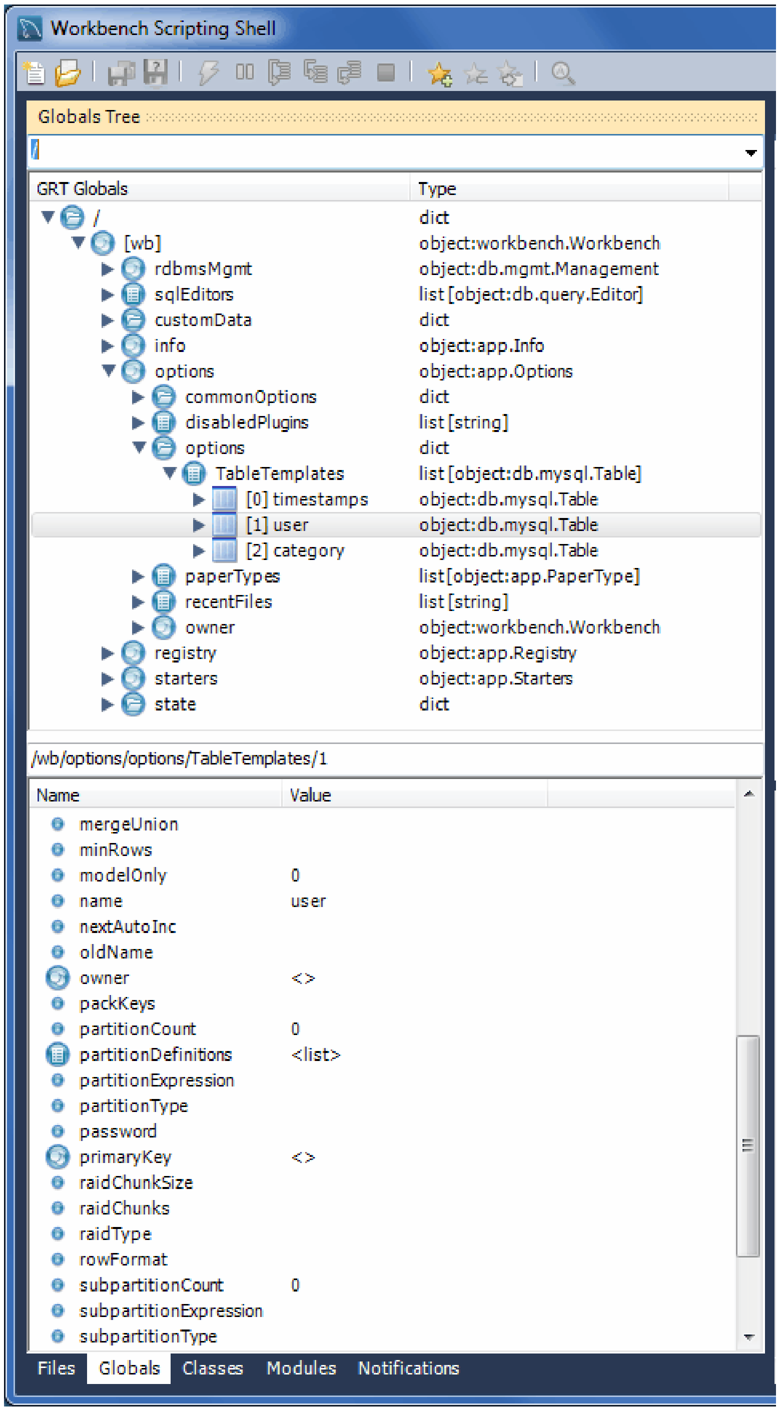
Table C.4 Default Scripts Location

Operating System	Default `scripts/` path
Linux	`~/.mysql/workbench/scripts`
OS X	`~/Library/Application\ Support/MySQL/Workbench/scripts/`
Windows 7	`C:\Users\[user]\AppData\Roaming\MySQL\Workbench\scripts\`

The Globals Tab

At the top of the window is a list that is used to select the starting point, or root, of the GRT Globals tree displayed beneath it. By default, this starting point is the root of the tree, that is, '/'. You can expand or collapse the GRT Globals tree as desired. The GRT Globals tree is the structure in which MySQL Workbench stores document data. Clicking any item results in its name and value being displayed in the panel below the tree.

Figure C.4 The Workbench Scripting Shell tab: Globals



GRT Globals	Type
/	dict
[wb]	object:workbench.Workbench
rdbmsMgmt	object:db.mgmt.Management
sqlEditors	list [object:db.query.Editor]
customData	dict
info	object:app.Info
options	object:app.Options
commonOptions	dict
disabledPlugins	list [string]
options	dict
TableTemplates	list [object:db.mysql.Table]
[0] timestamps	object:db.mysql.Table
[1] user	object:db.mysql.Table
[2] category	object:db.mysql.Table
paperTypes	list [object:app.PaperType]
recentFiles	list [string]
owner	object:workbench.Workbench
registry	object:app.Registry
starters	object:app.Starters
state	dict

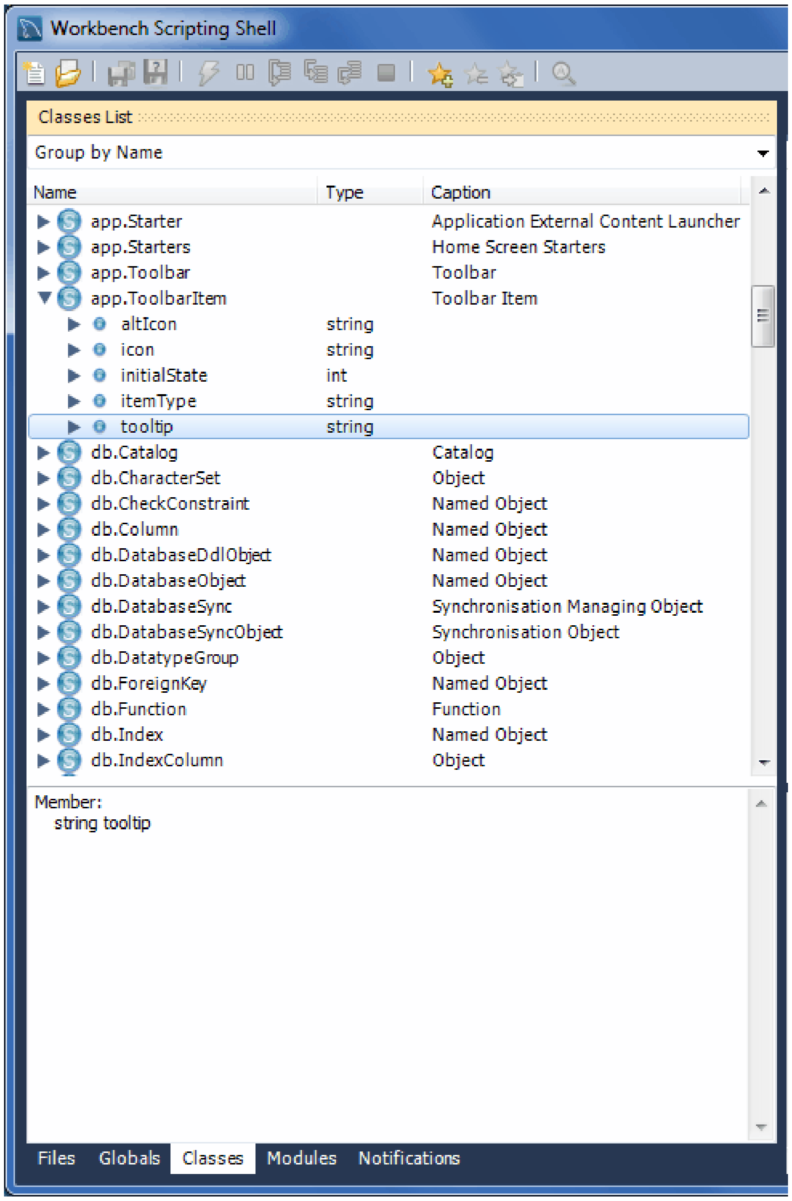
/wb/options/options/TableTemplates/1

Name	Value
mergeUnion	
minRows	
modelOnly	0
name	user
nextAutoInc	
oldName	
owner	<>
packKeys	
partitionCount	0
partitionDefinitions	<list>
partitionExpression	
partitionType	
password	
primaryKey	<>
raidChunkSize	
raidChunks	
raidType	
rowFormat	
subpartitionCount	0
subpartitionExpression	
subpartitionType	

The Classes Tab

A `class` is a user-defined data type formed by combining primitive data types: integers, doubles, strings, dicts, lists, and objects. This tab shows the definitions of the classes used by the objects in the **Modules** tab. Clicking a class causes a brief description of the class to be displayed in a panel below the classes explorer.

Figure C.5 The Workbench Scripting Shell tab: Classes



Group by Name

Name	Type	Caption
app.Starter		Application External Content Launcher
app.Starters		Home Screen Starters
app.Toolbar		Toolbar
app.ToolbarItem		Toolbar Item
altIcon	string	
icon	string	
initialState	int	
itemType	string	
tooltip	string	
db.Catalog		Catalog
db.CharacterSet		Object
db.CheckConstraint		Named Object
db.Column		Named Object
db.DatabaseDdlObject		Named Object
db.DatabaseObject		Named Object
db.DatabaseSync		Synchronisation Managing Object
db.DatabaseSyncObject		Synchronisation Object
db.DatatypeGroup		Object
db.ForeignKey		Named Object
db.Function		Function
db.Index		Named Object
db.IndexColumn		Object

Member:
string tooltip

When the **Classes** tab is selected, the list displays the following items:

- **Group by Name**: Group by the object name
- **Group by Hierarchy**: Group by inheritance
- **Group by Package**: Group by functionality

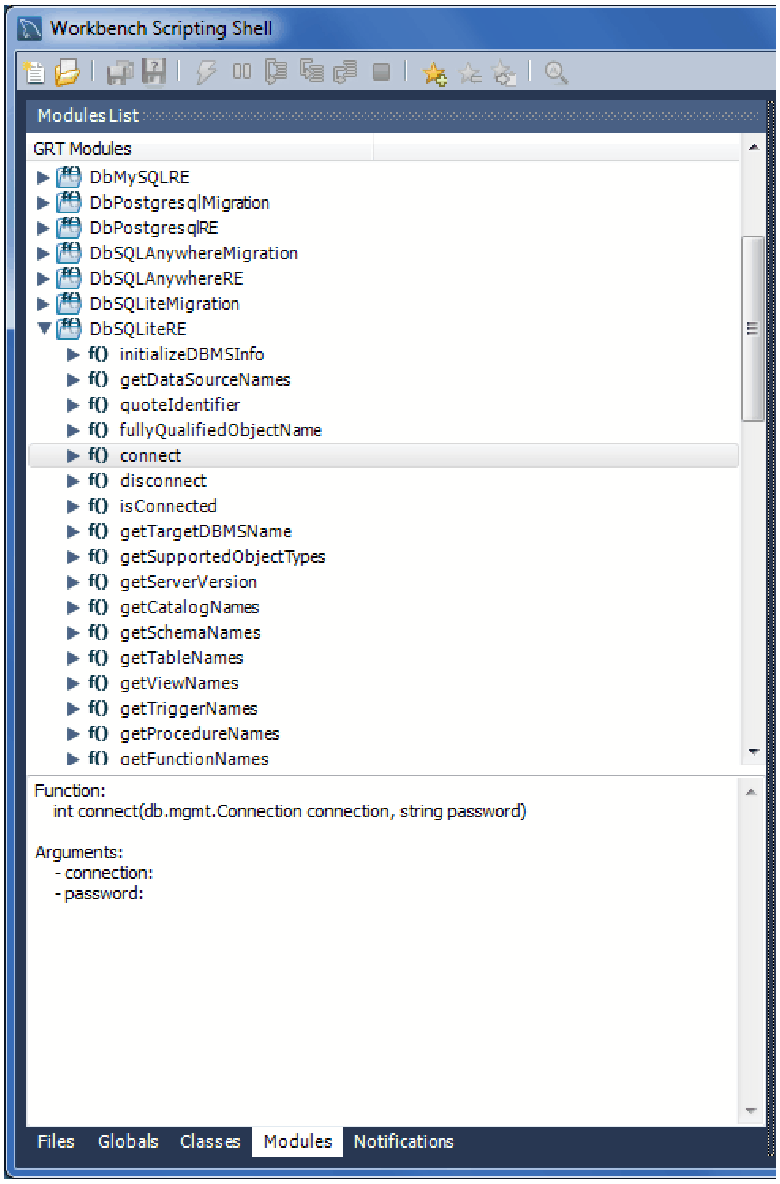
The default view for this tab is **Group By Name**. This view shows all the different objects arranged alphabetically. Click the **+** icon or double-click a package to show the properties of the struct.

If you switch to the hierarchical view, you will see `GrtObject`: the parent object from which all other objects are derived.

The Modules Tab

The **Modules** tab enables you to browse the MySQL Workbench installed modules and their functions. Clicking a module within the explorer causes its details to be displayed in a panel below the explorer. This facility is useful for exploring the available modules, and their supported functions. It is also a way to check whether custom modules have been correctly installed.

Figure C.6 The Workbench Scripting Shell tab: Modules

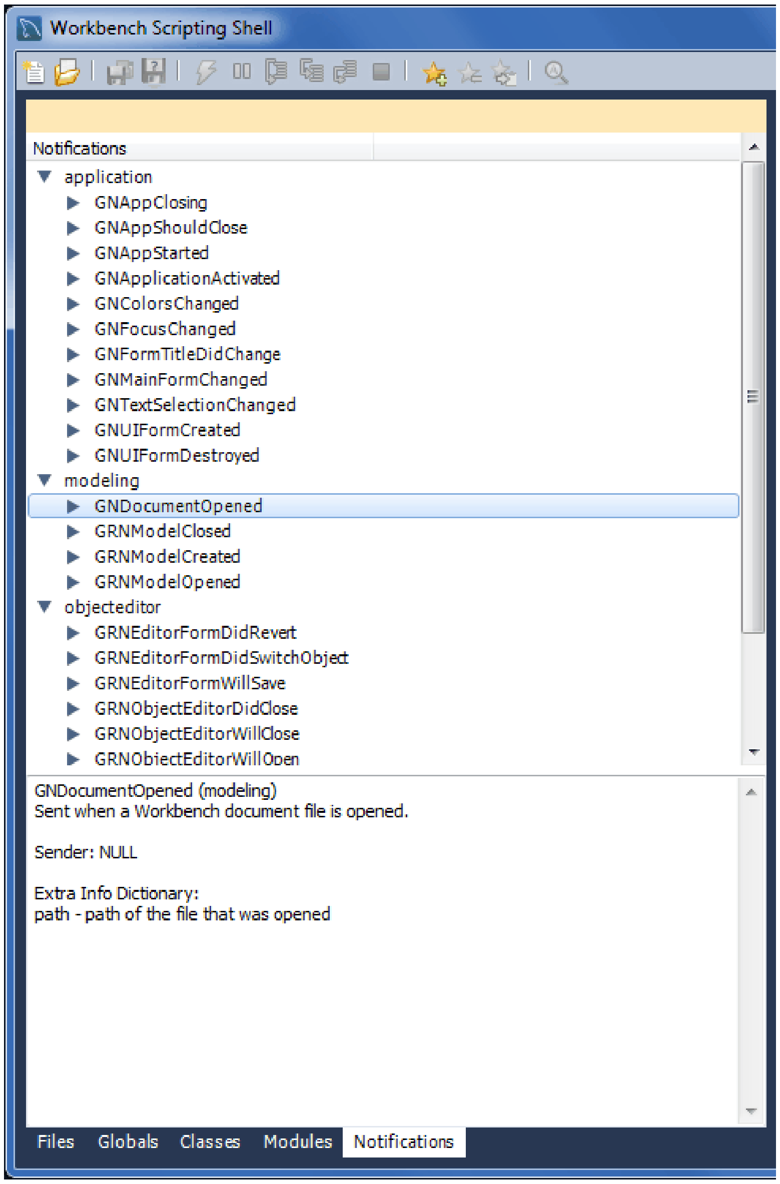


```
Function:
   int connect(db.mgmt.Connection connection, string password)

Arguments:
   - connection:
   - password:
```

The Notifications Tab

The set of `notification` classes used by MySQL Workbench modules. Click a notification class for a description of its use.

Figure C.7 The Workbench Scripting Shell tab: Notifications



```
▼ application
    ▶ GNAppClosing
    ▶ GNAppShouldClose
    ▶ GNAppStarted
    ▶ GNApplicationActivated
    ▶ GNColorsChanged
    ▶ GNFocusChanged
    ▶ GNFormTitleDidChange
    ▶ GNMainFormChanged
    ▶ GNTextSelectionChanged
    ▶ GNUIFormCreated
    ▶ GNUIFormDestroyed
▼ modeling
    ▶ GNDocumentOpened
    ▶ GRNModelClosed
    ▶ GRNModelCreated
    ▶ GRNModelOpened
▼ objecteditor
    ▶ GRNEditorFormDidRevert
    ▶ GRNEditorFormDidSwitchObject
    ▶ GRNEditorFormWillSave
    ▶ GRNObjectEditorDidClose
    ▶ GRNObjectEditorWillClose
    ▶ GRNObjectEditorWillOpen
```

```
GNDocumentOpened (modeling)
Sent when a Workbench document file is opened.

Sender: NULL

Extra Info Dictionary:
path - path of the file that was opened
```

C.6 Tutorial: Writing Plugins

These tutorials show you how to extend MySQL Workbench by creating custom plugins.

C.6.1 Tutorial: Generate PHP Code to Create a Connection with PDO_MySQL

MySQL Workbench includes a plugin that generates PHP code with the **mysqli** extension. This guide shows how to generate code using a different extension, and in this case we use PHP's PDO_MySQL extension. You might choose a different extension or a different language altogether, so adjust the generated code accordingly.

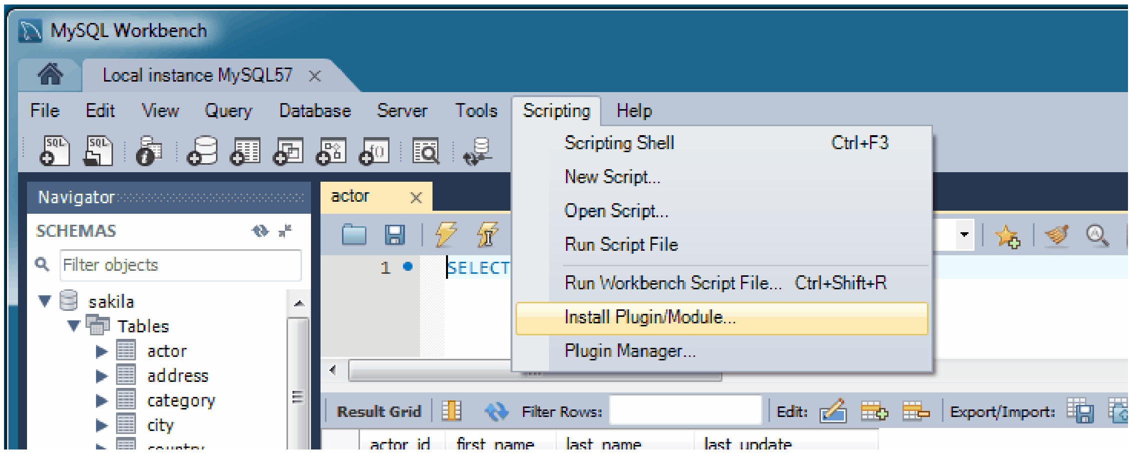
To begin, let us jump straight into the plugin code:

```
# import the wb module
from wb import DefineModule, wbinputs
# import the grt module
import grt
# import the mforms module for GUI stuff
import mforms

# define this Python module as a GRT module
ModuleInfo = DefineModule(name= "MySQLPDO", author= "Yours Truly", version="1.0")

@ModuleInfo.plugin("info.yourstruly.wb.mysqlpdo", caption= "MySQL PDO (Connect to Server)", input= [wbinputs.c
@ModuleInfo.export(grt.INT, grt.classes.db_query_Editor)

def mysqlpdo(editor):
    """Copies PHP code to connect to the active MySQL connection using PDO, to the clipboard.
    """
    # Values depend on the active connection type
    if editor.connection:
        conn = editor.connection

        if conn.driver.name == "MysqlNativeSocket":
            params = {
            "host" : "",
            "port" : "",
            "user" : conn.parameterValues["userName"],
            "socket" : conn.parameterValues["socket"],
            "dbname" : editor.defaultSchema,
            "dsn" : "mysql:unix_socket={$socket};dbname={$dbname}"
            }
        else:
            params = {
            "host" : conn.parameterValues["hostName"],
            "port" : conn.parameterValues["port"] if conn.parameterValues["port"] else 3306,
            "user" : conn.parameterValues["userName"],
            "socket" : "",
            "dbname" : editor.defaultSchema,
            "dsn" : "mysql:host={$host};port={$port};dbname={$dbname}"
            }
        text = """$host="%(host)s";
$port=%(port)s;
$socket="%(socket)s";
$user="%(user)s";
$password="";
$dbname="%(dbname)s";

try {
    $dbh = new PDO("%(dsn)s", $user, $password));
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

""" % params
        mforms.Utilities.set_clipboard_text(text)
        mforms.App.get().set_status_text("Copied PHP code to clipboard")
    return 0
```

This simple plugin generates PHP code to create a MySQL connection using PHP's PDO_MySQL extension. The DSN definition depends on the connection type in MySQL Workbench. The part you might want to modify is within the text definition. To demonstrate, first install the plugin:

1. Copy the plugin code into a new file. In our example, we name it `php-pdo-connect_grt.py` but you can use a different name as long as `_grt.py` is the suffix.

Figure C.8 Menu: Install Plugin/Module

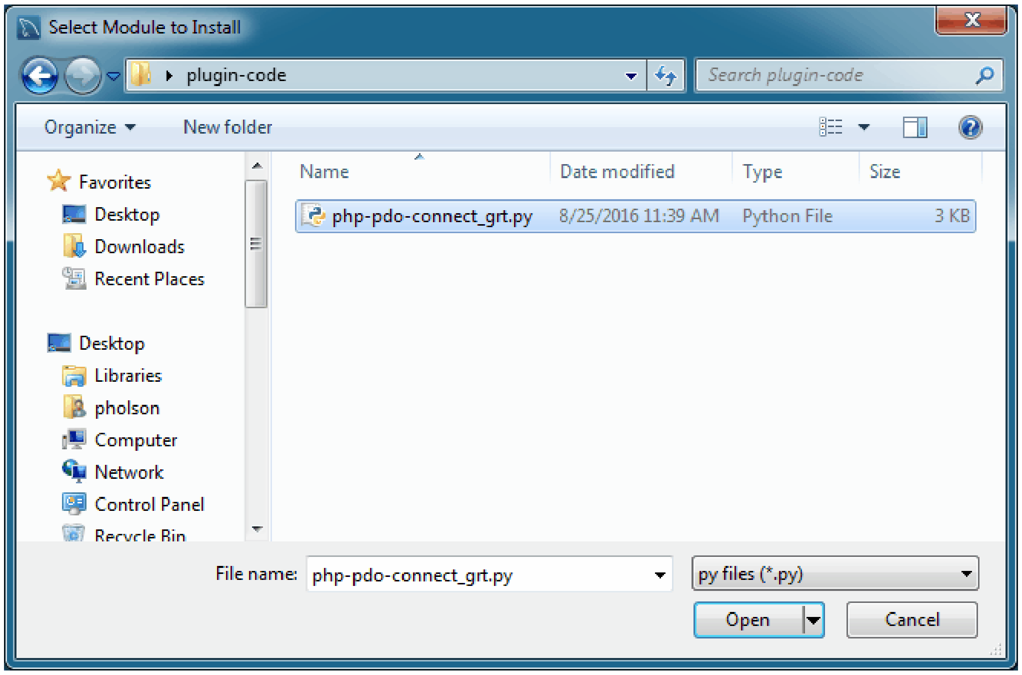


2. Open MySQL Workbench, choose **Scripting** from the main navigation menu, and then choose **Install Plugin/Module**. This opens a file menu titled **Select Module to Install**, so choose the plugin file created from the above code, such as `php-pdo-connect_grt.py`.

Note

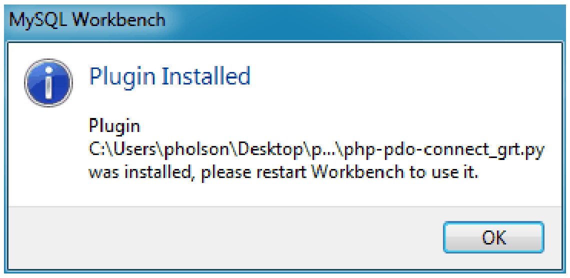
You could copy the file directly to the plugin folder instead of using the **Install Plugin/Module** interface. The result would be the same.

Figure C.9 Select Plugin File



3. Workbench must be restarted to install the plugin. Workbench will generate a compiled bytecode file (`.pyc`) from your source file. In our case, it generates `php-pdo-connect_grt.pyc`.

Figure C.10 Notice to Restart Workbench



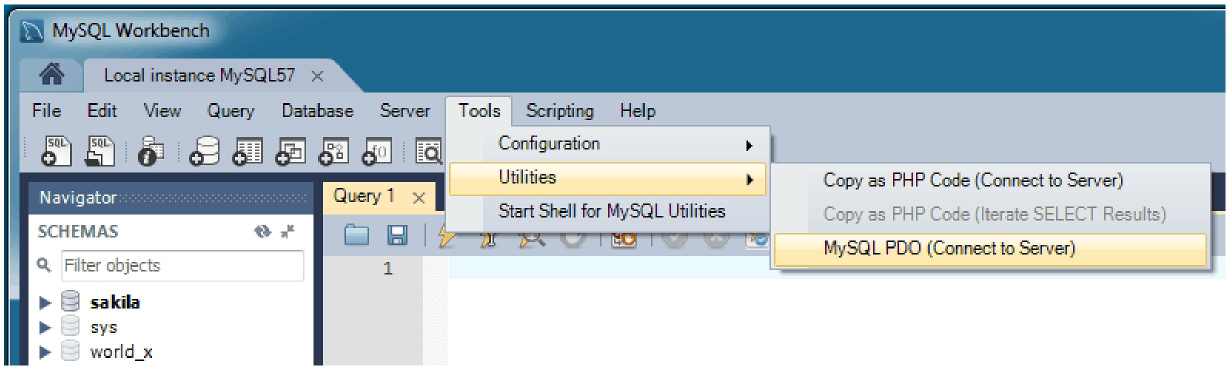
MySQL Workbench

Plugin Installed

Plugin
C:\Users\pholson\Desktop\p...\php-pdo-connect_grt.py
was installed, please restart Workbench to use it.

4. After restarting Workbench, load the MySQL connection that you want the PHP code to generate from. Then, choose **Scripting**, **Utilities**, and finally **MySQL PDO (Connect to Server)**, which is the `Caption` we defined in the plugin's code.

Figure C.11 Execute the Plugin



5. This copied the generated PHP code into your system's clipboard. Our example connection defines "sakila" as the default database, and here is what the generated code will look like:

```
$host="localhost";
$port=3306;
$socket="";
$user="root";
$password="";
$dbname="sakila";

try {
    $dbh = new PDO("mysql:host={$host};port={$port};dbname={$dbname}", $user, $password));
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}
```

C.6.2 Tutorial: Generating Foreign Keys with MyISAM

EER Diagrams are useful for visualizing complex database schemata. They are often created for existing databases, to clarify their purpose or document them. MySQL Workbench provides facilities for reverse engineering existing databases, and then creating an EER Diagram automatically. In this case, relationship lines between foreign keys in the table will automatically be drawn. This graphical representation makes the relationships between the tables much easier to understand. However, the older MyISAM storage engine does not include support for foreign keys. This means that MyISAM tables that are reverse engineered will not automatically have the relationship lines drawn between tables, making the database harder to understand. The plugin created in this tutorial gets around this problem by using the fact that a naming convention is often used for foreign keys: `tablename_primarykeyname`. Using this convention, foreign keys can automatically be created after a database is reverse engineered, which will result in relationship lines being drawn in the EER diagram.

Algorithm

The basic algorithm for this task would be as follows:

```
for each table in the schema
   for each column in the table
      look for another table whose name and primary key name match the current column name
      if such a table is found, add a foreign key referencing it
```

As iterating the complete table list to find a match can be slow for models with a large number of tables, it is necessary to optimize by pre-computing all possible foreign key names in a given schema.

```
import grt

def auto_create_fks(schema):
   fk_name_format = "%(table)s_%(pk)s"
   possible_fks = {}
   # create the list of possible foreign keys from the list of tables
   for table in schema.tables:
      if table.primaryKey:
         format_args = {'table':table.name, 'pk':table.primaryKey.name}
         fkname = fk_name_format % format_args
         possible_fks[fkname] = table

   # go through all tables in schema, this time to find columns that may be a fk
   for table in schema.tables:
      for column in table.columns:
         if possible_fks.has_key(column.name):
            ref_table = possible_fks[column.name]
            if ref_table.primaryKey.formattedType != column.type:
               continue
            fk = table.createForeignKey(column.name+"_fk")
            fk.referencedTable = ref_table
            fk.columns.append(column)
            fk.referencedColumn.append(ref_table.primaryKey)
            print "Created foreign key %s from %s.%s to %s.%s" \
            % (fk.name, table.name, column.name, ref_table.name, ref_table.primaryKey.name)

auto_create_fks(grt.root.wb.doc.physicalModels[0].catalog.schemata[0])
```

Creating a Plugin from a Script

To create a plugin from an arbitrary script, it is first necessary to make the file a module, and export the required function from it. It is then necessary to declare the module as a plugin, and specify the return type and input arguments.

```
from wb import *
import grt

ModuleInfo = DefineModule(name="AutoFK", author="John Doe", version="1.0")

@ModuleInfo.plugin("sample.createGuessedForeignKeys",
  caption="Create Foreign Keys from ColumnNames",
  input=[wbinputs.objectOfClass("db.mysql.schema")],
  groups=["Overview/Utility"])
```

```
@ModuleInfo.export(grt.INT, grt.classes.db_mysql_Schema)
def auto_create_fks(schema):
   ...
```

With the addition of the preceding code, the `auto_create_fks()` function is exported and will be added to the schema context menu in the model overview. When invoked, it receives the currently selected schema as its input.

Appendix D How To Report Bugs or Problems

The following is a list of tips and information that is helpful for reporting a MySQL Workbench bug.

A useful bug report includes:

- The exact steps taken to repeat the bug, ideally as a video if the bug is tricky to repeat
- A screenshot, if the bug is visual
- The error messages, which includes text sent to stdout and the GUI
- The MySQL Workbench Log file

  The log file location can be found using **Help**, **Locate Log Files** from within MySQL Workbench.

Bugs that cannot be reproduced are difficult and nearly impossible to fix, so it is important to provide the steps necessary to reproduce the bug.

Where to report a bug

Visit http://bugs.mysql.com/ and use one of the "MySQL Workbench" bug categories.

Log Levels

There are six different log levels, with increasing levels of verbosity: `error`, `warning`, `info`, `debug1`, `debug2`, and `debug3`. By default, the `error`, `warning` and `info` levels are enabled. There is also a "none" level that disables logging.

Important

Please enable the `debug3` level before generating a log for the report.

The enabled error log levels can be configured using an environment variable, or by using a command line parameter.

Both the environment variable and command line variants accept a single error level, but enabling a more verbose option will implicitly enable the levels below it. For example, passing in "info" will also enable the "error" and "warning" levels.

- Environment variable: `WB_LOG_LEVEL`

  Command line option: `--log-level` on OS X and Linux, and `-log-level` on Microsoft Windows

Note

If both the command line and environment variable are set, the command line takes precedence.

For example:

```
# Microsoft Windows
shell> cd "C:\Program Files (x86)\MySQL\MySQL Workbench CE 6.3.7\"
shell> MySQLWorkbench.exe -log-level=debug3
```

```
# OS X
shell> cd /Applications
shell> MySQLWorkbench --log-level=debug3

# Linux (Ubuntu)
shell> cd /usr/bin
shell> mysqlworkbench --log-level=debug3
```

If the `info` level is enabled, the system information and all paths used in the application are also logged. On Microsoft Windows, this also means that the log file contains the full set of current environment variables that are active for the program.

Operating System Specific Notes

Microsoft Windows

- Log file location: Near the user's app data folder, such as `C:\Users\[user]\AppData\Roaming\MySQL\Workbench\log` for Microsoft Windows 7.
- In case of errors (or exceptions), the log file contains the stack trace to the point MySQL Workbench can track it (usually only C# code, and not C++ code). Also, all warnings are added to the log if the warning (or greater) log level is enabled.
- If it is a crash and that cannot be replicated by the MySQL Workbench team, and the stack trace cannot be obtained, we will request a crashdump. Instructions for enabling a crashdump can be found here, and please also read the MSDN details for this as we need a full dump, and not the mini dump.
- For crashes related to display issues, start MySQL Workbench with the `-swrendering` parameter (and only then, as it switches off OpenGL rendering, which is of no use in WBA or WQE). This output will be added to the log file.
- If it is a crash when MySQL Workbench is started (especially if the error report includes something about `kernelbase.dll`), we will ask you to run `depends.exe` on the `MySQLWorkbench.exe` binary, and ask for the reported errors.
- If it is a crash when MySQL Workbench is started, and it is a 64-bit version of Microsoft Windows, check that the correct MSVC runtimes are installed. Often people install the 64-bit version of them, but only the 32-bit will function. More precisely: `MSVC 2010 runtime x86 (32-bit)`.

OS X

- Log File Location: `~/Library/Application Support/MySQL/Workbench/logs`
- System crash logs generated for Workbench are in `~/Library/Logs/DiagnosticReports/MySQLWorkbench*`

Linux

- Log File Location: `~/.mysql/workbench/logs`
- For a crash, we might ask for a stack trace that can generated by `gdb` by using the following steps:

Note

Because published MySQL Workbench builds lack debug symbols, this step is optional and will probably not be necessary.

    - In shell, execute `source /usr/bin/mysql-workbench`
    - Quit MySQL Workbench
    - In shell, execute `gdb /usr/bin/mysql-workbench-bin`
    - In the gdb interface, type `run`
    - In MySQL Workbench, repeat the crash
    - In the gdb interface, type `bt`
- If it is a crash, also run `glxinfo`. If that also crashes, then it is a driver/X server problem related to OpenGL that is not specific to MySQL Workbench.

Appendix E MySQL Enterprise Features

A MySQL Enterprise subscription is the most comprehensive offering of MySQL database software, services and support; it ensures that your business achieves the highest levels of reliability, security, and uptime.

An Enterprise Subscription includes a MySQL Workbench GUI for the following enterprise features:

- The MySQL Enterprise server: The most reliable, secure, and up-to-date version of the world's most popular open source database
- MySQL Enterprise Backup: Performs backup and restore operations for MySQL data, and MySQL Workbench offers a GUI for these operations
- MySQL Enterprise Audit: An easy to use auditing and compliance solution for applications that are governed by both internal and external regulatory guidelines
- MySQL Enterprise Firewall: regulatory guidelines
- DBDoc Model Reporting Templates: For accessing the Ctemplate System.
- Model Validation: Validation modules for testing models before implementing them, both with general RDMS and specific MySQL rules.
- **MySQL Production Support (MOS)**: Technical and consultative support when you need it, along with regularly scheduled service packs, and hot-fixes, and MySQL Workbench links to your My Oracle Support (MOS) service

For more information, visit http://www.mysql.com/enterprise

Appendix F MySQL Utilities

MySQL Utilities is a package of utilities that are used for maintenance and administration of MySQL servers. These utilities encapsulate a set of primitive commands, bundling them so they can be used to perform macro operations with a single command. MySQL Utilities can be installed with MySQL Workbench or as a standalone package.

They are a set of command-line utilities and a Python library for making the common tasks easy to accomplish. The library is written entirely in Python, meaning that it is not necessary to have any other tools or libraries installed to make it work. It is currently designed to work with Python v2.6 or later and there is no support (yet) for Python v3.1.

The utilities are available under the GPLv2 license, and are extendable using the supplied library.

For more information, see the MySQL Utilities manual at http://dev.mysql.com/doc/index-utils-fabric.html.

Installing The MySQL Utilities

MySQL Utilities development is managed elsewhere, and require a separate download. Attempting to start the MySQL Utilities when they are not installed will prompt for a download and their installation. See the MySQL Utilities manual for additional information.

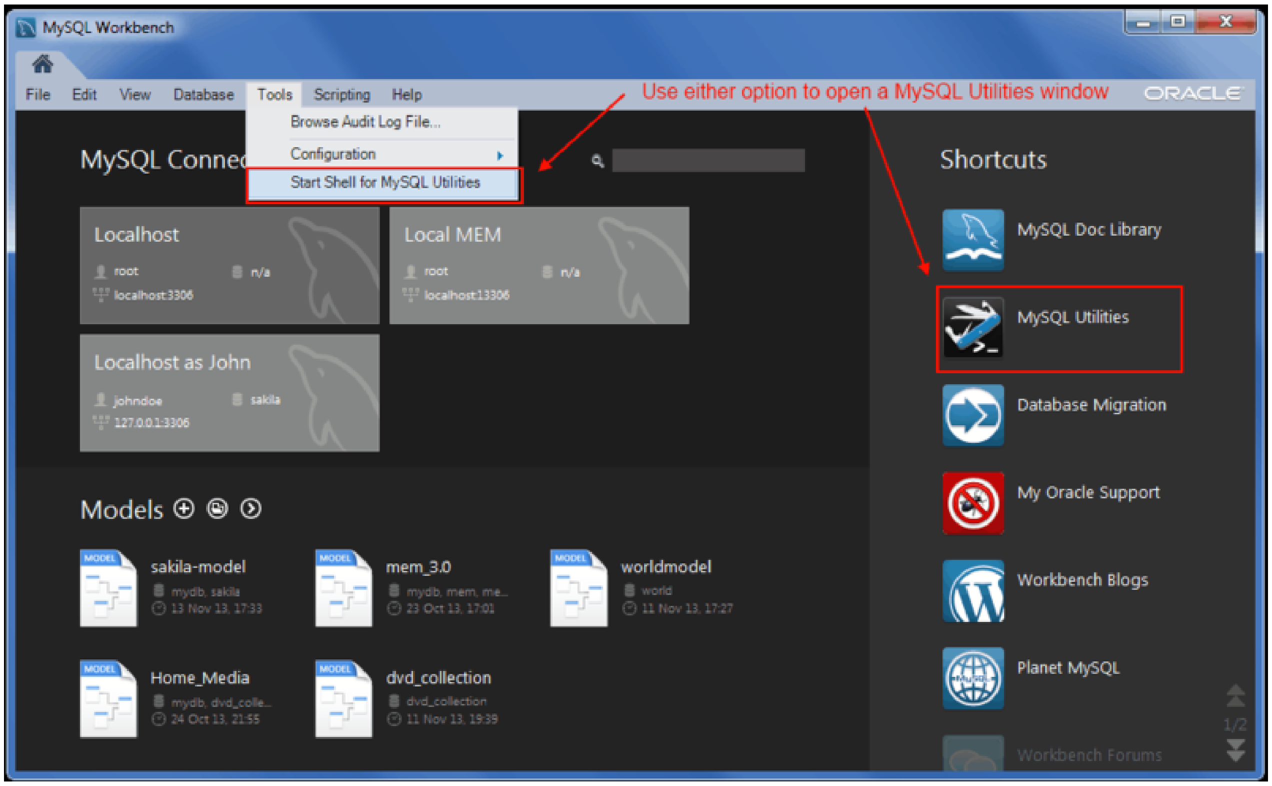
Note

MySQL Workbench searches for the `mysqluc` MySQL Utility in the system's `PATH` to determine if the MySQL Utilities are installed.

Opening MySQL Utilities From MySQL Workbench

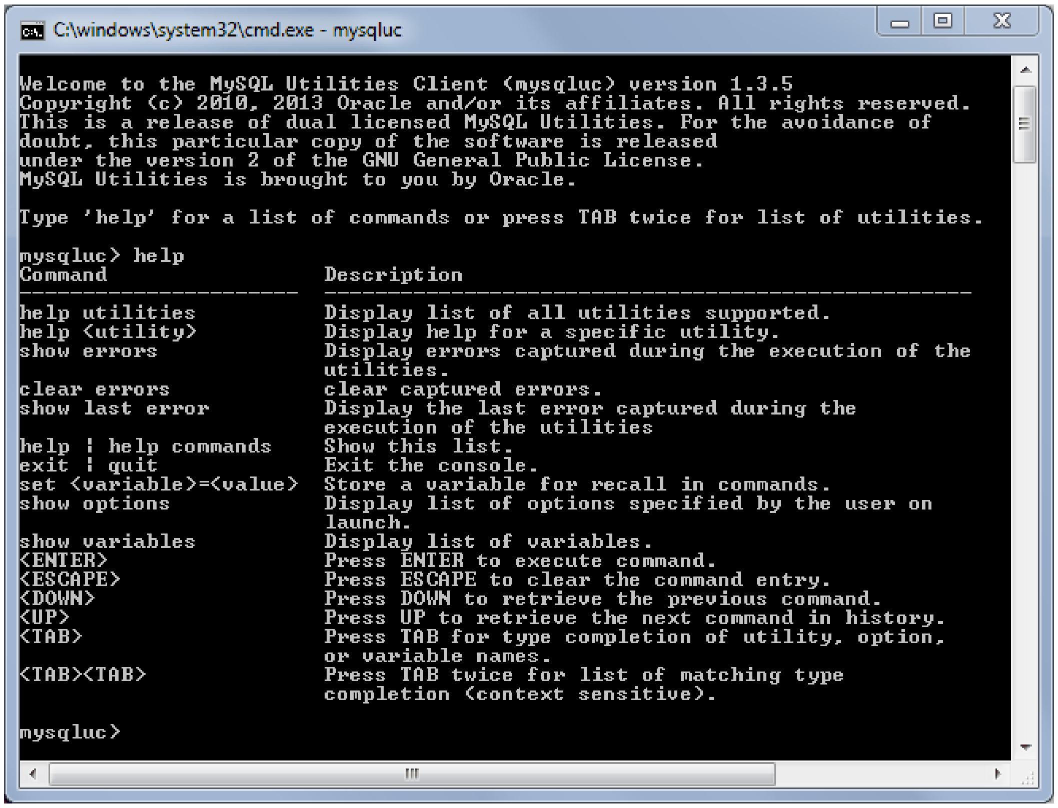
Open the MySQL Utility `mysqluc` (MySQL Utilities Unified Console) from MySQL Workbench from either the **Tools** menu, or from the MySQL Utilities shortcut on the Home page.

Figure F.1 Open MySQL Utilities From Workbench



The MySQL Utilities console window can be seen below, with the "help" command executed:

Figure F.2 The MySQL Utilities Console



```
Welcome to the MySQL Utilities Client (mysqluc) version 1.3.5
Copyright (c) 2010, 2013 Oracle and/or its affiliates. All rights reserved.
This is a release of dual licensed MySQL Utilities. For the avoidance of
doubt, this particular copy of the software is released
under the version 2 of the GNU General Public License.
MySQL Utilities is brought to you by Oracle.

Type 'help' for a list of commands or press TAB twice for list of utilities.

mysqluc> help
Command                  Description
----------------------   --------------------------------------------------
help utilities           Display list of all utilities supported.
help <utility>           Display help for a specific utility.
show errors              Display errors captured during the execution of the
                         utilities.
clear errors             clear captured errors.
show last error          Display the last error captured during the
                         execution of the utilities
help | help commands     Show this list.
exit | quit              Exit the console.
set <variable>=<value>   Store a variable for recall in commands.
show options             Display list of options specified by the user on
                         launch.
show variables           Display list of variables.
<ENTER>                  Press ENTER to execute command.
<ESCAPE>                 Press ESCAPE to clear the command entry.
<DOWN>                   Press DOWN to retrieve the previous command.
<UP>                     Press UP to retrieve the next command in history.
<TAB>                    Press TAB for type completion of utility, option,
                         or variable names.
<TAB><TAB>               Press TAB twice for list of matching type
                         completion (context sensitive).

mysqluc>
```