

MySQL Workbench

Abstract

This is the MySQL Workbench Reference Manual. It documents the MySQL Workbench Community and MySQL Workbench Commercial releases for versions 8.0 through 8.0.38.

MySQL Workbench is developed and tested with MySQL Server 8.0. MySQL Workbench may connect to MySQL Server 8.4 and higher but some MySQL Workbench features may not function with those later server versions.

MySQL Workbench platform support evolves over time. For the latest platform support information, see <https://www.mysql.com/support/supportedplatforms/workbench.html>.

For notes detailing the changes in each release, see the [MySQL Workbench Release Notes](#).

For legal information, including licensing information, see the [Preface and Legal Notices](#).

If you have not yet installed the MySQL Workbench Community release, please download your free copy from the [download site](#). The MySQL Workbench Community release is available for Microsoft Windows, macOS, and Linux.

For help with using MySQL, please visit the [MySQL Forums](#), where you can discuss your issues with other MySQL users.

Document generated on: 2024-07-31 (revision: 79173)

Table of Contents

Preface and Legal Notices	vii
1 General Information	1
1.1 What Is New in MySQL Workbench	1
1.1.1 New in MySQL Workbench 8.0 Release Series	1
1.1.2 New in MySQL Workbench 6.0 Release Series	6
1.2 MySQL Workbench Editions	53
2 Installation	55
2.1 System Requirements	55
2.2 Command-Line Options	57
2.3 MySQL Workbench on Windows	57
2.3.1 Installing	57
2.3.2 Launching	58
2.3.3 Uninstalling	59
2.4 MySQL Workbench on Linux	59
2.4.1 Installing	59
2.4.2 Launching	62
2.4.3 Uninstalling	63
2.5 MySQL Workbench on macOS	64
2.5.1 Installing	64
2.5.2 Launching	64
2.5.3 Uninstalling	65
3 Configuration	67
3.1 User Accessibility Options	67
3.2 Workbench Preferences	71
3.2.1 General Editors Preferences	72
3.2.2 SQL Editor Preferences	73
3.2.3 Administration Preferences	80
3.2.4 Modeling Preferences	81
3.2.5 Fonts and Colors Preferences	86
3.2.6 SSH Preferences	87
3.2.7 Other Preferences	89
3.3 MySQL Workbench Settings and Log Files	90
3.4 Common Preferences and Configurations	92
4 Home Screen Tab	95
5 Connections in MySQL Workbench	99
5.1 Creating A New MySQL Connection (Simple)	99
5.2 Creating A New MySQL Connection (Tutorial)	100
5.3 Manage Server Connections	111
5.3.1 Standard TCP/IP Connection Method	113
5.3.2 Local Socket/Pipe Connection Method	115
5.3.3 Standard TCP/IP over SSH Connection Method	116
5.3.4 LDAP and Kerberos Connection Methods	118
5.3.5 SSL Wizard (Certificates)	123
5.3.6 Remote Management	127
5.3.7 System Profile	128
5.3.8 Configure Server Management Wizard	129
5.3.9 The Password Storage Vault	132
5.3.10 Updating Old Authentication Protocol Passwords	132
5.4 Client Connections	136
6 Administrative Tasks	141
6.1 Server Administration	141

6.1.1	Server Logs	143
6.1.2	Service Control	144
6.1.3	Configuration (Options File)	145
6.2	Users and Privileges	146
6.3	Server Status	149
6.4	Status and System Variables	150
6.5	Data Export and Import	153
6.5.1	Table Data Export and Import Wizard	154
6.5.2	SQL Data Export and Import Wizard	162
6.5.3	Result Data Export and Import	166
6.6	MySQL Audit Inspector Interface	167
6.7	MySQL Enterprise Backup Interface	169
6.7.1	General Requirements	170
6.7.2	Online Backup	172
6.7.3	Backup Recovery	174
6.8	MySQL Enterprise Firewall Interface	178
6.9	wbcopytables Utility	180
7	Performance Tools	185
7.1	Performance Dashboard	185
7.2	Performance Schema Reports	186
7.3	Query Statistics	191
7.4	Visual Explain Plan	192
7.5	Tutorial: Using Explain to Improve Query Performance	195
8	Database Development	201
8.1	Visual SQL Editor	201
8.1.1	SQL Query Tab	203
8.1.2	SQL Query Toolbar	204
8.1.3	Query and Edit Menus	206
8.1.4	Result Grid	207
8.1.5	SQL Additions - Snippets Tab	211
8.1.6	SQL Additions - Context Help Tab	213
8.1.7	Output Panel	214
8.1.8	Table Data Search Tab	216
8.1.9	Export or Import a Table	217
8.1.10	MySQL Table Editor	218
8.1.11	Code Generation Overview	227
8.2	Object Management	232
8.2.1	Object Browser and Editor Navigator	232
8.2.2	Session and Object Information Panel	235
8.2.3	Schema and Table Inspector	235
9	Database Design and Modeling	239
9.1	Modeling Interface	240
9.1.1	Model Editor	241
9.1.2	EER Diagram Editor	256
9.1.3	Creating Tables	265
9.1.4	Creating Foreign Key Relationships	267
9.1.5	Creating Views	270
9.1.6	Creating Routines and Routine Groups	272
9.1.7	Creating Layers	276
9.1.8	Creating Notes	278
9.1.9	Creating Text Objects	279
9.1.10	Creating Images	280
9.2	Additional Modeling Tools	281
9.2.1	Printing Diagrams	281

9.2.2 DBDoc Model Reporting	281
9.2.3 Schema Validation Plugins	285
9.3 Modeling Tutorials	287
9.3.1 Creating a Model	287
9.3.2 Basic Modeling	294
9.3.3 Importing a Data Definition SQL Script	296
9.3.4 Using the Default Schema	298
9.3.5 Documenting the sakila Database	301
9.4 Forward and Reverse Engineering	303
9.4.1 Forward Engineering	303
9.4.2 Reverse Engineering	313
9.5 Schema Synchronization and Comparison	323
9.5.1 Database Synchronization	323
9.5.2 Compare and Report Differences in Catalogs	329
9.6 Table Templates	331
9.7 Customizing DBDoc Model Reporting Templates	334
9.7.1 Supported Template Markers	337
9.7.2 Creating a Custom Template	341
10 Database Migration Wizard	345
10.1 General Installation Requirements	346
10.1.1 ODBC Libraries	346
10.1.2 ODBC Drivers	347
10.2 Migration Overview	348
10.2.1 A Visual Guide to Performing a Database Migration	348
10.2.2 Migrating from Supported Databases	366
10.2.3 Migrating from Unsupported (Generic) Databases	367
10.3 Conceptual DBMS Equivalents	367
10.4 Microsoft Access Migration	369
10.5 Microsoft SQL Server Migration	386
10.5.1 Preparations	386
10.5.2 Drivers	386
10.5.3 Connection Setup	390
10.5.4 Microsoft SQL Server Type Mapping	393
10.6 PostgreSQL migration	394
10.6.1 Preparations	394
10.6.2 Drivers	395
10.6.3 Connection Setup	396
10.6.4 PostgreSQL Type Mapping	397
10.7 MySQL Migration	399
10.8 Using the MySQL Workbench Migration Wizard	403
10.8.1 Connecting to the Databases	403
10.8.2 Schema Retrieval and Selection	405
10.8.3 Reverse Engineering	407
10.8.4 Object Selection	408
10.8.5 Migration	409
10.8.6 Manual Editing	410
10.8.7 Target Creation Options	413
10.8.8 Schema Creation	414
10.8.9 Create Target Results	415
10.8.10 Data Transfer and Migration Setup	416
10.8.11 Bulk Data Transfer	417
10.8.12 Migration Report	418
10.9 MySQL Workbench Migration Wizard FAQ	419
A MySQL Workbench Frequently Asked Questions	421

B Keyboard Shortcuts	427
C Extending Workbench	431
C.1 GRT and Workbench Data Organization	431
C.2 Modules	432
C.3 Plugins and Tools	434
C.4 Adding a GUI to a Plugin Using MForms	435
C.5 The Workbench Scripting Shell	435
C.5.1 Exploring the Workbench Scripting Shell	436
C.5.2 The Shell Window	436
C.5.3 Files, Globals, Classes, Modules, and Notifications Tabs	438
C.6 Tutorial: Writing Plugins	443
C.6.1 Tutorial: Generate PHP Code to Create a Connection with PDO_MySQL	443
C.6.2 Tutorial: Generating Foreign Keys with MyISAM	445
D How To Report Bugs or Problems	447
E MySQL Enterprise Features	451
F MySQL Utilities	453

Preface and Legal Notices

This is the user manual for MySQL Workbench.

Licensing information. This product may include third-party software, used under license. If you are using a *Commercial* release of MySQL Workbench, see the [MySQL Workbench Commercial License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Commercial release. If you are using a *Community* release of MySQL Workbench, see the [MySQL Workbench Community License Information User Manual](#) for licensing information, including licensing information relating to third-party software that may be included in this Community release.

Legal Notices

Copyright © 2006, 2024, Oracle and/or its affiliates.

License Restrictions

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other

measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Third-Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Use of This Documentation

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Chapter 1 General Information

Table of Contents

1.1 What Is New in MySQL Workbench	1
1.1.1 New in MySQL Workbench 8.0 Release Series	1
1.1.2 New in MySQL Workbench 6.0 Release Series	6
1.2 MySQL Workbench Editions	53

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 8.0. Deprecated versions of MySQL Server may be incompatible with MySQL Workbench and should be upgraded before you attempt to make a connection. Versions after 8.0, such as 8.4, may connect but some features may not be supported.

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, PostgreSQL, 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

For notes detailing the changes in each point release, see the [MySQL Workbench Release Notes](#).

1.1.1 New in MySQL Workbench 8.0 Release Series

This section summarizes how the MySQL Workbench 8.0 release series progressed with each minor release. For the list of supported platforms, see <https://www.mysql.com/support/supportedplatforms/workbench.html>.

- [MySQL](#)

- [Character Set Changes](#)
- [Home Tab Changes](#)
- [MySQL Workbench Editors: Query, Object, and More](#)
- [SQL Export Options](#)
- [MySQL Model Changes](#)
- [MySQL Enterprise Backup \(MEB\)](#)
- [SET PERSIST and SET PERSIST ONLY Functionality](#)
- [Platform and Source Code Changes](#)
- [Security Changes](#)
- [Generic RunTime \(GRT\) Module Changes](#)

MySQL

- The following MySQL 8.0.19 and 8.0.20 server language features are supported in MySQL Workbench 8.0.21:
 - `DROP CHECK`, `ALTER CONSTRAINT`, and `DROP CONSTRAINT` in `ALTER TABLE` statements.
 - Locking clause for query expressions.
 - Table values constructor.
 - Short table select syntax.
 - New requirements for the `CHANGE MASTER TO` replication statement (row format, primary key check).
 - Failed login attempts and password lock time in `ALTER USER` statements.
 - Format support for `EXPLAIN ANALYZE` statements.
 - New and changed keywords:
 - `ARRAY`
 - `FAILED_LOGIN_ATTEMPTS`
 - `MASTER_COMPRESSION_ALGORITHM`
 - `MASTER_TLS_CIPHERSUITES`
 - `MASTER_ZSTD_COMPRESSION_LEVEL`
 - `MEMBER`
 - `OFF`
 - `PASSWORD_LOCK_TIME`
 - `PRIVILEGE_CHECKS_USER`
 - `RANDOM`

- [REQUIRE_ROW_FORMAT](#)
 - [REQUIRE_TABLE_PRIMARY_KEY_CHECK](#)
 - [STREAM](#)
 - [TIMESTAMP](#)
 - [TIME](#)
- The following MySQL server language features are supported in MySQL Workbench 8.0.19:
 - Value references in [INSERT](#) statements.
 - New options (TLS version, cipher suite, compression, and privilege check) in [CHANGE MASTER TO](#) replication statements.
 - Random passwords in [CREATE USER](#) and [ALTER USER](#) statements.
 - [EXPLAIN ANALYZE](#) (see [Obtaining Information with EXPLAIN ANALYZE](#)).
 - [binary](#) collation names.
 - Beginning with MySQL Workbench 8.0.17, the ANTLR4 grammar was updated to handle the language features of each supported MySQL server version from MySQL 5.6 to MySQL 8.0, including the following new language features in MySQL Server 8.0.17:
 - [CHECK](#) constraints with enforcement.
 - ODBC table references now require the [OJ](#) keyword (previously an identifier was allowed).
 - The equal sign (=) operator for aliases is no longer allowed.
 - [CHANGE MASTER](#) supports [NETWORK_NAMESPACE](#) and channels.
 - The [CREATE DATABASE](#) statement permits a default encryption to be specified.
 - Roles now support exception lists and [GRANT AS](#). Grant identifiers can be qualified with a schema name.
 - [MEMBER OF](#) is allowed in expressions.
 - [AS ARRAY](#) is allowed in [CAST](#) expressions.
 - Passwords in user statements can now also be hexadecimal numbers.
 - Hexadecimal numbers are also allowed in many numeric expressions.
 - Minor items, such as, permit data types in number expressions, and reorganize keywords to lower conflicts in the server parser generation.
 - Removed support for MySQL 5.5 in the MySQL Workbench 8.0 release series. Minimum version now is MySQL 5.6.

If you still need to use MySQL Workbench on a MySQL 5.5 server, you can use MySQL Workbench 6.3, which is available from [MySQL Product Archives](#).

Character Set Changes

- MySQL Workbench now uses `utf8mb4` as the connection and client character set, replacing `utf8mb3`.
- Support for the Chinese character set `gb18030` was added.

Home Tab Changes

- As of MySQL Workbench 8.0.14, keyboard access was added to the home screen tab to enable navigation using **Tab** and **Enter** keys. In addition, the screen view now scrolls to display a selected item if the item was off-screen when highlighted with the **Tab** key.

On Windows and Linux hosts, the Application key and **Ctrl+F10** now open a menu of commands (context menu) related to the selection.

MySQL Workbench Editors: Query, Object, and More

- Starting with MySQL Workbench 8.0.22, visual explain diagrams use the phrase `hash join` within the diamond symbol instead of `block nested loop` when the server version is 8.0.20 or higher.
- For consistency with other MySQL products, RapidJSON replaces the native JSON parser in the MySQL Workbench 8.0.18 release.
- **Important change:** MySQL model files last saved before MySQL Workbench 6.3 are no longer supported unless the models can be upgraded for use with the 6.3 release series.
- Beginning with MySQL Workbench 8.0.16, the script editor highlights matching pairs of parentheses when one of the pair is selected.
- SQL context help was enhanced to eliminate the requirement of having a valid MySQL connection to view the help topics and to improve the presentation of each help topic.
- A new auto-completion engine was added for use with object editors (triggers, views, stored procedures, and functions) in the visual SQL editor and in models.
- Geometry fields displayed in the result grid now include a context-menu item that opens the specific location value in a browser. The selected point opens in openstreetmap.org by default, but an alternative online service can be used (see [Section 3.2.7, “Other Preferences”](#)).
- Support for invisible indexes was added for use when the active server supports the feature and the index is neither a primary key index nor a unique column (see [Invisible Indexes](#)). A new option in the **Indexes** subtab of the table editor (for both the SQL and modeling editors) provides index visibility when it is selected.

SQL Export Options

- A new SQL export option in the Forward Engineering SQL Script wizard sorts tables alphabetically in the generated script, rather than sorting tables according to foreign-key references by default (see [Creating a Schema](#)).
- The `OmitSchemas` option replaces both the `UseShortNames` and `OmitSchemata` options to eliminate the schema name from table names when using the Python API to generate a schema from an `.mwb` file automatically.

MySQL Model Changes

- Output from schema validation plugins for MySQL models now is shown in a single location and reorganized to provide informational, warning, and error messages by category. A new **Validate** tab also

provides a simple way to reselect and rerun validation tests from the output area in the right side panel (see [Section 9.2.3, “Schema Validation Plugins”](#)).

MySQL Enterprise Backup (MEB)

- Support for the `--incremental-with-redo-log-only` option was added to create backups directly from the redo log (see [Options Tab](#)).

SET PERSIST and SET PERSIST ONLY Functionality

- MySQL Workbench now provides a simple way to enable or disable the persisted global system variable settings introduced in MySQL 8.0 (see [SET Syntax for Variable Assignment](#)). For variables that can be persisted, a new check box enables configuration changes at runtime that also persist across server restarts and applies the persisted value, if one exists. Persistent system variables can be reset (to not persist) individually or collectively. For additional information, see [Persist System Variables](#).

Platform and Source Code Changes

- With Python 2 reaching end-of-life, MySQL Workbench 8.0.23 is the first release to use Python 3 for scripting-related features, such as:
 - MySQL Workbench Migration Wizard
 - Workbench Scripting Shell
 - Administration: MySQL Enterprise Firewall, MySQL Enterprise Audit, MySQL Enterprise Audit, performance, startup/shutdown, server logs, options file, server status, client connections, users and privileges, status and system variables, data export, and data import
 - SQL IDE: power import/export, reformatter, run script, import spatial, text output, query analysis, and visual explain
- The `%cmake_build` macro replaces `%cmake` for running the `make` command to build MySQL Workbench from source code on Fedora 33 (and later) using the RPM package. For additional information about the change, see the [Fedora upstream documentation](#).
- The requirement to install the EPEL repository on enterprise Linux systems, such as Oracle Linux and Red Hat, is removed for general use with the MySQL Workbench 8.0.18 release. Working with spatial data is an exception and you can still install the repository if needed (see [Installing Oracle Enterprise Linux and Similar](#)).
- MySQL Workbench 8.0.18 switched to the C++17 programming language.
- MySQL Workbench source code has been reformatted according to Google style.
- MySQL Workbench 8.0.28 supports Apple macOS Monterey 12.
- Ubuntu 20.04 LTS is supported.
- Support for Microsoft Windows 11 and Microsoft Windows Server 2022 were added in the MySQL Workbench 8.0.28 release.
- Support for Microsoft Visual Studio 2019 was added in the MySQL Workbench 8.0.19 release. Microsoft Visual Studio support was upgraded from Visual Studio 15 to Visual Studio 17 in the MySQL Workbench 8.0.16 release.

Security Changes

- In MySQL Workbench 8.0.27, the following new connection methods for use with LDAP pluggable authentication and Kerberos pluggable authentication are supported:
 - The `LDAP User/Password` connection method for simple LDAP authorization on Linux and Windows hosts.
 - The `LDAP Sasl/Kerberos` connection method, which uses GSSAPI/Kerberos to authenticate users and passwords on Linux hosts.
 - The `Native Kerberos` connection method, using authentication tokens generated by the `kinit` command, on Linux and Windows hosts.

The new authentication-based connection methods are not supported on macOS hosts. For requirement and setup information, see [Section 5.3.4, “LDAP and Kerberos Connection Methods”](#).

- `libgnome-keyring` was deprecated and replaced with `libsecret` in the MySQL Workbench 8.0.12 release on Linux platforms. The `libsecret` library provides enhanced cross-platform password storage and lookup.



Important

Some users with existing stored passwords will be prompted to enter a password after upgrading.

- SSH tunneling support was added to the MySQL Workbench Migration Wizard and also to the `wbcopytables` command-line utility for copying data.
- Setting an encryption password is required to perform MySQL Enterprise Backup operations on encrypted tables (see [Options Tab](#)).
- The SSH implementation based on Paramiko was replaced with the one based on libssh.
- MySQL Workbench now supports the `caching_sha2_password` authentication plugin introduced in MySQL 8.0 (see [Caching SHA-2 Pluggable Authentication](#)).

Generic RunTime (GRT) Module Changes

As of MySQL Workbench 8.0.14, the following new functions are included in the Workbench GRT module:

- `activateDiagram(<Diagram>)`
Opens the selected EER diagram for use with the `exportPNG`, `exportSVG`, `exportPS`, and `exportPDF` functions.
- `exportDiagramToPng(<Diagram>, <path>)`
Performs a PNG export of an EER diagram to the path provided without activating it.

1.1.2 New in MySQL Workbench 6.0 Release Series

This section summarizes how the MySQL Workbench 6.0 release series progressed with each minor release.

1.1.2.1 New in MySQL Workbench 6.3

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

- [MySQL](#)
- [User Interface Changes](#)
- [User Preference Changes](#)
- [Package and Build Related Changes](#)
- [Fast Data Migration](#)
- [SSL Certificate Generator](#)
- [SQL Editor Auto-Completion](#)
- [MySQL Enterprise Firewall](#)
- [MySQL Enterprise Backup](#)
- [Table Data Export and Import Wizard](#)
- [Additional Changes](#)

MySQL

- Full MySQL 5.7 language support was added, which affects grammar, syntax highlighting, preferences, behavior, and more.
- The bundled `sakila_full` model now uses a dedicated 5.7 version to allow for 5.7 specific features, regardless of the version setting in the preferences.
- The JSON editor was improved with better parsing and error checking.
- The option to specify an alternative application data directory, instead of the default location, was added.
- Dropped support for MySQL 5.1. Minimum version is now MySQL 5.5.
- Dropped Fabric support in MySQL Workbench 6.3.9; support in older versions of MySQL Workbench is unchanged.
- Dropped support for DBDesigner 4.

User Interface Changes

- The home screen was modified: the connections, models, and starters were split into individual pages.

User Preference Changes

- A new **Log Level** preference (under **Others**) was added to alter the log verbosity level.

As before, this can still be set by passing in the `log-level` command-line argument into Workbench at runtime, and doing so overrides the **Log Level** setting.

Package and Build Related Changes

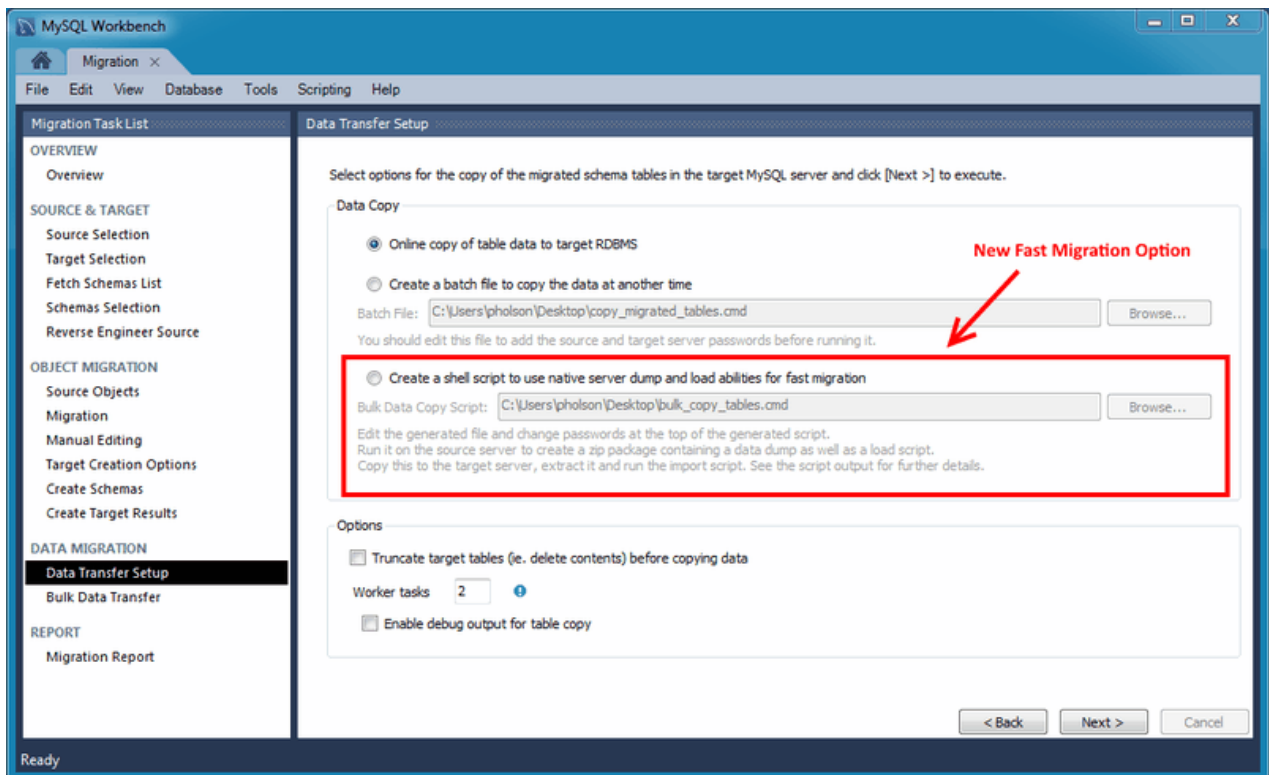
- Linux: Fedora 23 support dropped, Fedora 25 support added. Oracle Linux 6 support was dropped.
- Windows: Zip packages and 32-bit binaries are no longer published. The .NET Framework version 4.5 is now required.

- OS X / macOS: Version 10.7 (Lion) and 10.8 (Mountain Lion) support was dropped. Now supporting versions 10.9 (Mavericks), 10.10 (Yosemite), 10.11 (El Capitan), and 10.12 (Sierra).
- Changed to GTK 3 on Linux.
- Changed to C++11.

Fast Data Migration

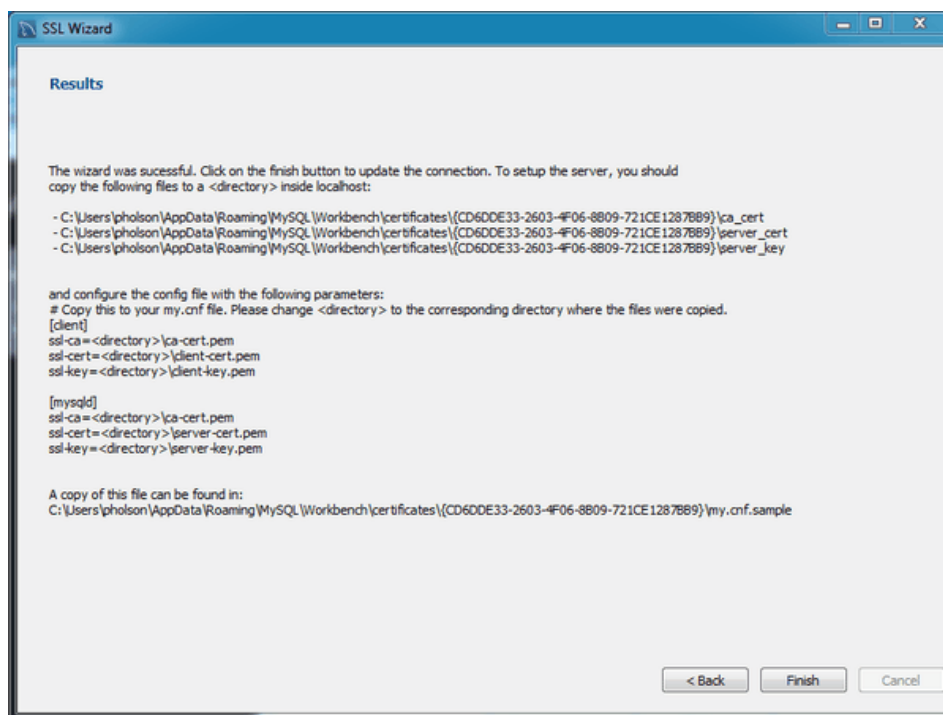
A new "fast migration" option was added to the migration wizard. This is another way to transfer data from one MySQL server to another while performing a migration, and it complements the existing solutions. The premise is to use a generated script on the source server to create a dump that you move 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



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

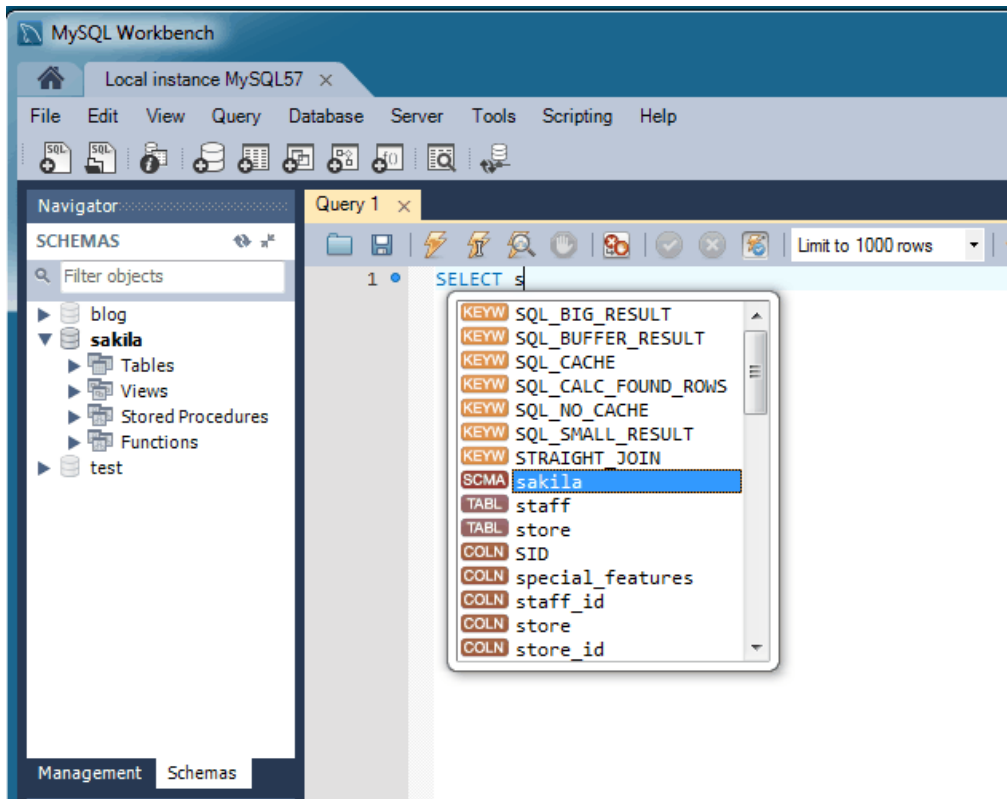
For additional details, see [Section 5.3.5, “SSL Wizard \(Certificates\)”](#).

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



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

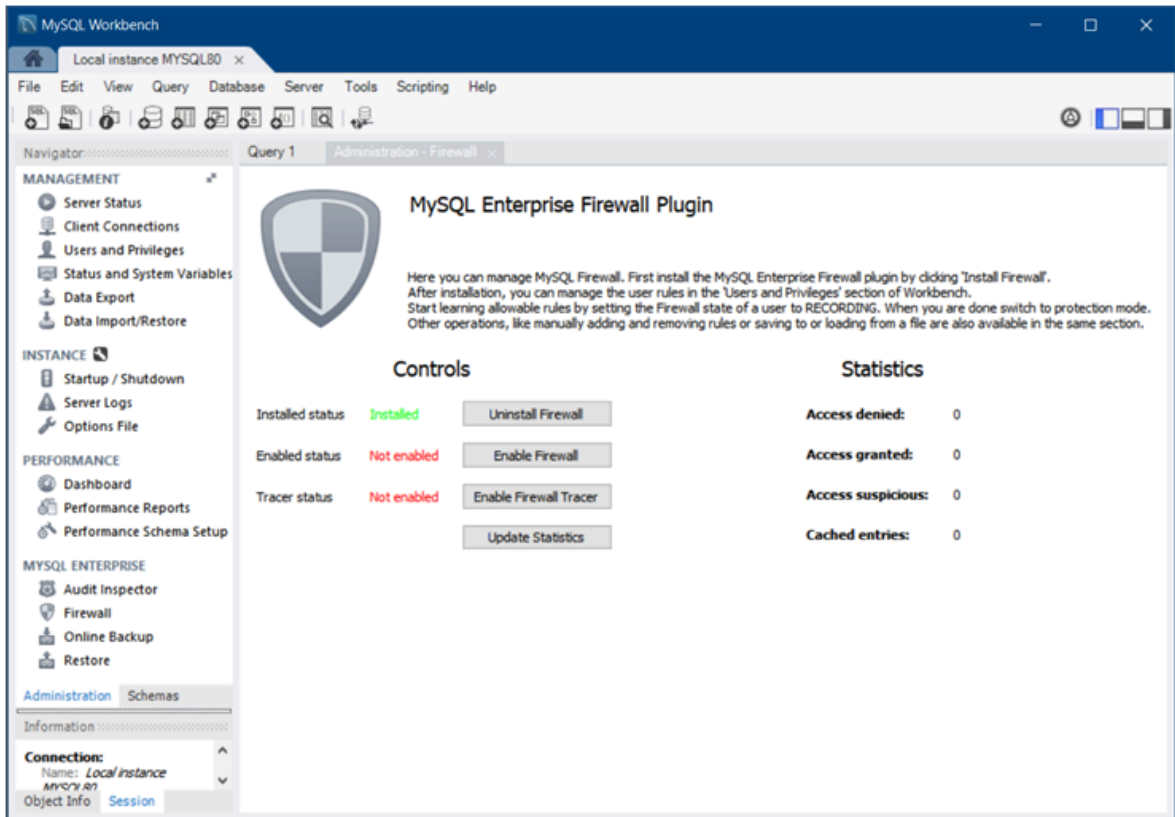
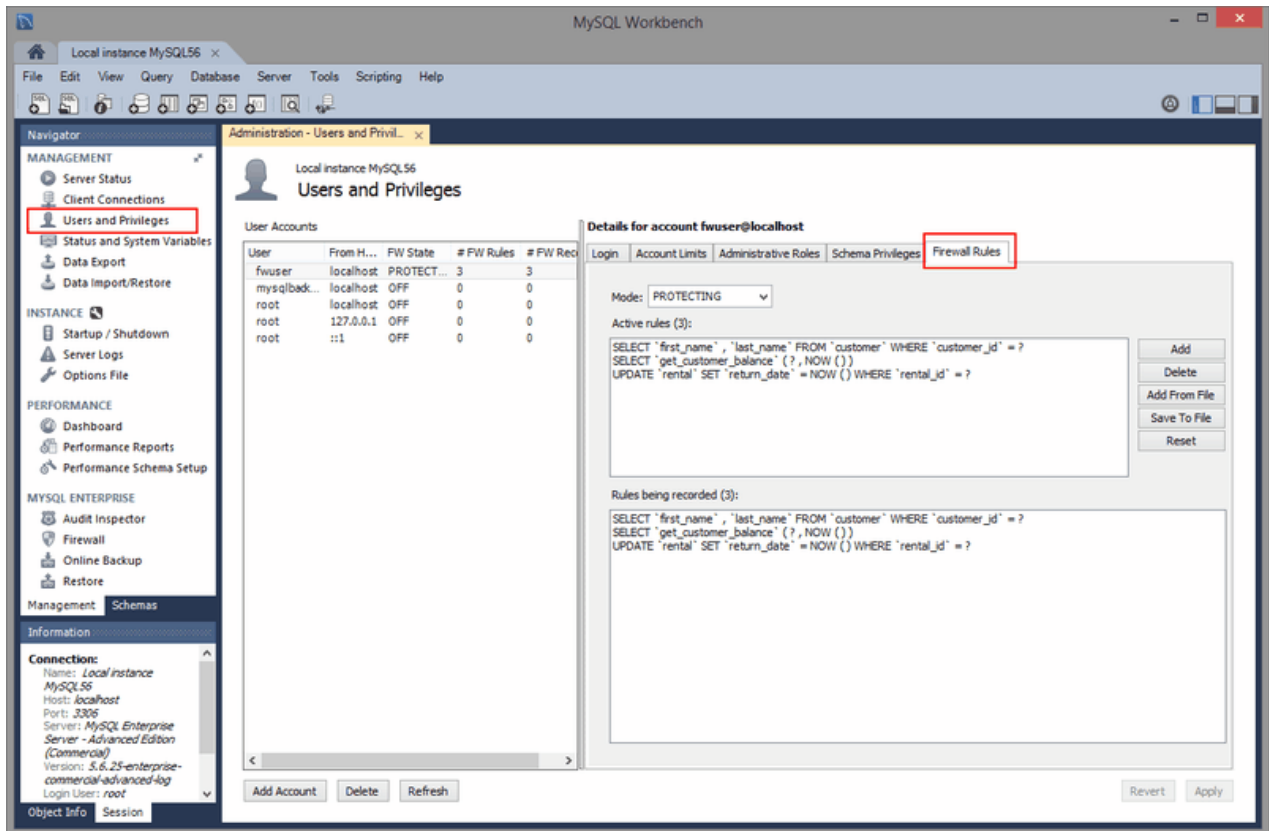


Figure 1.5 MySQL Enterprise Firewall Rules



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

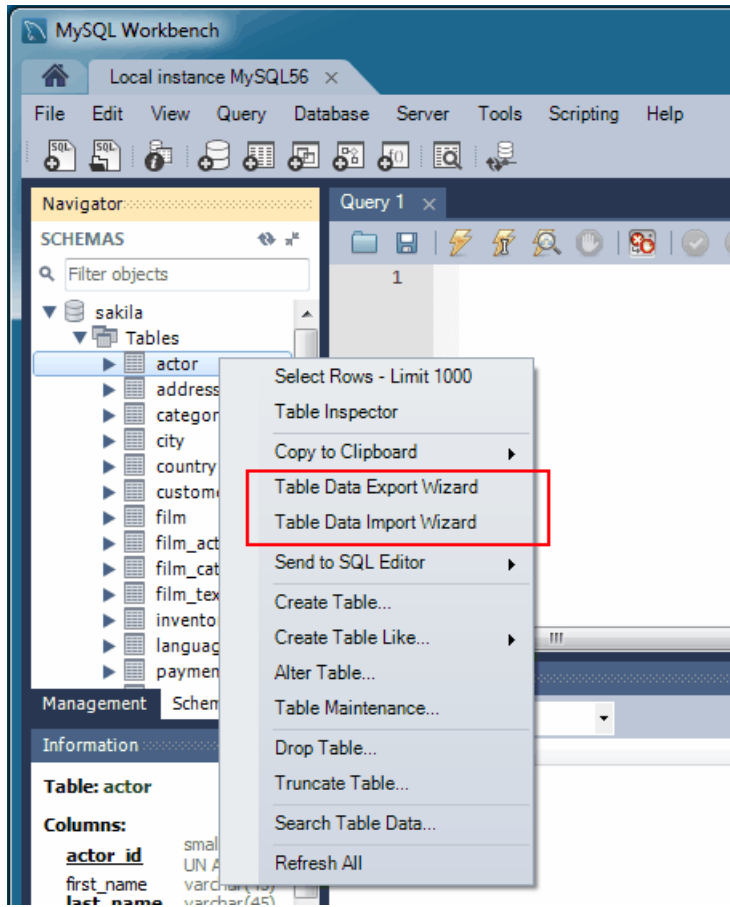


Figure 1.7 Table Data Import/Export Wizard CSV Configure

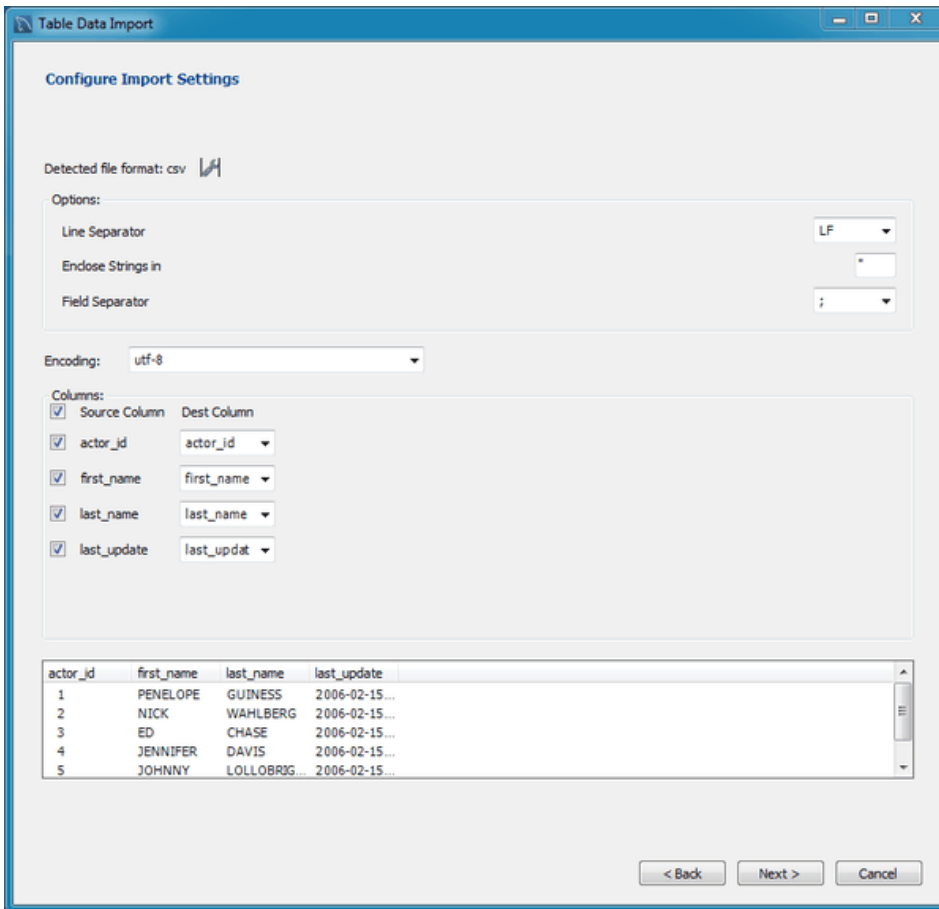
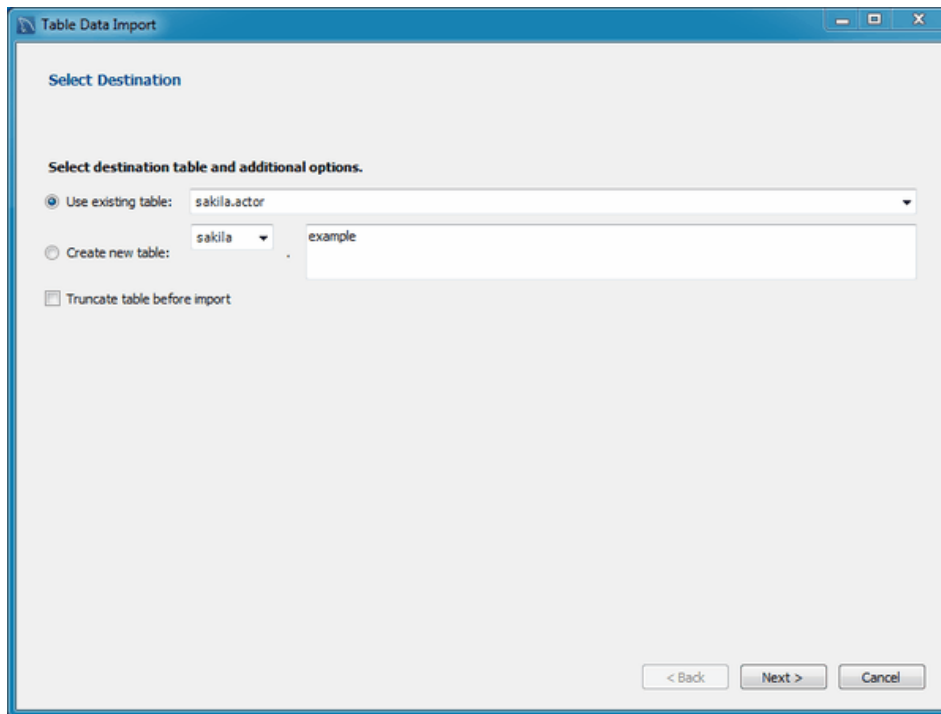


Figure 1.8 Table Data Import/Export Wizard Options

Additional Changes

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

OS X / macOS 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.

1.1.2.2 New in MySQL Workbench 6.2

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

- [SQL Editor](#)
- [Overlay Icons in the Object Viewer](#)
- [A "Pin Tab" Results Option](#)
- [Microsoft Access to MySQL Migration](#)
- [Visual Explain and Execution Plan Improvements](#)
- [Spatial View Panel](#)
- [Geometry Data Viewer](#)
- [Additional New SQL Editor Features](#)
- [Execute SQL Scripts](#)

- [Model Script Attachments](#)
- [Client Connections and Metadata Locks](#)
- [Additional New Features](#)

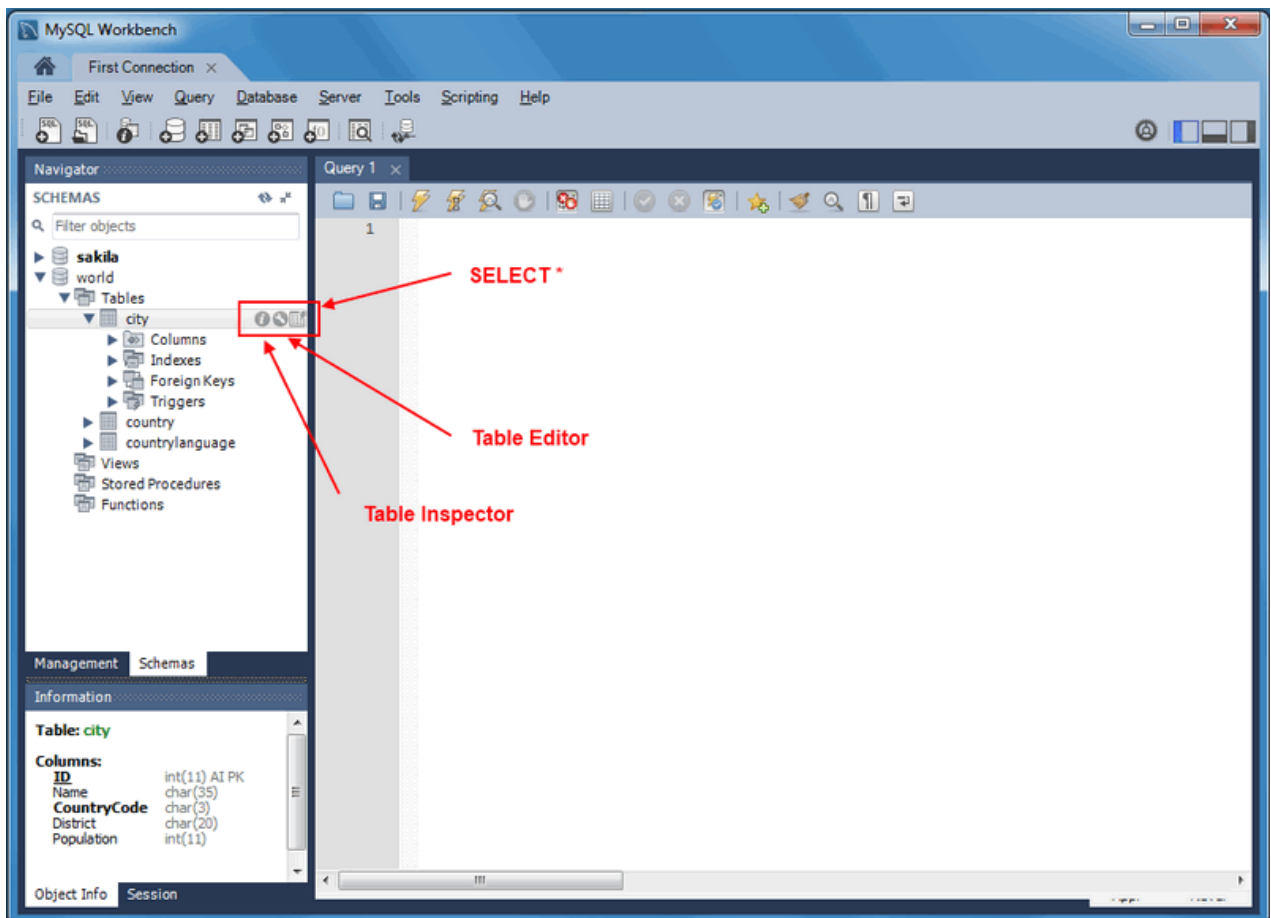
SQL Editor

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



A "Pin Tab" Results Option

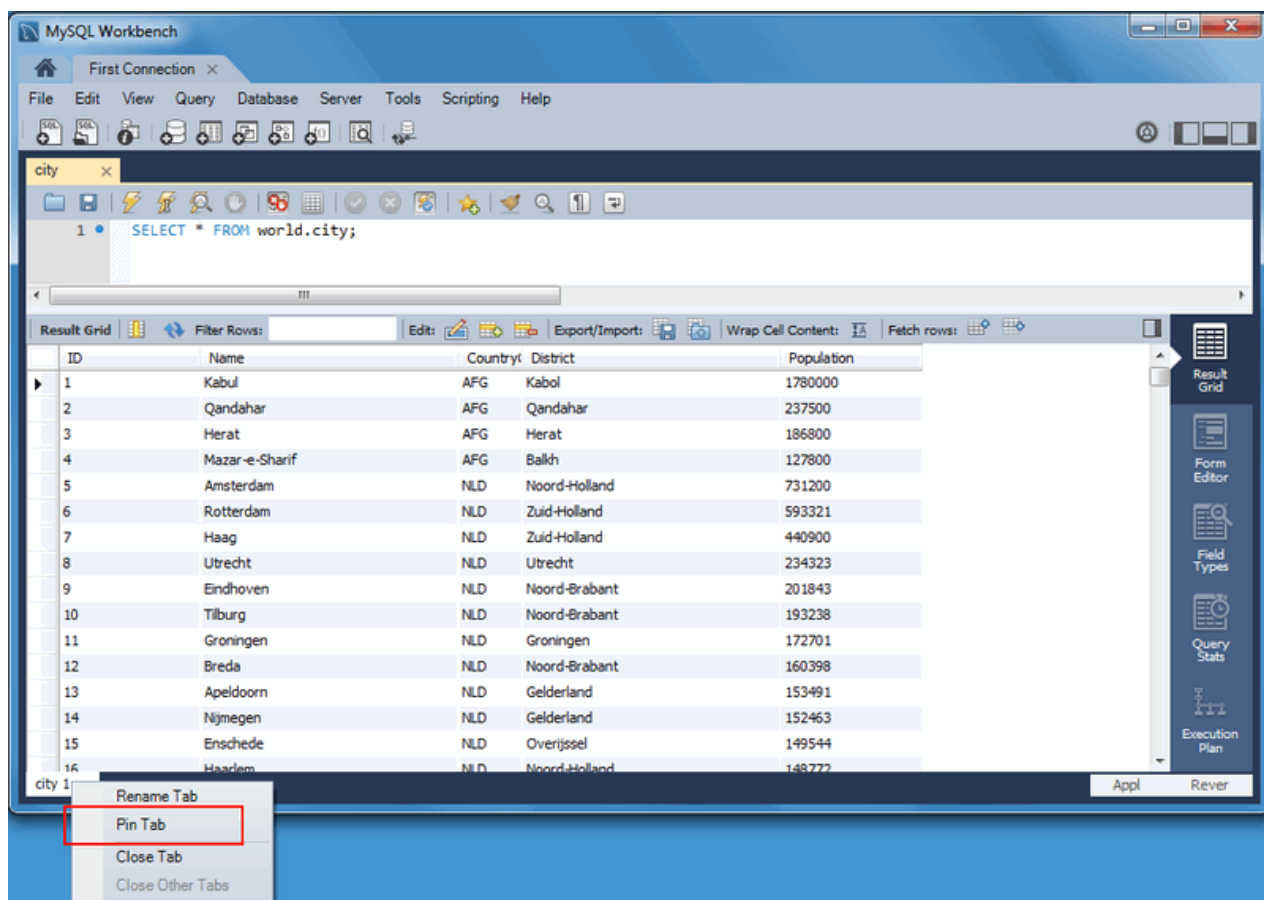
Result tabs can now be "pinned" to your result set window.



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



Microsoft Access to MySQL Migration

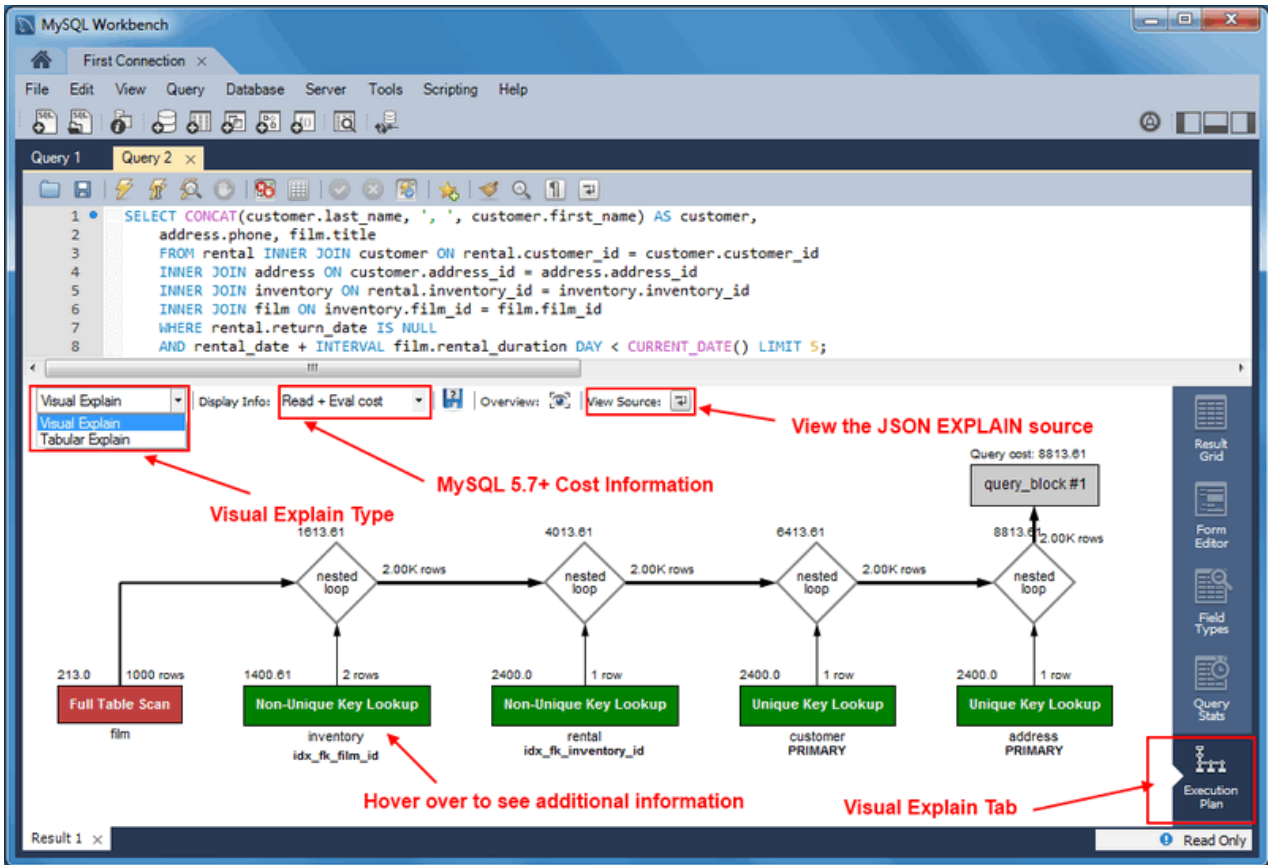
The migration wizard now supports Microsoft Access migration. Select "Microsoft Access" as your source database in the wizard, use MySQL as your target source database, and then execute. For additional information, see [Section 10.4, "Microsoft Access Migration"](#).

Visual Explain and 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 1.11 Execution Plan Explained



Spatial View Panel

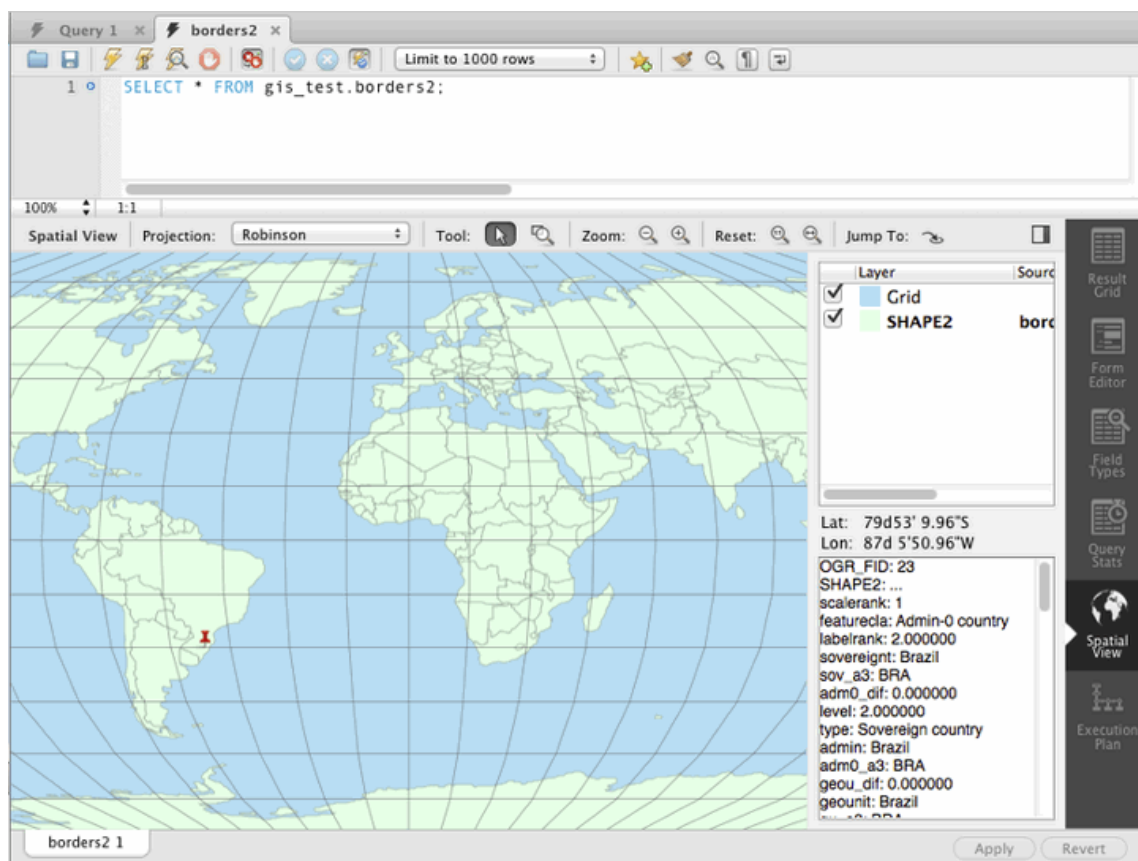
GIS support for InnoDB tables is now supported to make it easier to visualize spatial and geometry data in a geographic context. The new spatial view panel renders data from each row into a separate and selectable element. When clicked, you can view the rest of the data from that row in the text box. 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.

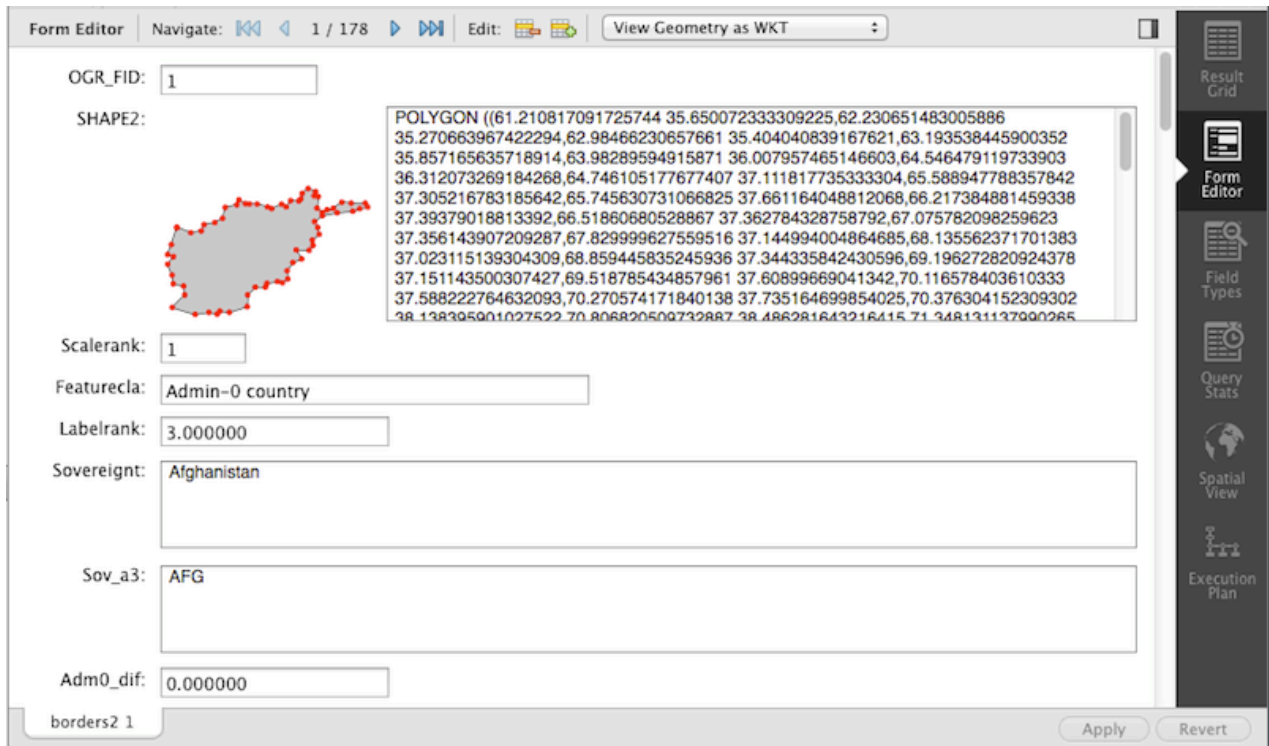
Figure 1.12 Spatial View Example



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 1.13 Geometry Data Viewer



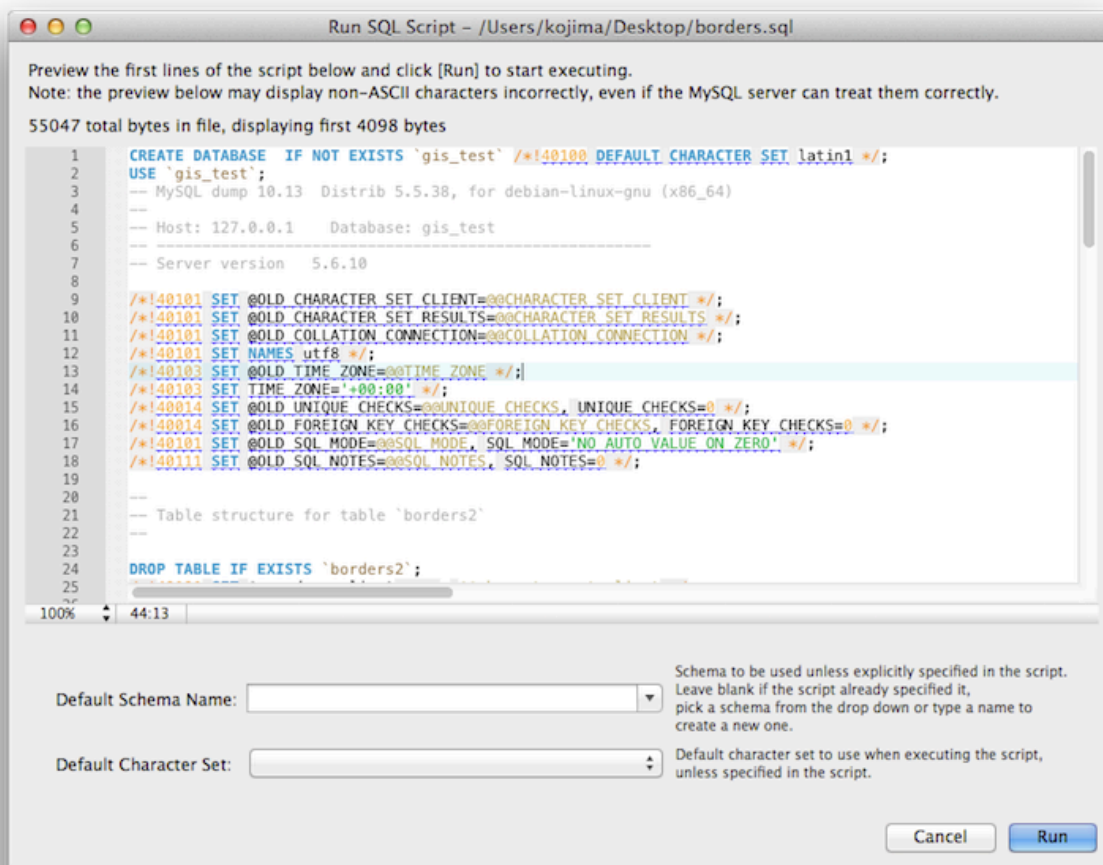
Additional New SQL Editor Features

- **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 Additions - 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.

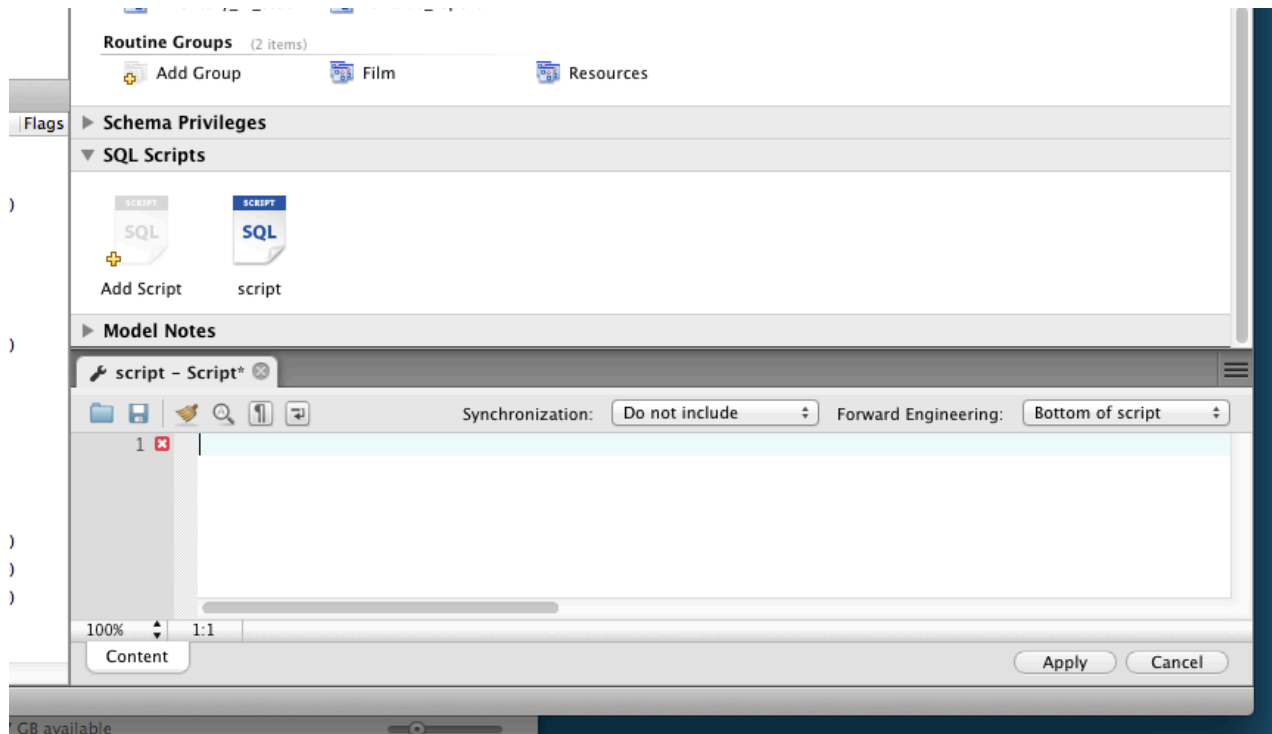
Figure 1.14 Run SQL Script



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 1.15 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 1.16 Metadata Locks Browser

The screenshot shows the MySQL Workbench Client Connections window for a local instance 3306. The window title is "Administration - Client Connections". It displays various connection statistics and a table of active connections. One connection (Thread Id 28) is highlighted as "Waiting for table metadata lock". The right-hand pane shows the "Locks" tab, which provides details about the lock. It indicates that the connection is waiting for an EXCLUSIVE lock on the sakila.actor table, which is currently held by threads 30 and 28. The lock type is EXCLUSIVE and the duration is TRANSACTION.

Client Connections
 Local instance 3306

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...	

Locks

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 Hide Details

Refresh Rate: Don't Refresh Kill Query(s) Kill Connection(s) Refresh

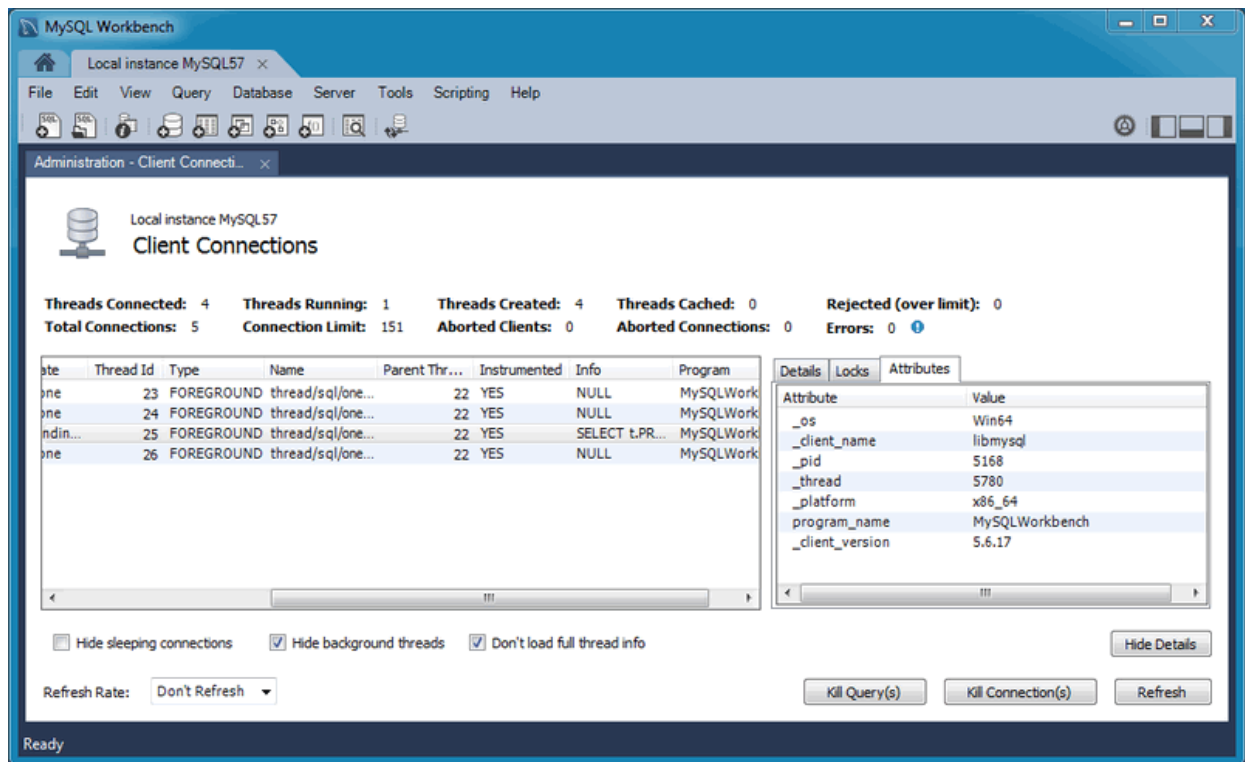


Note

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 1.17 Client Connection Attributes



Note

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

For additional information, see [Section 5.4, “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 1.18 Model Diagram Note Formatting



1.1.2.3 New in MySQL Workbench 6.1

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

- [New Navigator PERFORMANCE Section](#)
- [Server Variable Groupings](#)
- [SQL Editor Views](#)
- [Home Screen Features](#)
- [Visual Explain](#)
- [Table Inspector](#)
- [Additional Client-Connection Information](#)
- [Miscellaneous Additions](#)

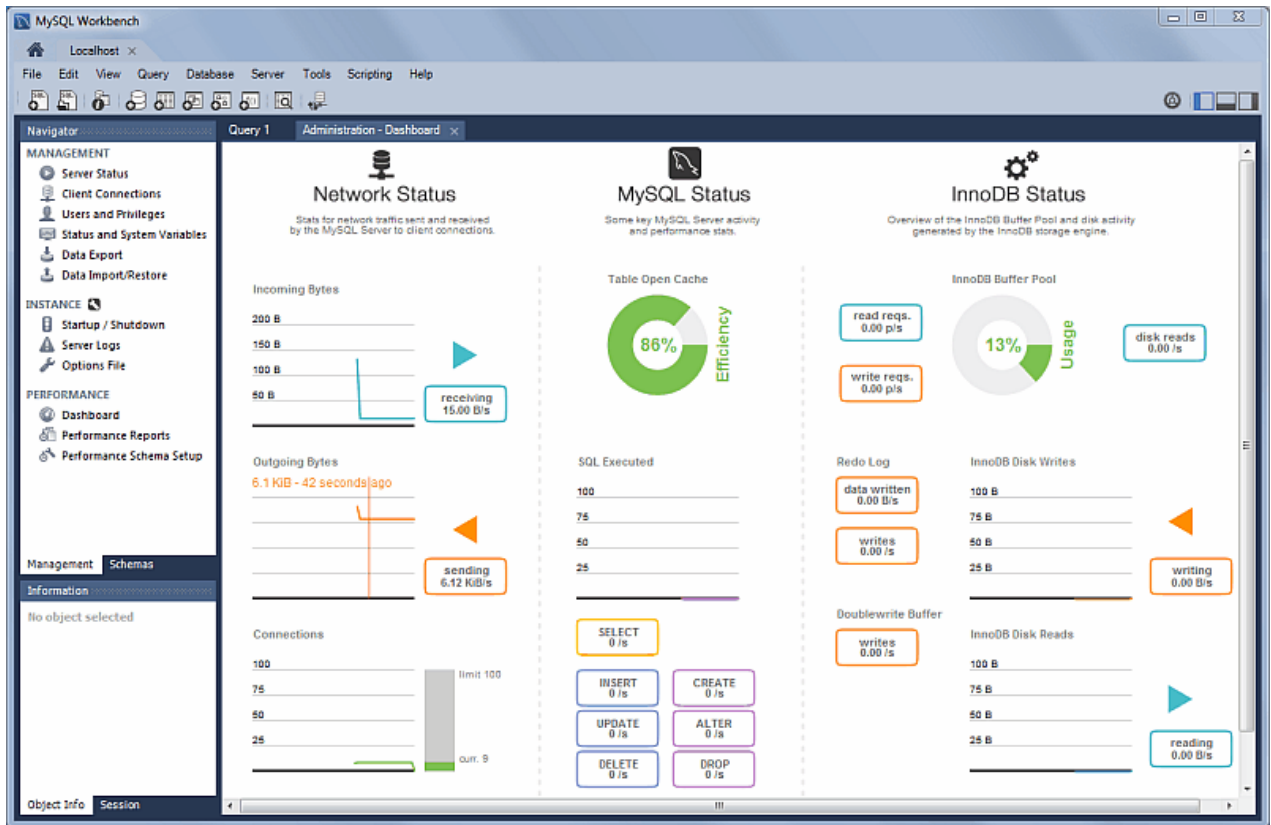
New Navigator PERFORMANCE Section

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.

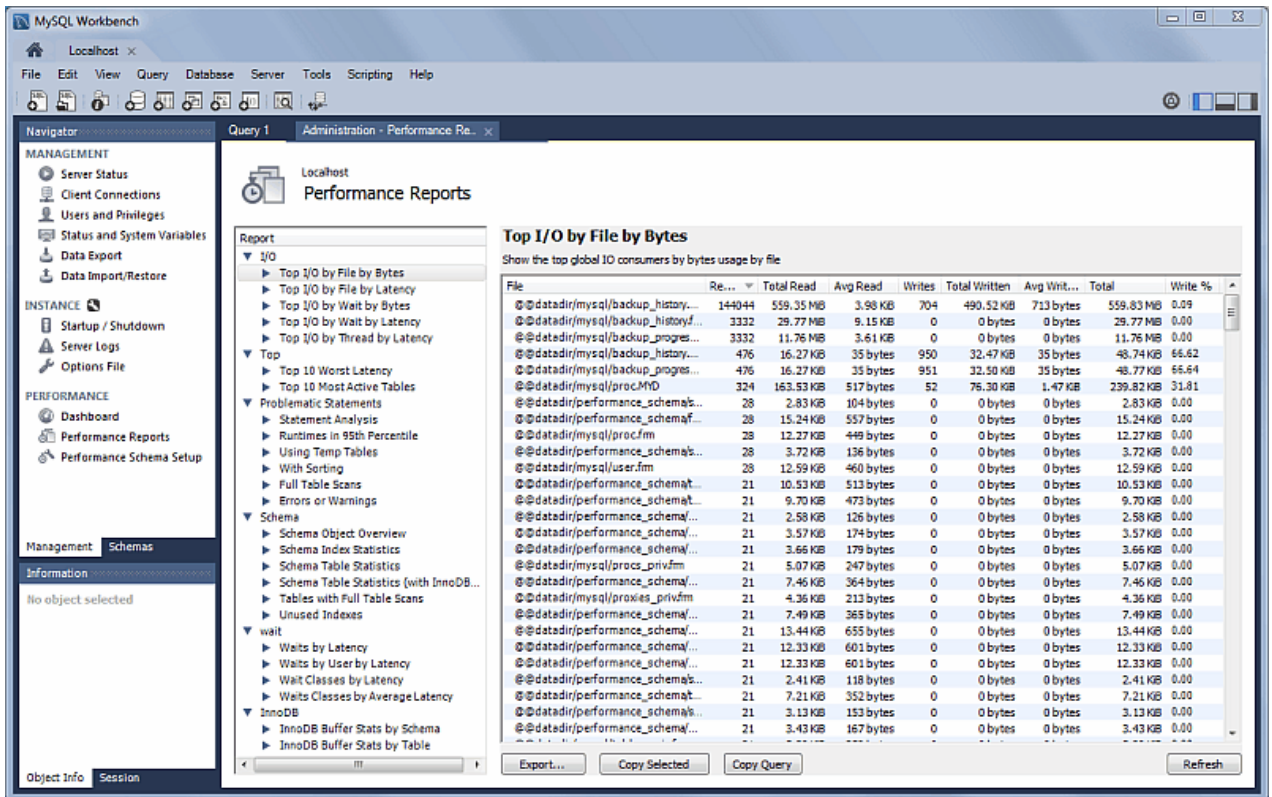
Figure 1.19 Performance Dashboard



Performance Reports

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

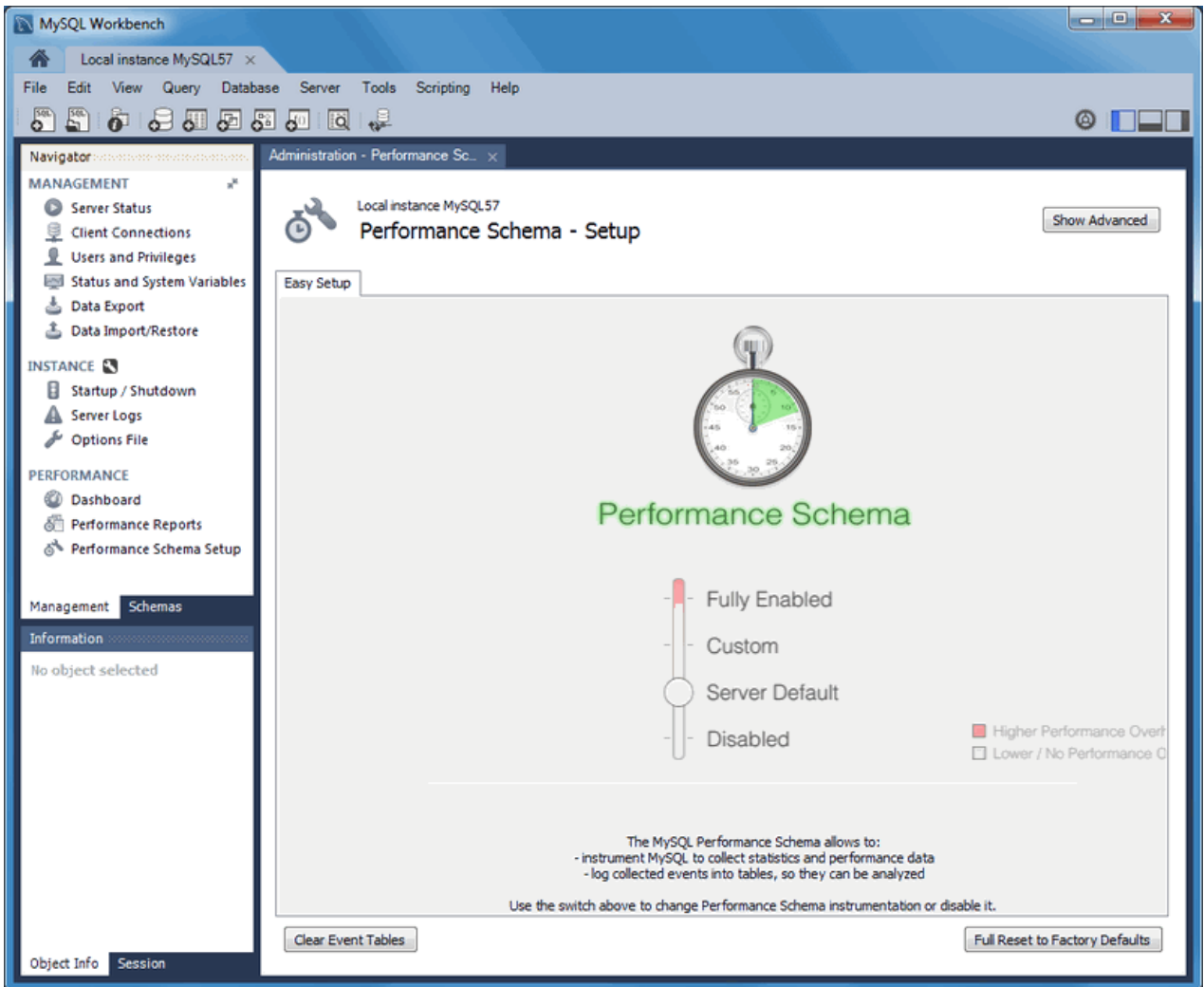
Figure 1.20 Performance Reports: Top I/O By Bytes



Performance Schema Setup

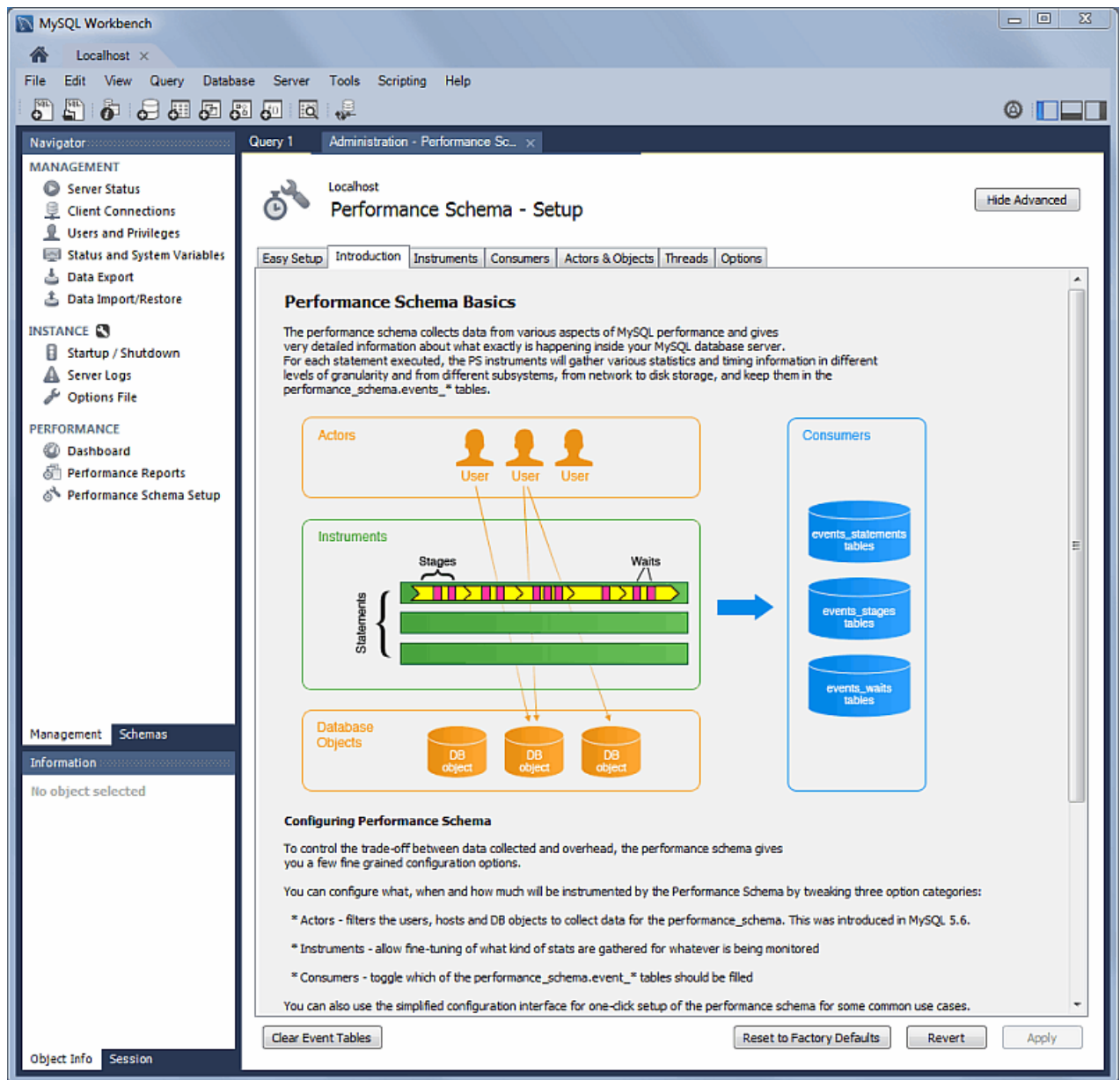
A GUI for configuring and fine tuning the Performance Schema instrumentation. Initially, this loads an "Easy Setup" page that is enough for most users. Slide the "Performance Schema Full Enabled" slider to **YES** to enable all available Performance Schema instruments.

Figure 1.21 Performance Schema Setup: Easy Setup



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

Figure 1.22 Performance Schema Setup: Introduction



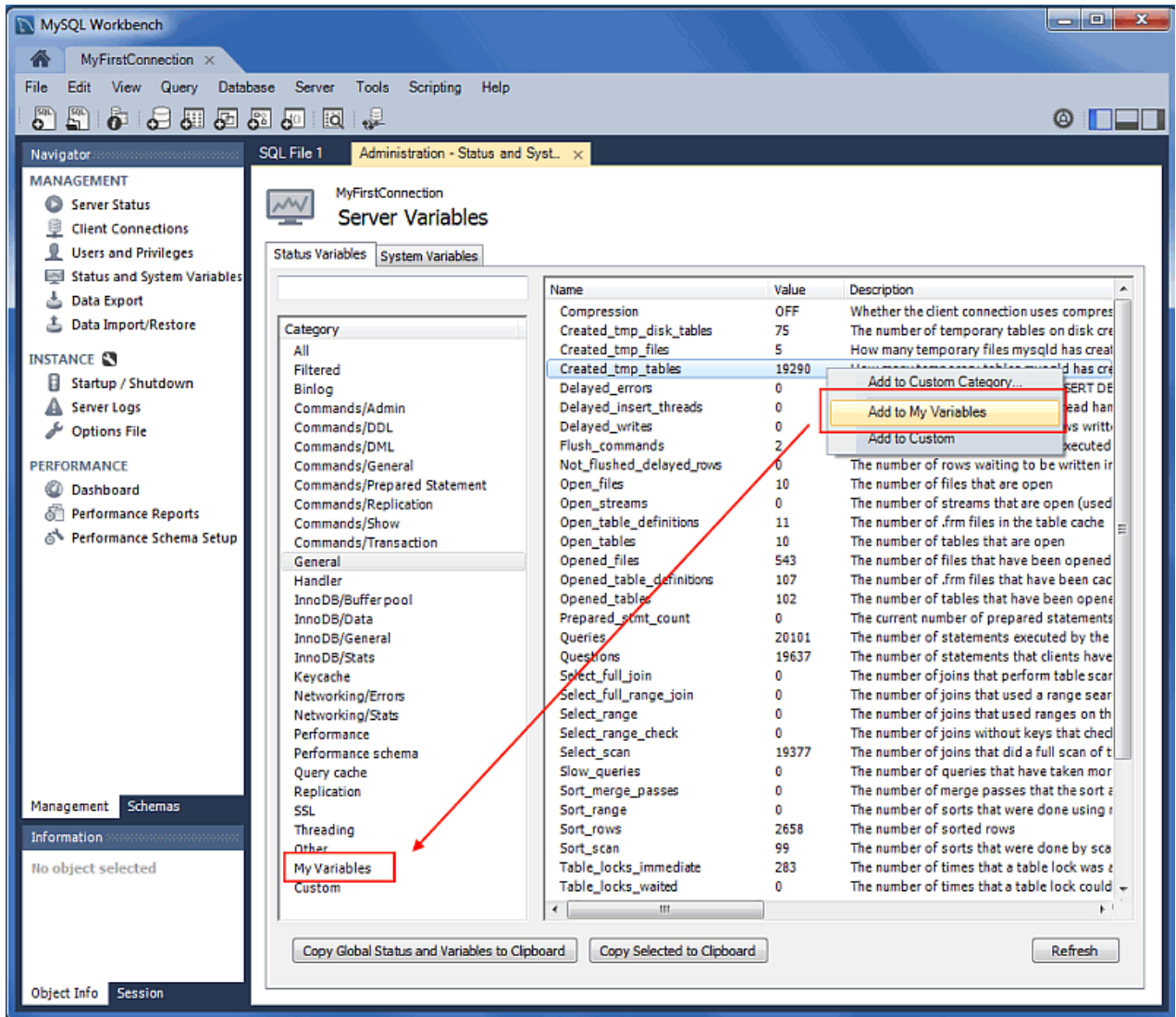
For additional information, see [Chapter 7, Performance Tools](#).

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 1.23 Status And System Variables: Custom



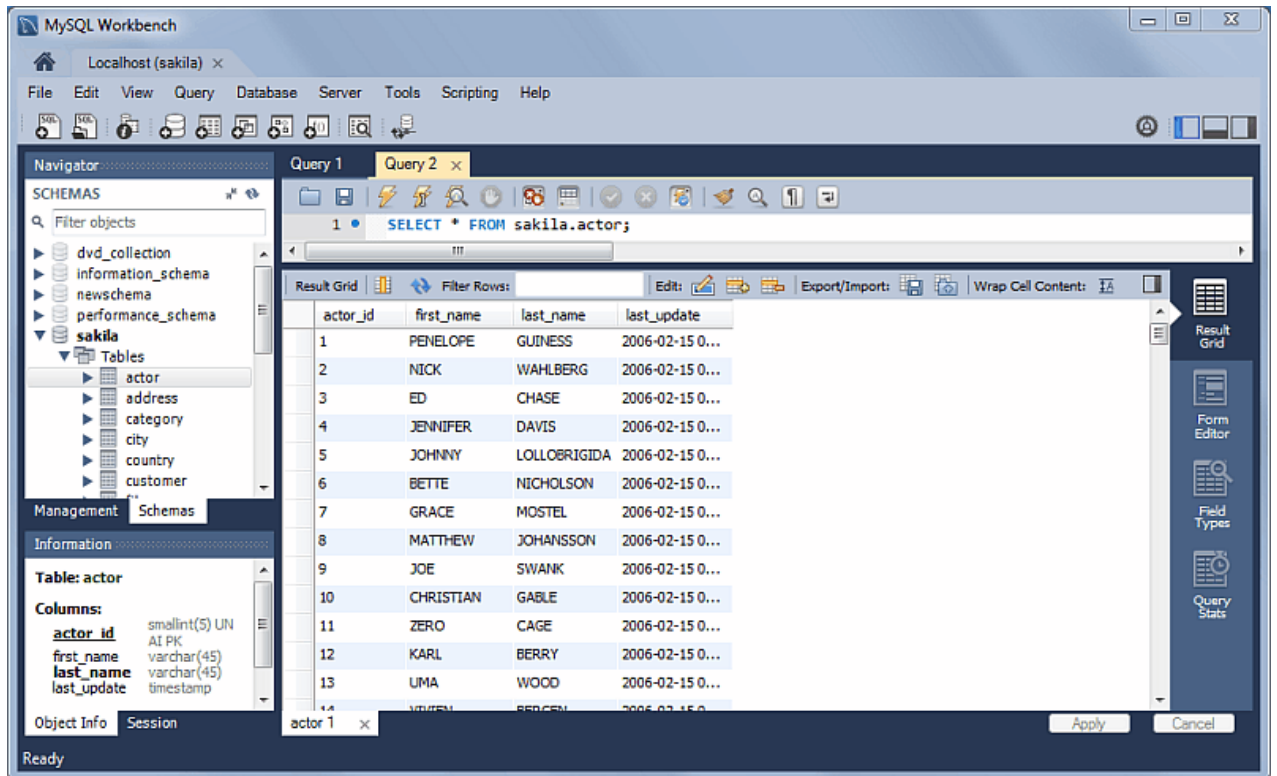
SQL Editor Views

Additional viewing options were added for executed statements:

Result Grid

Available previously, and it remains the default view.

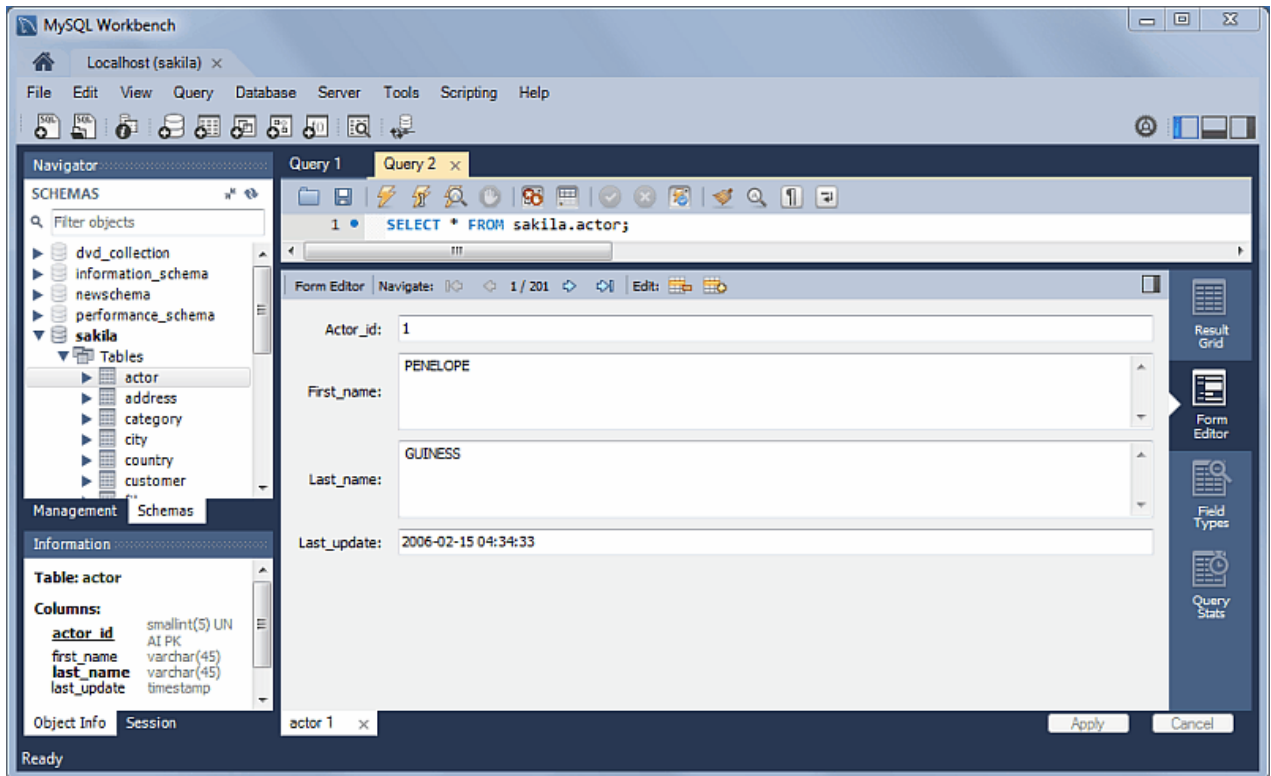
Figure 1.24 SQL Editor: Result Grid



Form Editor

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

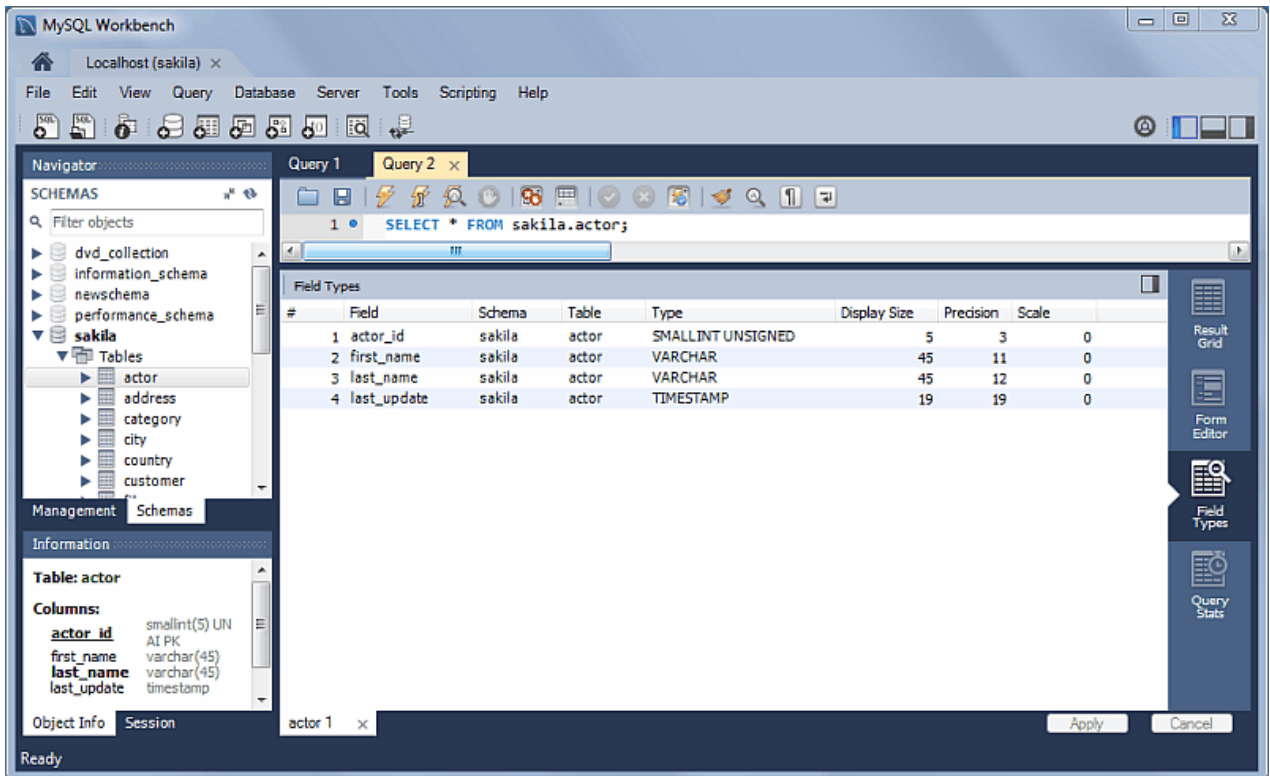
Figure 1.25 SQL Editor: Form Editor



Field Types

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

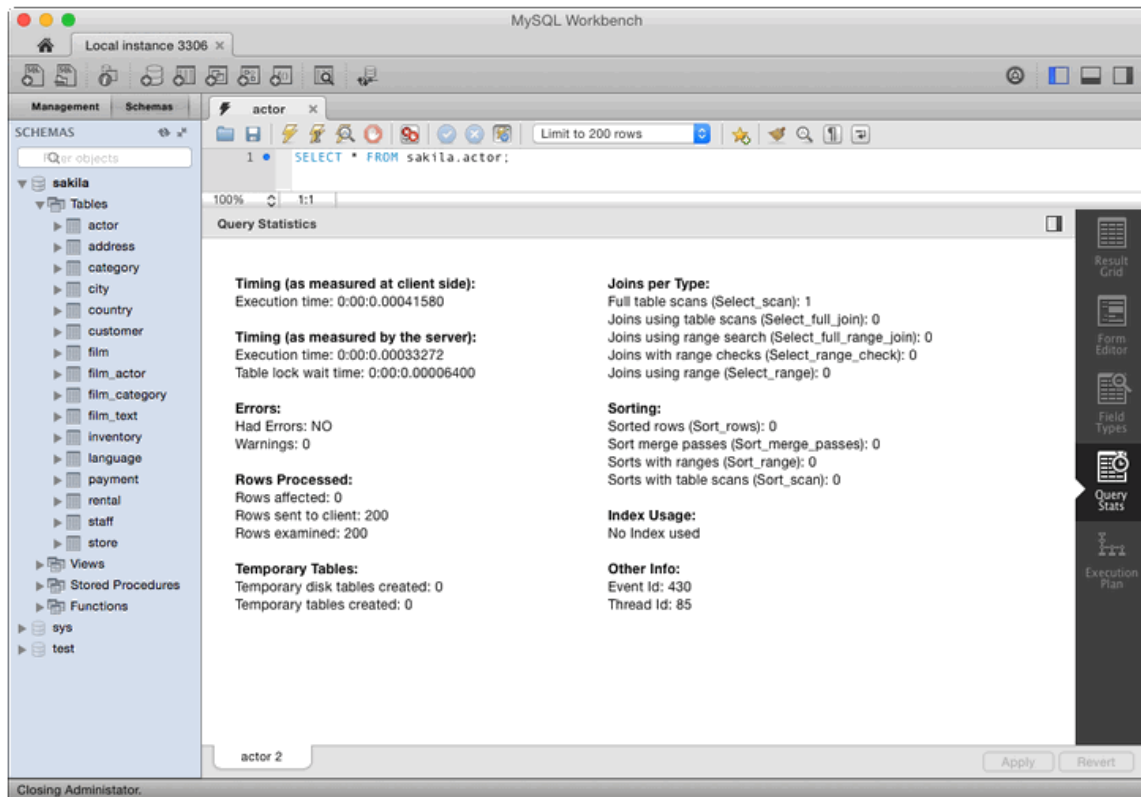
Figure 1.26 SQL Editor: Field Types



Query Stats

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

Figure 1.27 SQL Editor: Query Stats



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 1.28 Visual Explain: Workbench 6.0

The screenshot shows the MySQL Workbench interface with a SQL query and its Visual Explain execution plan. The query is as follows:

```

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

The Visual Explain execution plan shows a nested loop join starting with the **film (ALL)** table (1000 rows). It then joins the **inventory (ref)** table (2 rows) using the `idx_fk_film_id` key. Next, it joins the **rental (ref)** table (1000 rows) using the `idx_fk_inventory_id` key. Then, it joins the **customer (eq_ref)** table (1 row) using the `PRIMARY` key. Finally, it joins the **address (eq_ref)** table (1 row) using the `PRIMARY` key. The plan also shows the attached condition for the rental table: `(isnull('sakila`.`rental`.`return_date') and (('sakila`.`rental`.`rental_date' + interval 'sakila`.`film`.`rental_duration` day) < <cache>(curdate())))`.

Query finished.

Figure 1.29 Visual Explain: Workbench 6.1

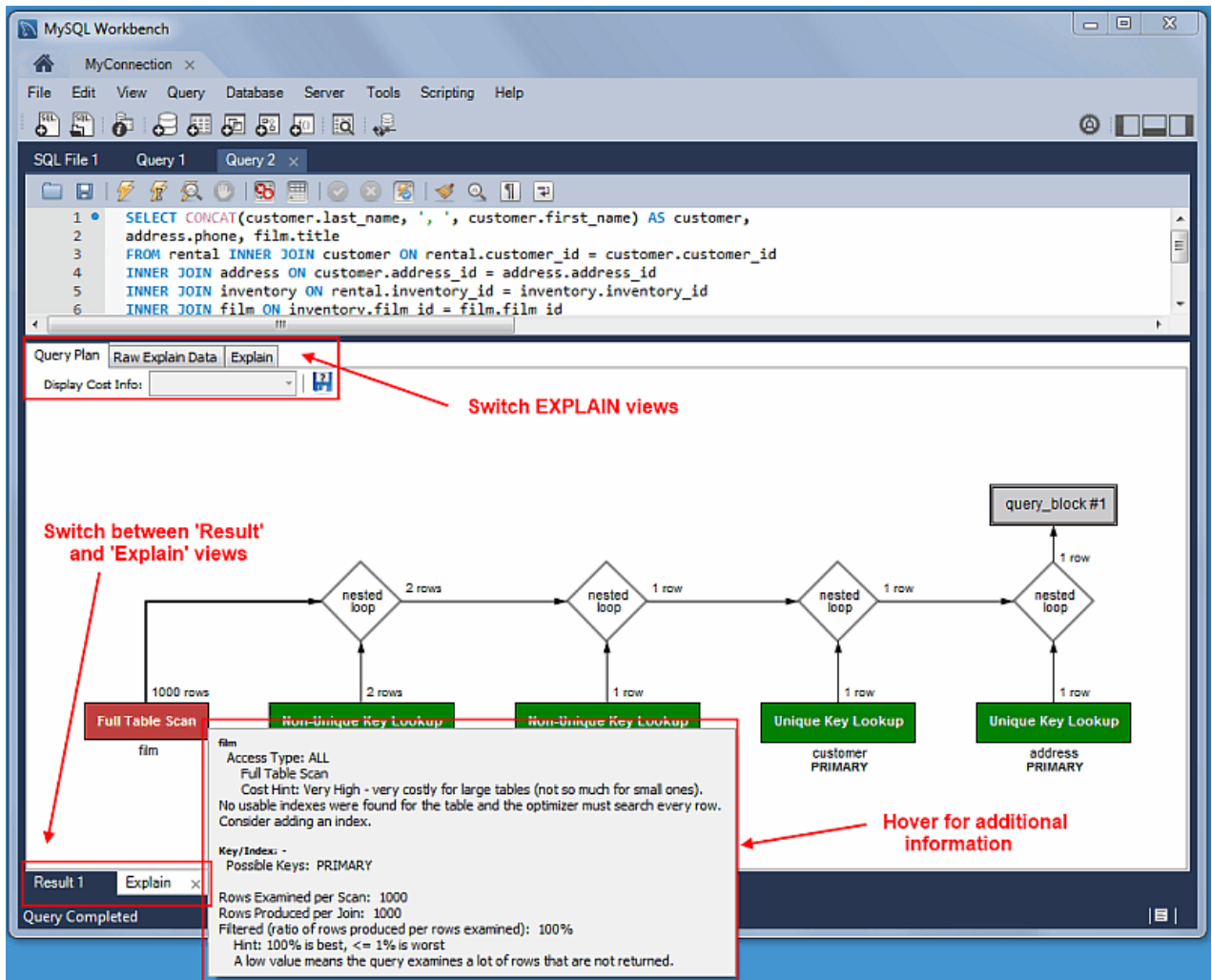
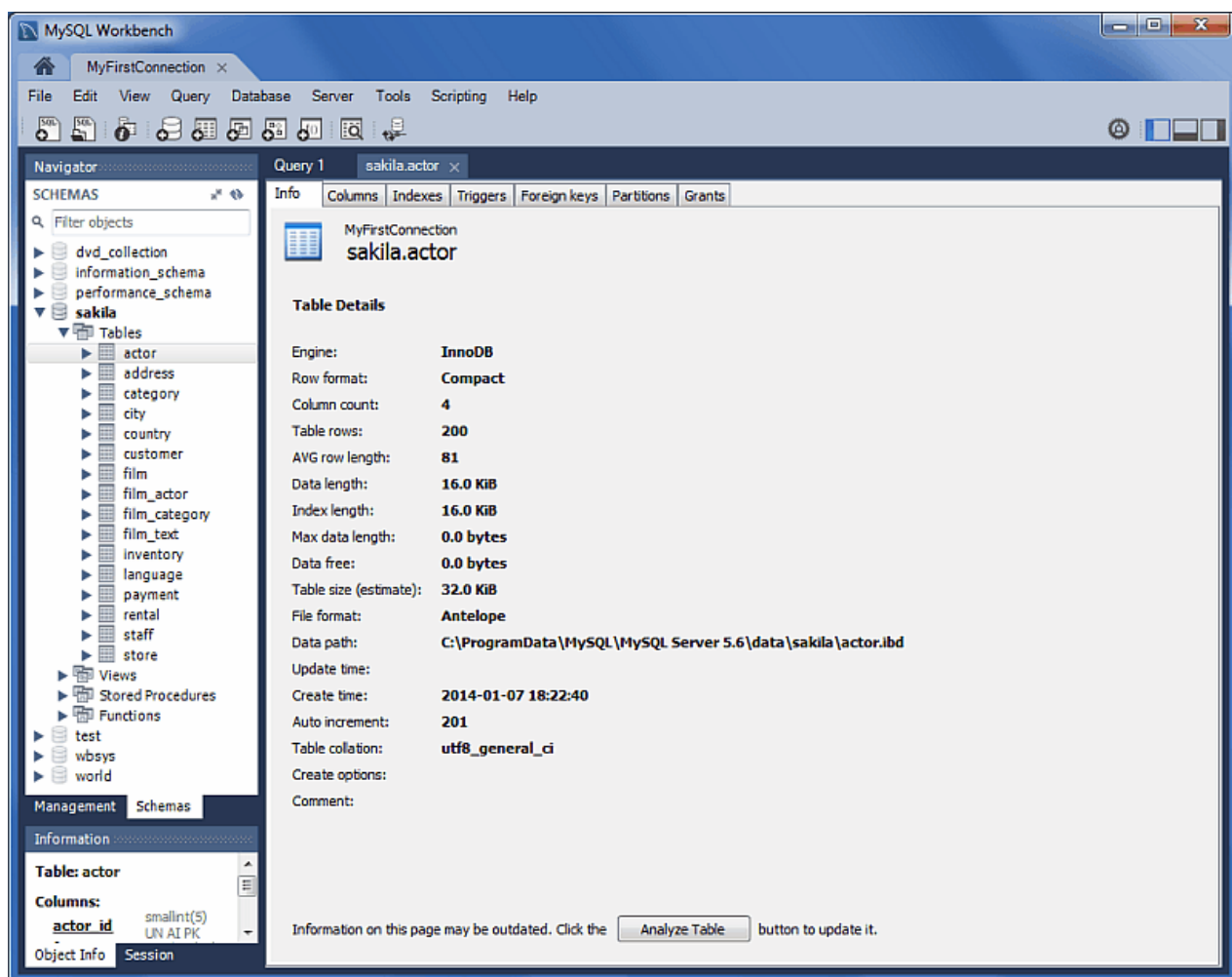


Table Inspector

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 1.30 Table Inspector



Additional Client-Connection Information

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

Figure 1.31 Client Connections: MySQL Workbench 6.0

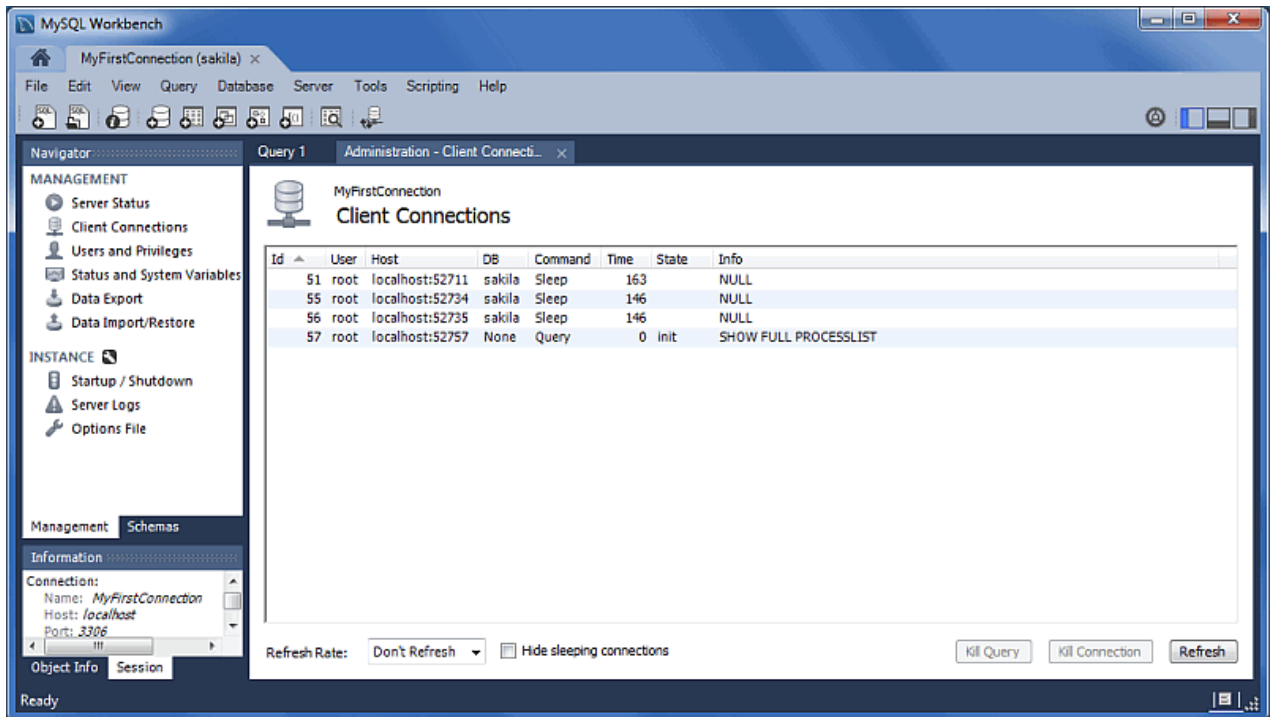
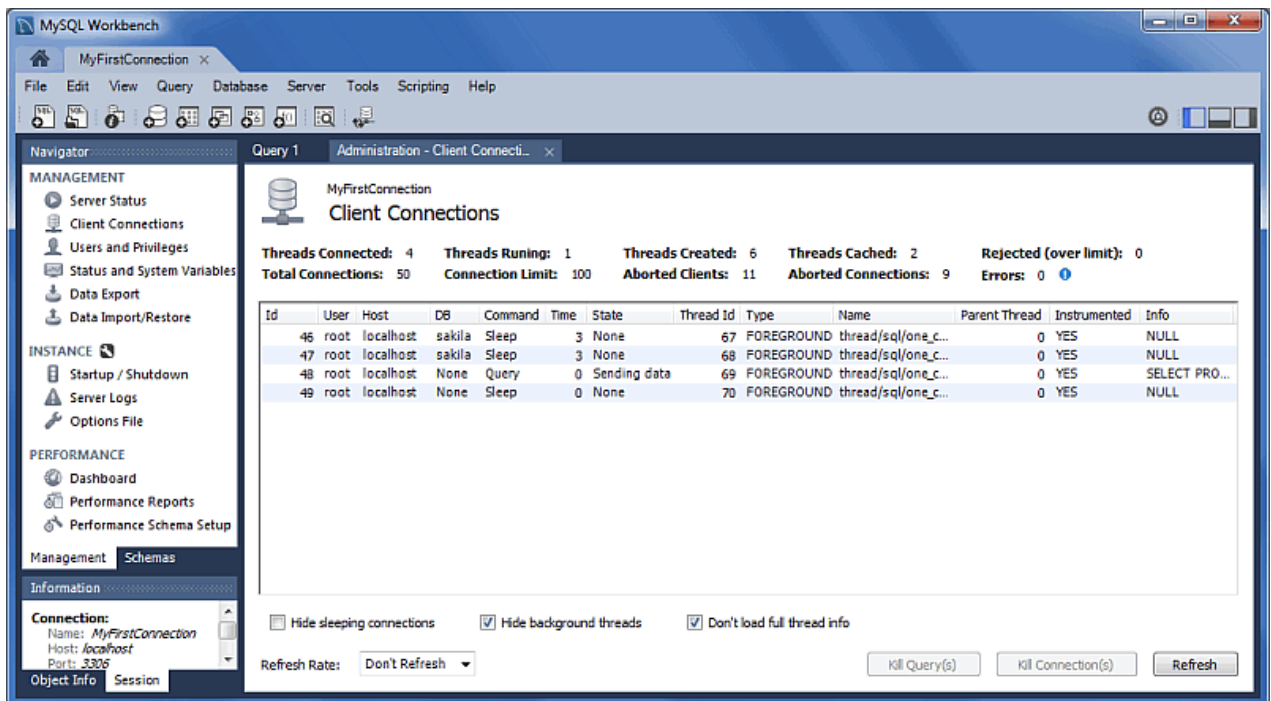


Figure 1.32 Client Connections: MySQL Workbench 6.1



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 1.33 Client Connections: View Thread Stack

Event Id	Event info	Type	Timer wait [us]
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
    
```

Close

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

1.1.2.4 New in MySQL Workbench 6.0

This section summarizes many of the new features added to MySQL Workbench 6.0, in relation to the MySQL Workbench 5.2 release.

- [A New Home Screen](#)
- [Unified SQL Editor and Administration Interface](#)
- [Table Data Search](#)
- [Context Help for the SQL Editor](#)
- [Schema Inspector](#)
- [Cascaded DELETE Statements Generator](#)
- [Table Templates](#)
- [Vertical Text](#)
- [Improved Visual Explain](#)
- [Improved Server Status](#)
- [Enterprise Features](#)
- [Database Migration Features](#)

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 1.34 Home Screen: Workbench 5.2

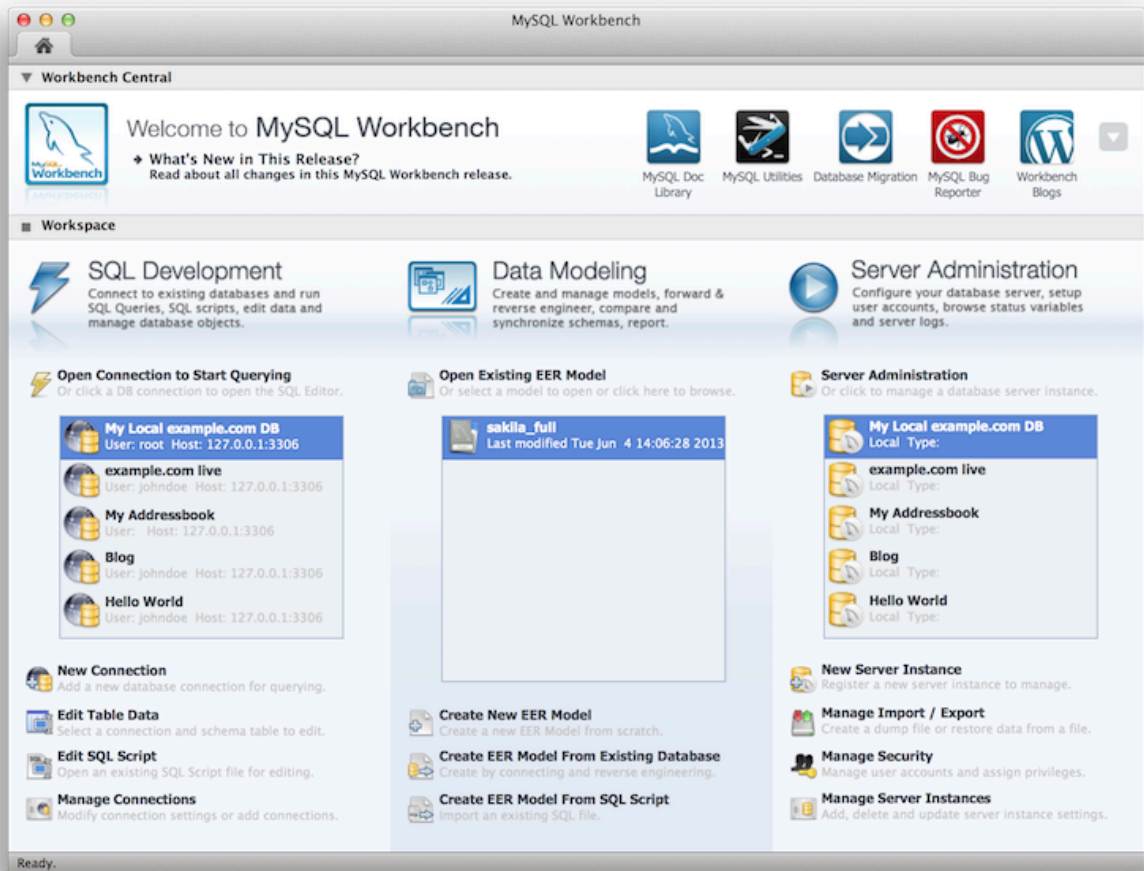
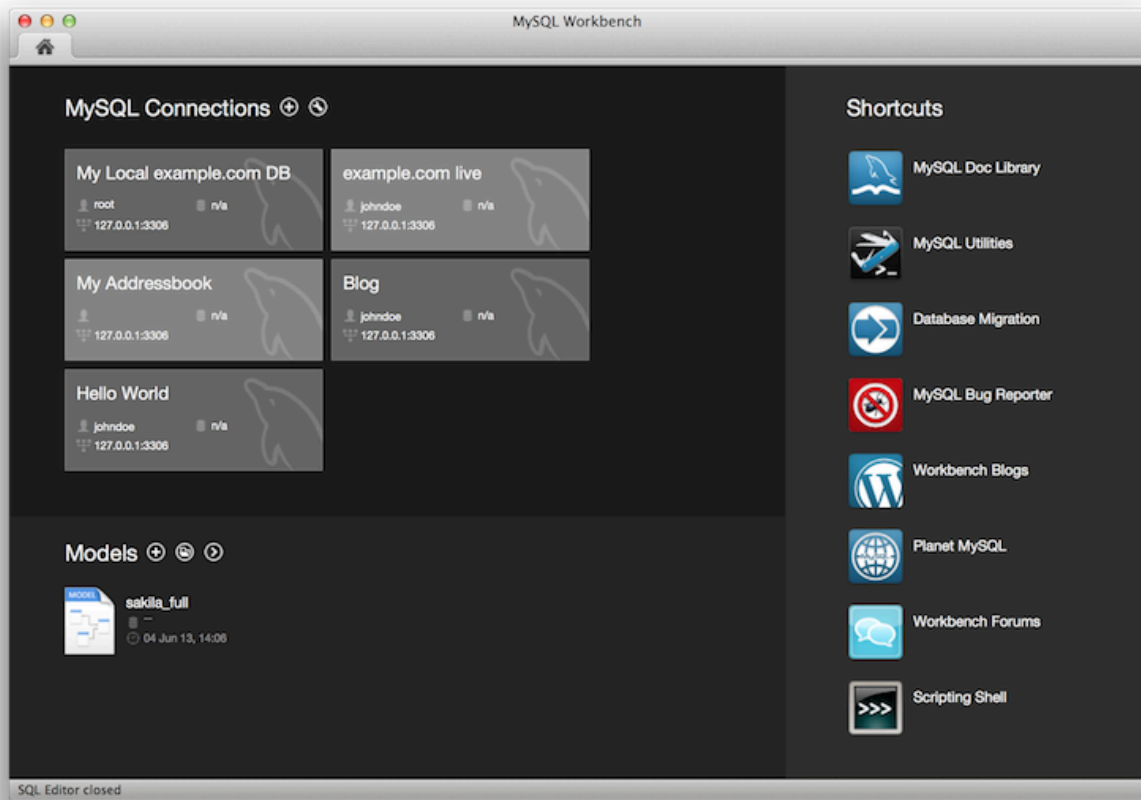


Figure 1.35 Home Screen: Workbench 6.0



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 following figure 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.

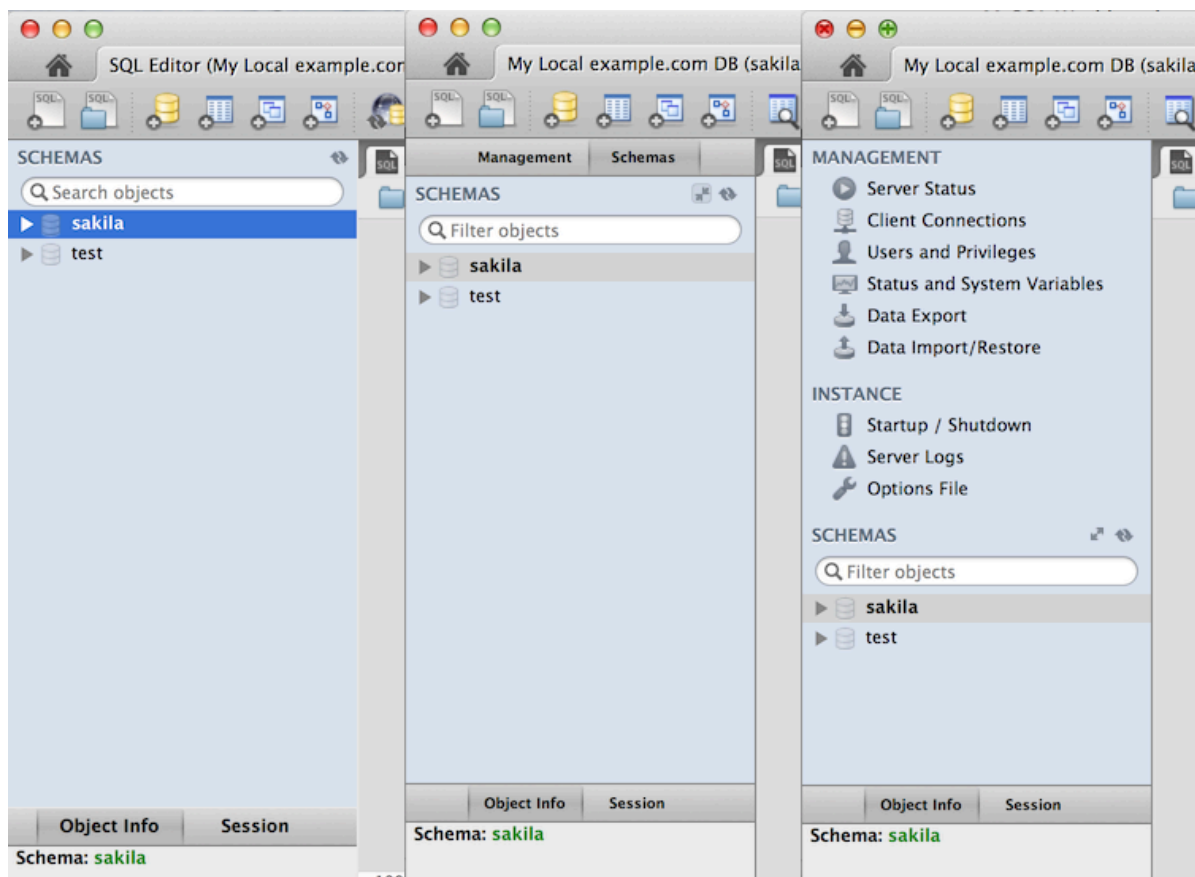
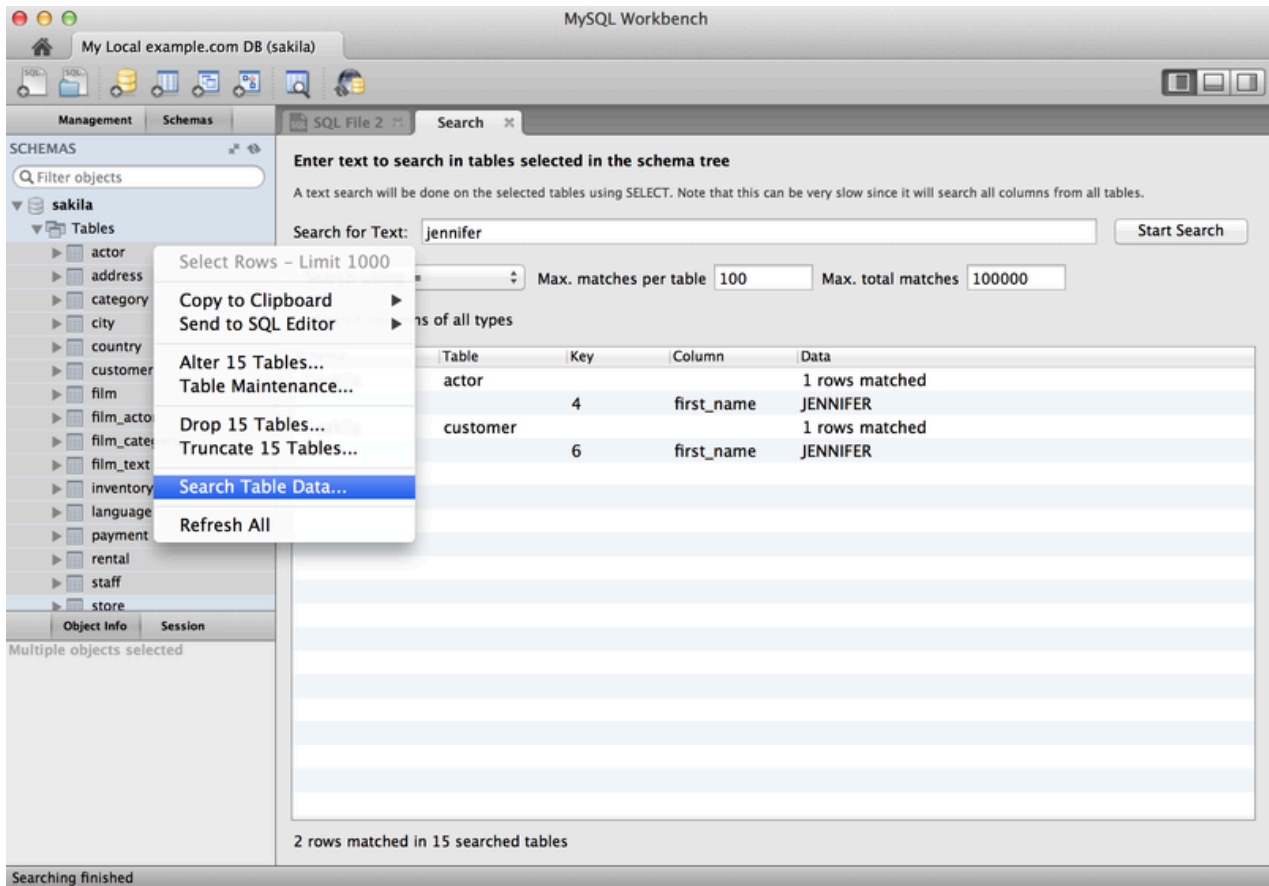
Figure 1.36 Comparing the SQL Editor interface for Workbench 5.2 and 6.0

Table Data Search

You can select schemas, tables, or both 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 1.37 Table search functionality

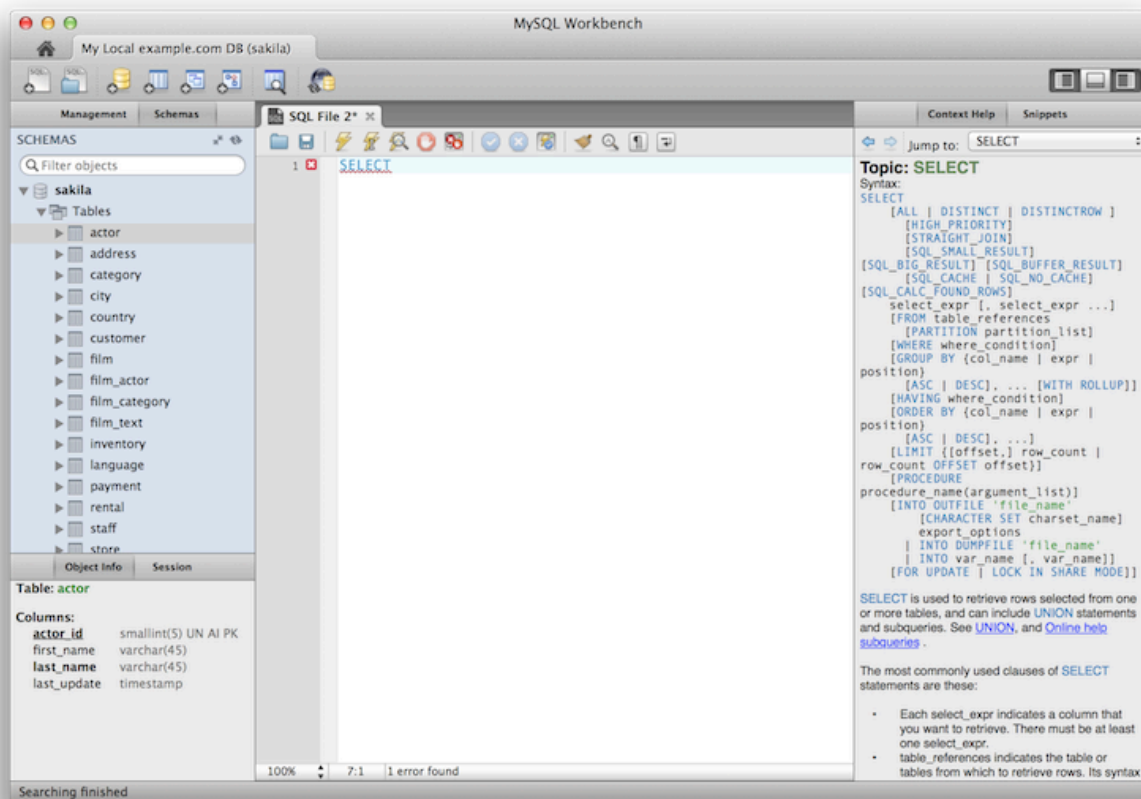


For additional information, see [Section 8.1.8, “Table Data Search Tab”](#).

Context 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 1.38 Context Sensitive Help

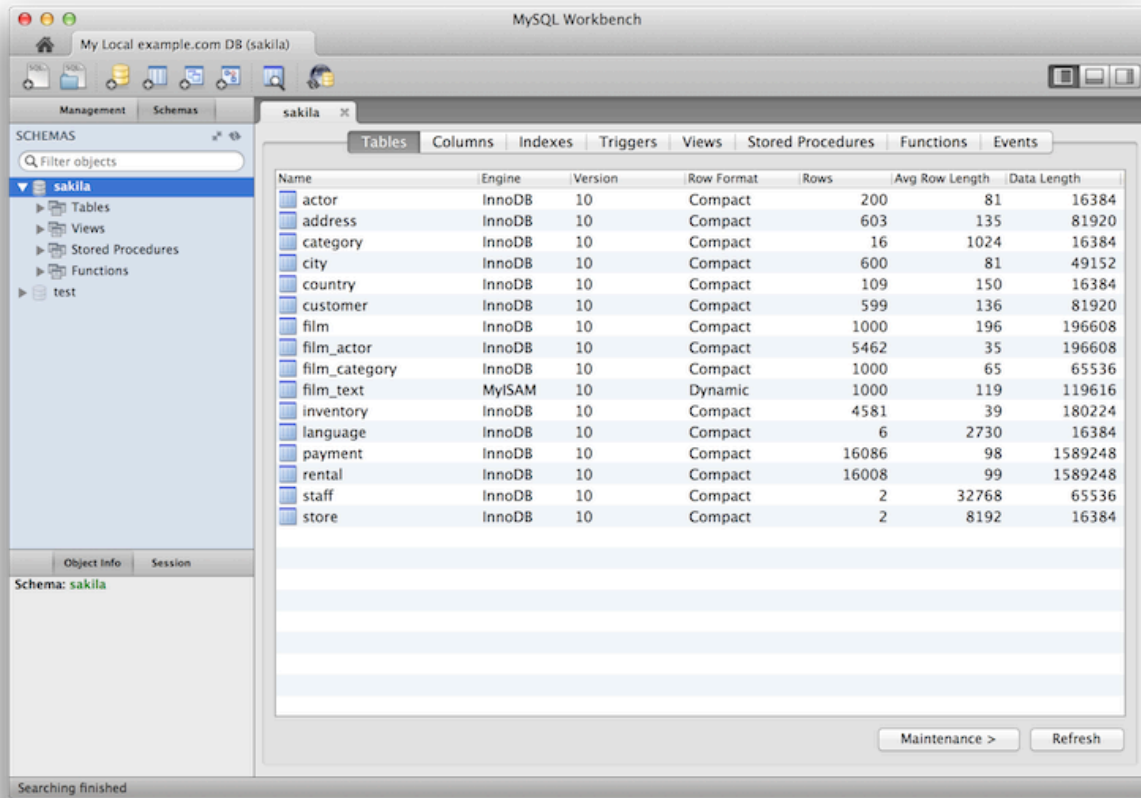


For additional information, see [Section 8.1.6, “SQL Additions - Context Help Tab”](#).

Schema Inspector

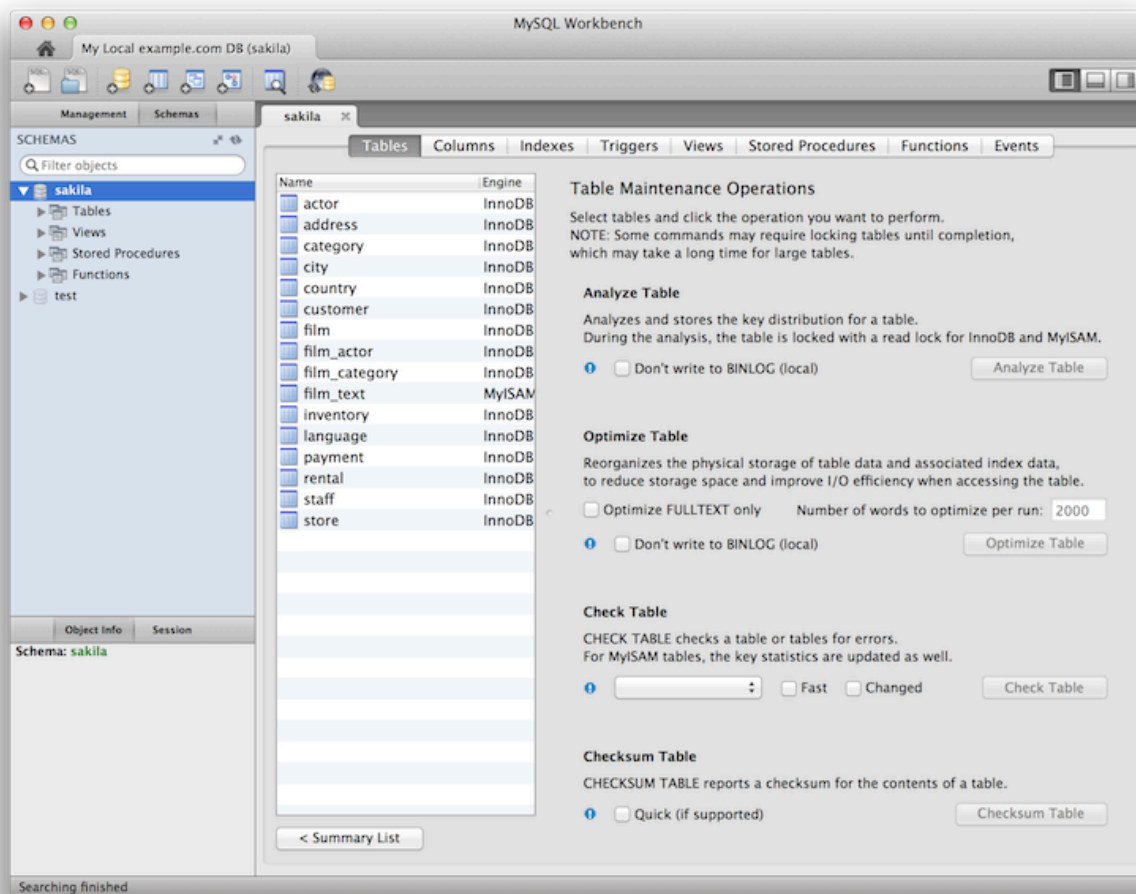
New Schema Inspector feature allows you to browse general information from schema objects. For tables, it's also possible to perform maintenance tasks such as [ANALYZE](#), [OPTIMIZE](#), [CHECK](#), and [CHECKSUM TABLE](#). To access the inspector, right-click a schema and select the **Schema Inspector**

Figure 1.39 Schema Inspector



And choosing **Maintenance** for a table:

Figure 1.40 Schema Inspector: Maintenance



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 1.41 Cascading SELECT

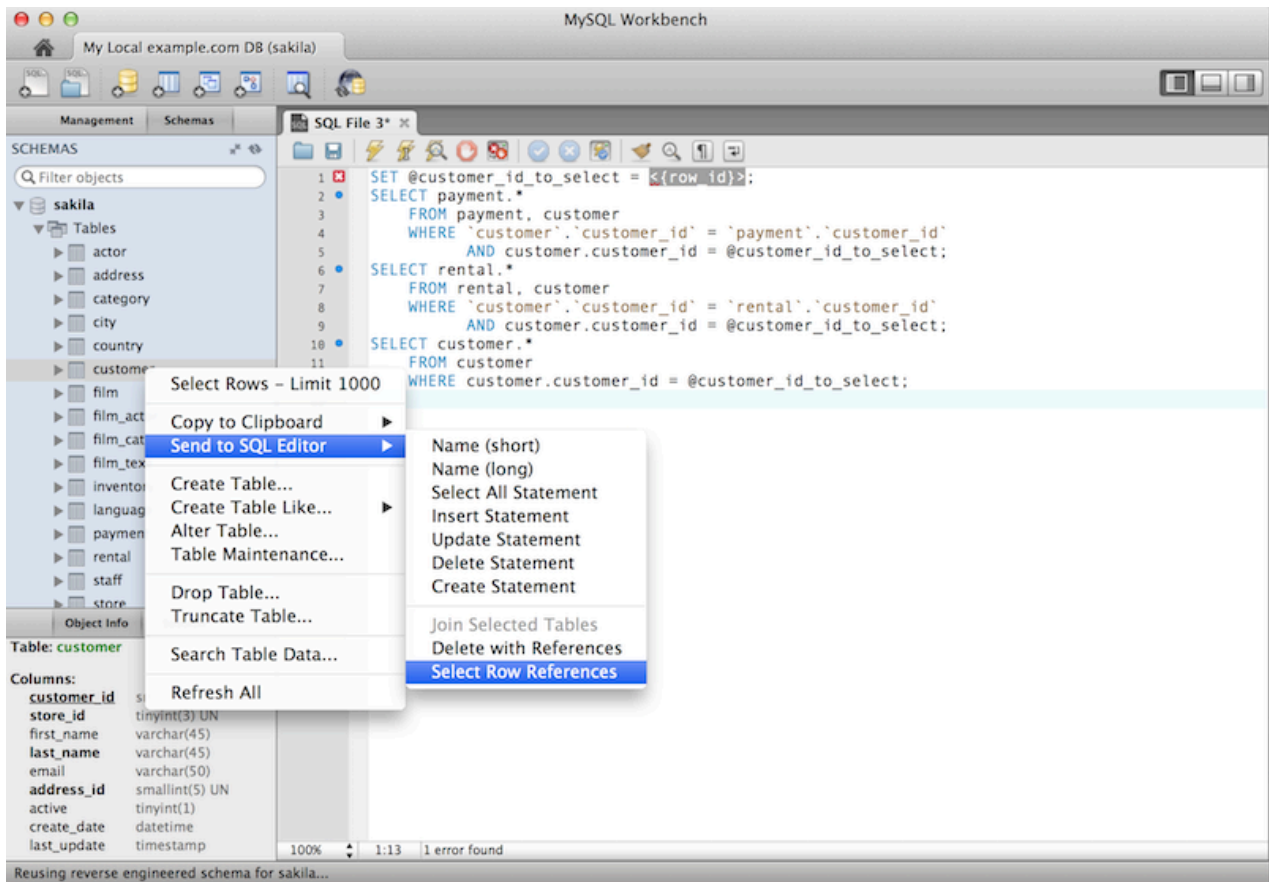


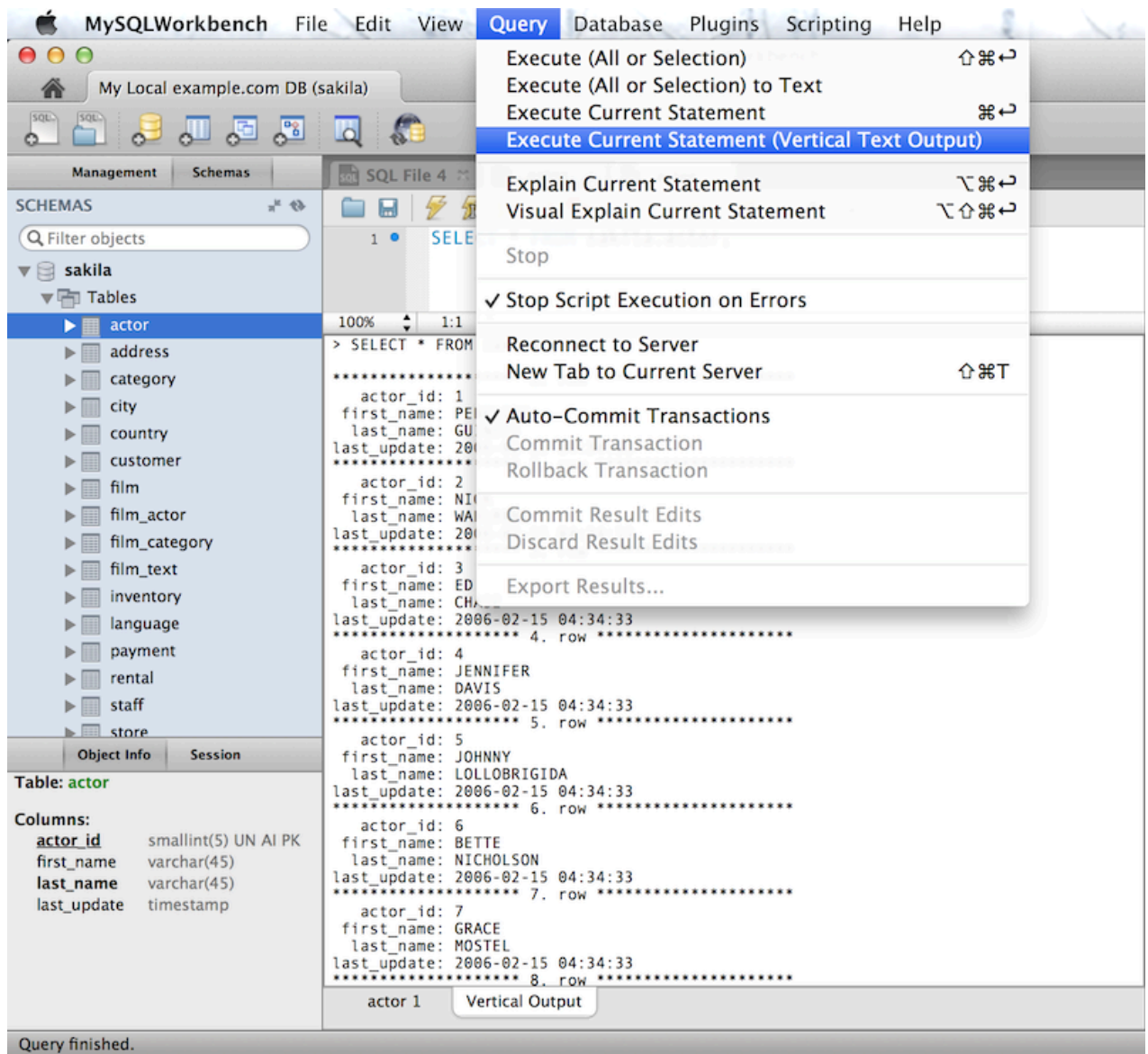
Table Templates

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 1.42 Vertical Text (⌘G)



Improved Visual Explain

The Visual Explain output was improved.

Figure 1.43 Visual Explain: Workbench 5.2

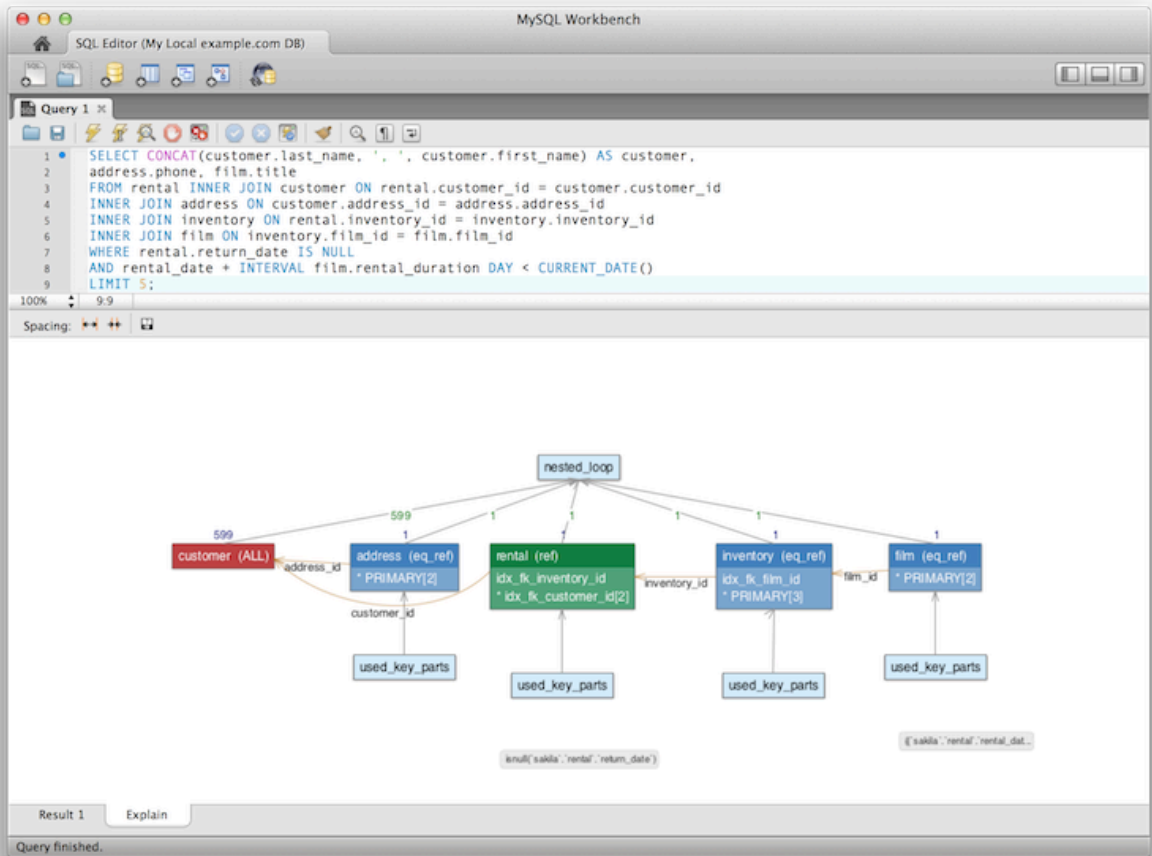
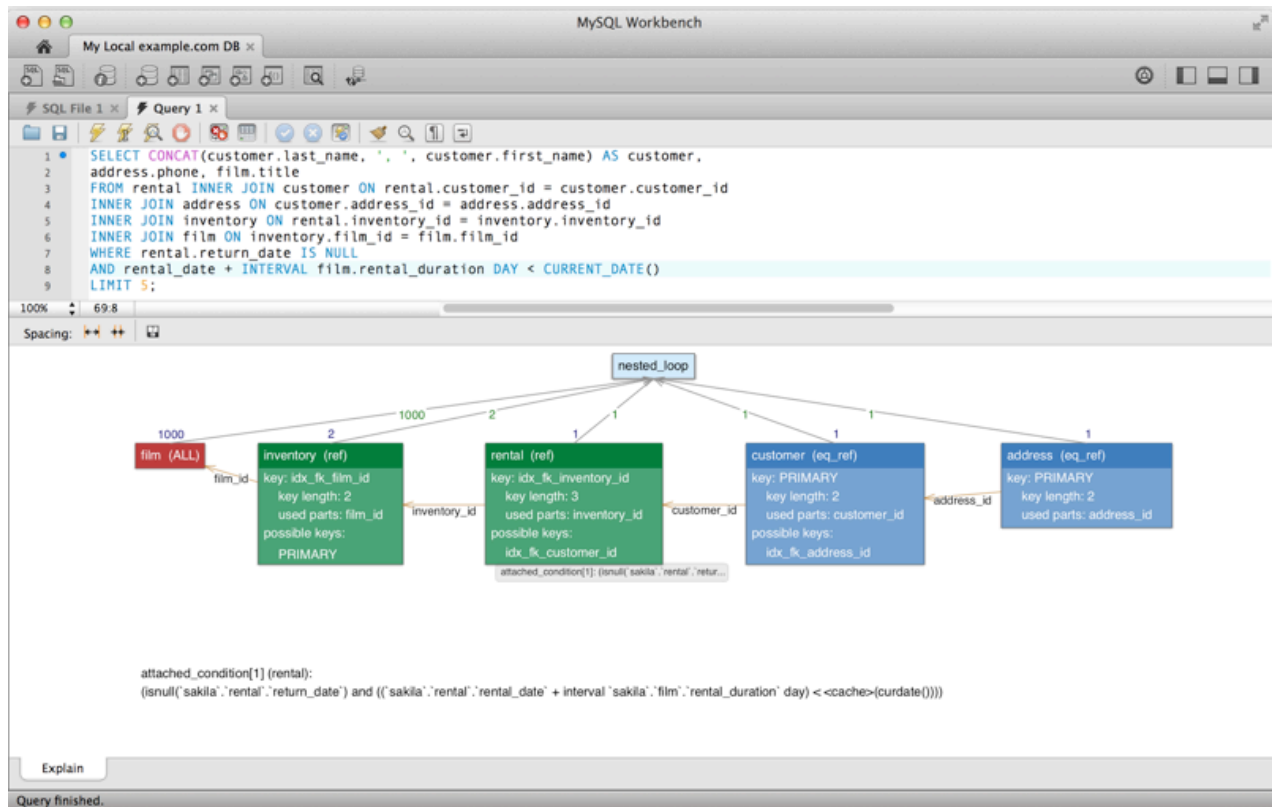


Figure 1.44 Visual Explain: Workbench 6.0



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 1.45 Server Status: Workbench 5.2

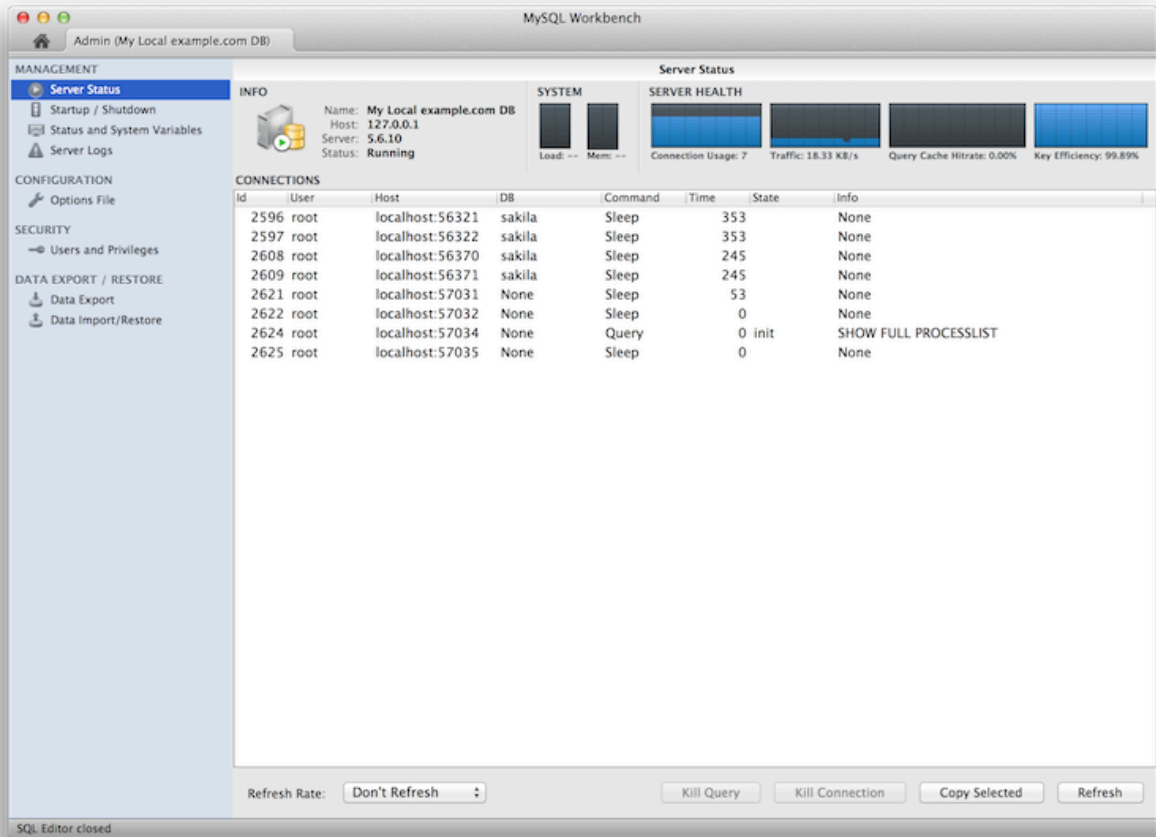
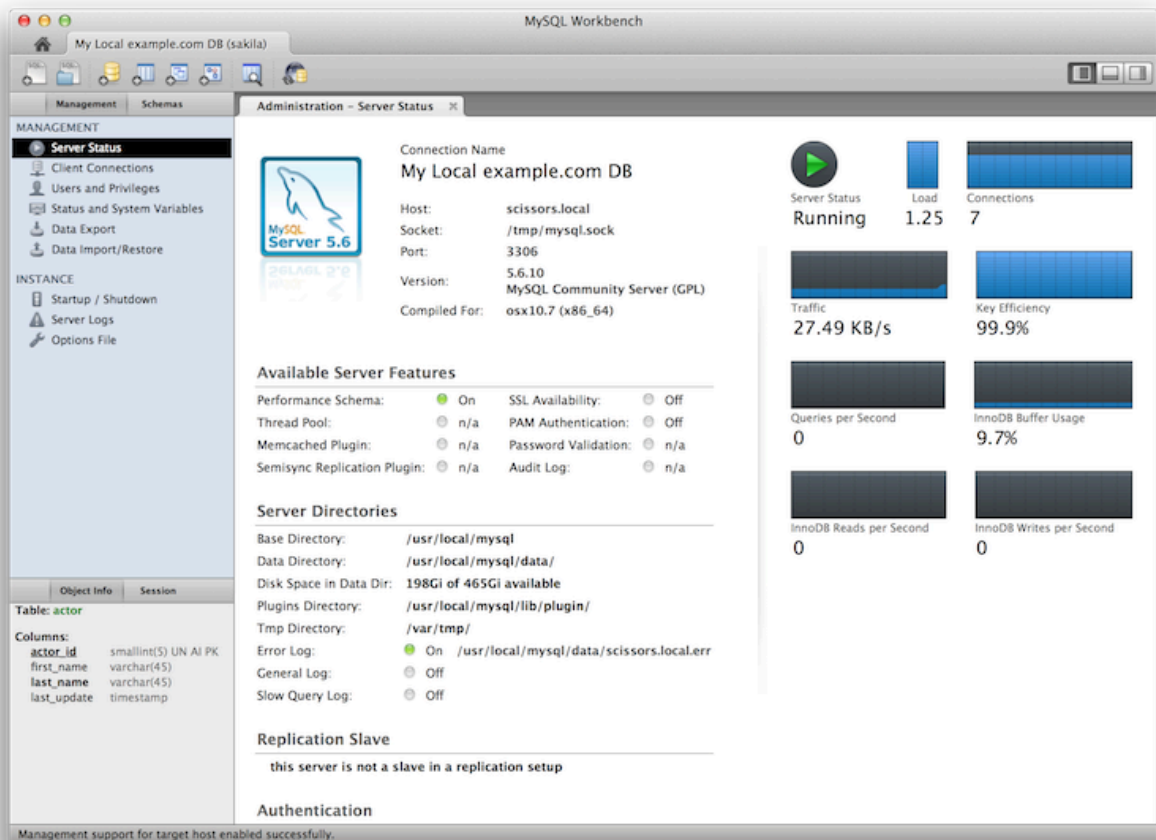


Figure 1.46 Server Status: Workbench 6.0



Enterprise Features

Support for MySQL Enterprise features in MySQL Workbench Commercial 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 MySQL Workbench Commercial and MySQL Enterprise Backup, MySQL Workbench will check for and handle the prerequisites. Backup recovery is also supported. This plugin supports MEB with local and remote installations of Linux and macOS, and locally for 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 macOS, and locally for Microsoft Windows.

Database Migration Features

SQL Anywhere and SQLite are now supported.

1.2 MySQL Workbench Editions

MySQL Workbench is available in the following editions:

- MySQL Workbench Community (Open Source, GPL)
- MySQL Workbench Commercial (Standard Edition and Enterprise Edition).

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

For more information about the MySQL Enterprise Edition, visit <http://www.mysql.com/enterprise>

Chapter 2 Installation

Table of Contents

2.1 System Requirements	55
2.2 Command-Line Options	57
2.3 MySQL Workbench on Windows	57
2.3.1 Installing	57
2.3.2 Launching	58
2.3.3 Uninstalling	59
2.4 MySQL Workbench on Linux	59
2.4.1 Installing	59
2.4.2 Launching	62
2.4.3 Uninstalling	63
2.5 MySQL Workbench on macOS	64
2.5.1 Installing	64
2.5.2 Launching	64
2.5.3 Uninstalling	65

MySQL Workbench is available for Windows, Linux, and macOS.

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 <https://dev.mysql.com/downloads/workbench/>. The source code is also [available on GitHub](#).

The sections in this chapter 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 <https://www.mysql.com/support/supportedplatforms/workbench.html> on the MySQL website.

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, Connections in MySQL Workbench](#). For additional information about installing a MySQL server, see [Installing 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 as 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 connections" 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 vault** functionality requires [gnome-keyring-daemon](#) to store the passwords. Note that KDE systems use their own [ksecret-service](#) implementation.
- For Linux and macOS, the MySQL server administration features require [sudo](#) command 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 entry to the [/etc/sudoers](#) file safely by using the [visudo](#) command:

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

For MySQL Workbench to execute MySQL Enterprise Backup commands, the [sudo](#) command user must also be able to execute the MySQL Enterprise Backup binary.

Requirements for Windows

The following prerequisites are available at the [Microsoft Download Center](#):

- Microsoft .NET Framework 4.5.2
- Microsoft Visual C++ 2015-2022 Redistributable
- Microsoft Windows 11 or Windows Server 2022

2.2 Command-Line Options

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

- `--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 Default location of generated MySQL Workbench log files

Platform	Default location
Linux	<code>~/.mysql/workbench/log/</code>
macOS	<code>~/Library/Application Support/Workbench/log/</code>
Microsoft Windows	<code>C:\Users\user_name\AppData\Roaming\MySQL\Workbench\log\</code>

- `--admin instance`: Opens an administration tab to the named connection.
- `--upgrade-mysql-dbs`: Opens the Schema Transfer Wizard.
- `--migration`: Opens the MySQL Workbench Migration Wizard.
- `--log-to-stderr`: Also sends the log to `stderr`.
- `--version`: Shows the MySQL Workbench version number and exits.
- `--verbose`, `-v`: Enables diagnostics output.
- `--query [connection]`: Opens a named connection.
- `--model modelfile`: Opens the given EER model file.
- `--script script`: Opens the given SQL file in a connection, typically used with the `--query` parameter.
- `--run code`: Executes the given code using the default language for GRT shell.
- `--run-python script`: Executes the given code in Python.
- `--run-script file`: Executes Python code from a file.
- `--open file`: Opens 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 or the standalone Windows MSI Installer package. For general requirements and specific installation instructions, see the sections that follow.

- [Requirements for Windows](#)

- [Installation Using MySQL Installer](#)
- [Installation Using the Windows MSI Installer Package](#)

Requirements for Windows

The following prerequisites are available at the [Microsoft Download Center](#):

- Microsoft .NET Framework 4.5.2
- Microsoft Visual C++ 2015-2022 Redistributable
- Microsoft Windows 11 or Windows Server 2022

Installation Using MySQL Installer

The general MySQL Installer download is available at <https://dev.mysql.com/downloads/windows/installer/>. The MySQL Installer application can install, upgrade, and manage most MySQL products, including MySQL Workbench. 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 MSI Installer package described in the next section.

Installation Using the Windows MSI Installer Package

The standalone download is available at <https://dev.mysql.com/downloads/workbench/>.



Important

Installing MySQL Workbench using a Windows MSI Installer package requires either Administrator or Power User privileges.

MySQL Workbench can be installed using the Windows MSI Installer 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 (winx64).

To install MySQL Workbench:

1. From an account with Administrator or Power User privileges, 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 page, select either a [Complete](#) or [Custom](#) installation. To use all features of MySQL Workbench choose the [Complete](#) setup type.
3. Unless you choose otherwise, MySQL Workbench is installed in `C:\%PROGRAMFILES%\MySQL\MySQL Workbench 8.0 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.

2.3.2 Launching

To start MySQL Workbench on Windows, select **MySQL** from the **Start** menu and then select MySQL Workbench. This sequence executes the `MySQLWorkbench.exe` file on your system. Alternatively, start MySQL Workbench from the command line, for example:

```
C:\Program Files\MySQL\MySQL Workbench 8.0\mysqlworkbench.exe
```

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.

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

To uninstall MySQL Workbench, do one of the following:

- From **Start**, select **Settings** and **Apps**.
- Search for [Add or Remove Programs](#).

Select the MySQL Workbench entry and click **Uninstall** to 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 remove some of the components manually, there is no command-line option for removing MySQL Workbench.

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

Removing MySQL Workbench After Installation from MySQL Installer

1. From the MySQL Installer dashboard, click [Remove](#) to open the Select Products to Remove page.
2. Select MySQL Workbench (the status changes to [Ready to remove](#)) and click **Next**.
3. Click **Execute** to uninstall all of the selected products.

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\` and the `C:\username\AppData\Roaming\` portion 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 Additions - 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. For general requirements and specific installation instructions, see the sections that follow.

- [Requirements for Linux](#)
- [Installing DEB Packages](#)
- [Installing RPM Packages](#)
- [Installing Oracle Enterprise Linux and Similar](#)

The procedure for installing on Linux depends on which Linux distribution you are using. Select one of the following installation methods:

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 <code>mysql-workbench-community</code> .
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 <code>mysql-workbench</code> .
Download official MySQL packages	Downloads are available at https://dev.mysql.com/downloads/workbench .
Download the source code and compile yourself	The source code is available at https://dev.mysql.com/downloads/workbench as a <code>tar.gz</code> or RPM package.



Note

A change in the `%cmake` macro causes an error to occur when MySQL Workbench is built from source code on Fedora 33 using the RPM package. To avoid the error, use the new `%cmake_build` macro.

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 vault** functionality requires `gnome-keyring-daemon` to store the passwords. Note that KDE systems use their own `ksecret-service` implementation.
- For Linux and macOS, the MySQL server administration features require `sudo` command 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 entry to the `/etc/sudoers` file safely by using the `visudo` command:

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

For MySQL Workbench to execute MySQL Enterprise Backup commands, the `sudo` command 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 for which the command `"apt-get install mysql-workbench"` installs 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"`.

1. Install the MySQL APT repository as described in the [MySQL APT Repository](#) documentation. For example:

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

2. Install the MySQL Workbench package. There might be multiple Workbench packages available, so specify the exact version to install such as `mysql-workbench-community`. For example:

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

Installing a Package Manually

You install MySQL Workbench using a command such as:

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

In the previous example, `package.deb` is the MySQL Workbench package name; for example, `mysql-workbench-community-version1ubul404-amd64.deb`, where `version` is the MySQL Workbench version number.



Note

You might be warned that certain libraries are not available, depending on what you already have installed. If such a warning interrupts the installation, install the required libraries first 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, may 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 for which the command "yum install mysql-workbench" installs 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".

1. Install the MySQL Yum repository as described in the [MySQL Yum Repository](#) documentation. For example:

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

2. Install the MySQL Workbench package. There might be multiple MySQL Workbench packages available, so specify the exact version to install such as `mysql-workbench-community`. For example:

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

Installing a Package Manually

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

In the previous example, `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 only if you are working with spatial data, but not for general use. Earlier versions of MySQL Workbench (before 8.0.18) require access to it for all use cases. EPEL is a repository with additional RPM packages that are not part of the core RHEL/OEL distribution. This includes packages (such as tinymce) 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:

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

$> 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 by using the following command:

```
$> /usr/bin/mysql-workbench
```

To show the available command-line options:

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

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:

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

Or, alternatively:

```
$> 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:

```
$> 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:

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

Or, alternatively:

```
$> 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 Additions - Snippets Tab"](#).

2.5 MySQL Workbench on macOS

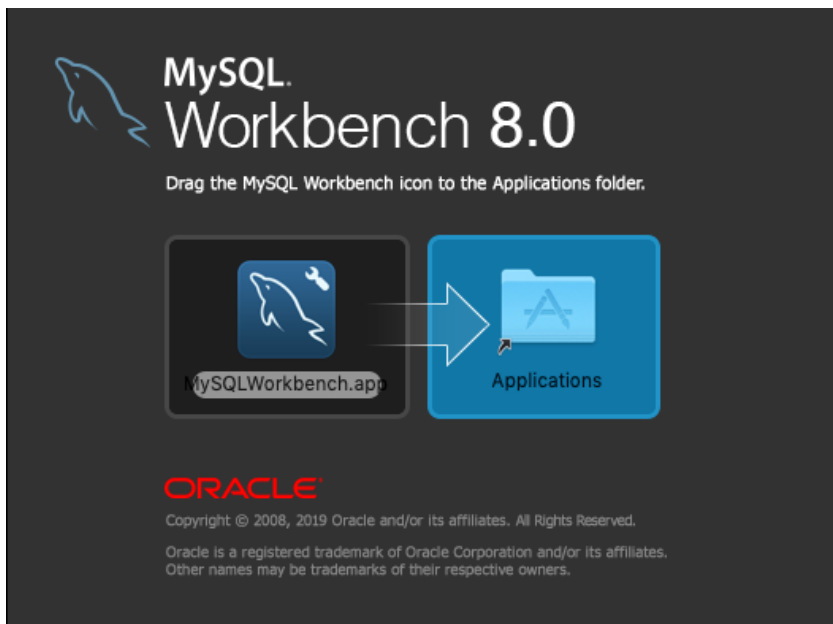
2.5.1 Installing

MySQL Workbench for macOS is distributed as a DMG file. The file is named `mysql-workbench-community-version-osx.dmg`, where `version` is the MySQL Workbench version. MySQL Workbench provides platform support for the latest version of macOS only (see [supported platforms](#)). The previous version continues to be supported for a transition period of six months after each new version of macOS is released.

Downloads are available at <https://dev.mysql.com/downloads/workbench/>.

To install MySQL Workbench on macOS, download the file. Double-click the downloaded file to open the installation window shown in the figure that follows.

Figure 2.1 MySQL Workbench macOS Installation Window



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.

2.5.2 Launching

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

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

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

Specifying options, a model file, or both is optional. For example:

```
$> /Applications/MySQLWorkbench.app/Contents/MacOS/MySQLWorkbench
```

To show the available command-line options:

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

2.5.3 Uninstalling

To uninstall MySQL Workbench for macOS, 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 Additions - Snippets Tab"](#).

Chapter 3 Configuration

Table of Contents

3.1 User Accessibility Options	67
3.2 Workbench Preferences	71
3.2.1 General Editors Preferences	72
3.2.2 SQL Editor Preferences	73
3.2.3 Administration Preferences	80
3.2.4 Modeling Preferences	81
3.2.5 Fonts and Colors Preferences	86
3.2.6 SSH Preferences	87
3.2.7 Other Preferences	89
3.3 MySQL Workbench Settings and Log Files	90
3.4 Common Preferences and Configurations	92

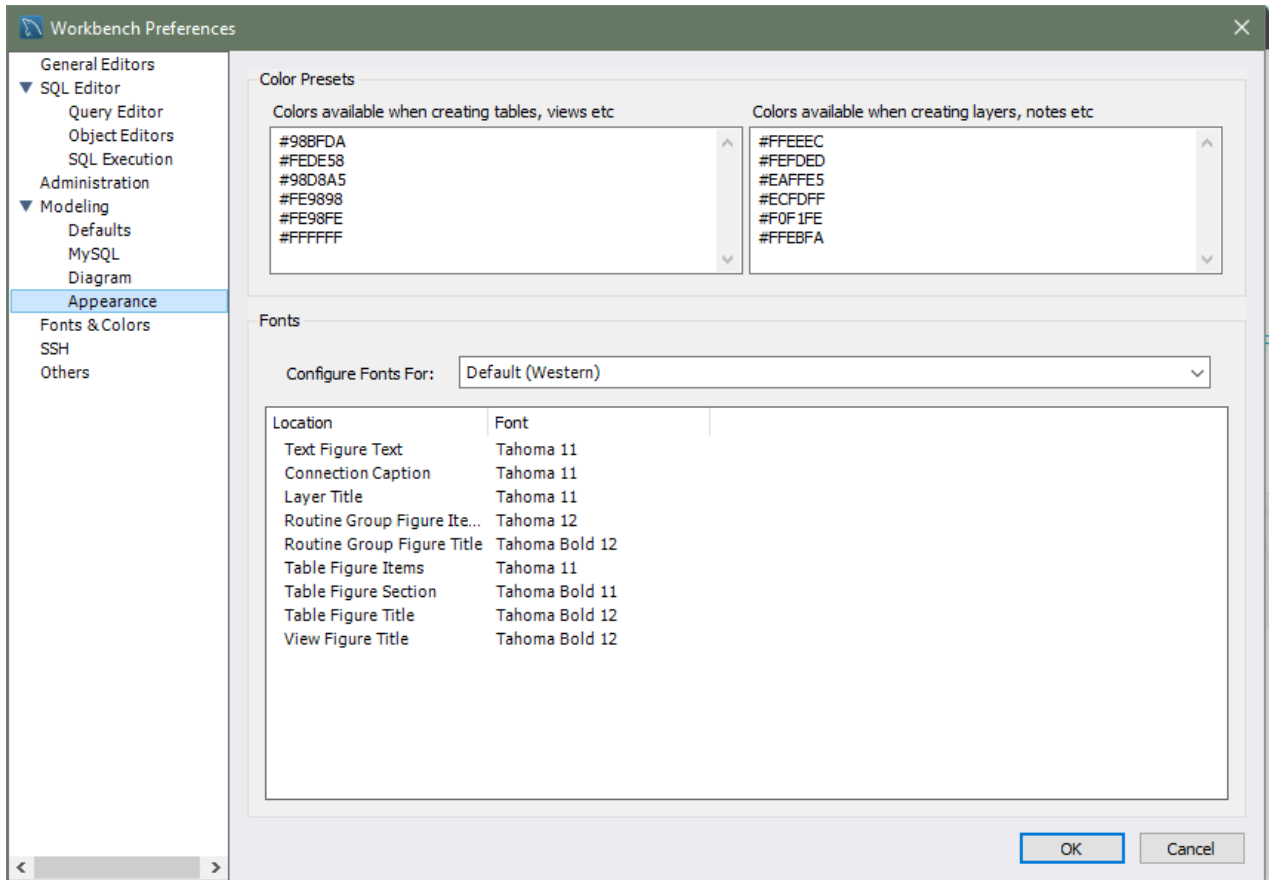
3.1 User Accessibility Options

MySQL Workbench includes options to improve user accessibility that you can select from the Workbench Preferences dialog. To open the dialog, click **Edit** and then **Preferences** from the menu.

Fonts

Modeling fonts are adjustable from the **Appearance** section of the **Modeling** list. The following figure shows the color presets and fonts.

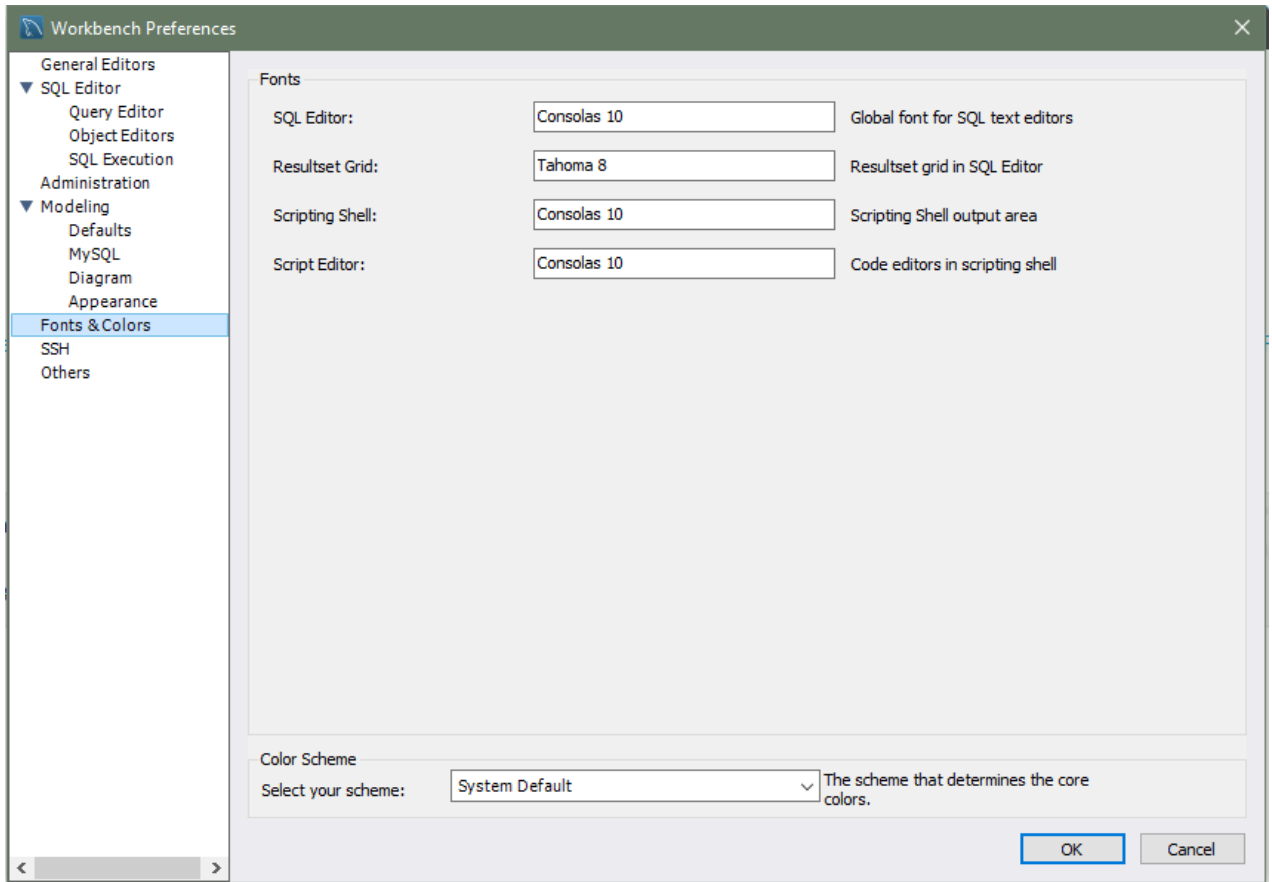
Figure 3.1 Appearance Preferences



Choose the character set from the **Configure Fonts For** list (or leave the default setting) and then adjust the model fonts to fit your requirements.

The font types and sizes for other screen elements are set from the **Fonts & Colors** preferences. The next figure shows the default fonts for the SQL Editor, Resultset Grid, Scripting Shell, and Script Editor.

Figure 3.2 Fonts & Color Preferences

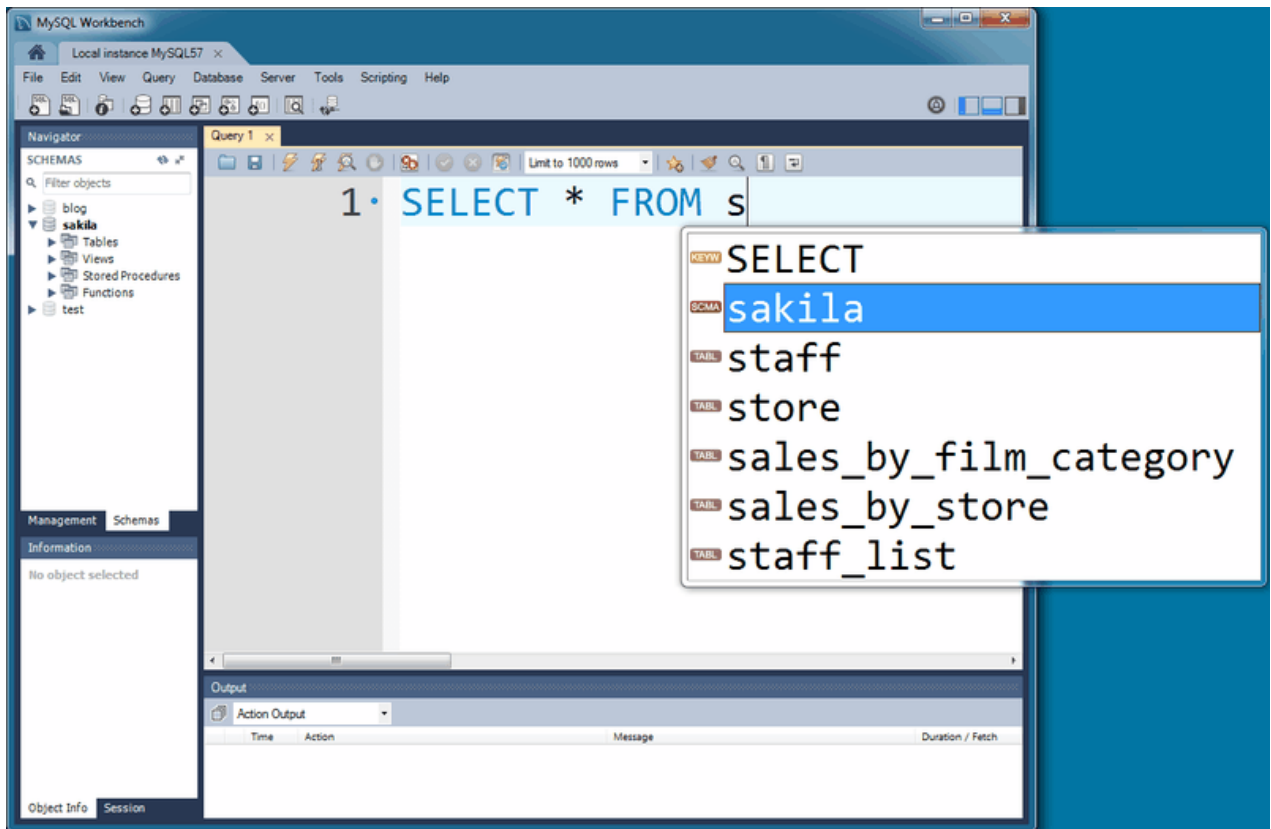


Note

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

The following figure shows an example of the SQL Editor after changing the **Editor** font size from 10 to 30.

Figure 3.3 SQL Editor with Font size 30



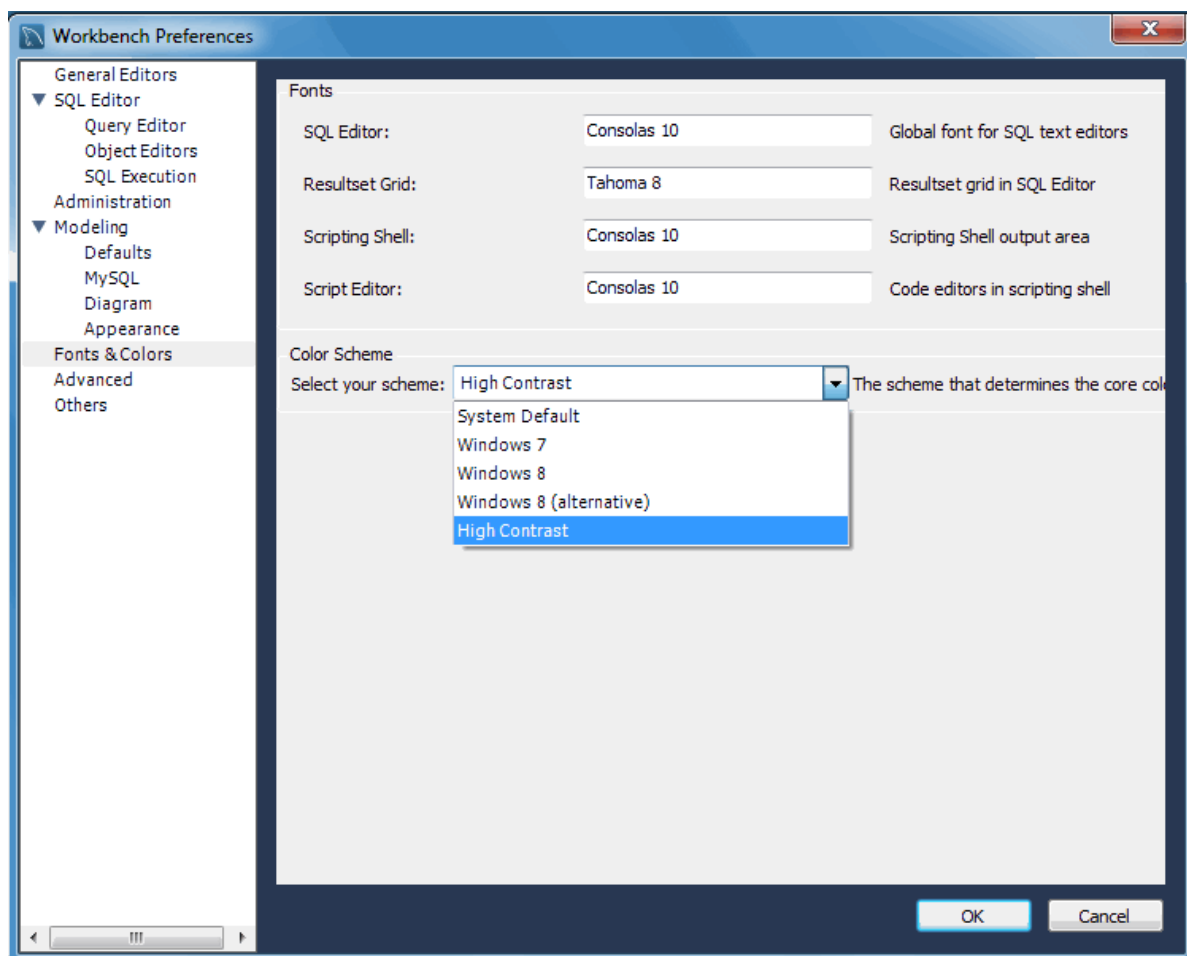
Color Presets

Color presets in the **Appearance** preferences enable you to 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 (see the figure that follows). This theme preference affects the MySQL Workbench GUI.

Figure 3.4 High Contrast Preference



Microsoft Active Accessibility (MSAA)

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

3.2 Workbench Preferences

Select **Preferences** from the **Edit** menu to configure MySQL Workbench to your specific needs. The **Workbench Preferences** sidebar menu is divided into the following sections:

- **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.
- **SSH:** Set secure connection timeouts and file locations.

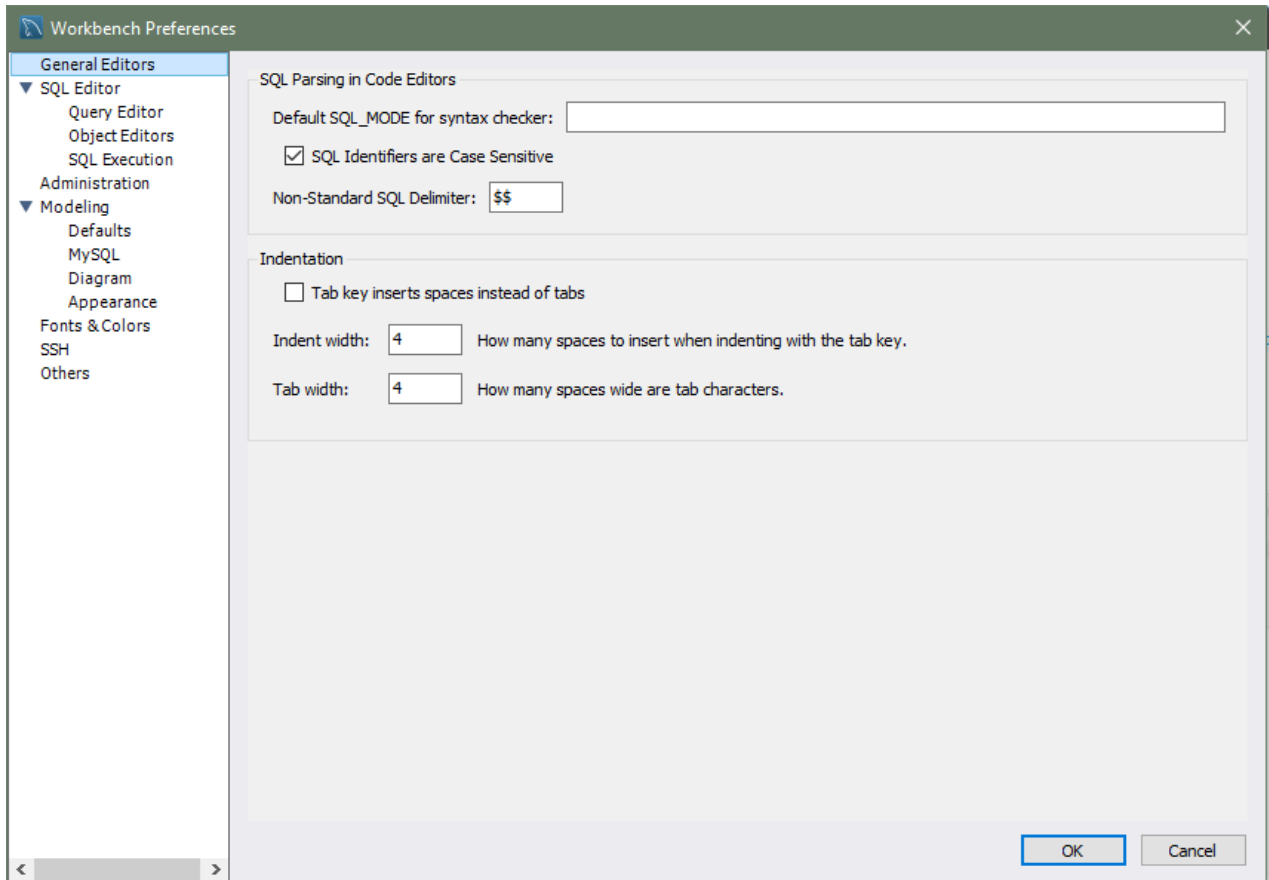
- **Others:** Miscellaneous options.

A more detailed discussion of these options follows.

3.2.1 General Editors Preferences

This section describes the preference options that apply to general-purpose editors (see the figure that follows).

Figure 3.5 Preferences: General Editors



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:** [`blank`]

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:** Enabled by default. Whether to treat identifiers separately if their names differ only in letter case.
- **Non-Standard SQL Delimiter:** `[$$]`. Define the SQL statement delimiter to be different from the normally used delimiter (such as the `;` character). Change this value if the delimiter you normally use, specifically in stored routines, happens to be the current setting.

Indentation

- **Tab key inserts spaces instead of tabs**
- **Indent width:** `[4]` How many spaces to insert when indenting with the tab key.

The number of spaces inserted after pressing the **Tab** key -- this assumes that the *Tab key inserts spaces instead of tabs* option is enabled.

- **Tab width:** `[4]` How many spaces wide are tab characters.

The number of spaces defined for tab characters within MySQL Workbench.

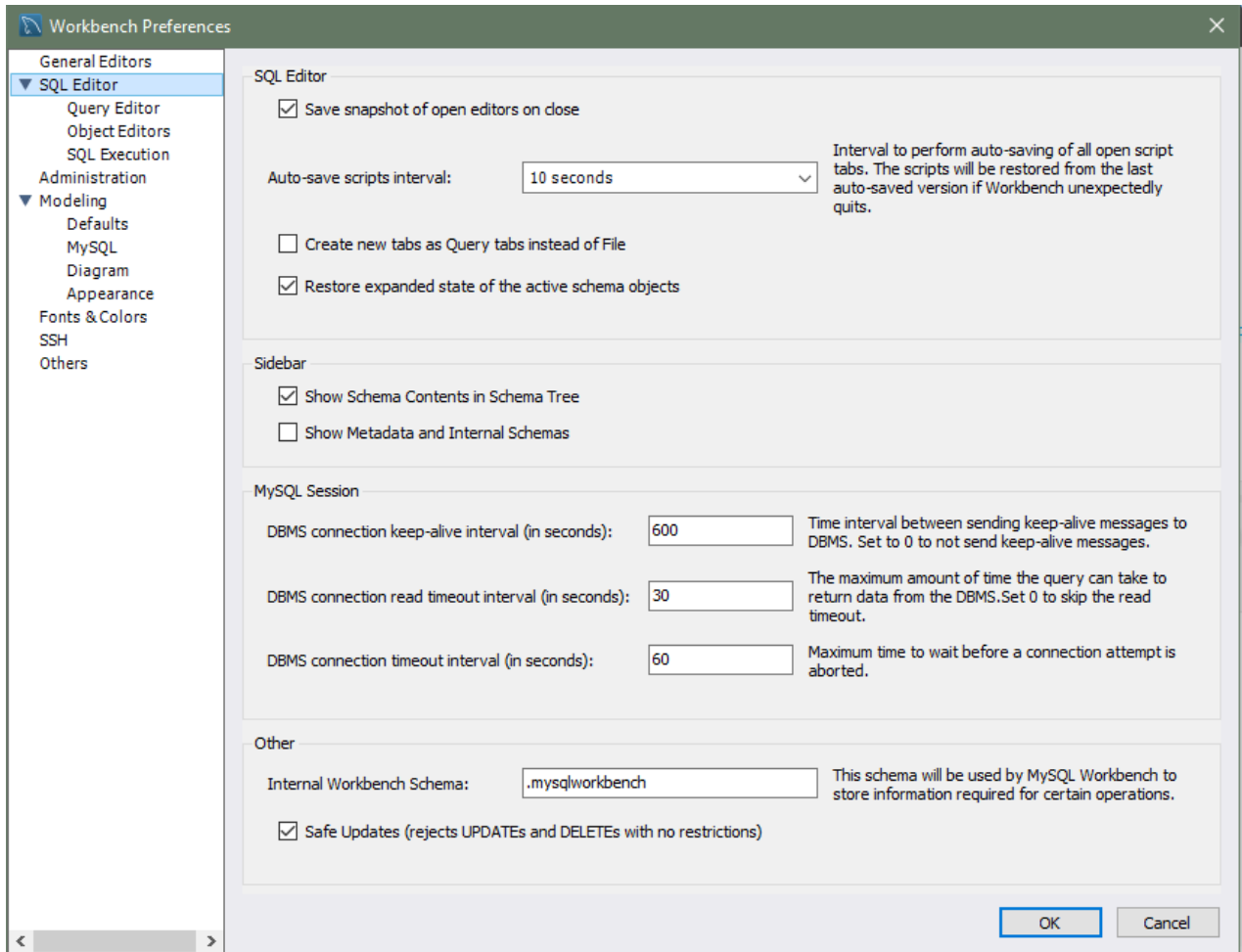
3.2.2 SQL Editor Preferences

This section provides configuration options that affect the SQL editor functionality in MySQL Workbench. As the following figure shows, the SQL editor preferences include both general options and others that apply to specific editors:

- Preferences: SQL Editor: Main
- Preferences: SQL Editor: Query Editor
- Preferences: SQL Editor: Object Editors
- Preferences: SQL Editor: SQL Execution

Preferences: SQL Editor: Main

Figure 3.6 Preferences: Main SQL Editor Section



SQL Editor

- **Save snapshot of open editors on close**

Enabled by default. Permits saving and reloading the SQL editor tabs after MySQL Workbench is closed and reopened (including after an unexpected shutdown).

- **Auto-save scripts interval: [10 seconds]**

Frequency of the automatic saves. Scripts are restored from the last saved version, if MySQL Workbench shuts down.

- **Create new tabs as Query tabs instead of File**

By default, opening a new SQL editor tab opens as an **SQL File** tab. Select this option if you prefer the simpler **Query** tabs that, for example, do not prompt to be saved when closed.

- **Restore expanded state of the active schema objects**

Enabled by default. Group nodes that were previously expanded in the active schema when the SQL editor was last closed are re-expanded and loaded.

Sidebar

- **Show Schema Contents in Schema Tree**

Enabled by default. 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 schemas and tables.

- **Show Metadata and Internal Schemas**

Whether to show metadata and internal schemas in the schema tree, such as `INFORMATION_SCHEMA`, `mysql`, and schemas starting with the period character (`.`).

MySQL Session

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

Time interval between sending keep-alive messages to the DBMS. Set the value to `0` to not send keep-alive messages.

- **DBMS connection read timeout interval (in seconds): [30]**

The maximum amount of time the query can take to return data from the DBMS. Set the value to `0` to skip the read timeout.

- **DBMS connection timeout interval (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 (rejects UPDATEs and DELETEs with no restrictions)**

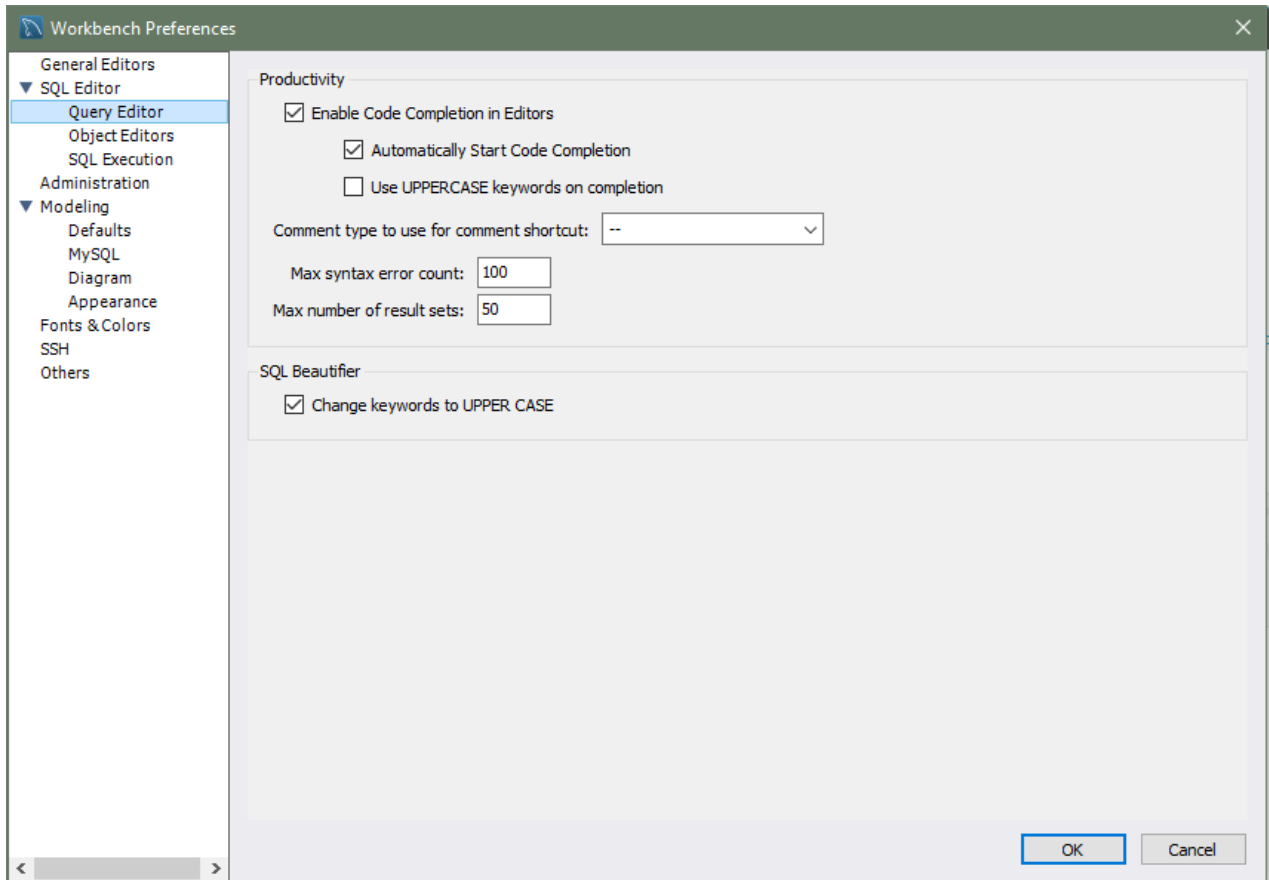
Enabled by default. Prevents `UPDATE` and `DELETE` queries that lack a corresponding key in a `WHERE` clause, or lack a `LIMIT` clause, from executing. This option requires a MySQL server reconnection.

When selected, this preference makes it possible to catch `UPDATE` and `DELETE` statements with keys that are not used properly and that can probably accidentally change or delete a large number of rows.

Preferences: SQL Editor: Query Editor

The following figure shows the preference options that apply to the query editor.

Figure 3.7 Preferences: SQL Editor: Query Editor



Productivity

- **Enable Code Completion in Editors**

The SQL Editor offers autocomplete 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 option automatically executes the code autocomplete feature while editing SQL in the SQL editor. If disabled, you can instead use the keyboard shortcut **Modifier + Space** to execute the autocomplete 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 the -- comment characters, with the # character as an alternative comment option.

- **Max syntax error count: [100]**

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:** [50]

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.

SQL Beautifier

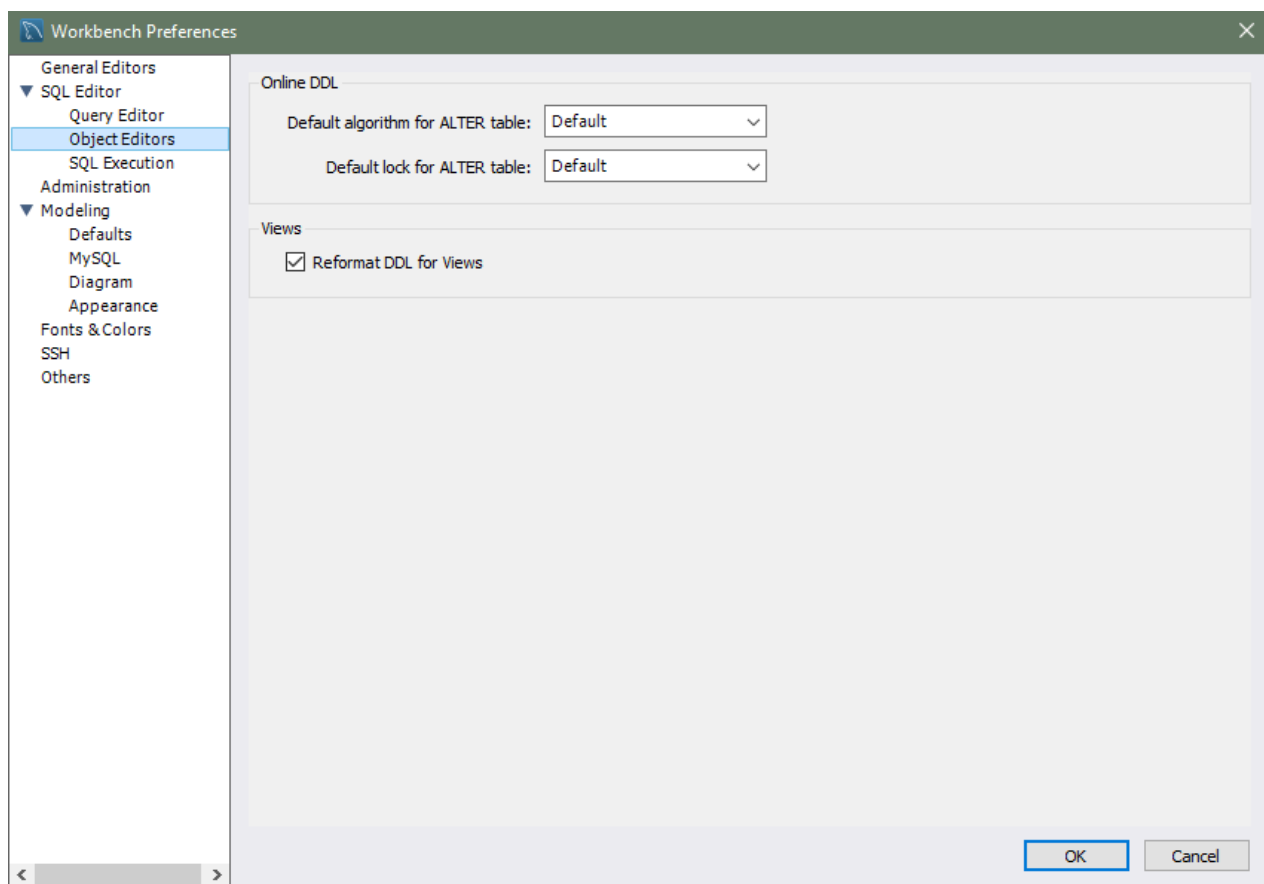
- **Change keywords to UPPER CASE**

Enabled by default. Executing the SQL beautifier sets all SQL keywords to uppercase.

Preferences: SQL Editor: Object Editors

The following figure shows the preference options that apply to all object editors.

Figure 3.8 Preferences: SQL Editor: Object Editors



Online DDL

- **Default algorithm for ALTER table:** [*Default*]

Sets the default algorithm when performing `ALTER TABLE` operations in MySQL Workbench. The setting can also be adjusted for each `ALTER TABLE` operation. Options include `Default`, `In-Place` (preferred), and `Copy`. See the [online DDL](#) documentation for more information.

- **Default lock for ALTER table:** [*Default*]

Sets the default lock setting to allow concurrent queries with `ALTER TABLE` in MySQL Workbench. This setting can also be adjusted for each `ALTER TABLE` operation. Options include `Default`, `None`, `Shared`, and `Exclusive`. See the [online DDL](#) documentation for more information.

Views

- **Reformat DDL for Views**

Enabled by default. Determines whether to automatically reformat the View DDL that is returned by the MySQL server.



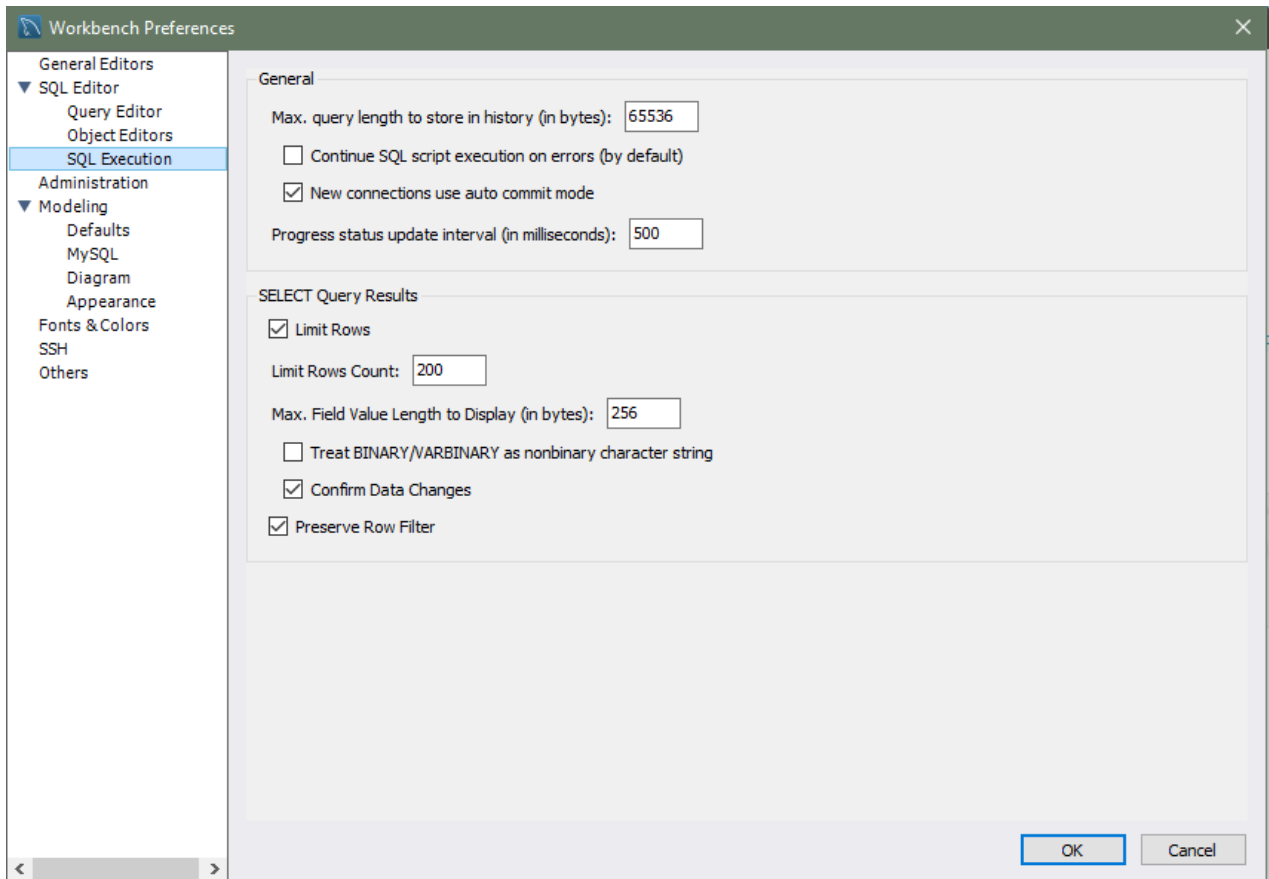
Note

MySQL server instances do not store the formatting information for View definitions.

Preferences: SQL Editor: SQL Execution

The following figure shows the preference options that apply to SQL execution.

Figure 3.9 Preferences: SQL Editor: SQL Execution



General

- **Max. query length to store in history (in bytes):** [*65536*]

Queries that exceed this size are not be saved in the history when executed. Setting this value to 0 eliminates the limit (all queries are saved).

- **Continue on SQL script execution on errors (by default)**

Should an error occur while executing a script, this option causes the execution to continue for the remainder of the script.

- **New connections use auto commit mode**

Enabled by default. Toggles the default autocommit mode for new connections. When enabled, the editor commits each statement 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 (in milliseconds): [500]**

When executing long running queries over a slow connection, you may need to increase this value to prevent excess load on the connection.

SELECT Query Results

- **Limit Rows**

Enabled by default. 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: [200]**

Specify the maximum number of result rows to return.

- **Max. Field Value Length to Display (in bytes): [256]**

To avoid display problems due to excessive field length, it is possible to set the maximum field length to display (in bytes).

- **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**

Enabled by default. If you edit table data in the SQL Editor and then click **Applying changes to data**, MySQL Workbench launches a wizard to step through your changes before applying them. If this option is deselected, the changes are applied to the server without the wizard being displayed and without giving you a chance to review the changes.

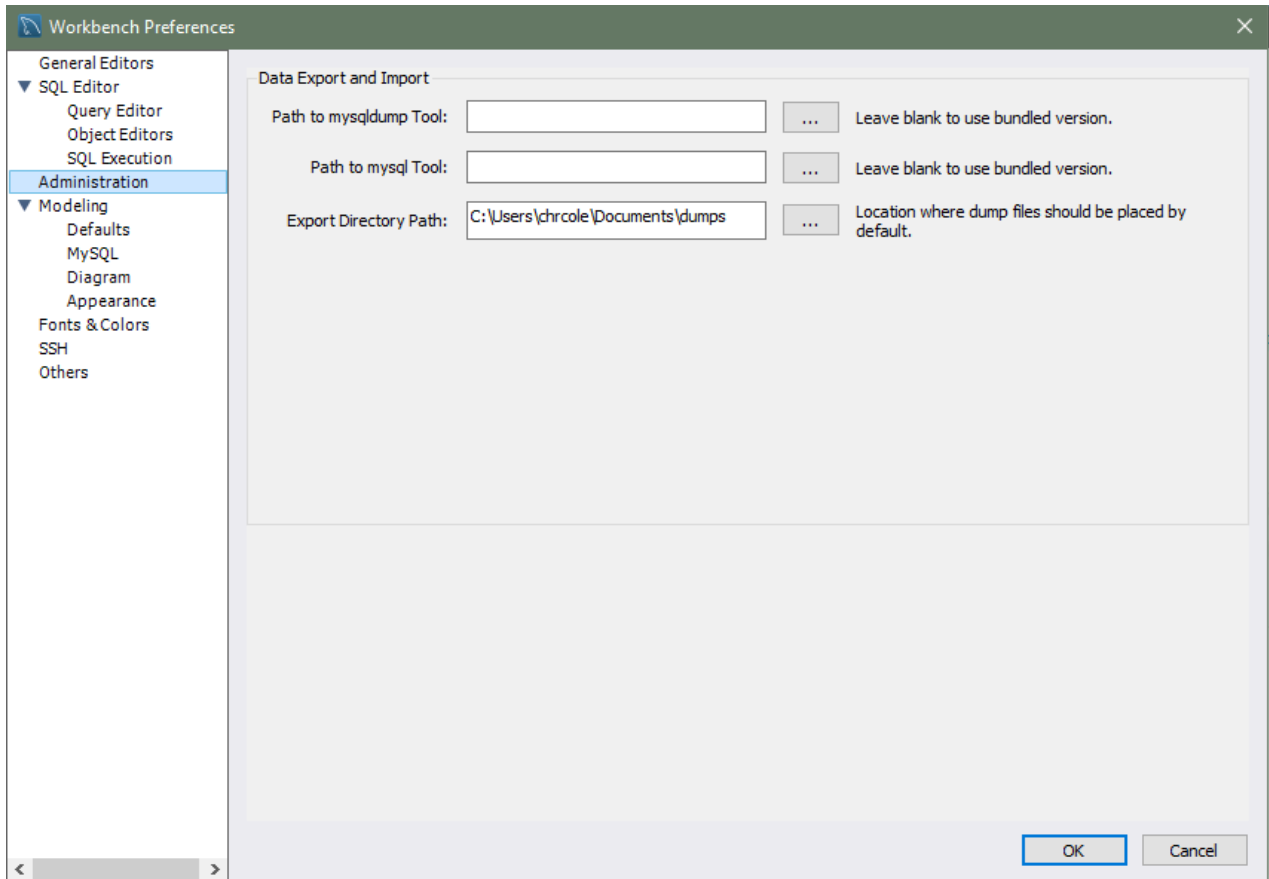
- **Preserve Row Filter**

Enabled by default. This option preserves the active filter on resultset changes. The filter is reset when the option is disabled.

3.2.3 Administration Preferences

This section provides configuration options that affect the administration functionality in MySQL Workbench (see the figure that follows).

Figure 3.10 Preferences: Administration



Data Export and Import

- **Path to mysqldump Tool:** [*blank*]

Path to your local `mysqldump` binary. Leave it blank to use the bundled `mysqldump` binary.

- **Path to mysql Tool:** [*blank*]

Path to your local `mysql` client binary. Leave it blank to use the bundled `mysql` binary.

- **Export Directory Path:** [*default directory*]

Directory where your exported `mysql` dumps are located.

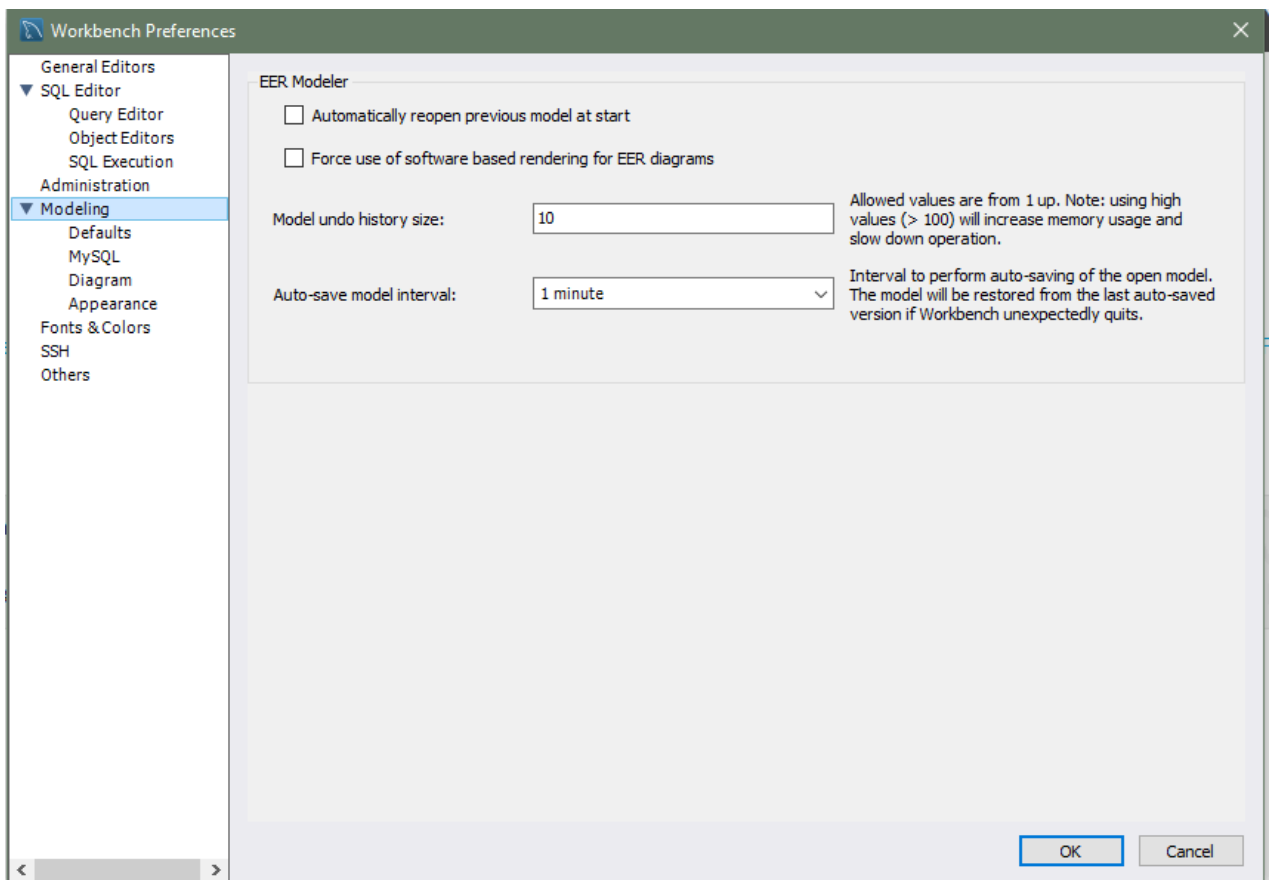
3.2.4 Modeling Preferences

This section provides configuration options that affect the modeling functionality in MySQL Workbench (see the figure that follows). Modeling preferences include specific categories of options:

- [Preferences: Modeling: Main](#)
- [Preferences: Modeling: Defaults](#)
- [Preferences: Modeling: MySQL](#)
- [Preferences: Modeling: Diagram](#)
- [Preferences: Modeling: Appearance](#)

Preferences: Modeling: Main

Figure 3.11 Preferences: Modeling



EER Modeler

- **Automatically reopen previous model at start**

Select this check box 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 uses 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:** [10]

You can limit the size of the undo history here. Set this value to 0 to have an unlimited undo history. Defaults to 10 undo history operations.

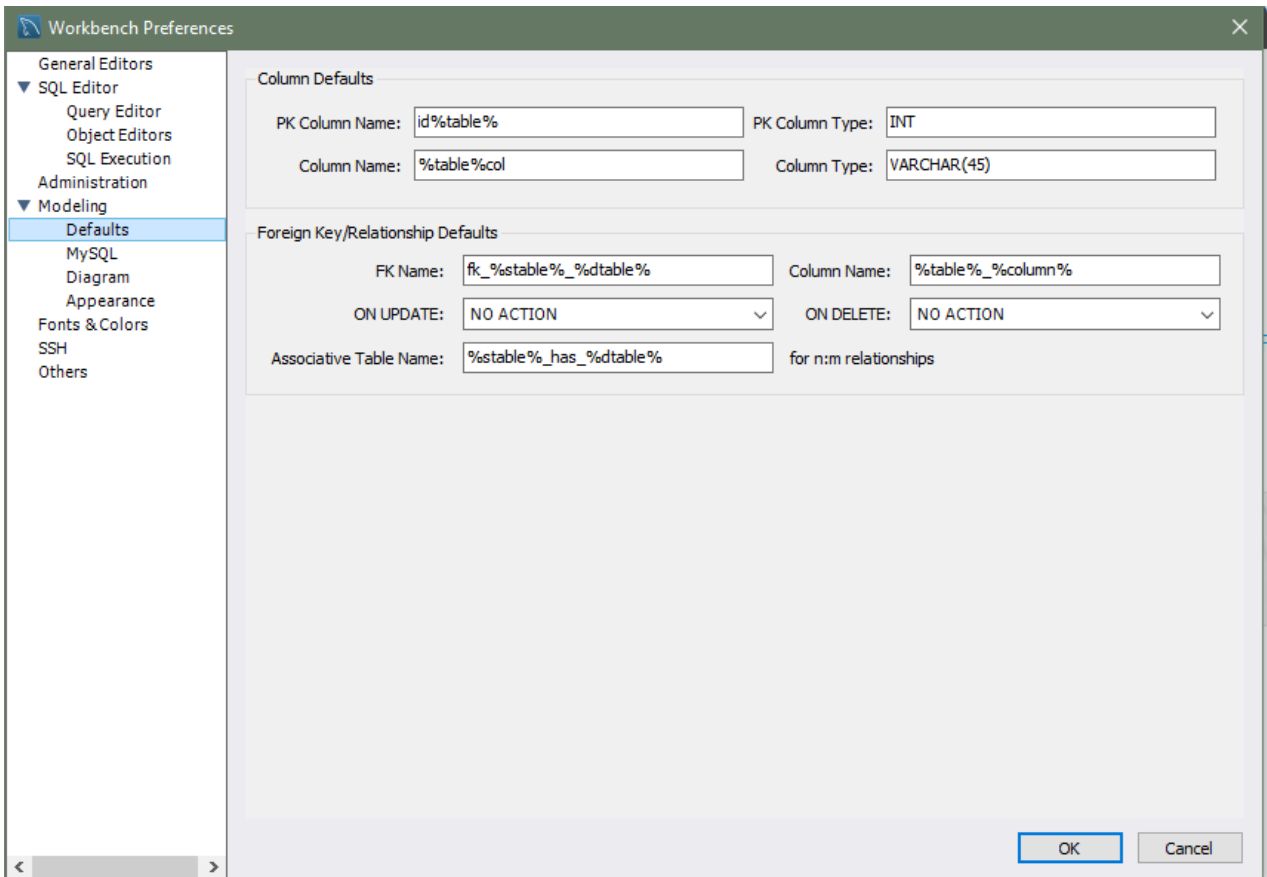
- **Auto-save model interval:** [1 minute]

An open model that has not been saved will automatically be saved after this period. On loading a model file, MySQL Workbench notifies you if the file was not previously saved correctly (possibly due to an unexpected shutdown or power failure). MySQL Workbench can then attempt to recover the last autosaved version. For automatic recovery to be available on a new file, save the file at least one time.

Preferences: Modeling: Defaults

Sets default values for modeling object names (see the figure that follows).

Figure 3.12 Preferences: Modeling: Defaults



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)

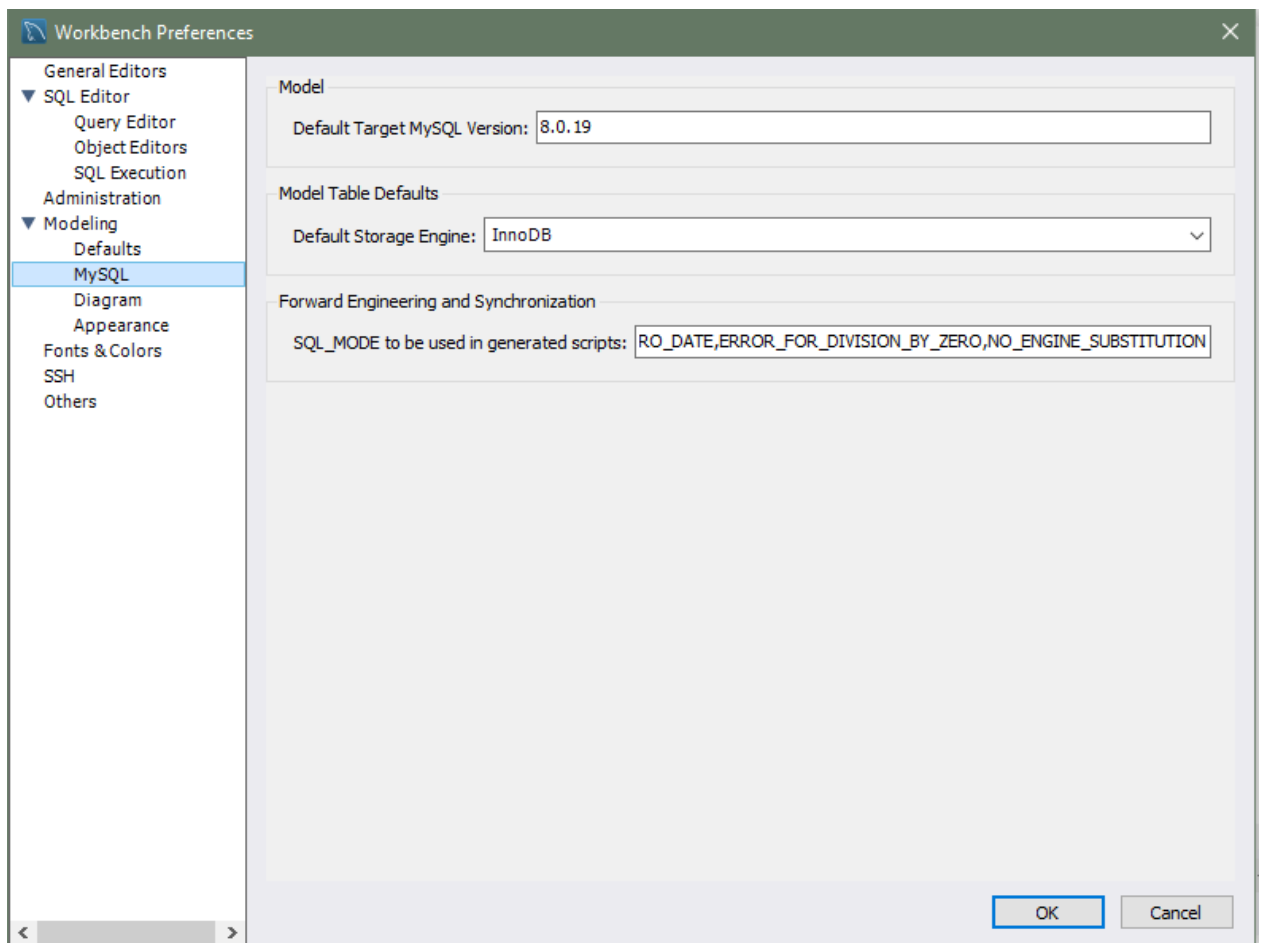
Foreign Key/Relationship Defaults

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 preference group enables you to set model-related options specific to your MySQL version (see the figure that follows).

Figure 3.13 Preferences: Modeling: MySQL



Model

- **Default Target MySQL Version:** [*version*]

A limited subset of validation procedures and table editor options are affected by this MySQL version number. Specify the version number in either *MAJOR.MINOR* (8.0) or *MAJOR.MINOR.RELEASE* (8.0.38) format.

Model Table Defaults

- **Default Storage Engine:**

Tables created in MySQL Workbench are defined using this default storage engine. Values include: *InnoDB*, *MyISAM*, *ndbcluster*, *MEMORY*, *FEDERATED*, *ARCHIVE*, *CSV*, *BLACKHOLE*, and *MRG_MyISAM*.

Forward Engineering and Synchronization

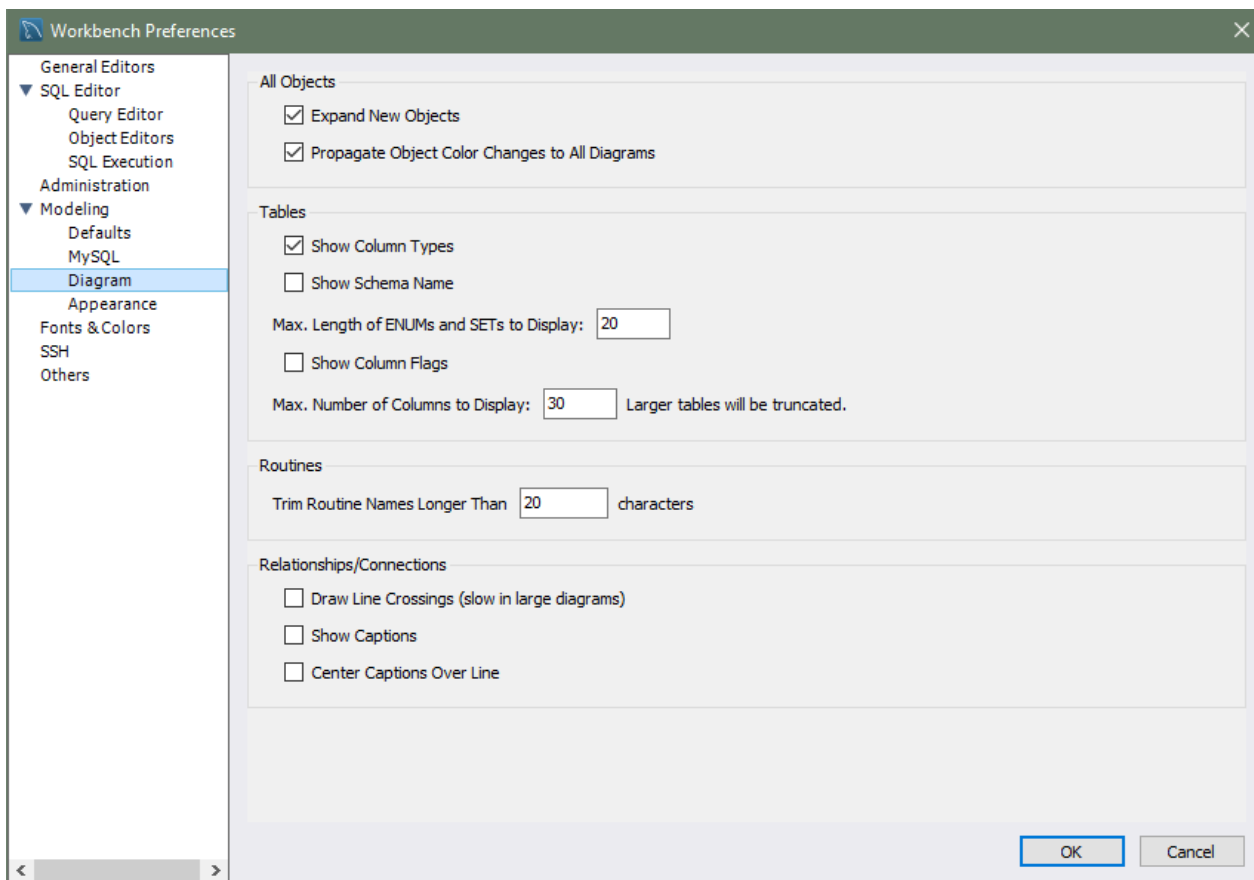
- **SQL_MODE to be used in generated scripts:**

Defines the *SQL_MODE* used by Forward Engineering and Synchronization. Defaults to *ONLY_FULL_GROUP_BY*, *STRICT_TRANS_TABLES*, *NO_ZERO_IN_DATE*, *NO_ZERO_DATE*, *ERROR_FOR_DIVISION*

Preferences: Modeling: Diagram

The following figure shows the preference options that apply to model-related diagrams.

Figure 3.14 Preferences: Modeling: Diagram



All Objects

- **Expand New Objects**

Enabled by default. Sets the initial state of newly created objects to expanded (or collapsed, if disabled).

- **Propagate Object Color Changes to All Diagrams**

Enabled by default. If an object's [Figure](#) color is changed, all figures in all diagrams that represent the same object are also updated.

Tables

- **Show Column Types**

Enabled by default. Shows the column types along their names in table figures.

- **Show Schema Name**

Shows the owning schema name in the title bar of table figures.

- **Max. Length of ENUMs and SETs to Display:** [\[20\]](#)

- **Show Column Flags**

Shows column flags, such as NOT NULL and UNSIGNED, along their names in table figures.

- **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**

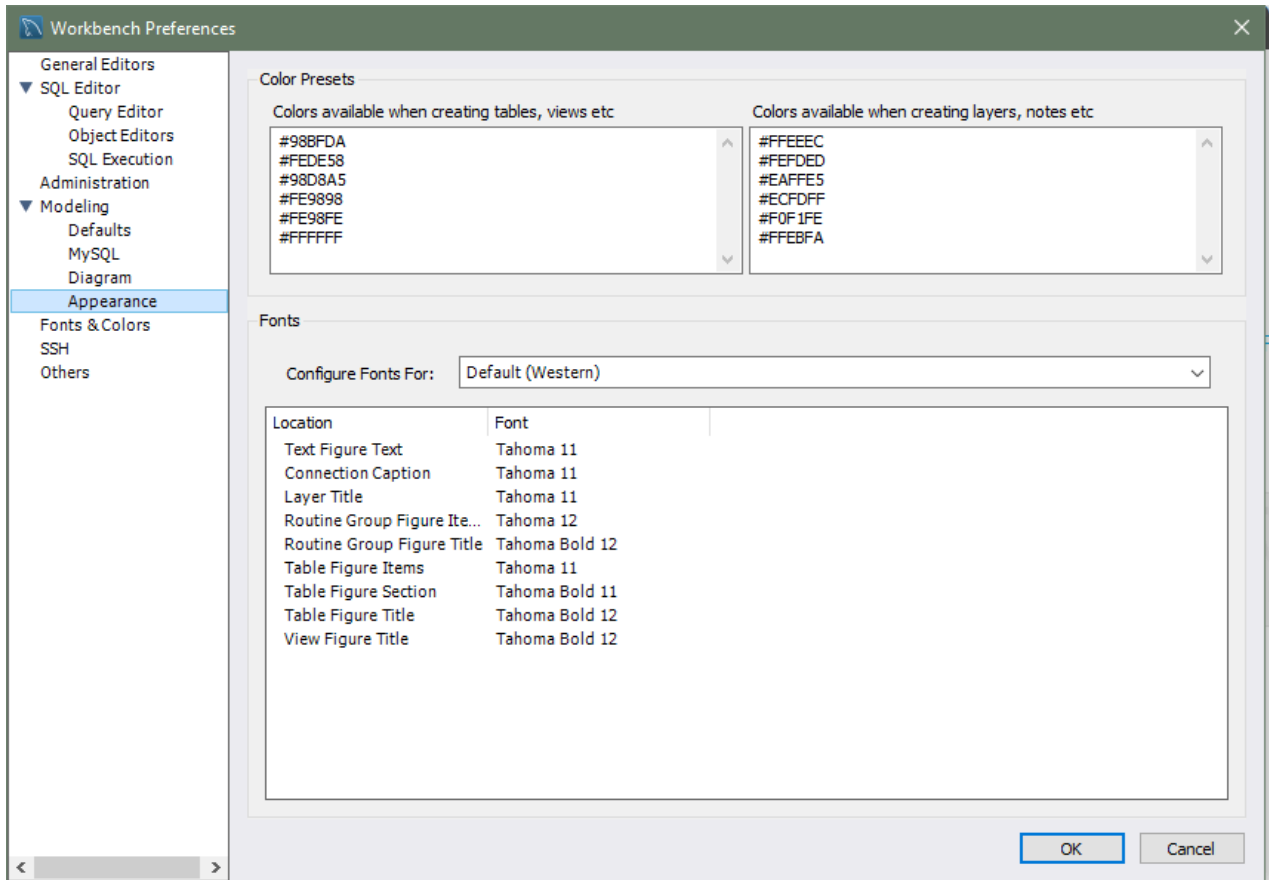
- **Center Captions Over Line**

Preferences: Modeling: Appearance

Use this preference group to set the available colors for the objects that appear on an EER diagram canvas. As the following figure shows, you can also add colors as needed.

For related information, see [Section 3.1, "User Accessibility Options"](#).

Figure 3.15 Preferences: Modeling: Appearance



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.

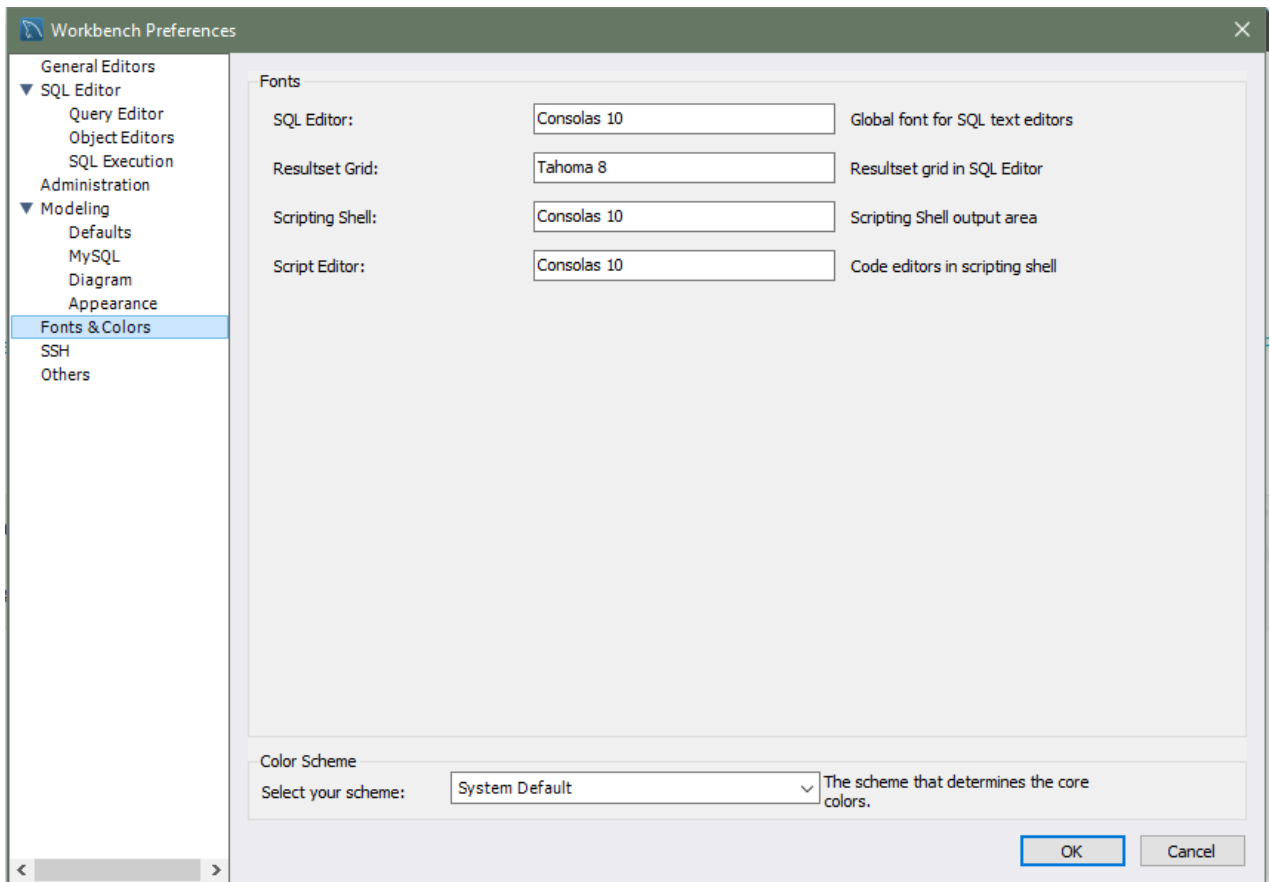
Fonts

These define the fonts and font sizes used while modeling.

3.2.5 Fonts and Colors Preferences

The following figure shows the preference options that apply to fonts and colors.

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.

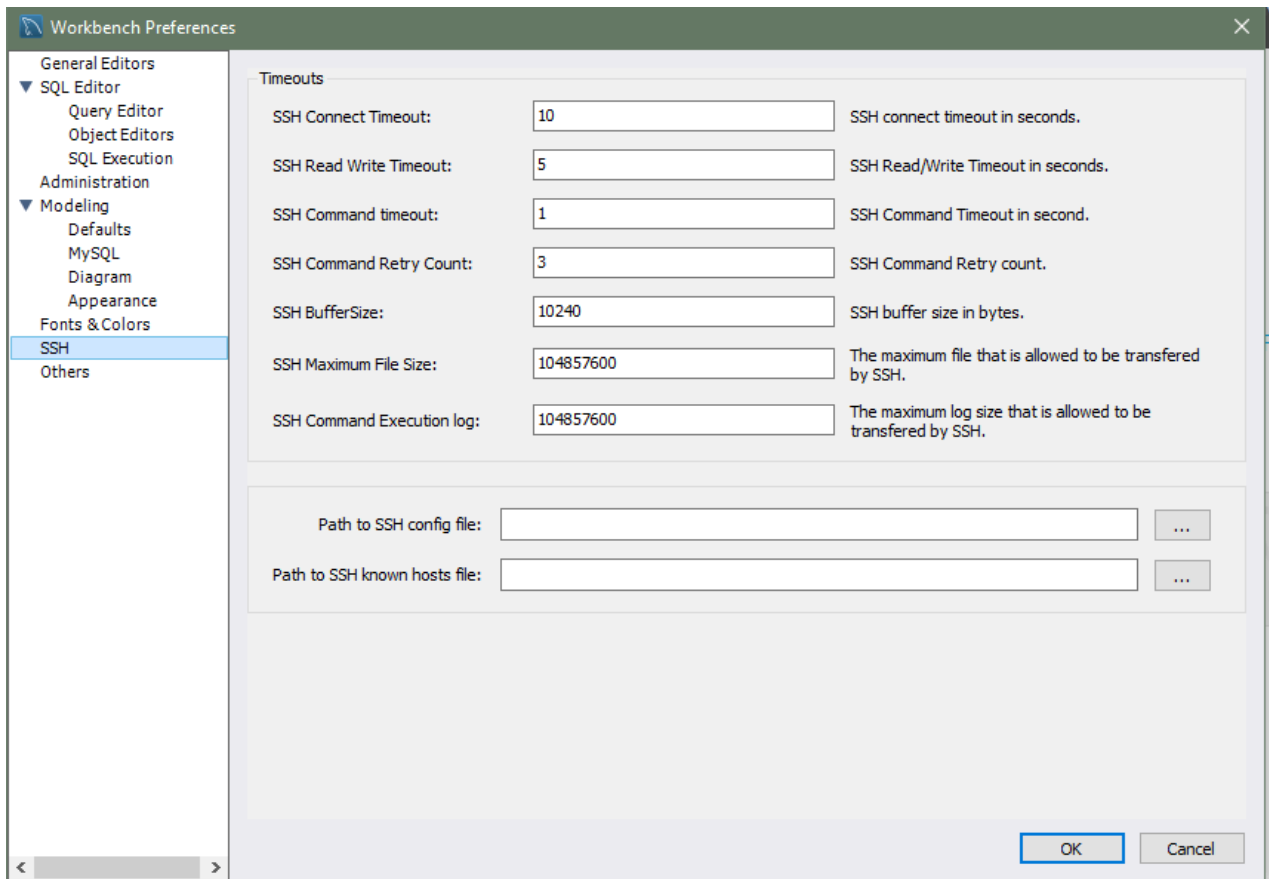
Color Scheme

The scheme that determines the core colors. On Microsoft Windows, select the scheme that determines the user-interface colors.

3.2.6 SSH Preferences

The following figure shows the preference options that apply to SSH items, such as timeouts.

Figure 3.17 Preferences: SSH



Timeouts

- **SSH Connect Timeout:** [10]
SSH connect timeout interval in seconds.
- **SSH Read Write Timeout:** [5]
SSH read and write timeout interval in seconds.
- **SSH Command timeout:** [1]
SSH command timeout interval in seconds.
- **SSH Command Retry Count:** [3]
SSH command retry count.
- **SSH BufferSize:** [10240]
SSH buffer size in bytes.
- **SSH Maximum File Size:** [104857600]
The maximum file size that is allowed to be transferred by SSH.
- **SSH Command Execution log:** [104857600]

The maximum file size that is allowed to be transferred by SSH.

Use the following options to set SSH file paths:

- **Path to SSH config file:**

Click browse to select a configuration file.

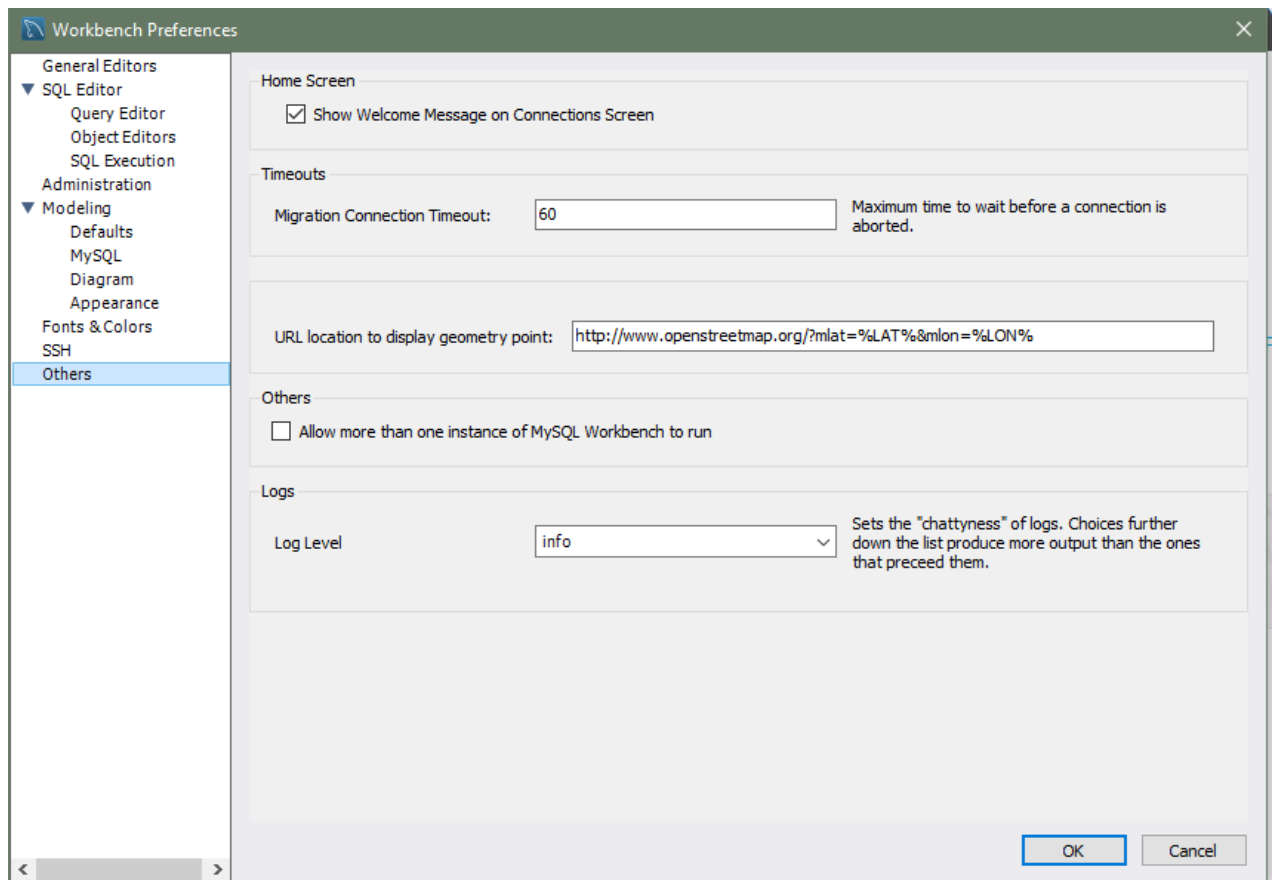
- **Path to SSH known hosts file:**

Click browse to select a hosts file.

3.2.7 Other Preferences

The following figure shows the preference options that apply to general items, such as the home screen and logs.

Figure 3.18 Preferences: Others



Home Screen

- **Show Welcome Message on Connections Screen**

Enabled by default. This option displays or hides the text and links of the welcome message when **MySQL Connections** is selected from the side panel and MySQL Workbench is restarted.

Timeouts

- **Migration Connection Timeout:** [60]

Maximum time to wait before a connection is aborted.

To set an alternative online service:

- **URL location to display geometry point:** [<http://www.openstreetmap.org/?mlat=%LAT%&mlon=%LON%>]

Replace the default value with an alternative URL. For example, to use Google Maps rather than OpenStreetMap, enter [https://www.google.com/maps/?q=%LAT% , %LON%](https://www.google.com/maps/?q=%LAT%,%LON%).

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. This setting is available only on Windows.



Note

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

Logs

- **Log Level:** [info]

Determines how serious a message has to be before it gets logged. For example, an error is more serious than a warning, a warning is more serious than an informational message, and so on. From lowest to highest, the levels are: `none`, `error`, `warning`, `info`, `debug1`, `debug2`, and `debug3`. Defaults to the `info` level.

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 a directory assigned to the current user as defined by MySQL Workbench. The following table shows the default file path by platform.

Table 3.1 Default Local Configuration Base File Path

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

The next table provides a brief description of these directories and files.

Table 3.2 Local MySQL Workbench Files and Directory Descriptions

Directory or File	Description
cache/	General behaviors are stored per-connection in <code>*.cache</code> files, and column widths as <code>*.column_widths</code> files.
log/	Log files include MySQL Workbench startup information, and also per-connection SQL action results performed in MySQL Workbench.

Directory or File	Description
scripts/, modules/, and libraries/	Saved user scripts, modules, and libraries that enables you to extend MySQL Workbench capabilities.
sql_history/	Queries executed in MySQL Workbench are stored here, and are available from within MySQL Workbench.
sql_workspaces/	Configuration details, such as the tab order or schema tree, are stored here by connection instance.
snippets/	Saved SQL snippets are stored here. For additional information, see Section 8.1.5, "SQL Additions - 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 tab . 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.
wb_state.xml	Stores the previous user-interface state. You can delete this file if you encounter a problem with the MySQL Workbench user interface.

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.

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, select **Show Log Files** from the **Help** menu.

- **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 rotate 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.

scripts/, modules/, and Libraries/ Directories

Custom user script, module, and library files are stored in the `scripts`, `modules`, and `libraries` directories. These user files are accessible from the file browser in the Workbench Scripting Shell. For additional information about user scripts, see [Section C.5, "The Workbench Scripting Shell"](#). For more information about user modules, see [Section C.2, "Modules"](#) and [Section C.3, "Plugins and Tools"](#).

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 Panel"](#).

sql_workspaces/ Directory

Workspace information is saved to the `sql_workspaces` directory by connection automatically. MySQL Workbench generates a subdirectory that persists between work sessions for each connection tab that you open. It uses the name of the connection, appended with a sequential number, to represent the order of each connection tab. For example, opening two connection tabs for a connection named `Local instance MySQL80` creates two subdirectories: `local_instance_MySQL80-1.autosave` and `local_instance_MySQL80-2.autosave`. The file extension changes from `.autosave` to `.workspace` when you close the connection tab.

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 Additions - Snippets Tab"](#).

3.4 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), MySQL Workbench will not execute `UPDATE` or `DELETE` statements if a key is not defined in the `WHERE` clause. In other words, MySQL 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, MySQL Workbench saves all query tabs and reopens them when you restart it. 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`.

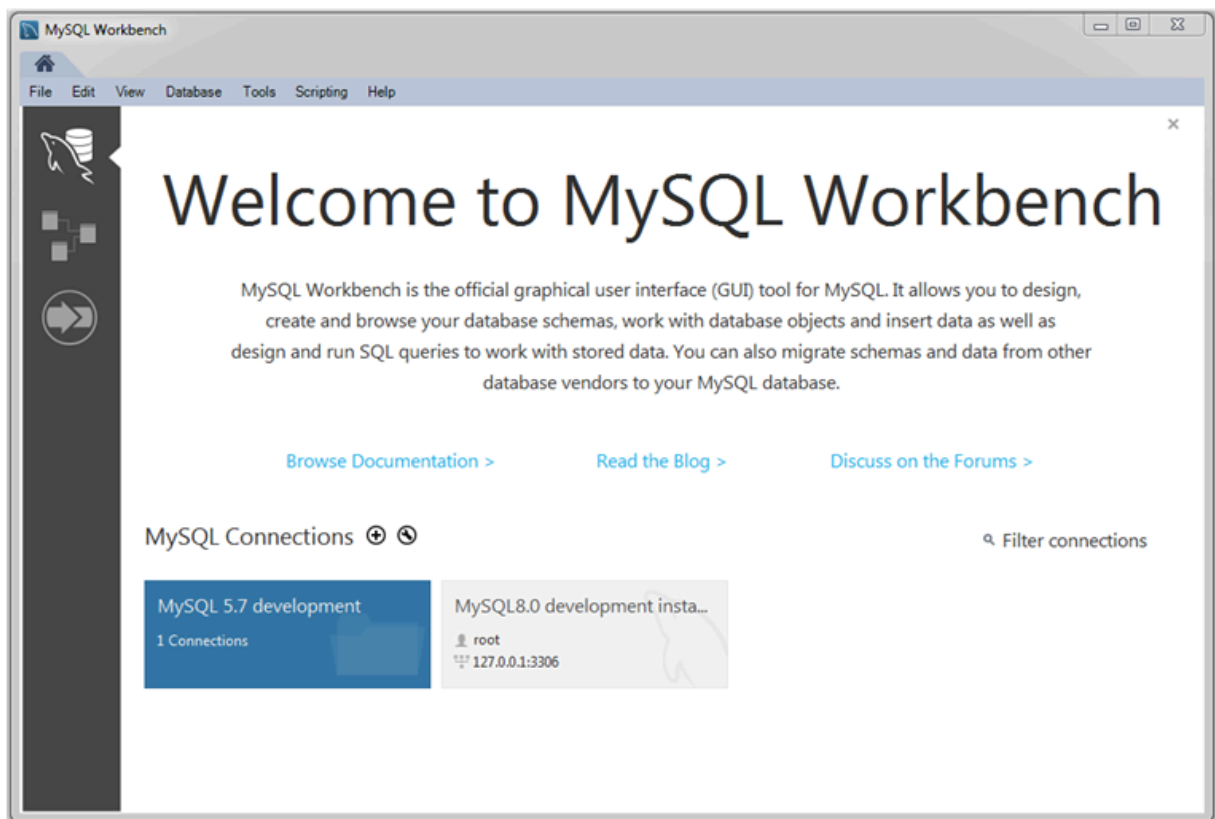
Related behavior: The **Context Help** right panel in the SQL editor displays documentation for SQL statements, and is disabled/enabled from the right panel. For example, typing `INSERT` will load documentation for the `INSERT` statement in the right panel.

Chapter 4 Home Screen Tab

When started, MySQL Workbench opens to the home screen tab. Initially, the screen displays a welcome message and links to **Browse Documentation >**, **Read the Blog >**, and **Discuss on the Forums >**. In addition, the home screen provides quick access to MySQL connections, models, and the MySQL Workbench Migration Wizard.

As depicted in the following figure, a home-screen side panel enables you to toggle between MySQL Connections (selected in the figure) and Models within the home tab. The last option in the side panel opens the MySQL Workbench Migration Wizard in a new tab.

Figure 4.1 Home Screen Tab



Note


In MySQL Workbench 6.3.8 and earlier, the home screen combined three sections titled **MySQL Connections**, **Models**, and external **Shortcuts**.

Workbench Welcome Message. The welcome text and links are optional and can be removed or restored to the home screen when MySQL Connections is selected from the side panel. Links to the documentation, blog, and forums are also accessible from the **Help** menu.

- To remove the text and links, click the **X** icon above the welcome message.
- To restore the text and links:
 1. Click **Edit, Preferences**, and then select **Other**.

2. Select the **Show Welcome Message on Connections Screen** check box and then restart MySQL Workbench.


MySQL Connections

The connections view () in the side panel, when selected, displays a list of established connections to local and remote instances of MySQL. It enables you to load, configure, group, and view information about each MySQL connection. For more information, see [Chapter 5, *Connections in MySQL Workbench*](#) and [Section 5.2, “Creating A New MySQL Connection \(Tutorial\)”](#).

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.

Connection Information. Click **Manage Connections** from the **Database** menu (or **Edit Connection** from the context menu of each connection) to view connection details.

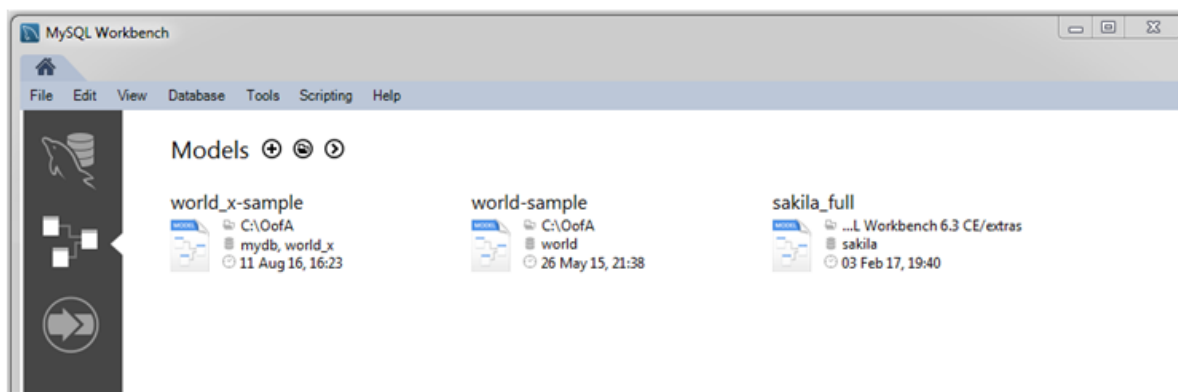
Models

The models view () in the side panel, when selected, displays your most recently used models. As the next figure shows, each entry lists the date and time that the model was last opened and shows its associated database. To the right of the **Models** title are the following options:


- Plus-sign button (+) adds a new model.
- Folder button (see the figure that follows) enables you to browse for and open saved models.
- More button (>) opens a context menu with additional commands, such as **Create EER Model from Database**.

For additional information about modeling, see [Chapter 9, *Database Design and Modeling*](#).

Figure 4.2 Selecting Models from the Side Panel



MySQL Migration Wizard

The migration view () in the side panel, when selected, opens the **Migration** tab and displays an overview of prerequisites for using the wizard. From the **Migration** tab, you can start a migration process,

open the ODBC administrator, or view documentation. For detailed instructions on using the wizard, see [Chapter 10, *Database Migration Wizard*](#).

Chapter 5 Connections in MySQL Workbench

Table of Contents

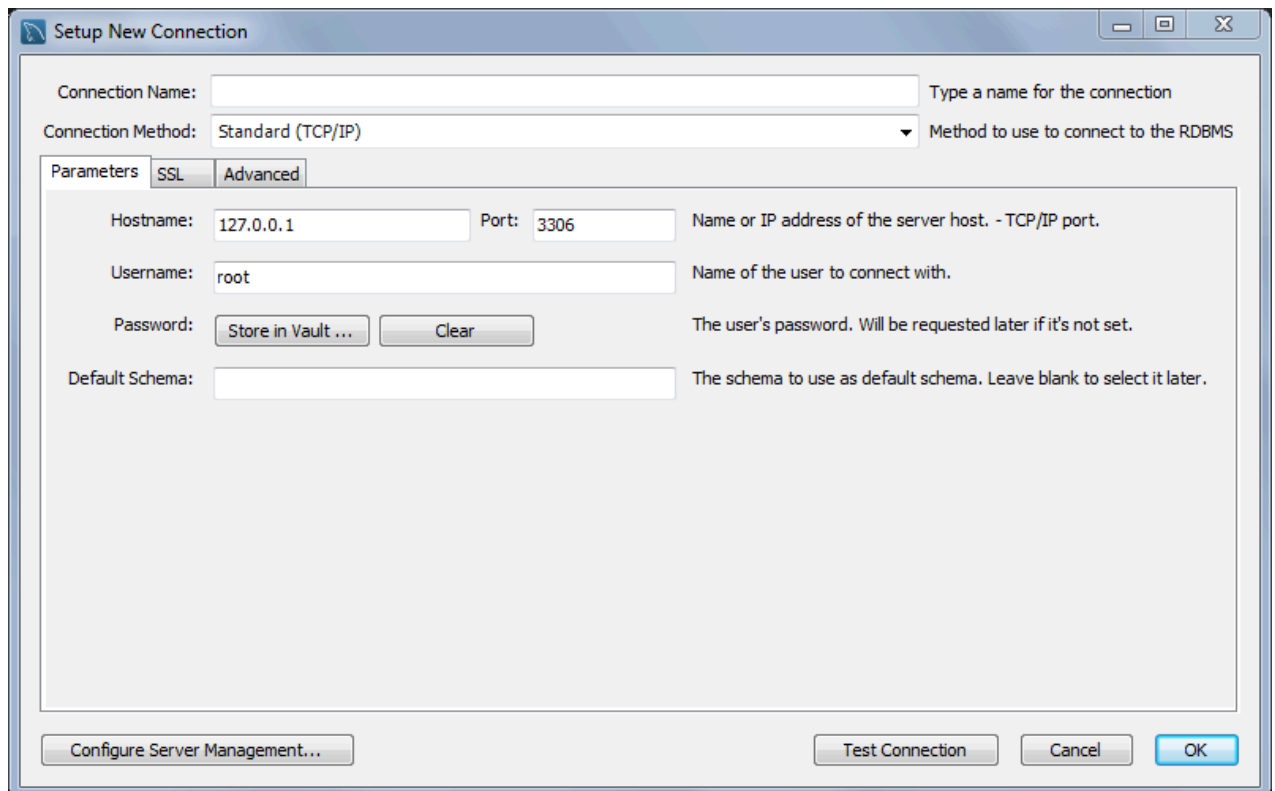
5.1 Creating A New MySQL Connection (Simple)	99
5.2 Creating A New MySQL Connection (Tutorial)	100
5.3 Manage Server Connections	111
5.3.1 Standard TCP/IP Connection Method	113
5.3.2 Local Socket/Pipe Connection Method	115
5.3.3 Standard TCP/IP over SSH Connection Method	116
5.3.4 LDAP and Kerberos Connection Methods	118
5.3.5 SSL Wizard (Certificates)	123
5.3.6 Remote Management	127
5.3.7 System Profile	128
5.3.8 Configure Server Management Wizard	129
5.3.9 The Password Storage Vault	132
5.3.10 Updating Old Authentication Protocol Passwords	132
5.4 Client Connections	136

This chapter describes how to create and manage 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, as the following figure shows.

Figure 5.1 Setup New Connection Form





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 5.3.8, “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 a connection, 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 \(Tutorial\)”](#).

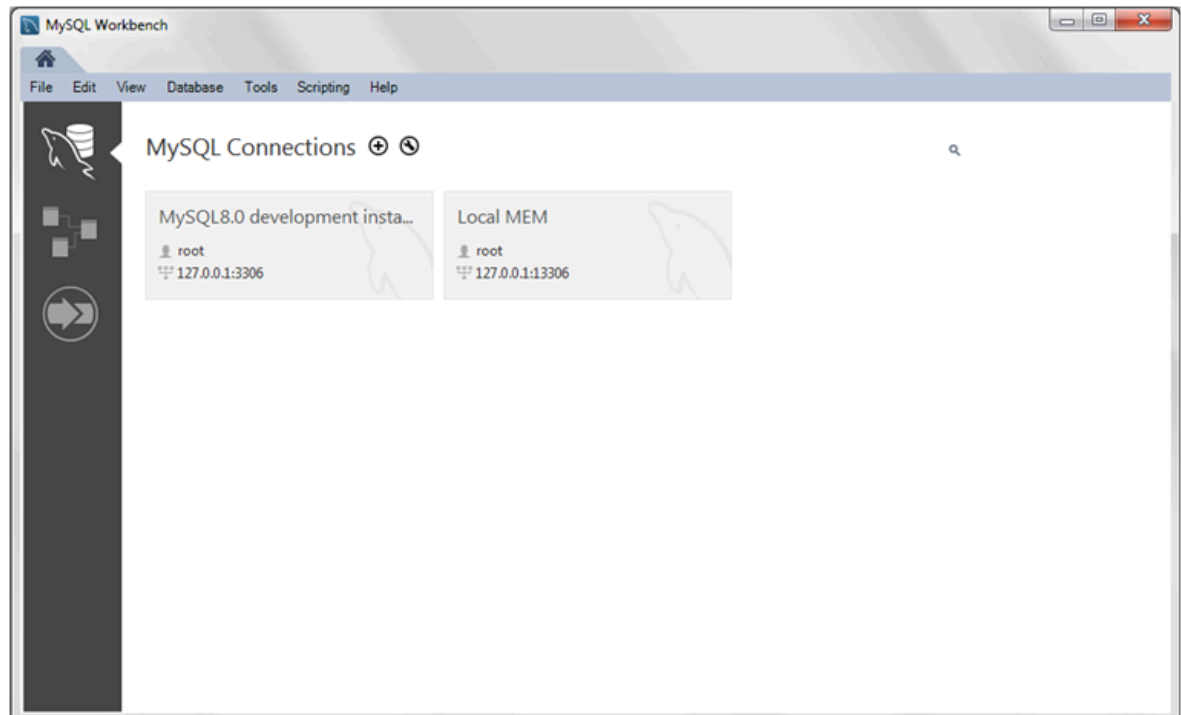
5.2 Creating A New MySQL Connection (Tutorial)

This tutorial adds a new connection that can be either an initial connection or an additional connection. An instance of MySQL server must be installed, started, and accessible to MySQL Workbench before you begin.

To create a new connection, follow these steps:

1. Launch MySQL Workbench to open the home screen. Existing connections are shown when you click the **MySQL Connections** view from the sidebar. No connections exist for first-time users.

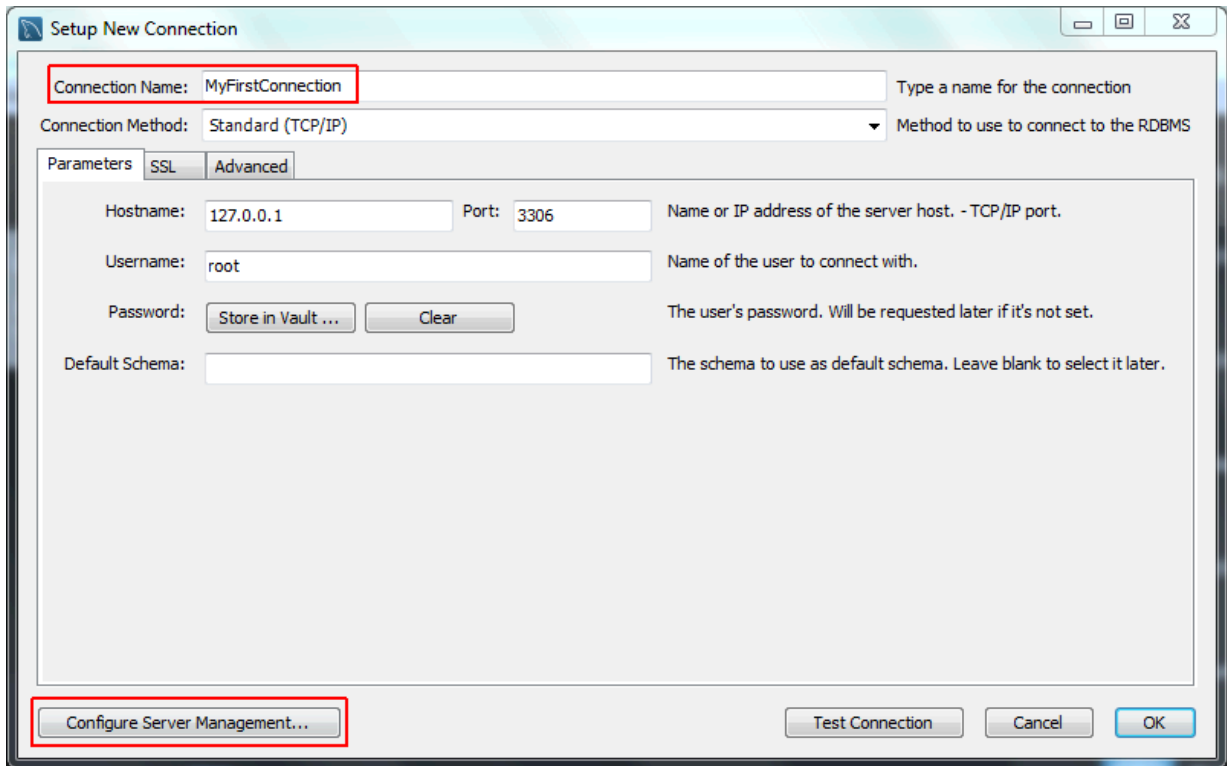
Figure 5.2 Getting Started Tutorial - Home Screen



2. From the MySQL Workbench home screen shown in the previous figure, click the **[+]** icon near the **MySQL Connections** label to open the **Setup New Connection** wizard.

3. Define the **Connection Name** value, such as `MyFirstConnection` as the next figure shows.

Figure 5.3 Getting Started Tutorial - Setup New Connection: MyFirstConnection

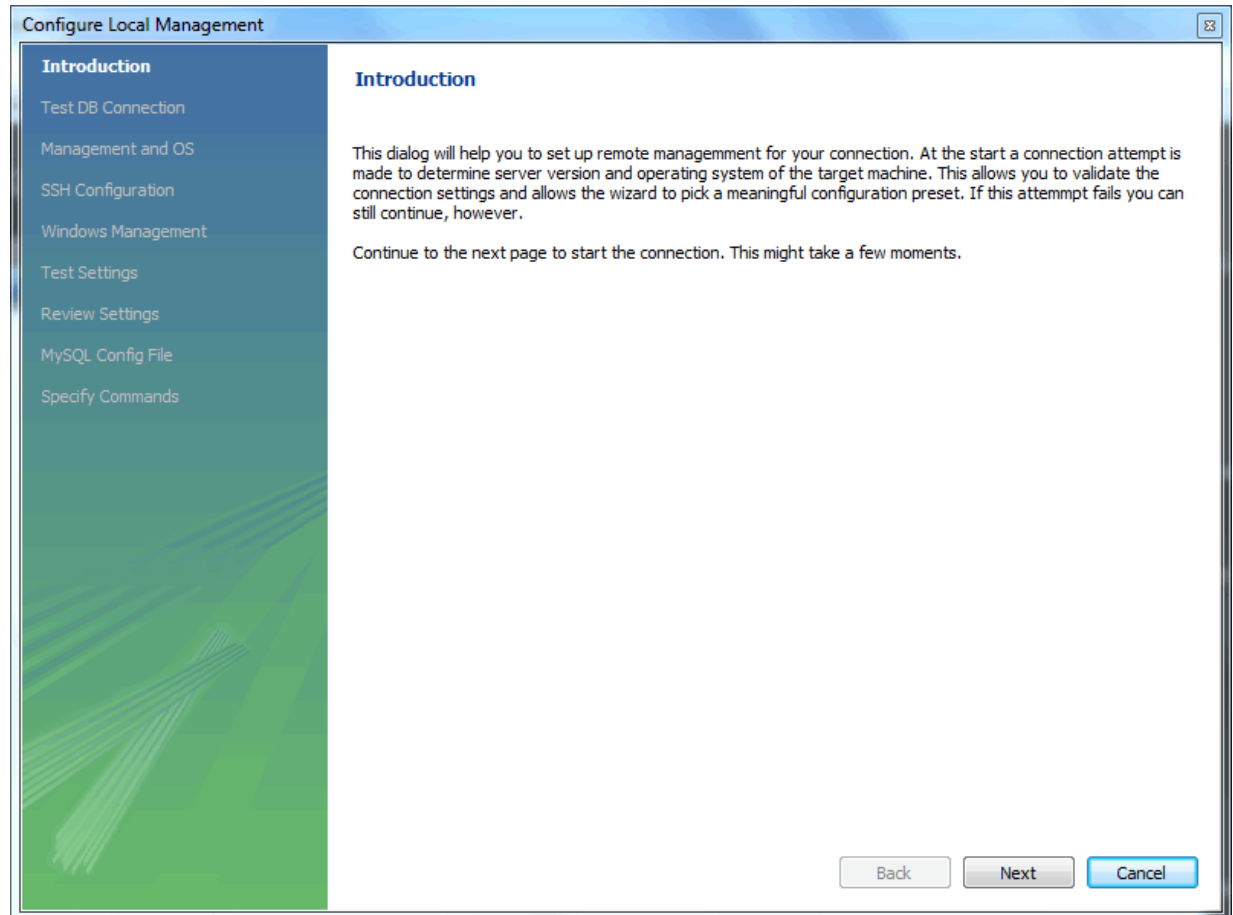


The default connection values are for a typical local setup, so check them and enter the appropriate values. If you are unsure, click the **Test Connection** button to check the connection parameters. Do not press **OK**.

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

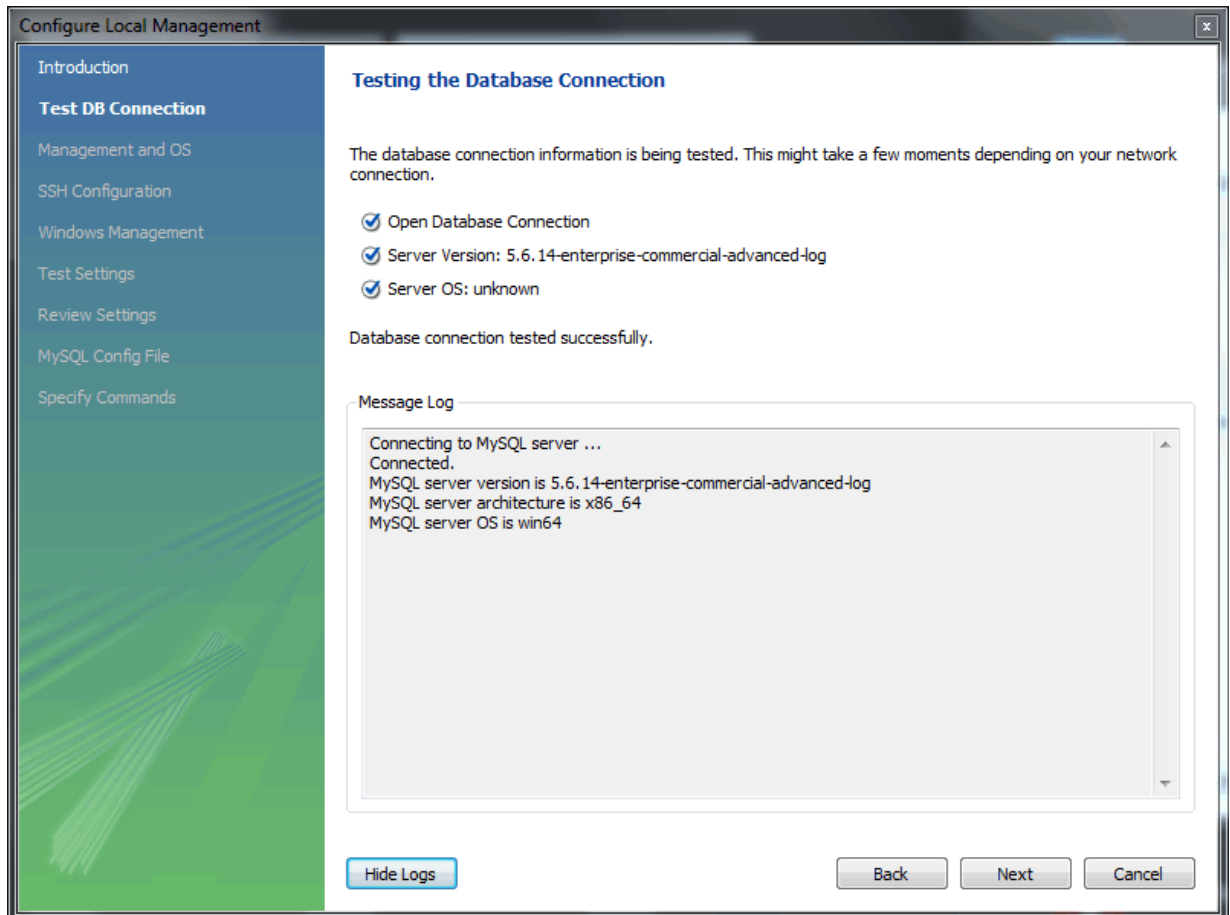
4. Read the **Configure Local Management** introduction (shown in the next figure), and press **Next** to begin defining the new connection parameters.

Figure 5.4 Getting Started Tutorial - Configure Local Management Introduction



- 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. The following figure shows a database connection that tested successfully.

Figure 5.5 Getting Started Tutorial - Test Database Connection



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

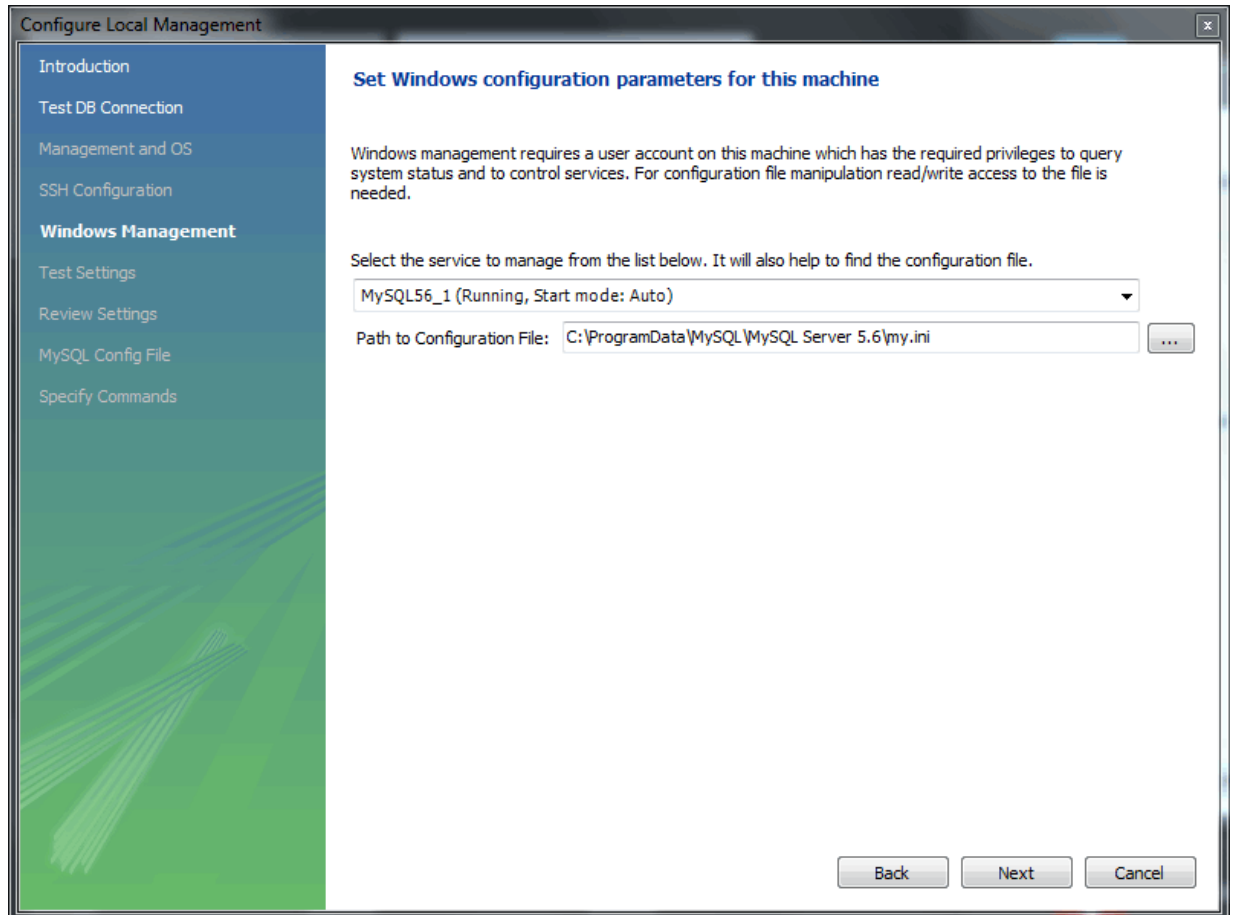
- Optionally, you may configure a method for remote management if a Remote Host was specified. Setting these options enables MySQL Workbench to determine the location of configuration files, and the correct start and stop commands to use for the connection.

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.

This step creates a local MySQL connection, so you can skip the **Management and OS** and **SSH Configuration** options, which are used for configuring a remote MySQL connection.

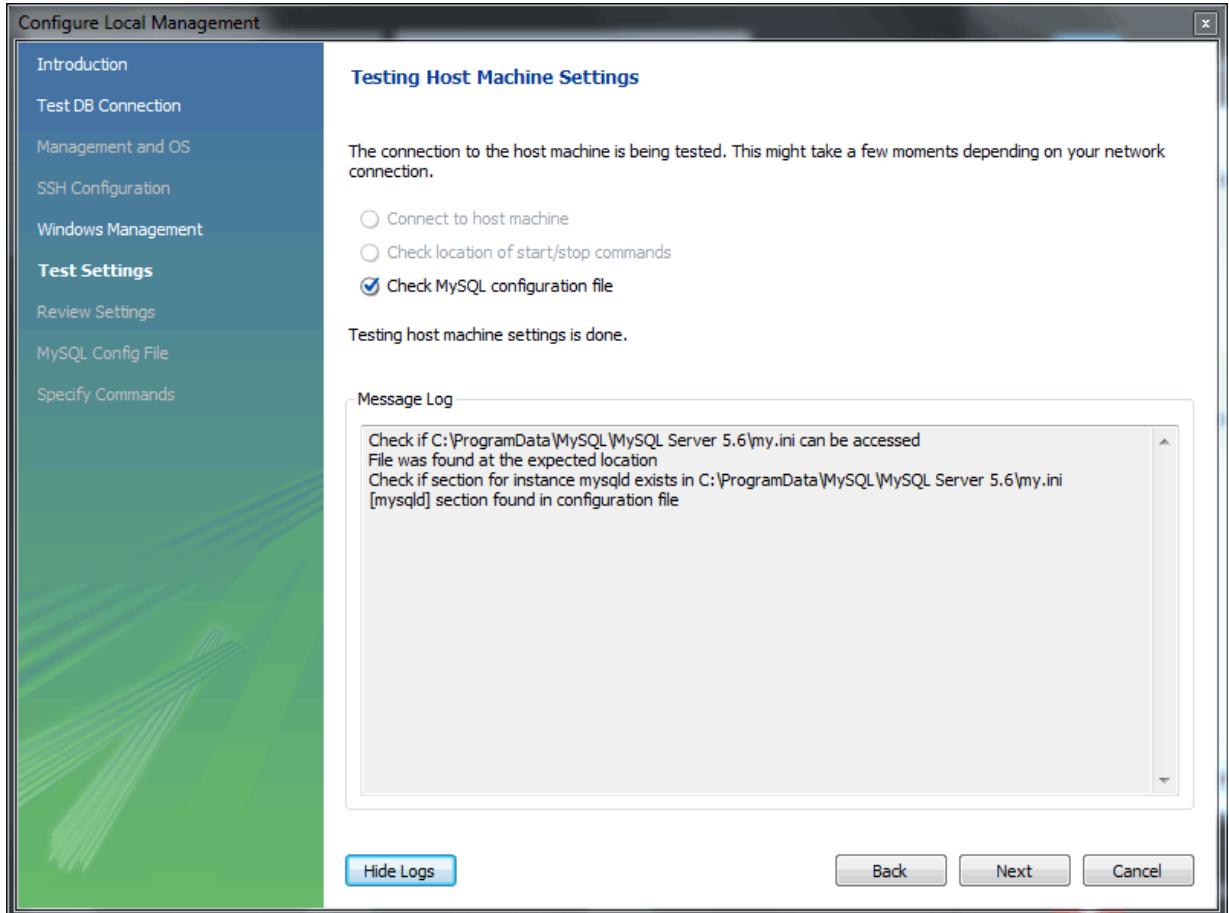
7. On Microsoft Windows, select the appropriate MySQL service for the MySQL connection, as shown in the figure that follows.

Figure 5.6 Getting Started Tutorial - Windows Management



8. The wizard will now check its ability to access the start and stop commands and then check access to the MySQL Server configuration file as the next figure shows.

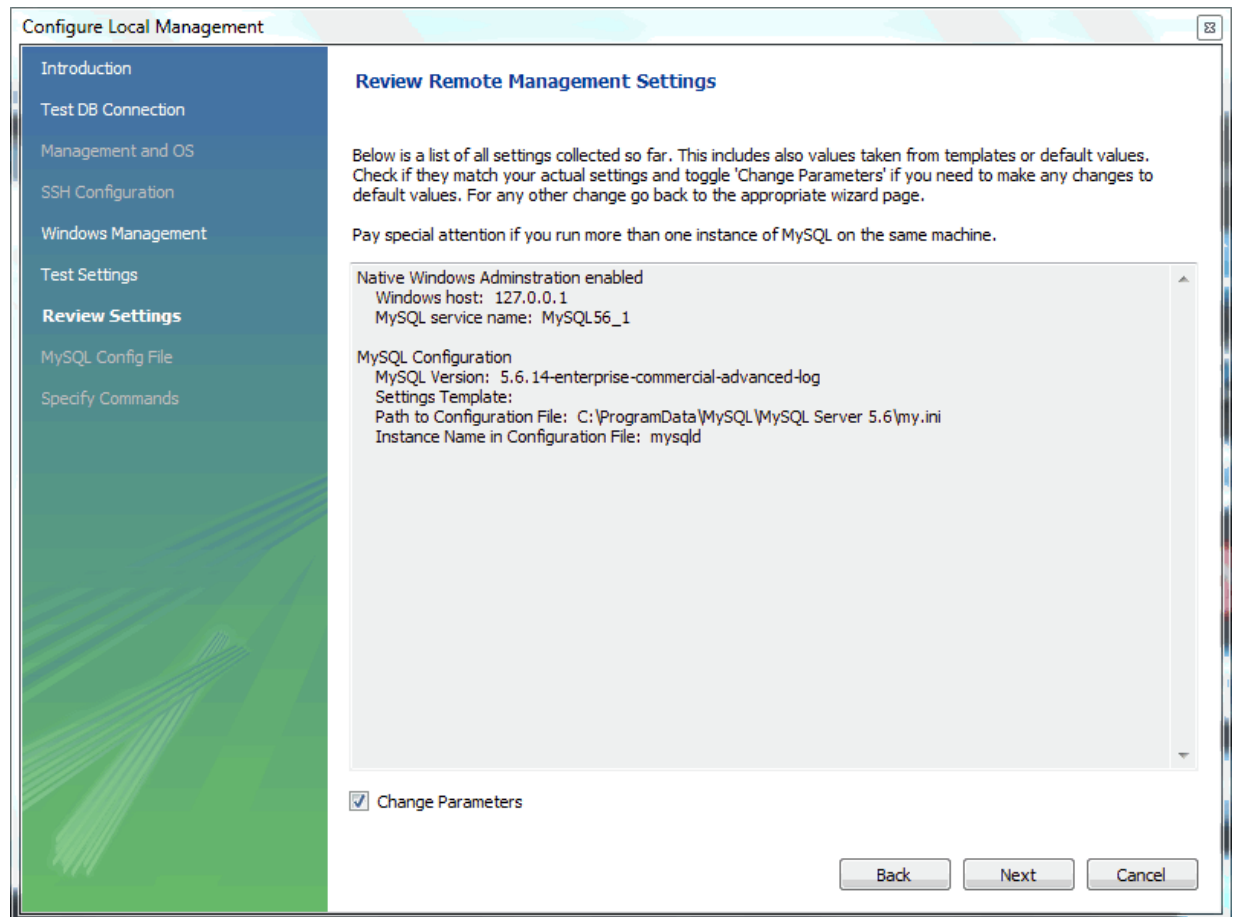
Figure 5.7 Getting Started Tutorial - Test Settings



- 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 as shown in the next figure. Choosing **Continue** closes the Configure Server Management dialog.

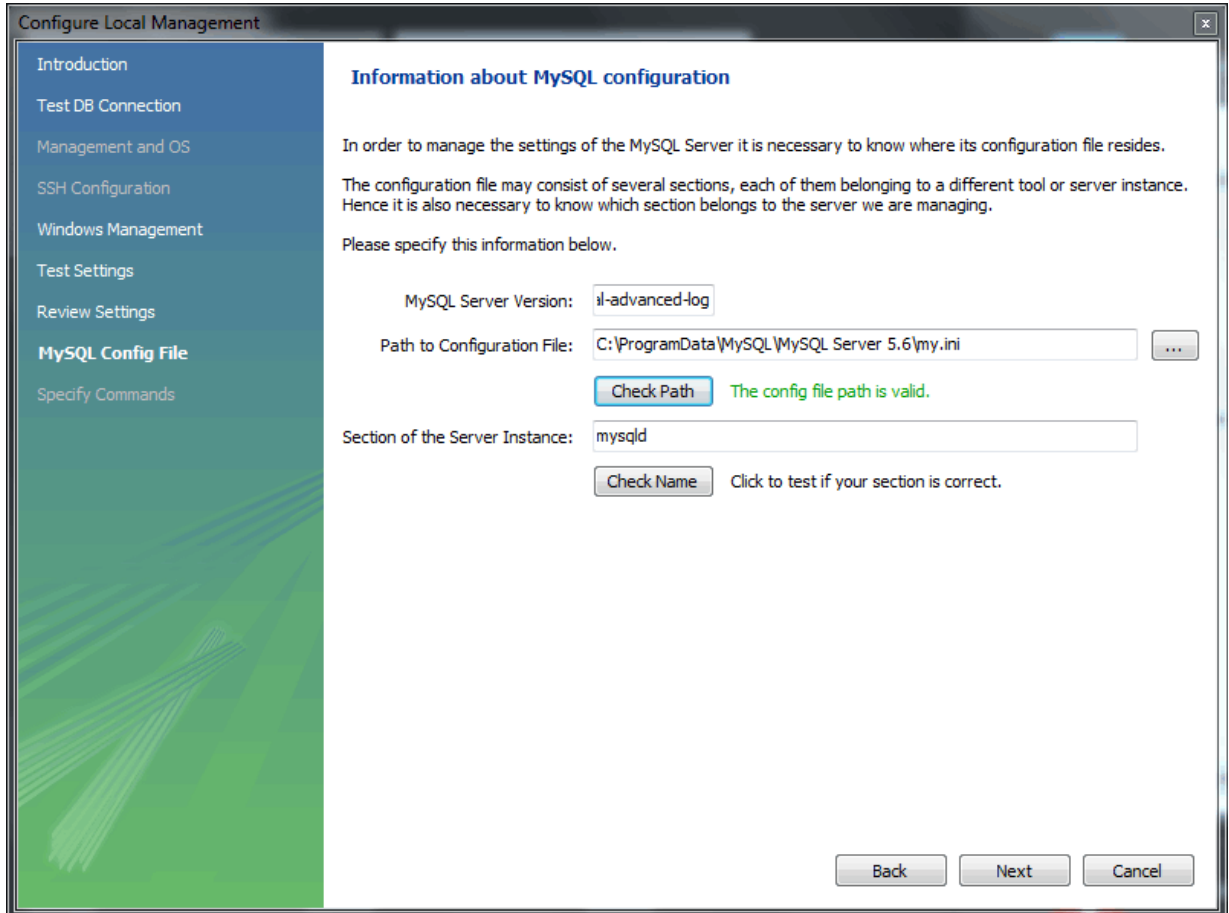
Figure 5.8 Getting Started Tutorial - Review Settings



Check the **Change Parameters** if you want to check or edit information about the MySQL configuration file. For this example, select the check box and click **Next** to continue.

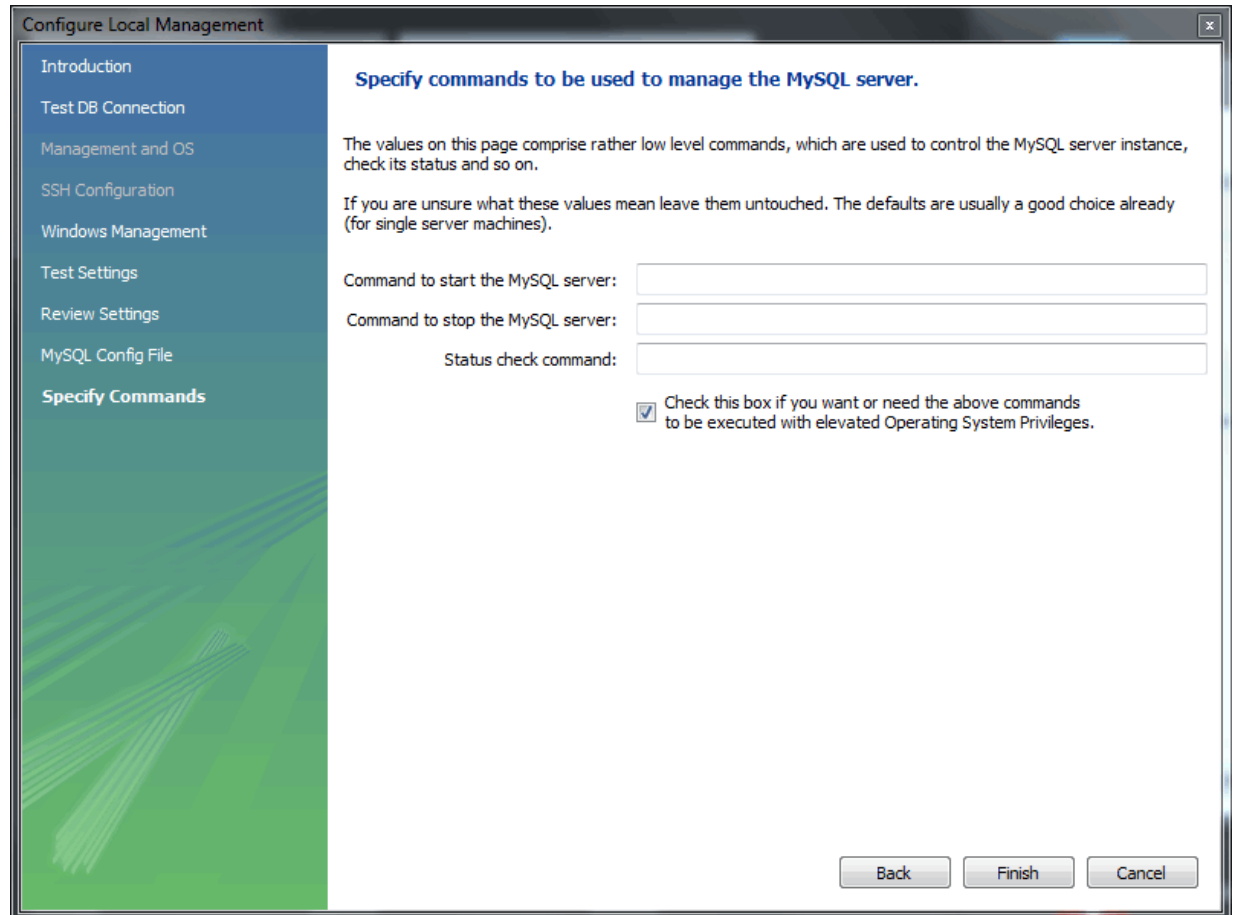
10. Review the MySQL configuration file information shown in the next figure. Click **Check Path** and **Check Name** to perform the described checks, or optionally change the configuration file path.

Figure 5.9 Getting Started Tutorial - MySQL Configuration File



11. Optionally, enter your own commands for starting, stopping, and checking the MySQL connection. To apply the default values, leave these optional values blank as the following figure shows.

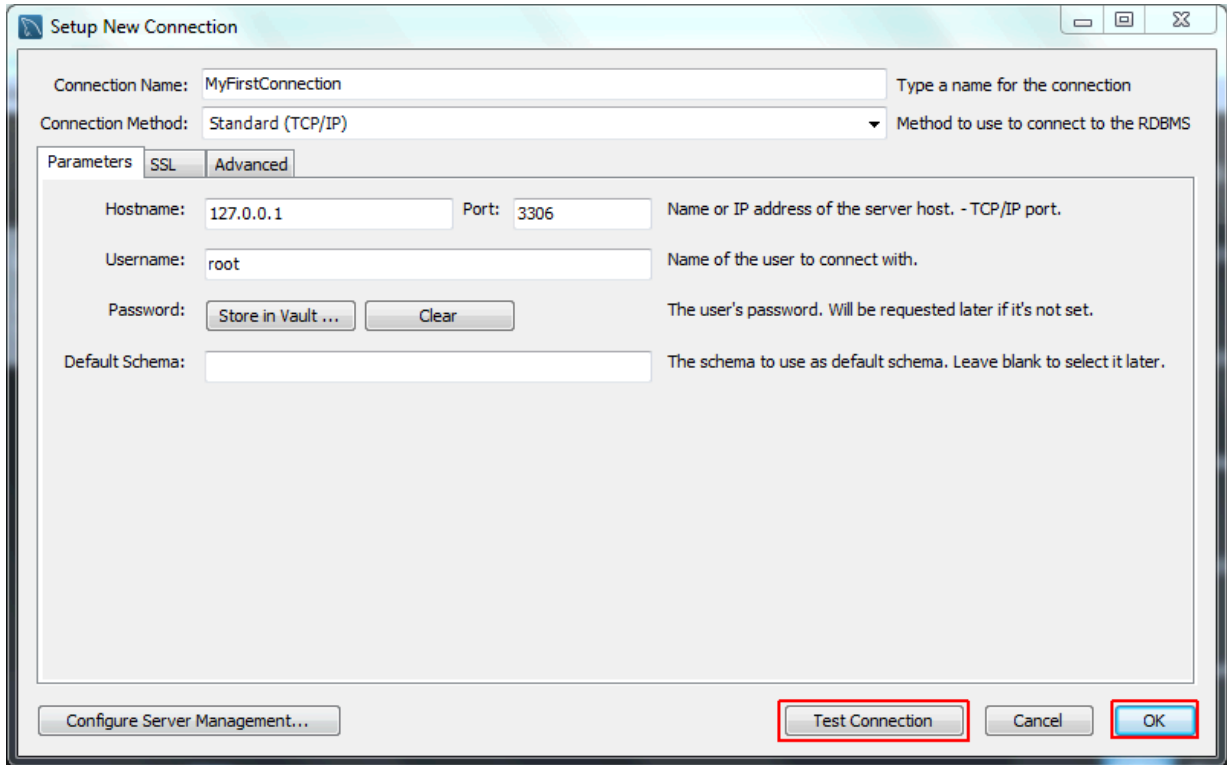
Figure 5.10 Getting Started Tutorial - Specify Commands



Click **Finish** to close the Configure Server Management dialog, which returns to the original Setup New Connection step.

12. After reviewing the **Setup New Connection** information (see the figure that follows), click **Test Connection** again to make sure it still functions and then click **OK** to create the new MySQL connection.

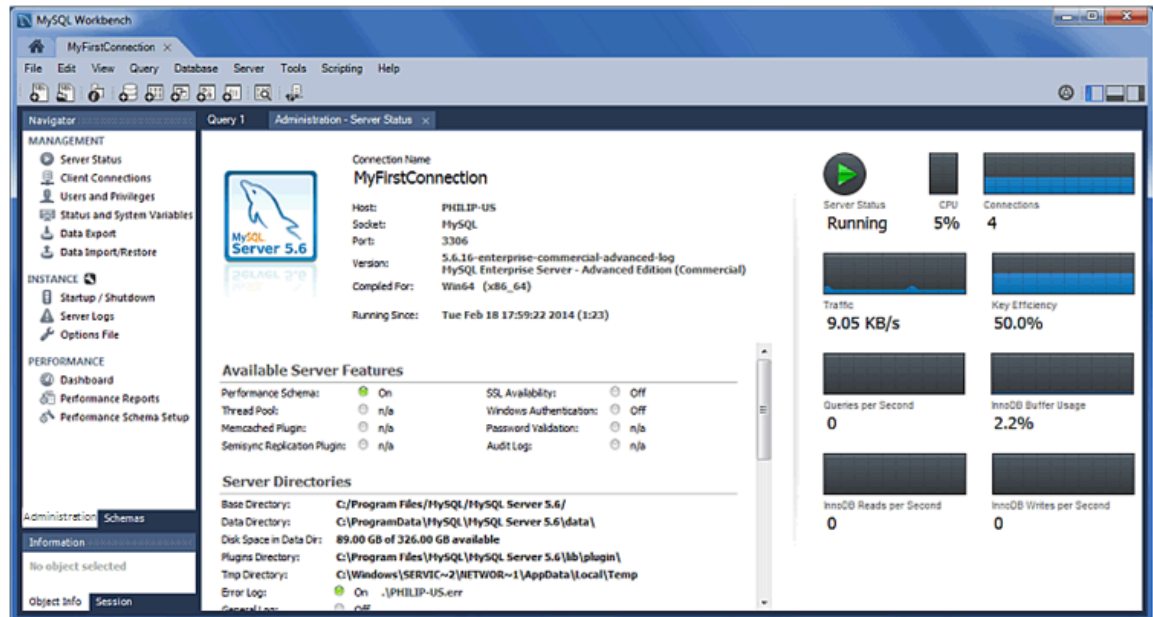
Figure 5.11 Getting Started Tutorial - Setup New Connection




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

- From the home screen, click the new MySQL connection to open the SQL editor for this connection. The SQL editor is the default page. Click **Server Status** from the Navigator area of the sidebar to display the current status of the connected MySQL server instance (see the figure that follows).

Figure 5.12 Getting Started Tutorial - Server Status




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

Notice the **Administration** and **Schemas** tabs in the Navigator area. The **Schemas** view displays the schemas that are associated with your new MySQL connection. Alternatively, you can merge the content of the tabs by either clicking merge () or by enabling the **Show Management Tools and Schema Tree in a single tab** SQL editor preference.

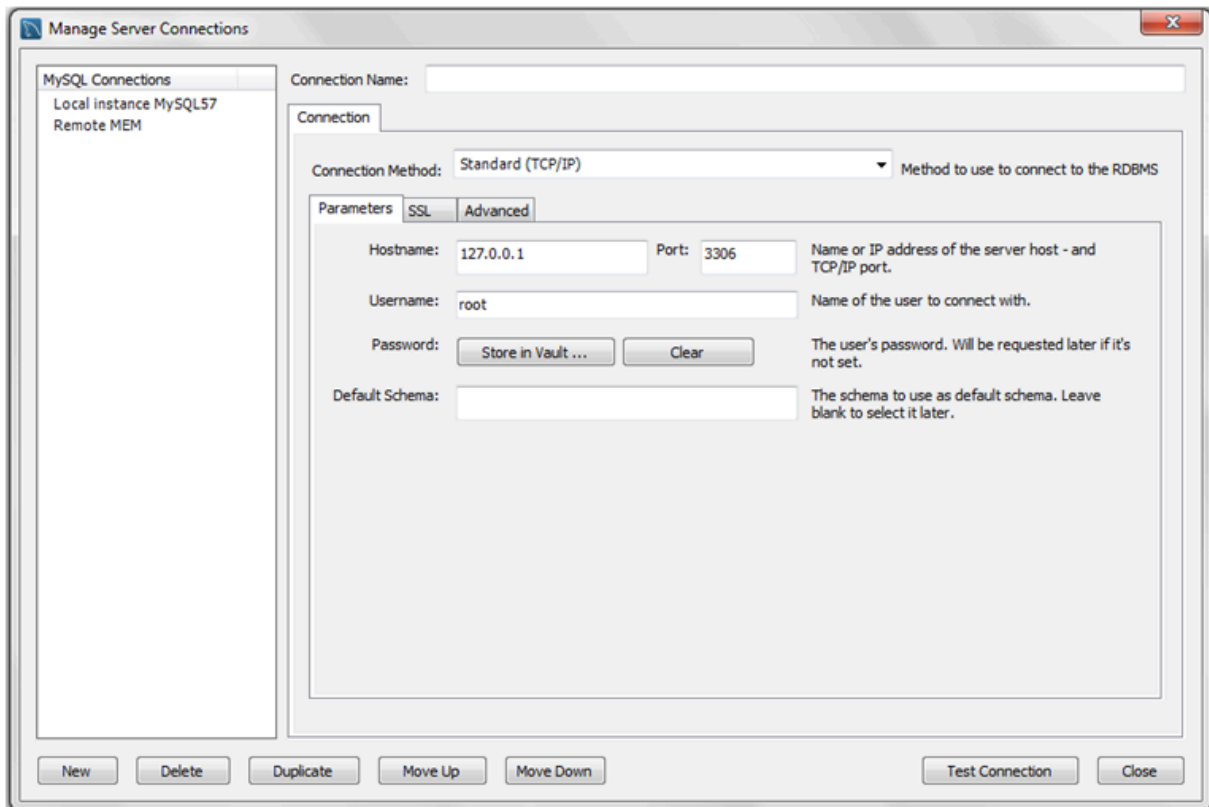
For additional information about MySQL connections, see [Chapter 5, Connections in MySQL Workbench](#).

5.3 Manage Server Connections

The Manage Server Connections dialog is another way to manage MySQL connections. This dialog is invoked by either clicking the manage connections icon () on the home screen or by selecting **Database** and then **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 **Connection** tab of the Manage Server Connections dialog. The following figure shows the initial connection options for a standard TCP/IP connection.

Figure 5.13 Manage Server Connections: Connection Tab



Elements of the MySQL Connection Manager

Connection input fields:

- **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. For more information about these options and parameters, see the section describing each connection method.

Connection actions:

- **New**

Opens a separate **Manage Server Connections** dialog to create a new connection. This dialog provides two tabs in addition to the **Connection** tab: [Remote Management](#) and [System Profile](#)

- **Delete, Duplicate, Move Up, and Move Down**

Operations used to manage the existing connections.

- **Test Connection**

Tests the selected MySQL connection and reports the connection status. It also reports which encryption protocol and cipher MySQL Workbench uses to enable SSL.

For testing remote connections, you might also use [ping](#) to check the host name, 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.

Notes on MySQL Connection Manager



Note

Simultaneous client connections: Opening a MySQL connection from the MySQL Workbench home screen opens a new connection tab in MySQL Workbench for that connection. Each of these tabs requires two MySQL connections to perform basic tasks, such as 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. In addition to naming your new connection in the field provided, you can select from the following tabs to specify connection values: **Parameters**, **SSL**, and **Advanced**.



Note

The [skip_networking](#) MySQL system variable affects the TCP/IP connection method. If disabled, use named pipes or shared memory (on Windows) or Unix socket files (on Unix).

Parameters Tab

The parameters for standard TCP/IP connections are:

- **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 are prompted to enter the password when MySQL Workbench attempts to establish the connection. MySQL Workbench can store the password in a vault (see [Section 5.3.9, “The Password Storage Vault”](#)).
- **Default Schema:** When the connection to the server is established, this option sets the schema that becomes the default schema for use in other parts of MySQL Workbench. For simplicity, you can leave the default schema value blank during the initial setup and set the default value later, if needed.

SSL Tab

SSL parameters are:

- **Use SSL:** SSL encryption is configurable, enabling you to adjust your connection to the conditions determined by the server. The values are:
 - **No** – establishes an unencrypted connection.
 - **If available** (default) – establishes an encrypted connection if the server supports encrypted connections, falling back to an unencrypted connection if an encrypted connection cannot be established.
 - **Require** – causes the connection attempt to fail if an encrypted connection cannot be established.
 - **Require and Verify CA** – requires an encrypted connection and also performs verification against the server CA certificate.
 - **Require and Verify Identity** – performs verification against the server CA certificate and against the server host name in its certificate.
- **SSL Key File:** Path to the Key file for SSL.
- **SSL CERT File:** Path the Certificate file for SSL.
- **SSL CA File:** Path to the Certification Authority file for SSL.
- **SSL Cipher:** Optional list of permissible ciphers to use for SSL encryption.

Actions in this tab:

- **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 5.3.5, “SSL Wizard \(Certificates\)”](#).
- **Files**

Opens a file browser that points to the SSL files generated by the SSL Wizard. For additional information, see [Section 5.3.5, “SSL Wizard \(Certificates\)”](#).

Advanced Tab

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 as text that is not encrypted. Required for some authentication methods. This option is unchecked by default.

It also includes these options:

Timeout: Maximum time to wait before the connection is aborted. The connection times out in 60 seconds by default.

SQL_MODE: Override the default [SQL_MODE](#) used by the server.

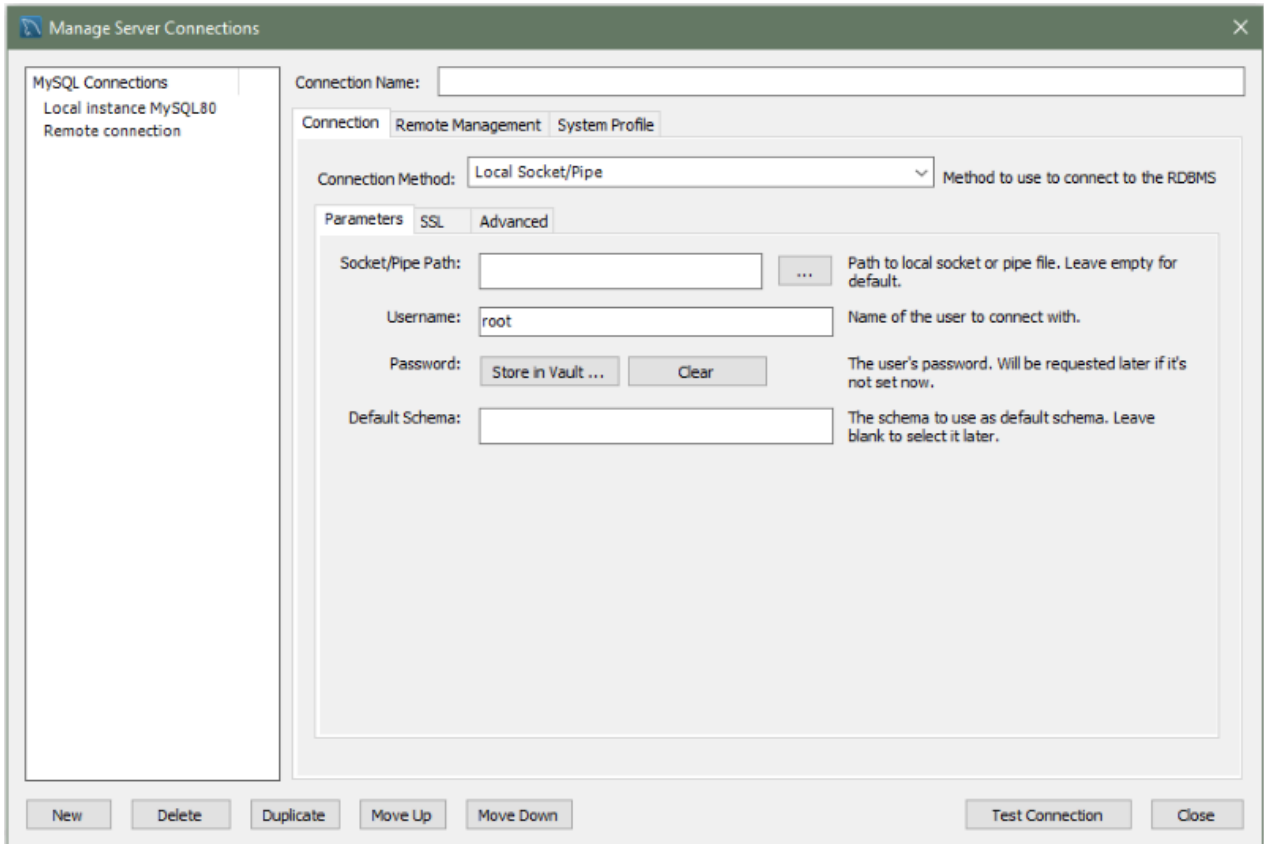
Others: Other options for Connector/C++ as option=value pairs, one per line.

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 Tab

In addition to a number of parameters that are in common with Standard TCP/IP connections, this connection method includes a unique field to configure the socket path or pipe name. As the following figure shows, you can supply the path to the socket file or pipe name within this dialog. 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 5.14 Manage Server Connections - Local Socket/Pipe Parameters

SSL Tab

The SSL options for this connection method are the same as [Standard TCP/IP](#) (see [SSL Tab](#)).

Advanced Tab

The advanced options for this connection method are similar to [Standard TCP/IP](#) (see [Advanced Tab](#)). The **Use compression protocol** and **Timeout** options do not apply to this 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 Tab

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 options are:

- **SSH Hostname:** The name of the SSH server. An optional port number can also be provided. For example, `localhost:22`.
- **SSH Username:** The name of the SSH user to use to make a connection.
- **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.



Note

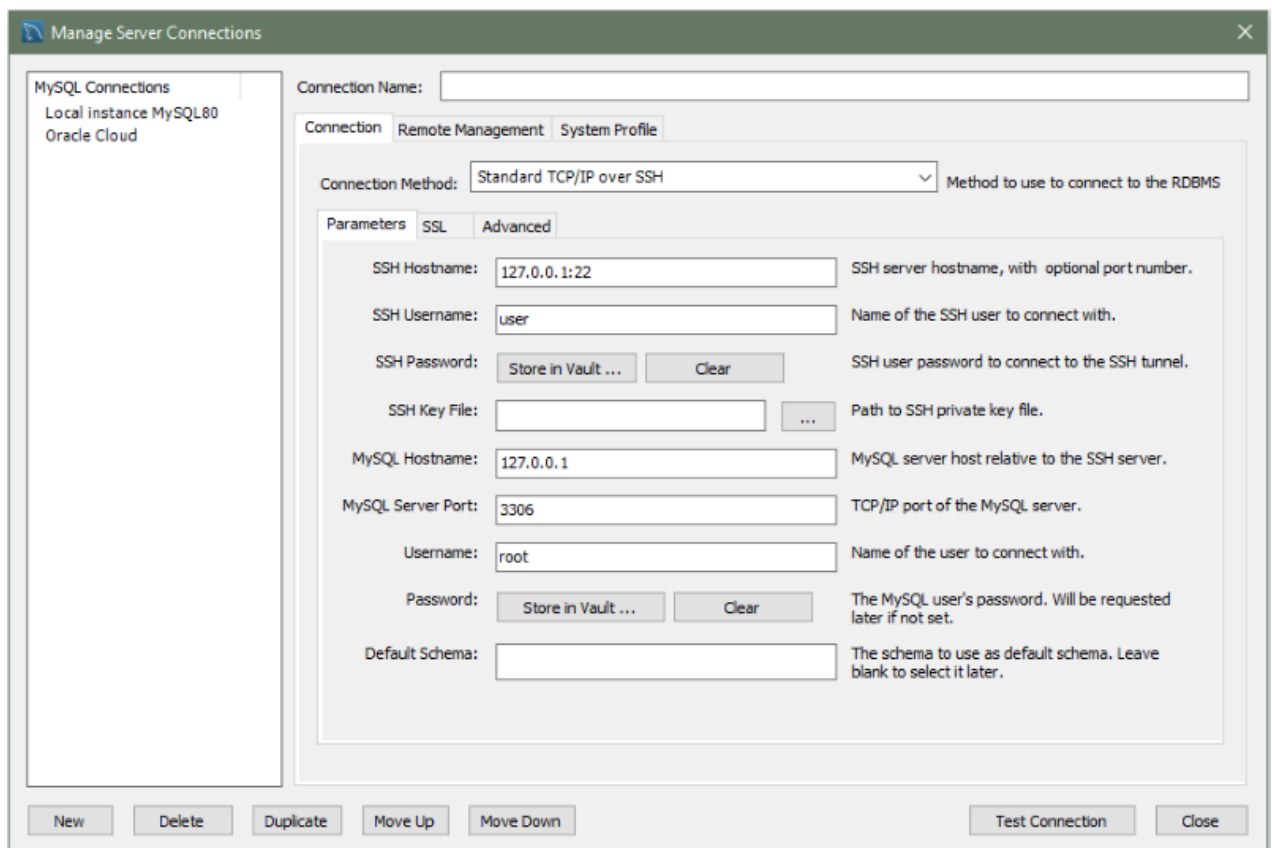
MySQL Workbench does not accept default PuTTY keys directly. You can convert an existing PuTTY Private Key (ppk) file to OpenSSH format by using the PuTTY Key Generator (PuTTYGen) utility.

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.

On Linux and macOS, 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 folder, such as `C:\Users\username\.ssh\known_hosts`. The path to the SSH known hosts file is configurable (see [Section 3.2.6, "SSH Preferences"](#)).

The following figure shows the initial settings of a new connection using the Standard TCP/IP over SSH connection method.

Figure 5.15 Manage Server Connections - Standard TCP/IP over SSH Parameters



SSL Tab

The SSL options for this connection method are the same as [Standard TCP/IP](#) (see [SSL Tab](#)).

Advanced Tab

The advanced options for this connection method are similar to [Standard TCP/IP](#) (see [Advanced Tab](#)). The **Timeout** option does not apply to this connection method.

5.3.4 LDAP and Kerberos Connection Methods

MySQL Enterprise Edition supports authentication methods that enable MySQL Server to use LDAP (Lightweight Directory Access Protocol), LDAP with Kerberos, or native Kerberos to authenticate MySQL users. MySQL Workbench 8.0.27 (and later) provides several connection methods that permit you to use LDAP and Kerberos authentication.



Note

The server-side LDAP and Kerberos authentication plugins are included only in MySQL Enterprise Edition. These server-side plugins are not included in MySQL community distributions. The client-side plugins are included in all MySQL Workbench distributions, including community distributions. This enables users from any MySQL Workbench distribution to connect to a server that has the server-side plugin loaded.

Although MySQL Workbench includes the client-side plugins in all distributions, specific support for LDAP and Kerberos authentication is platform dependent. For example, authentication is not supported when MySQL Workbench is running on macOS.



Note

This implementation of Kerberos authentication does not support MIT Kerberos on any platform.

The following table shows the platform support for each connection method.

Table 5.1 Connection Methods and Supported Platforms

Connection Method	Windows	Linux	macOS
LDAP User/Password	Supported	Supported	Not supported
LDAP Sasl/Kerberos	Not supported	Supported	Not supported
Native Kerberos	Supported	Supported	Not supported

In general, the following requirements must be satisfied to use LDAP or Kerberos pluggable authentication:

- Server-side and client-side plugins need to be compatible, and the server-side plugin must be installed. To minimize the potential for incompatibilities, regularly upgrade the server and MySQL Workbench on a timely basis.
- MySQL Enterprise Edition must be configured for the type of authentication protocol in use. Specific libraries, services, and servers that apply to each authentication method must be available to MySQL Server.



Note

MySQL Workbench restricts the use of SASL-based LDAP authentication to configurations using the Generic Security Service Application Program Interface (GSSAPI)/Kerberos authentication method. The exclusive use of SASL messages for secure transmission of credentials within the LDAP protocol is not supported.

- A MySQL user account must be created or altered with syntax that specifies how the account authenticates. For example, to create an account for `skylar` using simple LDAP authentication (and the [LDAP User/Password](#) connection method), use a form of syntax similar to:

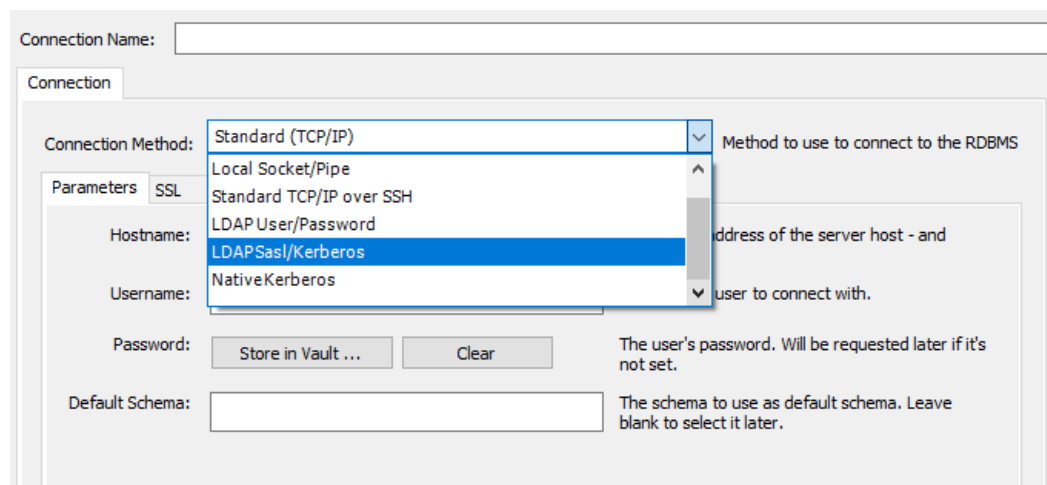
```
CREATE USER 'skylar'@'localhost'
  IDENTIFIED WITH authentication_ldap_simple
  [BY 'LDAP user DN'];
```

The `BY` clause in this example indicates which LDAP entry the MySQL account authenticates against. Specific attributes of the distinguished name (DN) may vary depending on the LDAP server.

To set up the client-side connection from MySQL Workbench:

1. Click the manage connections icon (🔑) on the home screen to open the Manage Server Connections dialog.
2. From the **Connection** tab, select the [LDAP User/Password](#), [LDAP Sasl/Kerberos](#), or [Native Kerberos](#) connection method from the list.

Figure 5.16 Manage Server Connections - LDAP and Kerberos Authentication



With the server requirements satisfied, you can configure the parameters of the named connection from MySQL Workbench. The [Section 5.3.4.1, “LDAP User/Password”](#), [Section 5.3.4.2, “LDAP Sasl/Kerberos”](#), and [Section 5.3.4.3, “Native Kerberos”](#) sections identify the settings for each connection method.

5.3.4.1 LDAP User/Password

MySQL uses LDAP to fetch user, credential, and group information. MySQL Workbench users connect to the MySQL server by providing the MySQL user name and LDAP password.

The [LDAP User/Password](#) connection method supports simple LDAP authentication. The `mysql_clear_password` client-side plugin is compatible with the `authentication_ldap_simple` server-side plugin. MySQL Workbench enables the client-side `mysql_clear_password` authentication plugin explicitly for this connection method.

The server-side plugin must be loaded before you begin (see [Installing LDAP Pluggable Authentication](#)). For an overview of how LDAP authentication works with MySQL, see [How LDAP Authentication of MySQL Users Works](#).

Connection values for the [LDAP User/Password](#) connection method include:

Parameters Tab

- **Hostname:** The host name or IP address of the MySQL server that communicates with the LDAP server.
- **Port:** The TCP/IP port number of the server host, such as 3306.
- **Username:** User name of the account to be authenticated. LDAP users to be authenticated by MySQL must be present in the directory managed by the LDAP server.
- **Password:** The LDAP password of the user account. If you enter no password here, you are prompted to enter the password when MySQL Workbench attempts to establish the connection. MySQL Workbench can store the password in a vault.
- **Default Schema:** When the connection to the server is established, this option sets the schema that becomes the default schema for use in other parts of MySQL Workbench. For simplicity, you can leave the default schema value blank during the initial setup and set the default value later, if needed.

SSL Tab

The SSL options for this connection method are the same as [Standard TCP/IP](#) (see [SSL Tab](#)).

**Important**

The client-side and server-side plugins for this connection method communicate the password as cleartext. No password hashing or encryption is used, so a secure connection between MySQL Workbench and the server is recommended to prevent password exposure.

Advanced Tab

The advanced options for this connection method are similar to [Standard TCP/IP](#) (see [Advanced Tab](#)). The **Enable Cleartext Authentication Plugin** option does not apply to this connection method.

5.3.4.2 LDAP Sasl/Kerberos

The [LDAP Sasl/Kerberos](#) connection method is supported as an LDAP authentication method for MySQL servers and MySQL Workbench on Linux only. Using the GSSAPI security abstraction interface, a connection of this type authenticates to Kerberos to obtain service credentials, then uses those credentials in turn to enable secure access to other services. A GSSAPI library and Kerberos services must be available to MySQL Server (see [The GSSAPI/Kerberos Authentication Method](#)).

**Tip**

If the Linux environment hosting MySQL Workbench has access to LDAP through Microsoft Active directory, then Kerberos is enabled by default.

MySQL Workbench provides the `authentication_ldap_sasl_client` client-side plugin to support this connection method. It is compatible with the `authentication_ldap_sasl` server-side plugin, which must be installed on the MySQL server hosting the connection (see [Installing LDAP Pluggable Authentication](#)). Also, the `authentication_ldap_sasl_auth_method_name` system variable must be set to use the `GSSAPI` method. For additional variables that can (or should) be configured when using the server-side plugin, see [Configure the Server-Side SASL LDAP Authentication Plugin for GSSAPI/Kerberos](#).

Connection values for the [LDAP Sasl/Kerberos](#) connection method include:

Parameters Tab

- **Hostname:** The host name or IP address of the MySQL server with an account that has the Kerberos principal name as the user name and that authenticates using the SASL LDAP plugin.
- **Port:** The TCP/IP port number of the server host, such as 3306.
- **Username:** User name of the Kerberos principal associated with the MySQL account. For LDAP Kerberos authentication, the user part of the account name includes the principal domain, so `user@default_realm` (for example, `skylar@MYSQL.LOCAL`) is the user name.
- **Password:** Password of the Kerberos principal associated with the MySQL account. If you enter no password here, you are prompted to enter the password when MySQL Workbench attempts to establish the connection. MySQL Workbench can store the password in a vault.
- **Default Schema:** When the connection to the server is established, this option sets the schema that becomes the default schema for use in other parts of MySQL Workbench. For simplicity, you can leave the default schema value blank during the initial setup and set the default value later, if needed.

SSL Tab

The SSL options for this connection method are the same as [Standard TCP/IP](#) (see [SSL Tab](#)).

Advanced Tab

The advanced options for this connection method are similar to [Standard TCP/IP](#) (see [Advanced Tab](#)), but also include the following options:

- **Path to plugin directory:**

An alternative path might be necessary to ensure that the client-side and server-side plugins remain compatible.

- **Kerberos configuration path:**

Full path name to the Kerberos configuration information on Linux.

- **Kerberos credentials cache:**

Location of the Kerberos credentials (ticket) cache on Linux.

5.3.4.3 Native Kerberos

The [Native Kerberos](#) connections method authenticates a MySQL user with authentication tokens generated by the `kinit` command. Using this connection method, MySQL Workbench and MySQL servers are able to use the Kerberos authentication protocol to mutually authenticate users and MySQL services. This way both the user and the server are able to verify each other's identity. No passwords are sent over the network and Kerberos protocol messages are protected against eavesdropping and replay attacks.

MySQL Workbench provides the `authentication_kerberos_client` client-side plugin to support this connection method. It is compatible with the `authentication_kerberos` server-side plugin, which must be installed and loaded on the MySQL server hosting the connection (see [Installing Kerberos Pluggable Authentication](#)).

For server configuration setup details and an operational overview of Kerberos authentication, see:

- [Prerequisites for Kerberos Pluggable Authentication](#)

- [How Kerberos Authentication of MySQL Users Works](#)

Connection values for the [Native Kerberos](#) connection method include:

Parameters Tab

- **Hostname:** The host name or IP address of the MySQL server with an account that has the Kerberos principal name as the user name and that authenticates using the Kerberos plugin.
- **Port:** The TCP/IP port number of the server host, such as 3306.
- **Username:** User name associated with the MySQL account.

The client-side Kerberos authentication plugin combines the user name you provide (for example, `skylar`) and the realm specified in the user account (for example, `MYSQL.LOCAL`) to construct the user principal name (UPN), such as `skylar@MYSQL.LOCAL`. The client-side plugin uses the UPN and password to obtain a ticket-granting ticket (TGT), uses the TGT to obtain a MySQL service ticket (ST), and uses the ST to authenticate to the MySQL server.

- **Password:** Password associated with the MySQL account. If you enter no password, you might be prompted to enter the password when MySQL Workbench attempts to establish the connection. MySQL Workbench can store the password in a vault.



Note

If the `kinit` command is used to authenticate a Kerberos principal name (outside of MySQL Workbench), MySQL Workbench authorizes the user without checking (or prompting) for a password. This behavior applies even when the password is stored in a vault.

- **Default Schema:** When the connection to the server is established, this option sets the schema that becomes the default schema for use in other parts of MySQL Workbench. For simplicity, you can leave the default schema value blank during the initial setup and set the default value later, if needed.
- **Kerberos Mode** On Windows, select between Kerberos authentication using the Windows SSPI Kerberos library or GSSAPI through the MIT Kerberos library. Only GSSAPI is permitted on Linux.

The mode values are:

- **GSS API Authentication (MIT Native)** (default value) – The MIT Kerberos cache can be populated using the `kinit` command. In `GSSAPI` mode, the ticket search on Windows hosts is restricted to the MIT Kerberos cache only. If the cache has no ticket, the connection fails even if the Windows ticket is valid.
- **SSPI API Authentication (Windows)** – The SSPI Kerberos library is not compatible with Java SE security tools (`klist`, `kinit`, and so on). In `SSPI` mode, the authentication method considers the Windows single sign-on ticket only. If the ticket is missing or invalid, the connection fails even if the MIT Kerberos cache contains a valid ticket.

SSL Tab

The SSL options for this connection method are the same as [Standard TCP/IP](#) (see [SSL Tab](#)).

Advanced Tab

The advanced options for this connection method are similar to [Standard TCP/IP](#) (see [Advanced Tab](#)), but also include the following options:

- **Path to plugin directory:**

An alternative path might be necessary to ensure that the client-side and server-side plugins remain compatible.

- **Kerberos configuration path:**

Full path name to the Kerberos configuration information on Linux, or on Windows with the [GSS API Authentication \(MIT Native\)](#) Kerberos mode option selected.

- **Kerberos credentials cache:**

Location of the Kerberos credentials (ticket) cache on Linux, or the MIT Kerberos cache on Windows with the [GSS API Authentication \(MIT Native\)](#) Kerberos mode option selected.

5.3.5 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.

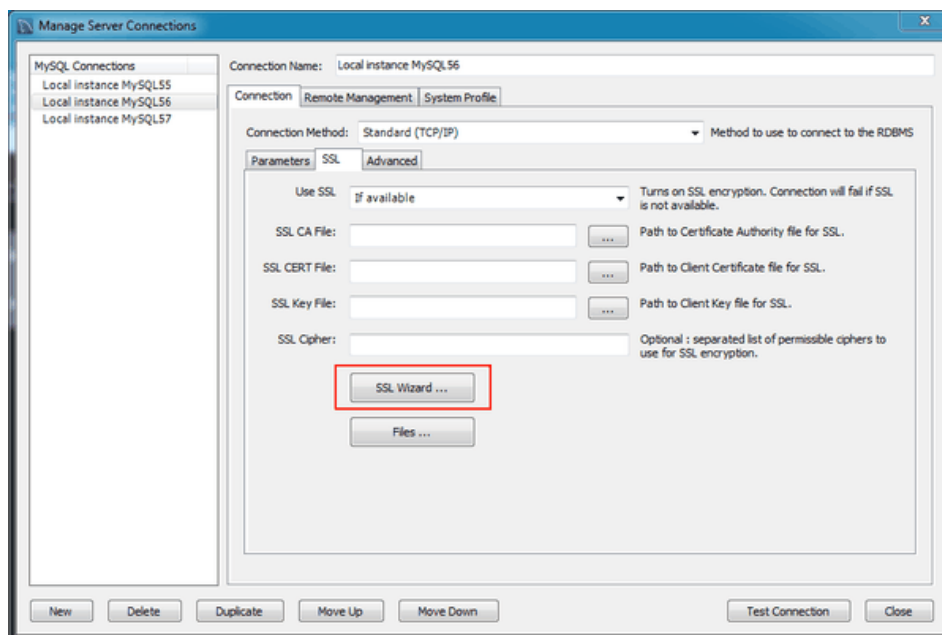


Note

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

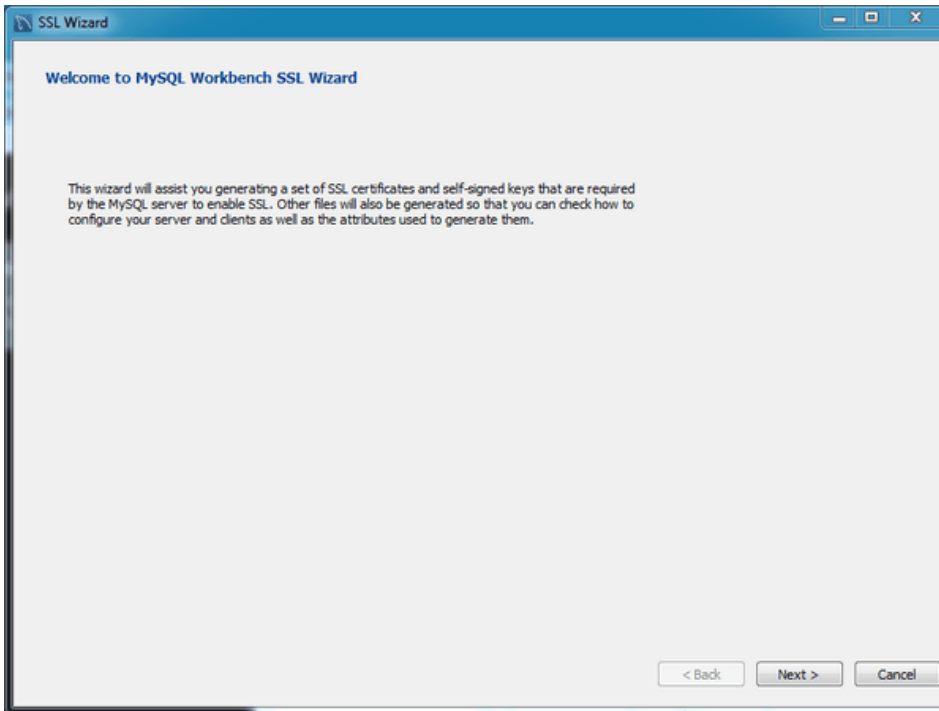
Start the SSL wizard from the **SSL** tab of a MySQL server connection. Locate this tab in the MySQL connection editor. Click **SSL Wizard** to execute the wizard, as the following figure shows.

Figure 5.17 SSL Wizard: Start



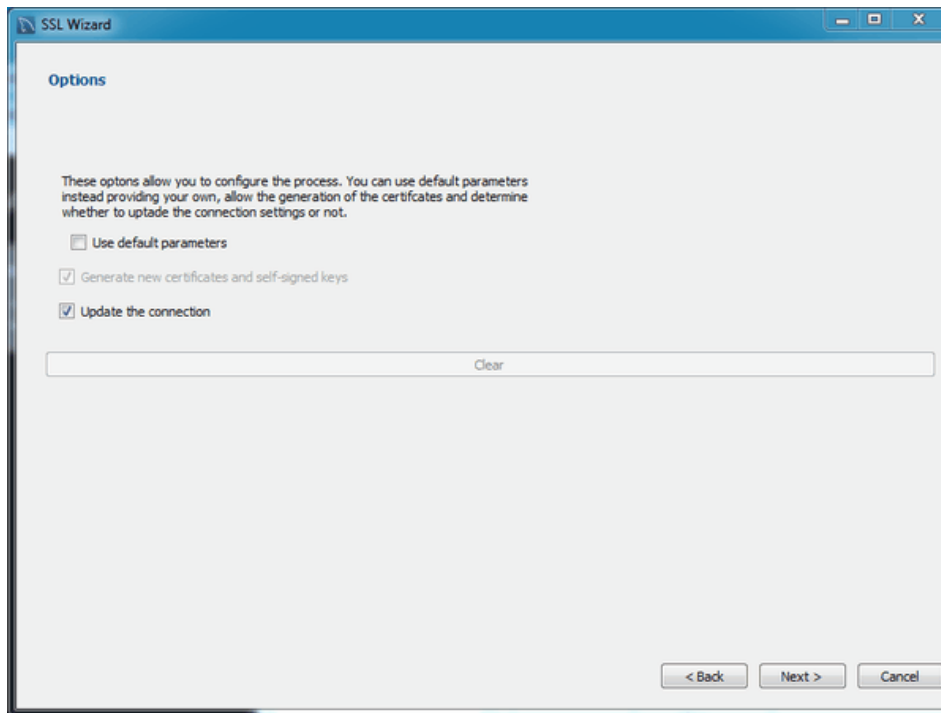
Read the informative text on the welcome screen (displayed in the following figure), and then click **Next**.

Figure 5.18 SSL Wizard: Welcome



Check the options that apply. The following figure shows an example of the available options.

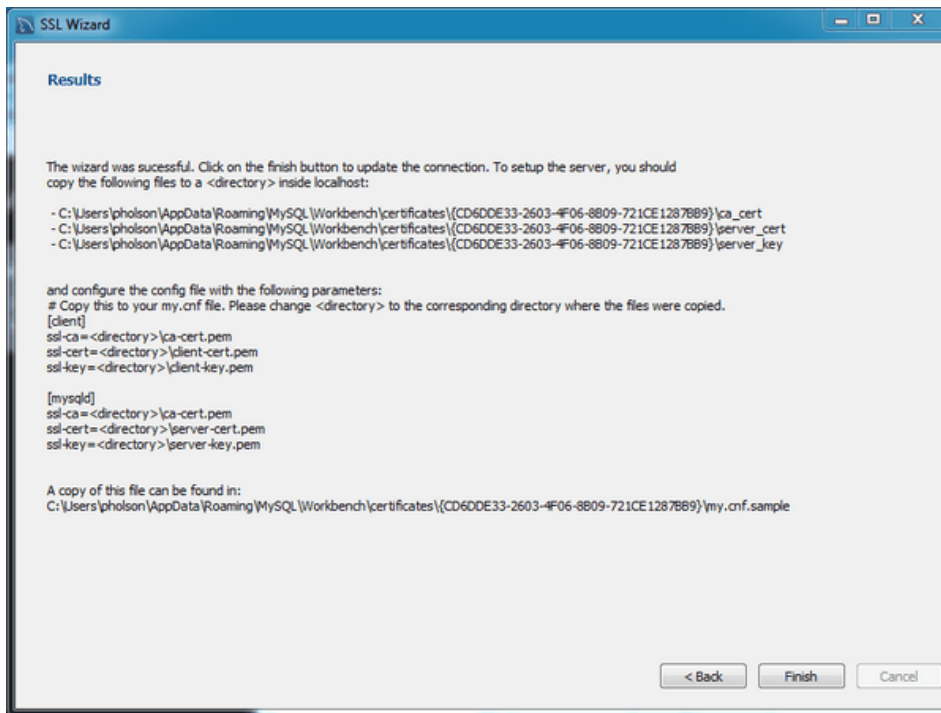
- **Use default parameters:** Check this to skip entering the optional attributes, such as Country, State, Organization, and so on. By default, these fields are empty.
- **Generate new certificates and self-signed keys:** Check this to generate new files, otherwise the existing files are used. You might 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 MySQL Workbench) with the generated certificate information.

Figure 5.19 SSL Wizard: Options

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.

The following figure shows an example Results screen. 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 5.20 SSL Wizard: Results



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 this simple example, MySQL Workbench is installed on the same host as the MySQL Server, and we created "C:\certs" on the 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 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
```

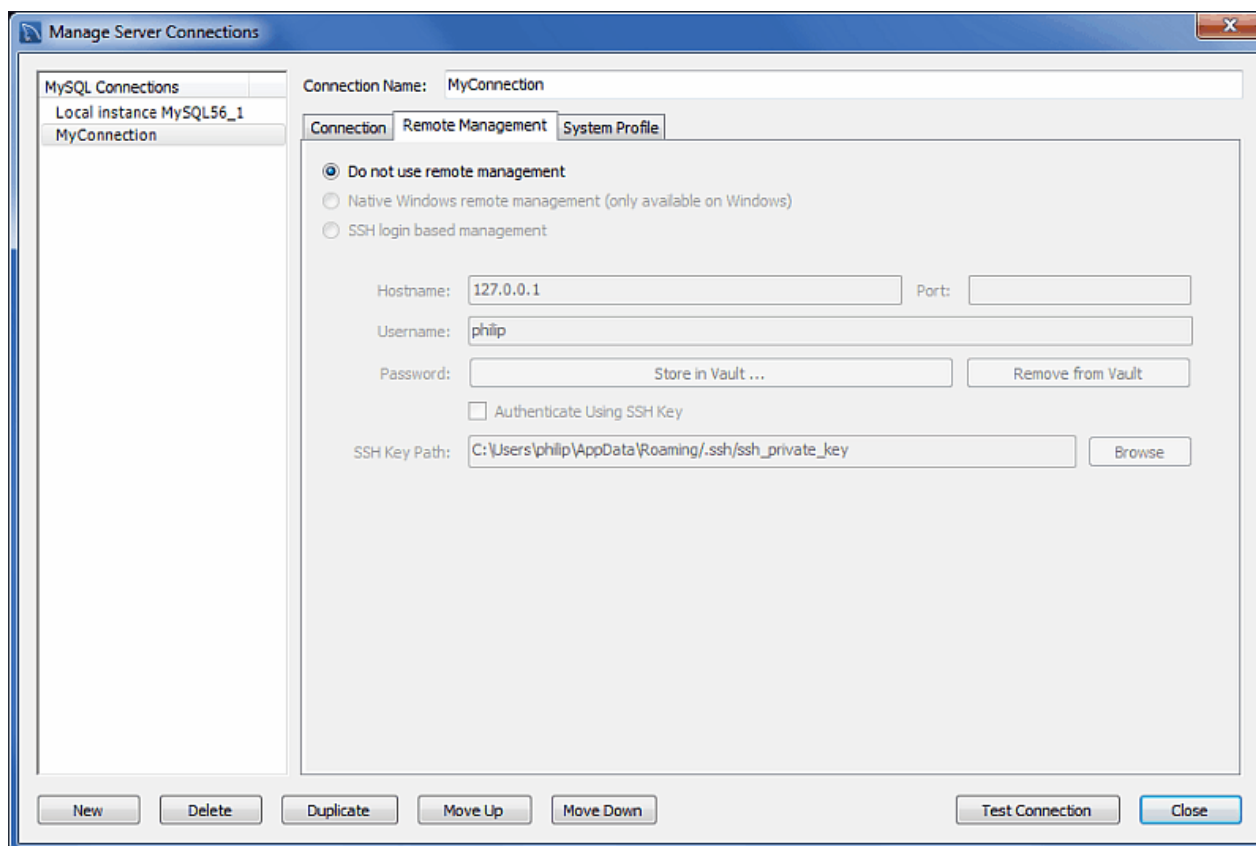
5. Update the paths to the SSL client certificates in your MySQL connection, under the SSL tab. There are three paths to update.
6. Restart the MySQL Server. In the log, you should see something like "Warning CA certificate `C:\certs\ca-cert.pem` is self signed."
7. In the MySQL connection editor, clicking **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.

5.3.6 Remote Management

The **Remote Management** tab is available when connecting to MySQL remotely, as the following figure shows. To access this tab, select a remote connection from the **MySQL Connections** pane or click **New** to create a new connection.

Figure 5.21 Manage Server Connections: Remote Management Tab



5.3.7 System Profile

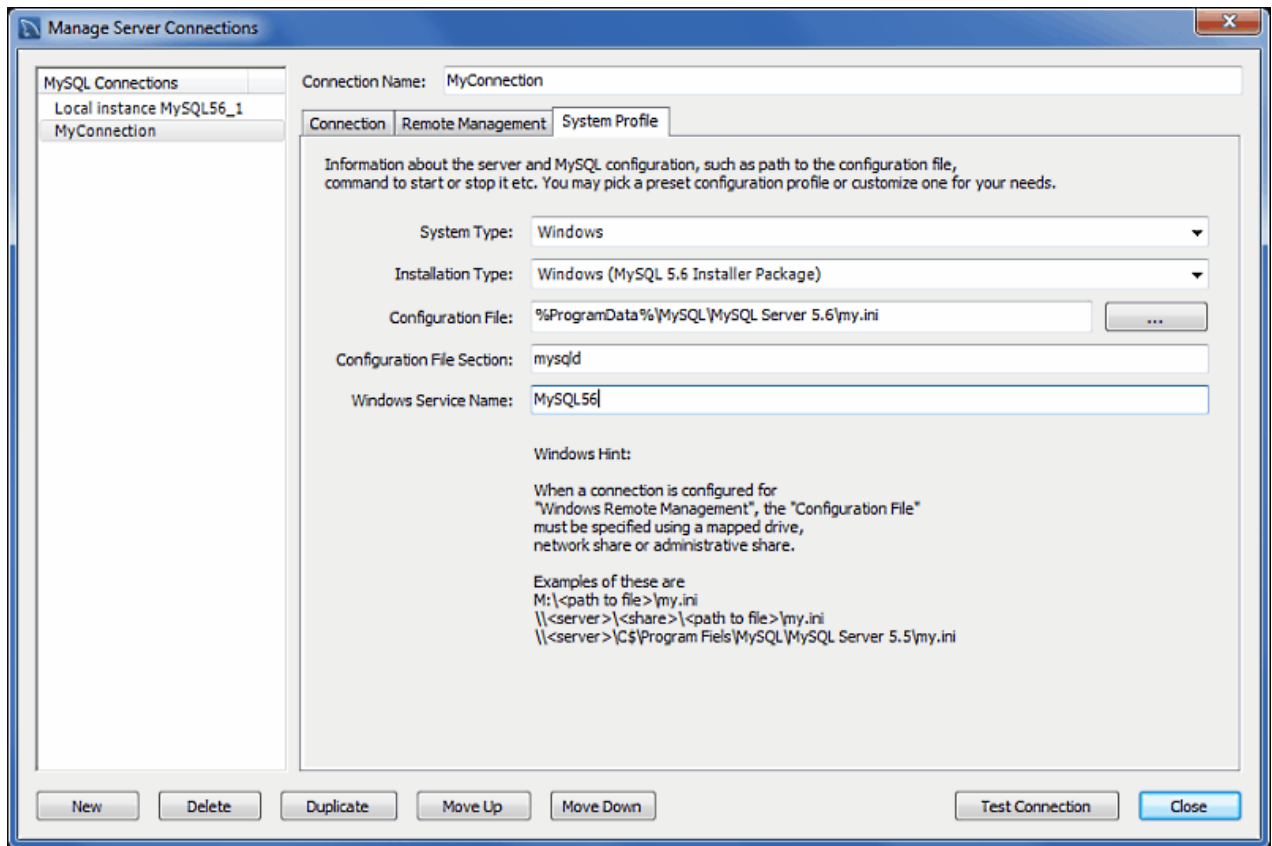
The **System Profile** tab enables you to specify host-specific information. This is achieved primarily through selecting a **System Type**, along with its corresponding **Installation Type**. These profile settings contain standard information that is used in managing the host's MySQL instance.

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
- macOS, 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. As the following figure shows, these default values are customizable.

Figure 5.22 Manage Server Connections: System Profile Tab

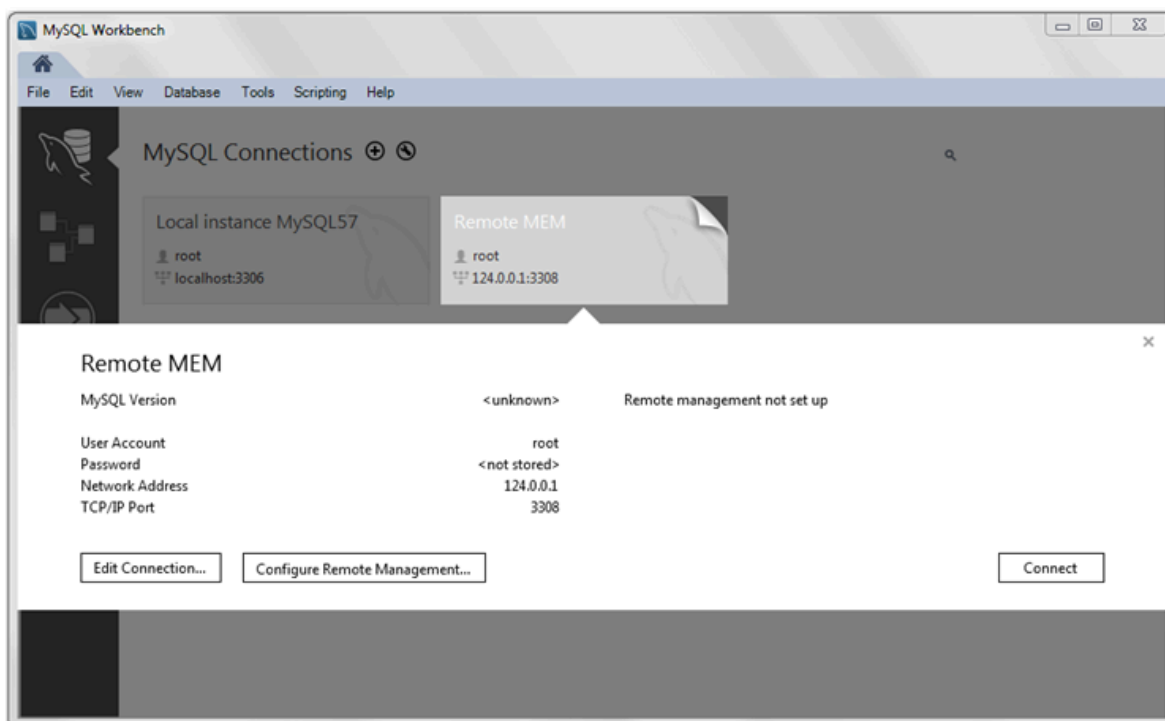


5.3.8 Configure Server Management Wizard

Clicking the [+] icon from the home screen 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 option can also be executed later (on remote connections) from the home screen by clicking the top right corner of a MySQL remote connection, as the following figure shows.

Figure 5.23 Configure Remote Management



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 following general steps, see [Section 5.2, “Creating A New MySQL Connection \(Tutorial\)”](#).

The steps presented in the wizard are:

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. Select the **Change Parameters** check box 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.

5.3.9 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 host name 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.
- **macOS:** The vault is implemented using the Secure Keychain. The keychain content is also viewable from the native `Keychain Access.app` utility.
- **Linux:** The vault works by storing passwords using the `libsecret` library, which communicates with Secret Service. For systems with the GNOME desktop environment, such as Ubuntu, the Secret Service is `gnome-keyring-daemon`. Systems with the KDE desktop environment, for example Kubuntu, use their own `ksecret-service` implementation.

5.3.10 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 MySQL 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 MySQL Workbench upgrades passwords, it upgrades `mysql_old_password` to `mysql_native_password`. For additional information about authentication plugins, see [Pluggable Authentication](#).

**Note**

The `mysql_native_password` authentication plugin is deprecated as of MySQL Server 8.0.34, disabled by default as of MySQL Server 8.4.0, and removed as of MySQL Server 9.0.0.

MySQL Workbench uses `mysql_native_password` by default, so for example enabling it is required to create a MySQL Server 8.4 user. Include `mysql_native_password=ON` in the `[mysqld]` section of your MySQL Server 8.4 `my.cnf` to enable this plugin.

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 MySQL 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 MySQL Workbench to Update Your Password

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

Open the [Users and Privileges](#) tab from the Management navigator. Select the user account 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 figures demonstrate the sequence of steps used in both methods:

Figure 5.24 Upgrade Old Password: Setting a New Password

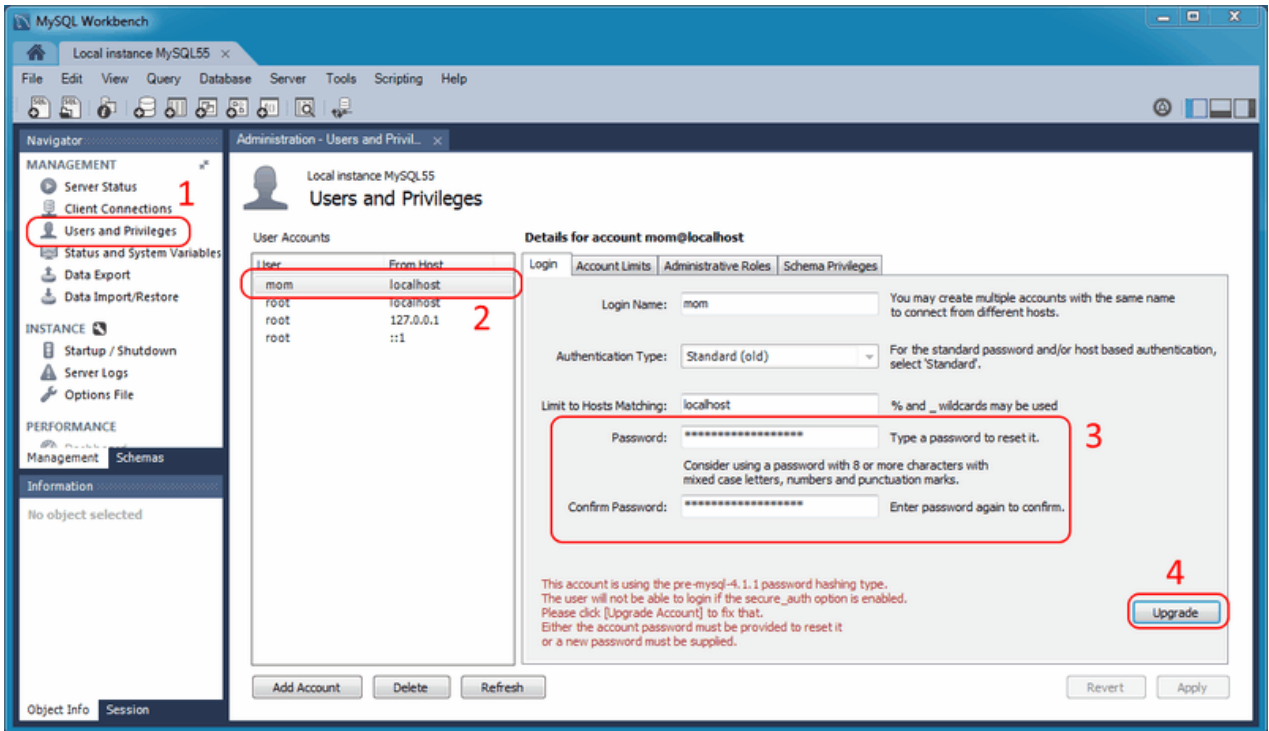
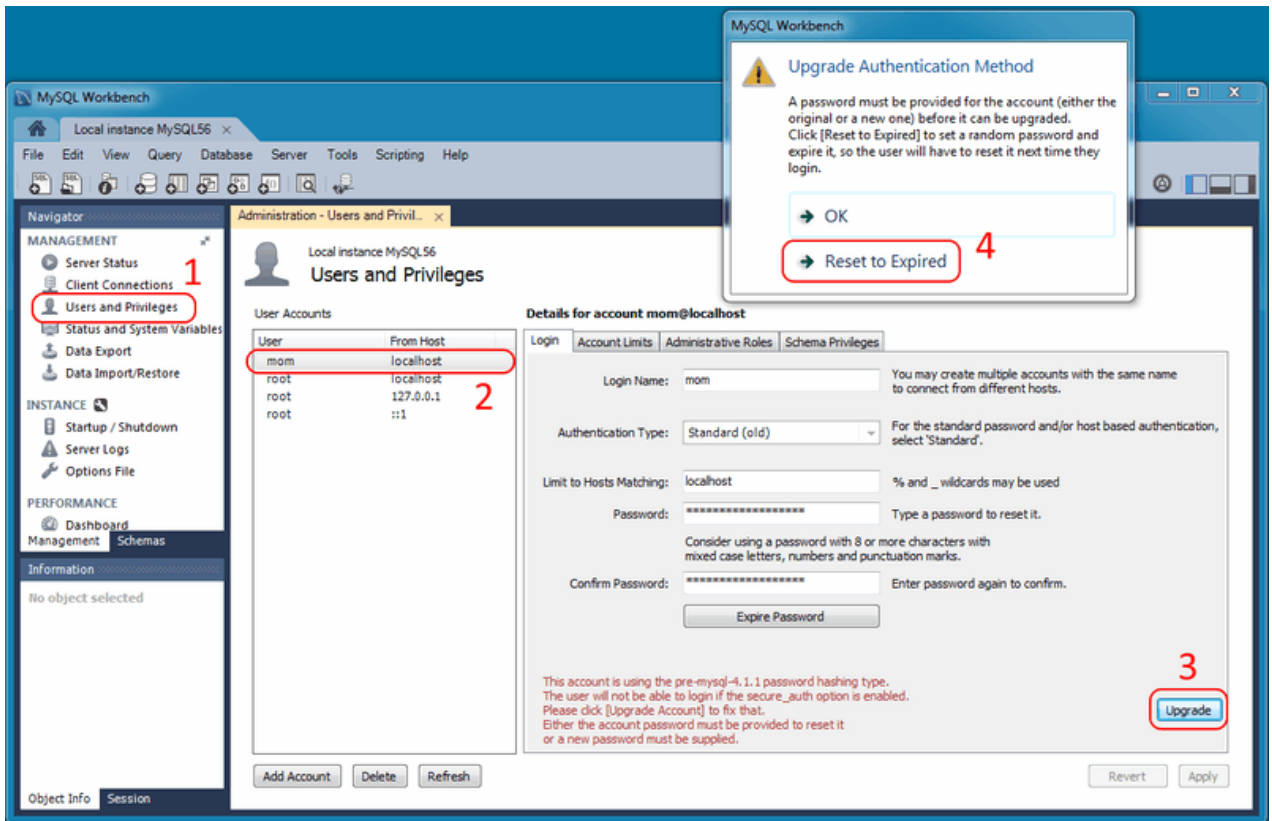
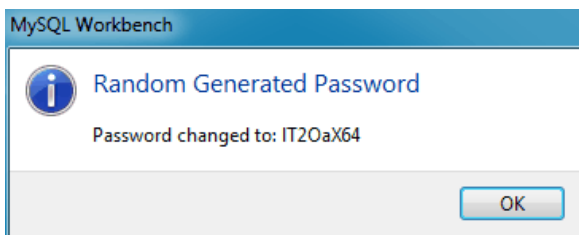


Figure 5.25 Upgrade Old Password: Reset to Random Expired Password



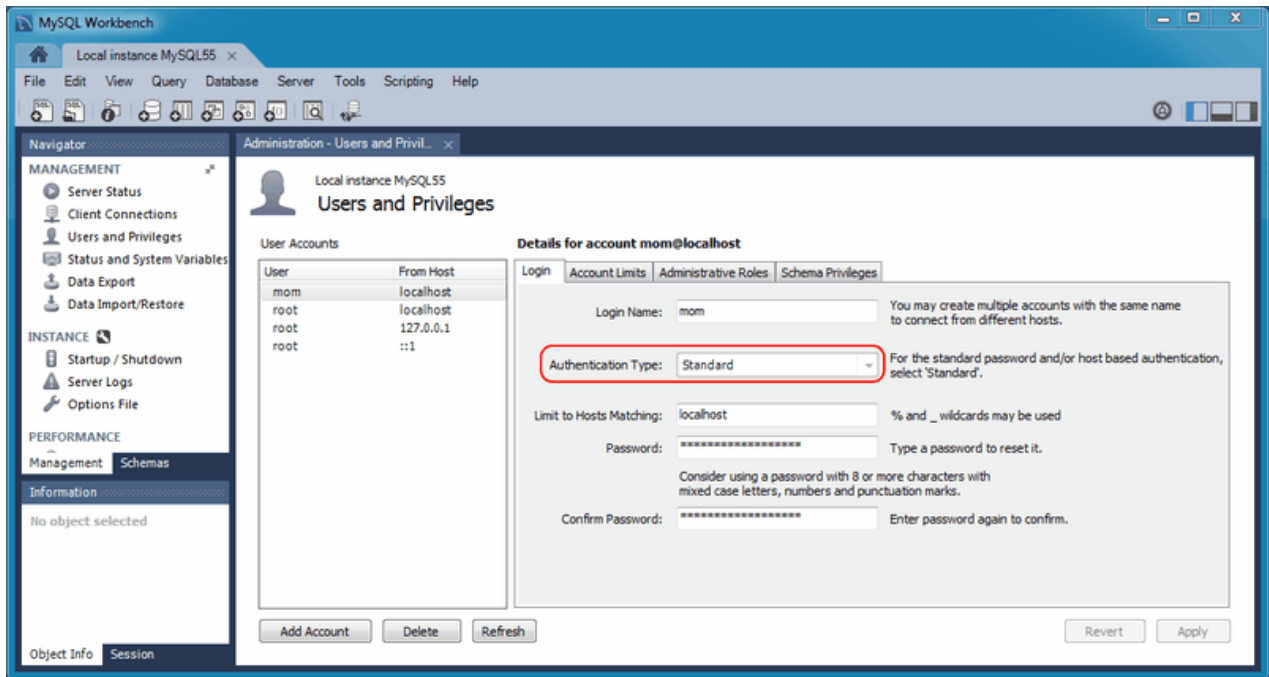
When resetting to a random password, you must save the password and give it to the user. You will find the random password in the new popup window that is similar to the following figure.

Figure 5.26 Upgrade Old Password: Random Password Popup



After completing the upgrade, notice the new **Authentication Type** for the connection. In the example shown in the next figure, the value changed from **Standard (old)** to **Standard**. In other words, the authentication type changed from `mysql_old_password` to `mysql_native_password`.

Figure 5.27 Upgraded Password: Standard (old) to Standard



5.4 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. The following figure shows an example of client connection information for a local host.



Note

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

Figure 5.28 Client Connection Overview

The screenshot shows the MySQL Workbench interface with the 'Client Connections' window open. The window displays a summary of connection statistics and a table of active connections.

Summary Statistics:

- Threads Connected: 4
- Threads Running: 2
- Threads Created: 4
- Threads Cached: 0
- Rejected (over limit): 0
- Total Connections: 32
- Connection Limit: 151
- Aborted Clients: 0
- Aborted Connections: 20
- Errors: 0

Table of Connections:

Id	User	Host	DB	Command	Time	State	Threa...	Type	Name	Paren...	Instrumented	Info	Program
32	root	localhost	None	Sleep	1	None	61	FOREGROUND	thread/sql/one_connection	35	YES	NULL	MySQLWorkbench
31	root	localhost	None	Query	0	Sending d...	60	FOREGROUND	thread/sql/one_connection	35	YES	SELECT...	MySQLWorkbench
30	root	localhost	None	Sleep	213	None	59	FOREGROUND	thread/sql/one_connection	35	YES	NULL	MySQLWorkbench
29	root	localhost	None	Sleep	214	None	58	FOREGROUND	thread/sql/one_connection	0	YES	NULL	MySQLWorkbench
6	None	None	None	Daemon	577	Suspending	34	FOREGROUND	thread/sql/compress_gtid_table	1	YES	NULL	None
4	even...	None	None	Sleep	0	Waiting on...	33	FOREGROUND	thread/sql/event_scheduler	1	YES	NULL	None

At the bottom of the window, there are controls for 'Refresh Rate' (set to 'Don't Refresh'), 'Kill Query(s)', 'Kill Connection(s)', 'Refresh', and 'Show Details'. There are also checkboxes for 'Hide sleeping connections', 'Hide background threads', and 'Don't load full thread info'.

Client Connections and Metadata Locks

The **Client Connections** management window includes a **Show Details** for connections to MySQL 5.6 and higher, as the following figure shows. These details are separated into three tabs:

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

Figure 5.29 Client Connections Details

The screenshot shows the MySQL Workbench interface for 'Local instance MySQL57'. The 'Client Connections' window displays a table of threads and a details panel for a selected thread.

Summary Statistics:

- 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...	

Details Panel (Thread 27):

- 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'
```

Buttons at the bottom: Kill Query(s), Kill Connection(s), Refresh, Hide Details.

- **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 (shown in the following figure) 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 5.30 Metadata Locks Browser

The screenshot shows the MySQL Administration Client Connections browser for a local instance 3306. The interface displays connection statistics and a table of active connections. One connection (Thread ID 28) is highlighted as 'Waiting for table metadata lock'. A detailed view on the right shows the lock information for this connection.

Client Connections Statistics:

- 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...	

Locks Details:

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	Duratic
<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

Options:

- Hide sleeping connections
- Hide background threads
- Don't load full thread info
- Hide Details
- Refresh Rate: Don't Refresh
- Kill Query(s)
- Kill Connection(s)
- Refresh

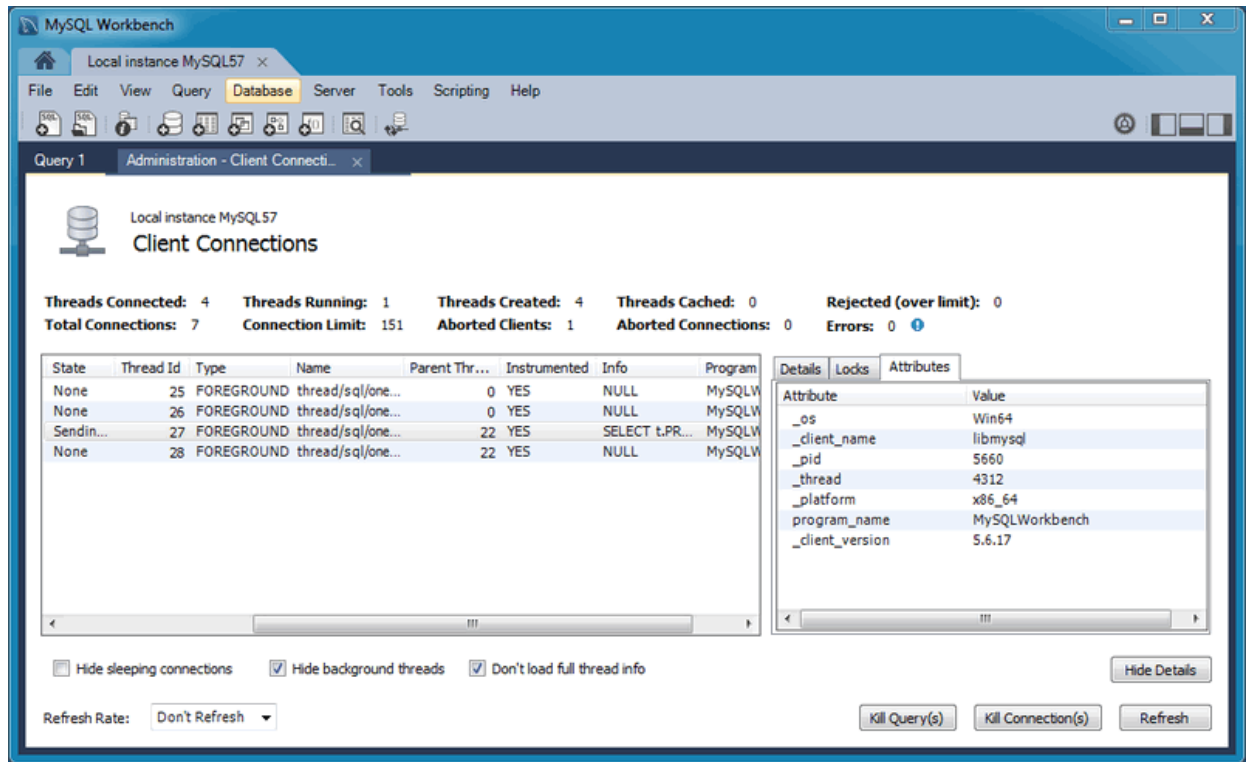


Note

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. The following figure shows a set of attributes and values.

Figure 5.31 Client Connection Attributes



Chapter 6 Administrative Tasks

Table of Contents

6.1 Server Administration	141
6.1.1 Server Logs	143
6.1.2 Service Control	144
6.1.3 Configuration (Options File)	145
6.2 Users and Privileges	146
6.3 Server Status	149
6.4 Status and System Variables	150
6.5 Data Export and Import	153
6.5.1 Table Data Export and Import Wizard	154
6.5.2 SQL Data Export and Import Wizard	162
6.5.3 Result Data Export and Import	166
6.6 MySQL Audit Inspector Interface	167
6.7 MySQL Enterprise Backup Interface	169
6.7.1 General Requirements	170
6.7.2 Online Backup	172
6.7.3 Backup Recovery	174
6.8 MySQL Enterprise Firewall Interface	178
6.9 wbcopytables Utility	180

MySQL Workbench provides a visual interface 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.

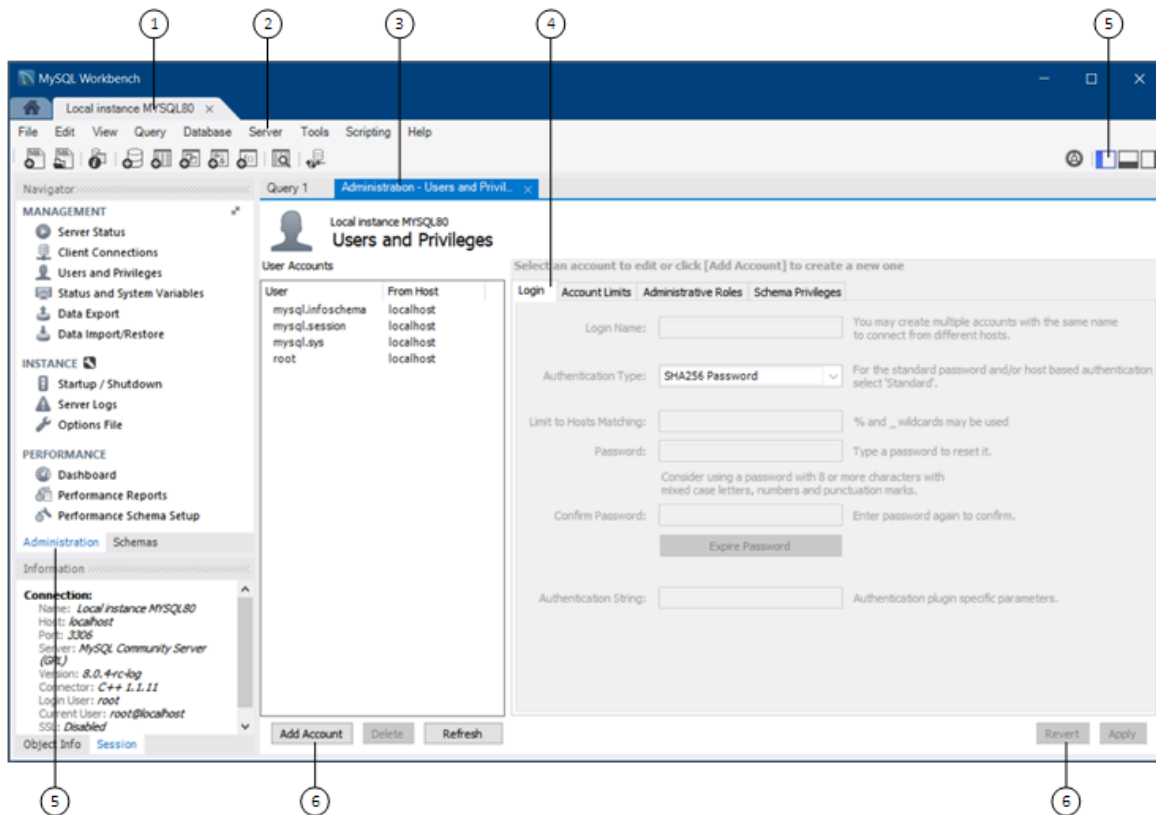
6.1 Server Administration

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.

MySQL Connection Navigator

The Navigator area of the sidebar panel has an **Administration** secondary tab (previously named **Management**) with functionality to monitor and configure your selected MySQL connection. On some hosts, the Navigator and Information labels are not shown. The following figure shows the main elements that apply to server administration.

Figure 6.1 Navigator Administration Tab



Description of the Server Administration Elements

1. Connection tab. Each connection made to the MySQL server is represented by a separate connection tab. A server can be active or inactive when the connection tab for it is opened. For assistance in creating and managing MySQL connections, see [Chapter 5, Connections in MySQL Workbench](#).
2. Server menu. The Server menu on the main menu bar is one way to open each category of administration secondary tab. The menu includes the same set of items listed in the **Administration** tab within the Navigator area, plus two additional items: **Management Access Settings** and **Reset Saved Passwords for Connection**.
To open the **Administration - Users and Privileges** secondary tab (see the previous figure), click **Server** and then **Users and Privileges**.
3. Administration tab. You can display only one administration tab at a time; subsequent administration tabs replace the active tab. To close an open tab, click the **x** on the tab.
4. Operation tabs. Some administration secondary tabs also include subtabs that separate the available operations within that category. For example, the **Administration - Users and Privileges** secondary tab (see the previous figure) has four subtabs: **Login**, **Account Limits**, **Administrative Roles**, and **Schema Privileges**.
5. Sidebar panel. The sidebar panel includes the Navigator and Information areas. With the **Administration** tab selected, click any item in the Navigator area to open the related administration tab in the workspace. You can hide or show the panel using the shortcut action in the main toolbar. The previous figure shows only the sidebar panel. The secondary sidebar and output area panels are hidden in this example.

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

The Navigator area also has a **Schemas** secondary tab for managing databases using your MySQL Connection. For information about the **Schemas** tab, see [Section 8.2.1, “Object Browser and Editor Navigator”](#).

6. Operation buttons. All buttons within the administration tabs are located in same area of the workspace. However, the set of operation buttons changes depending on the selected tab.

6.1.1 Server Logs

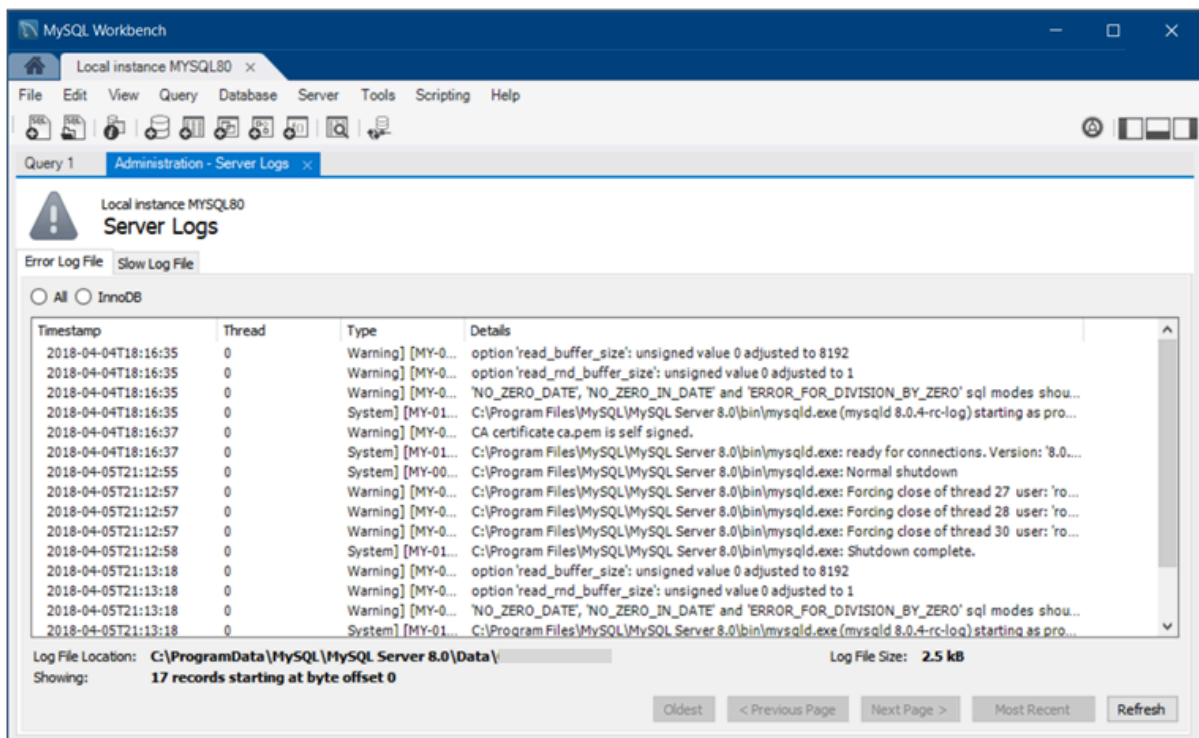
The **Administration - Server Logs** tab displays log information for the MySQL server represented by each connection tab. For each connection, the **Administration - Server Logs** tab includes additional tabs for the general error logs and the slow logs (if available).

With a valid connection established, and the connection tab for it open, you can access log information from either the Navigator area or by clicking **Server** and then **Server Logs** from the menu.

Error Log File

The following figure shows an example of entries within the **Error Log File** tab. For more information, see [The Error Log](#).

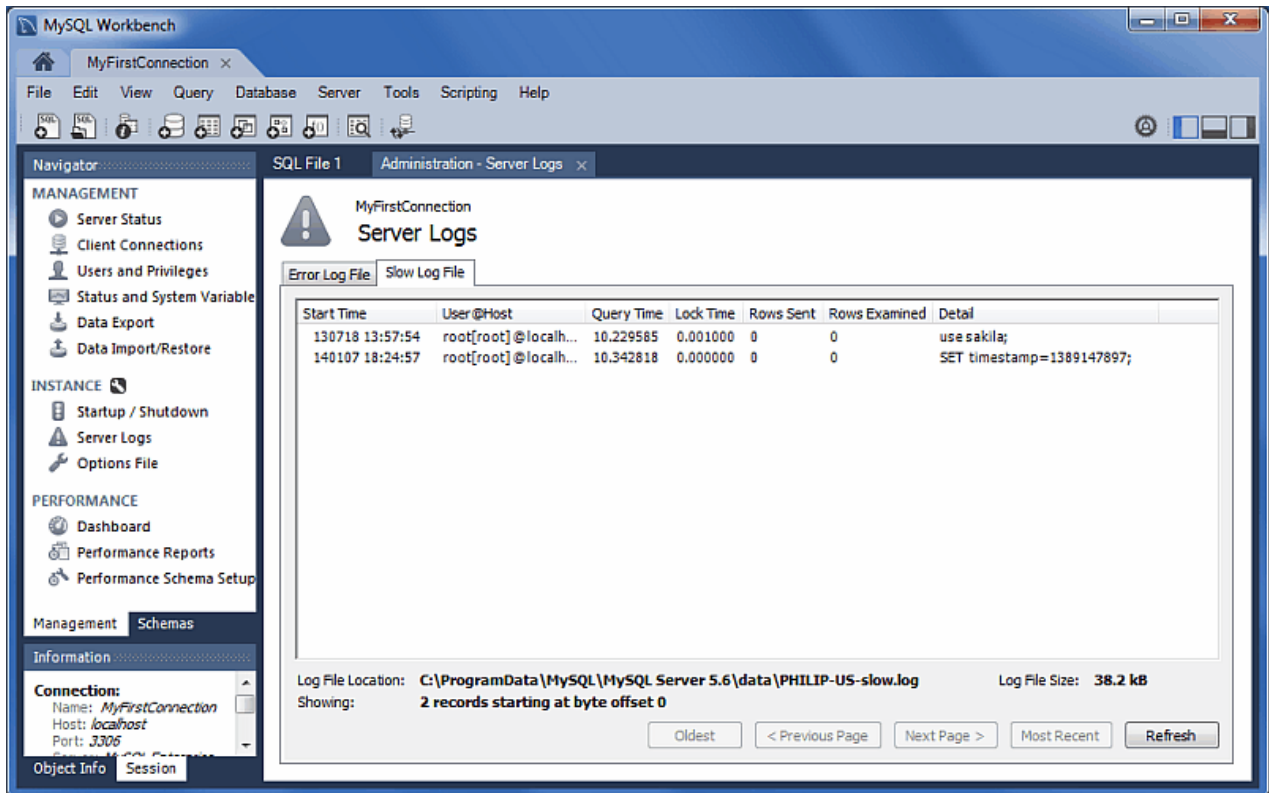
Figure 6.2 Navigator Administration: Instance: Server Logs: Error Log



Slow Log File

The next figure shows an example of entries within the **Slow Log File** tab. For more information, see [The Slow Query Log](#)

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



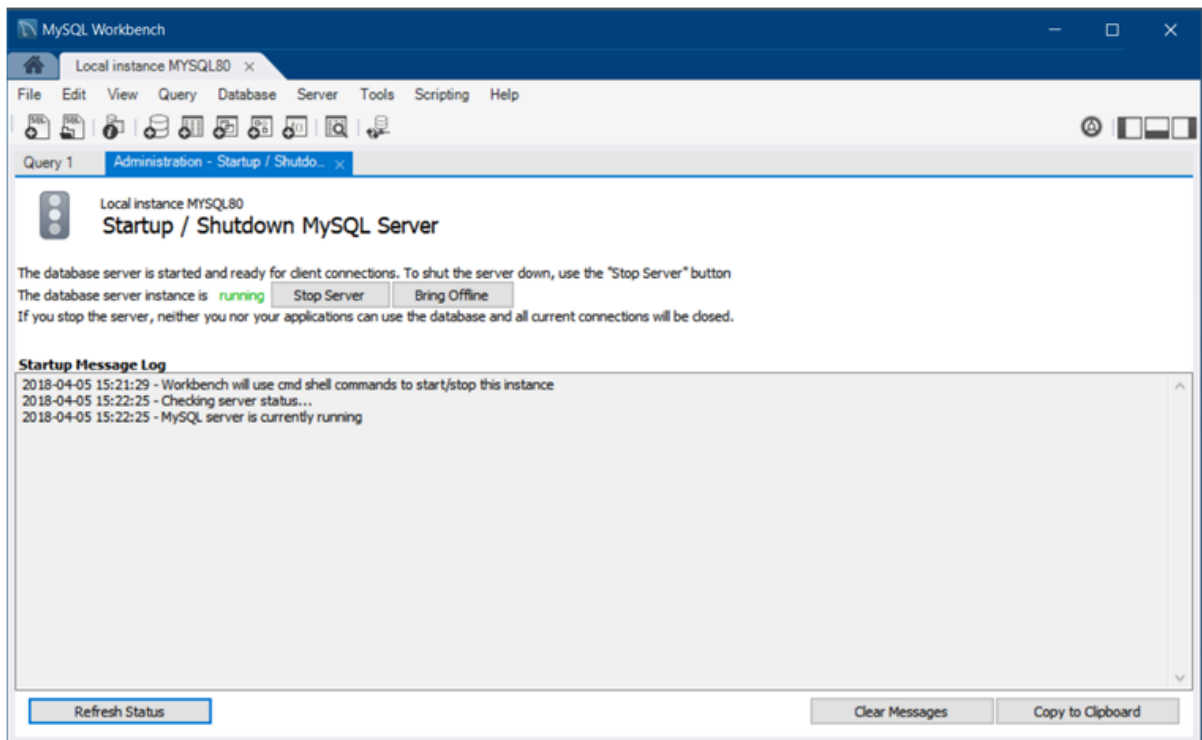
6.1.2 Service Control

The **Administration - Startup / Shutdown** tab enables you to perform the following service-control actions:

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

With a valid connection established, and the connection tab for it open, you can access the control actions from either the Navigator panel or by clicking **Server** and then **Startup/Shutdown** from the menu. The following figure shows an example of the **Administration - Startup / Shutdown** tab with the database server instance running and the **Stop Server** and **Bring Offline** buttons displayed.

Figure 6.4 Navigator Administration: Instance: Startup / Shutdown



6.1.3 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 and macOS) 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 so on). Make an edit and click **Apply** to commit the changes.

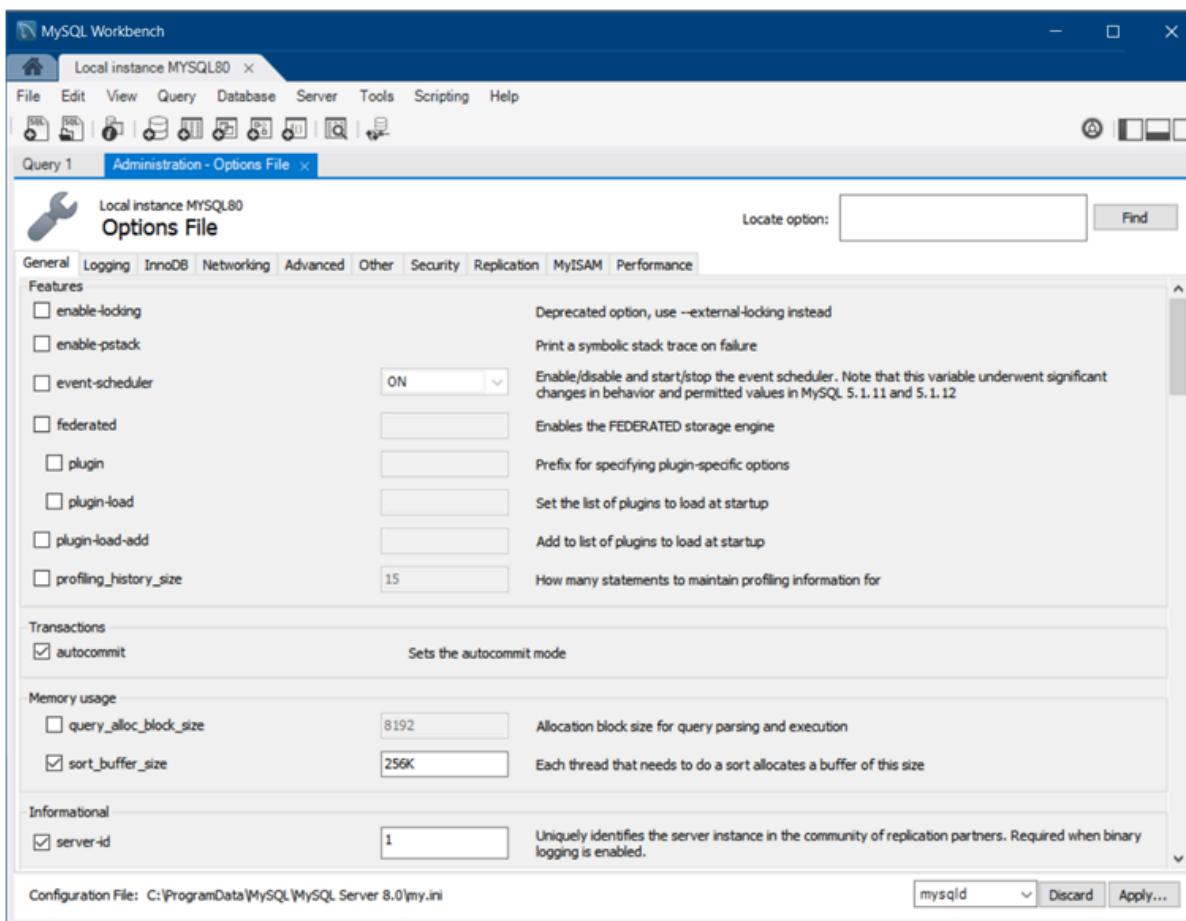
You can open the **Administration - Options File** secondary tab from either the Navigator area of the sidebar panel or by clicking **Server** and then **Options File** from the menu.

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](#).

The following figure shows an instance of the **Options File** with the **General** tab selected.

Figure 6.5 Navigator Administration: Instance: Options File: General



6.2 Users and Privileges

The **Administration - Users and Privileges** tab provides a list of all users and privileges that relate to an active MySQL server instance. From this tab, you can add and manage user accounts, adjust privileges, and expire passwords.

To open the **Administration - Users and Privileges** tab:

1. Establish a connection to an active MySQL server instance.
2. Within the connection tab, do one of the following:
 - Click **Users and Privileges** from the **Management** list within the Navigator area.
 - Click **Server** and then **Users and Privileges** from the menu.

The **Administration - Users and Privileges** tab has several task areas, which are described in the following sections:

- [User Accounts](#)
- [Login Tab](#)
- [Account Limits Tab](#)

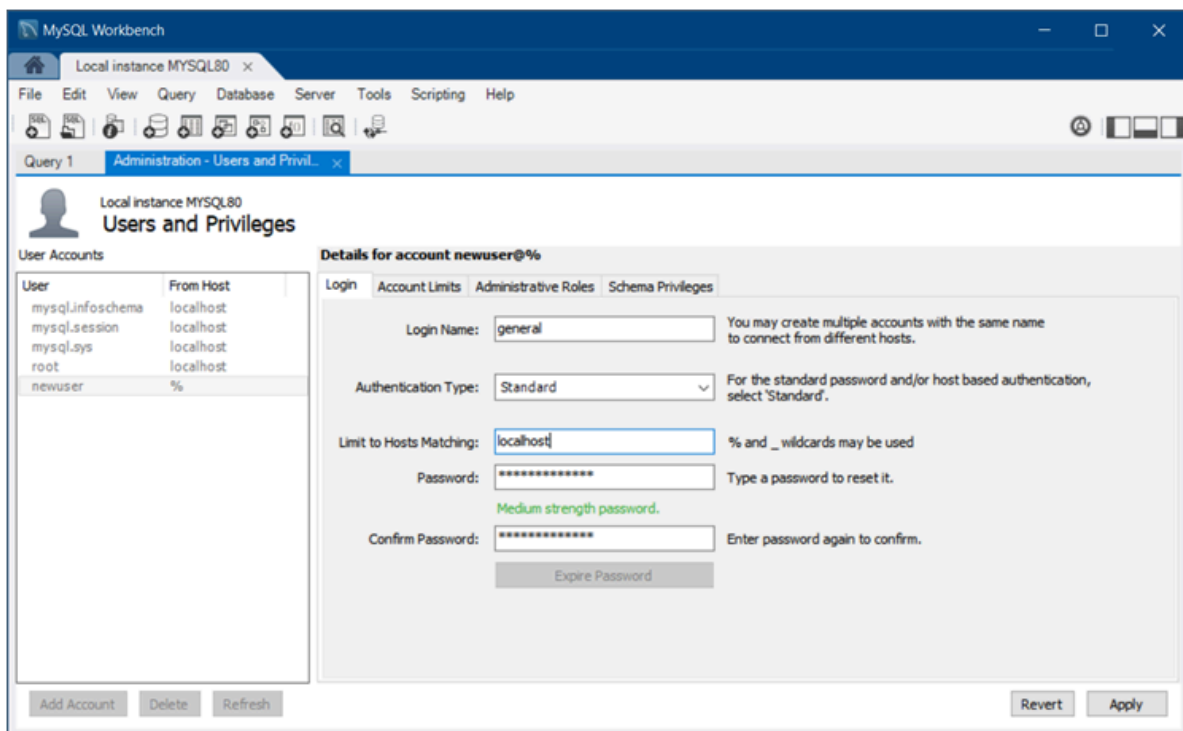
- [Administrative Roles Tab](#)
- [Schema Privileges Tab](#)

User Accounts

User Accounts consists of a vertical box that lists each user account associated with the active MySQL connection. The list contains each user name and the host name where the account resides. Use the **Add Account**, **Delete**, and **Refresh** buttons to manage the list of user accounts. Selecting an account from the list focuses the account details, which appear in set of tabs, onto the selected user account.

The figure that follows shows the layout of the **Administration - Users and Privileges** tab with the **Login** detail tab open to show an example of general account information.

Figure 6.6 Navigator Administration: User And Privileges with Login Tab Open



For a description of the **Login**, **Account Limits**, **Administrative Roles**, and **Schema Privileges** tabs, see the related sections.

Login Tab

The **Login** tab provides the following information related to the selected user account:

- **Login Name:** You may create multiple accounts with the same name to connect from different hosts.
- **Authentication Type:** For standard password or host-based authentication, select [Standard](#). The [caching_sha2_password](#) and [SHA256_Password](#) authentication types provide more secure password encryption than the [Standard](#) authentication type.

Starting with MySQL 8.0.4, the [caching_sha2_password](#) plugin is the default authentication plugin for the server. An account that authenticates with [caching_sha2_password](#) must use either a secure connection or an unencrypted connection that supports password exchange using an RSA key pair.

- **Limit to Hosts Matching:** The % and _ characters may be used as wildcards. The percent sign (%) matches zero or more characters and the underscore (_) matches a single character.
- **Password and Confirm Password:** To reset a password, type in the new password and then confirm it. Consider using a password of eight or more characters with mixed-case letters, numbers, and punctuation marks.

Use **Expire Password** to require a change of password to use the account.

Account Limits Tab

The **Account Limits** tab defines the following limits on the selected user account:

- **Max. Queries:** The number of queries the account can execute within one hour.
- **Max. Updates:** The number of updates the account can execute within one hour.
- **Max. Connections:** The number of times the account can connect to the server within an hour.
- **Concurrent Connections:** The number of simultaneous connections to the server the account can have.

Administrative Roles Tab

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 account or to assign privileges directly to an account without first assigning roles.

After you select a role for a user account, you will see the accumulated privileges in the **Global Privileges** panel. For example, if you select the role `BackupAdmin`, the privileges granted include `EVENT`, `LOCK TABLES`, `SELECT`, `SHOW DATABASES`. For a complete list of privileges, see [Privileges Provided by MySQL](#)

The **Administrative Roles** tab includes the following roles:

- `DBA`: Grants the rights to perform all tasks.
- `MaintenanceAdmin`: Grants rights to maintain the server.
- `ProcessAdmin`: Grants rights to assess, monitor, and kill user processes.
- `UserAdmin`: Grants rights to create user logins and reset passwords.
- `SecurityAdmin`: Grants rights to manage logins and grant and revoke server privileges.
- `MonitorAdmin`: Grants the minimum rights to monitor the server.
- `DBManager`: Grants full rights on all databases.
- `DBDesigner`: Grants rights to create and reverse-engineer any database schema.
- `ReplicationAdmin`: Grants rights needed to set up and manage replication.
- `BackupAdmin`: Grants minimum rights required to back up any database.
- `Custom`: Lists other (custom) privileges that are assigned to the user account. This role is not available for all default accounts, such as `root`. If you select a user account and then select one or more privileges directly that are outside of any selected roles, the **Custom** role is added (and selected) to the list of roles.

To remove all of the rights assigned to the selected user account, click **Revoke All Privileges**.

Schema Privileges Tab

The **Schema Privileges** tab refines the way you assign access rights to one or more schemas by user account. To assign privileges to the selected account by schema, do the following:

1. Add a schema entry (or rule) that specifies which schema or schemas apply the selected user account. Click **Add Entry** to open the New Schema Privilege Definition dialog. The dialog provides the following independent options to select:
 - **All Schema (%)** - This rule applies to any schema name.
 - **Schemas matching pattern: *pattern*** - Apply this rule to schemas that match the given name or pattern. You may use `_` and `%` as wildcards in the pattern; however, to use the literal value you must escape each wildcard character with a backslash (`\`).
 - **Selected schema: *schema name*** - Apply the rule to the specific schema name selected from the list.

Use **Delete Entry** to remove an entry and the privileges associated with it from the list. When you click **Revoke All Privileges**, you are prompted remove all privileges assigned to the selected user account.

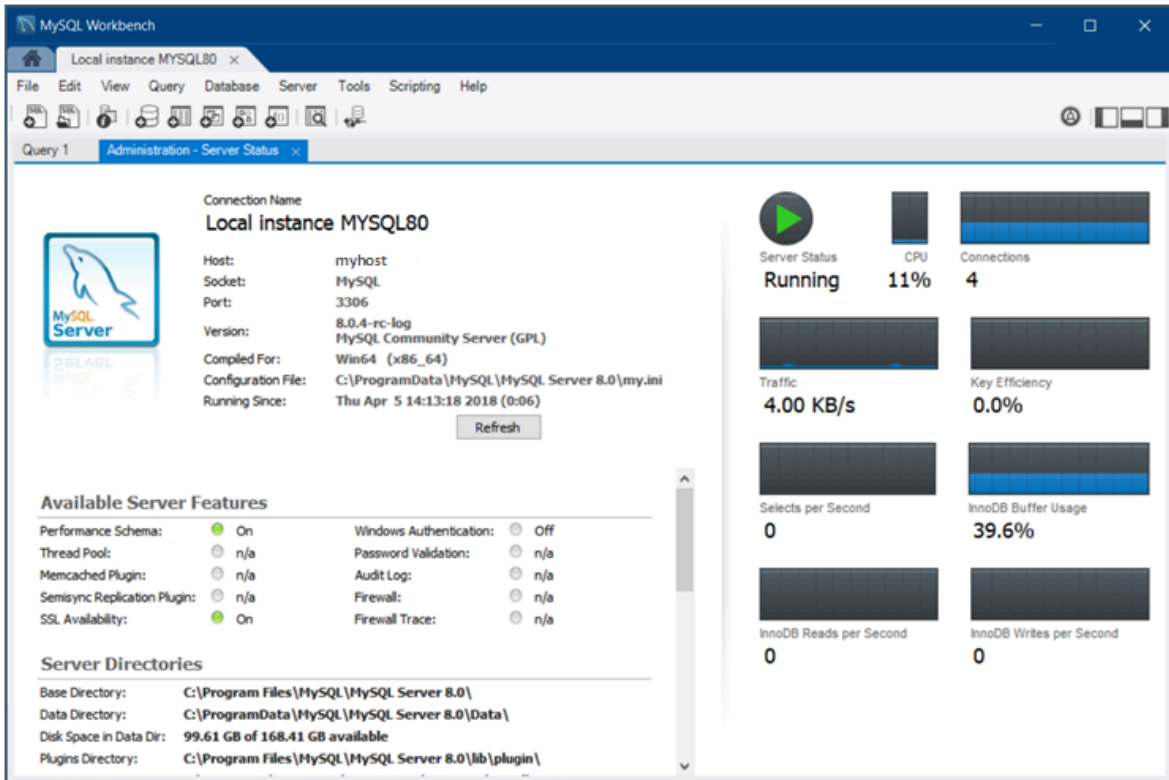
2. With an entry selected, mark the individual access rights that apply only to the schema or schemas defined in the entry. The access rights are categorized as Object Rights, DDL Rights, and Other Rights. Each right that you select appears in the **Privileges** column of the schema entry.

6.3 Server Status

Get an immediate view into the basic health indicators and counters for your MySQL environment. As the next figure shows, 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.

You can access this window from either the Navigator area, or by selecting **Server** and then **Server Status** from the menu.

Figure 6.7 Navigator Administration: Server Status

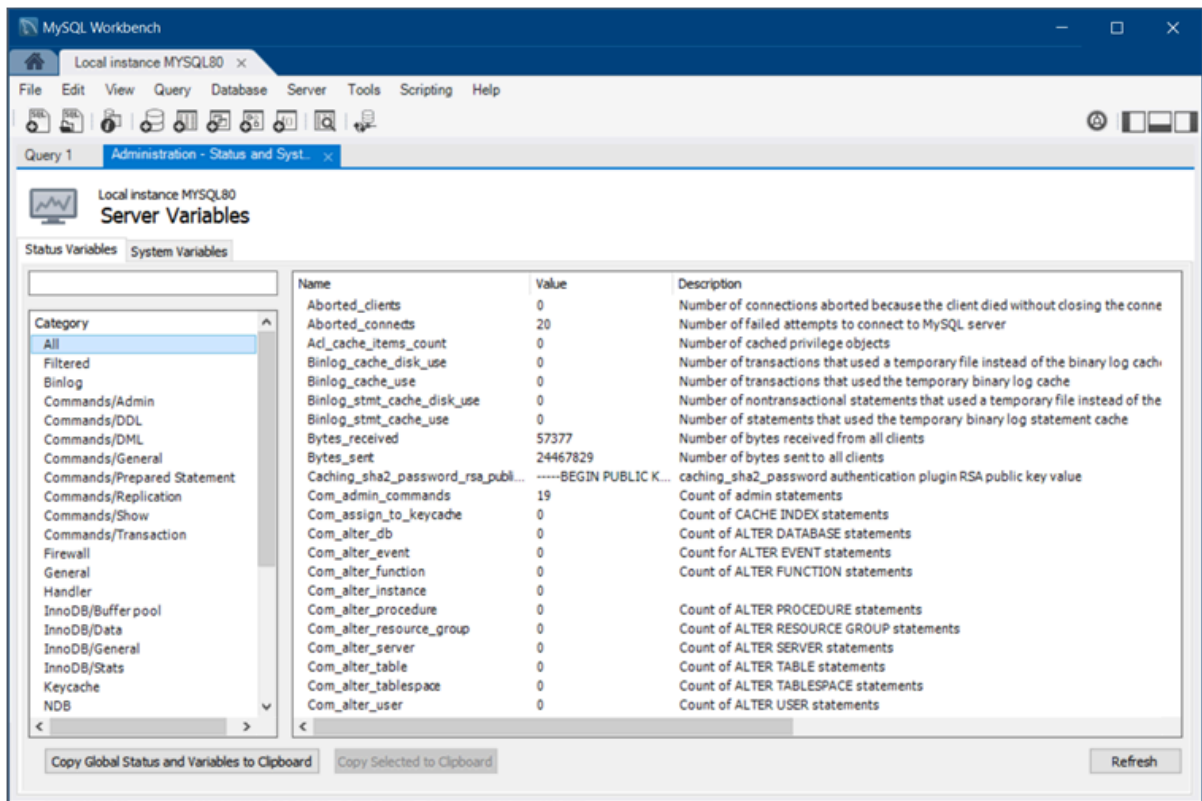


6.4 Status and System Variables

The **Administration - Status and System Variables** secondary tab lists the full set of server variables for the active MySQL connection. You may also copy all or selected variables to your clipboard.

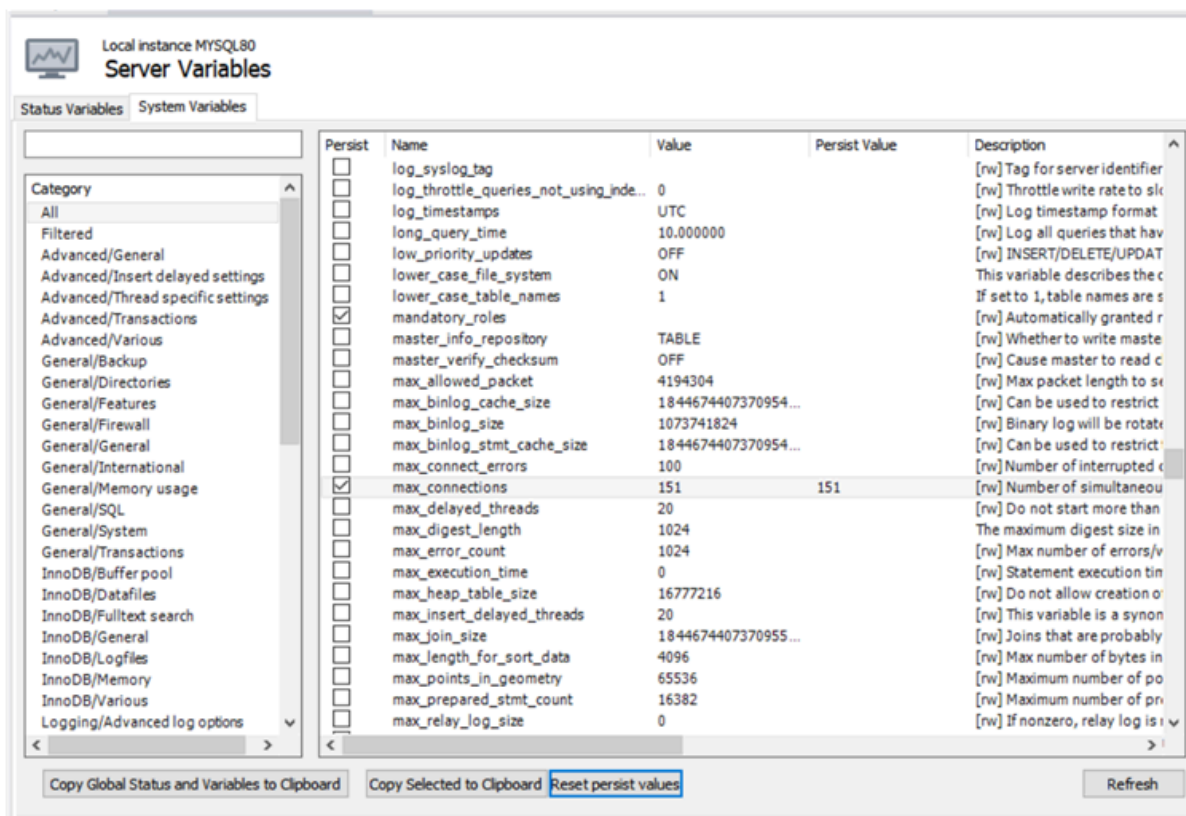
You can open this secondary tab from either the Navigator area (see MANAGEMENT) or by clicking **Server** and then **Status and System Variables** from the menu. The following figure shows the **Status Variables** subtab selected with all of the status variable listed by name. Each variable has a value, if applicable, and a description.

Figure 6.8 Navigator Administration: Status Variables



The next figure shows the **System Variables** subtab selected with all of the global system variables for the active server listed by name. You can refine the list of status and system variables by typing the variable name into the text box provided or by selecting a category, such as [InnoDB/General](#).

Figure 6.9 Navigator Administration: System Variables



Persist System Variables

Beginning with MySQL Workbench 8.0.11, you can set one or more global system variables to persist across server restarts. To persist a variable, select the **Persist** check box next to the name (see the previous figure). For system variables that include a value, the value is displayed in the **Persisted Value** column of the list after you select the check box. If a variable is not eligible to be persisted, an informational dialog box appears when you select the check box.

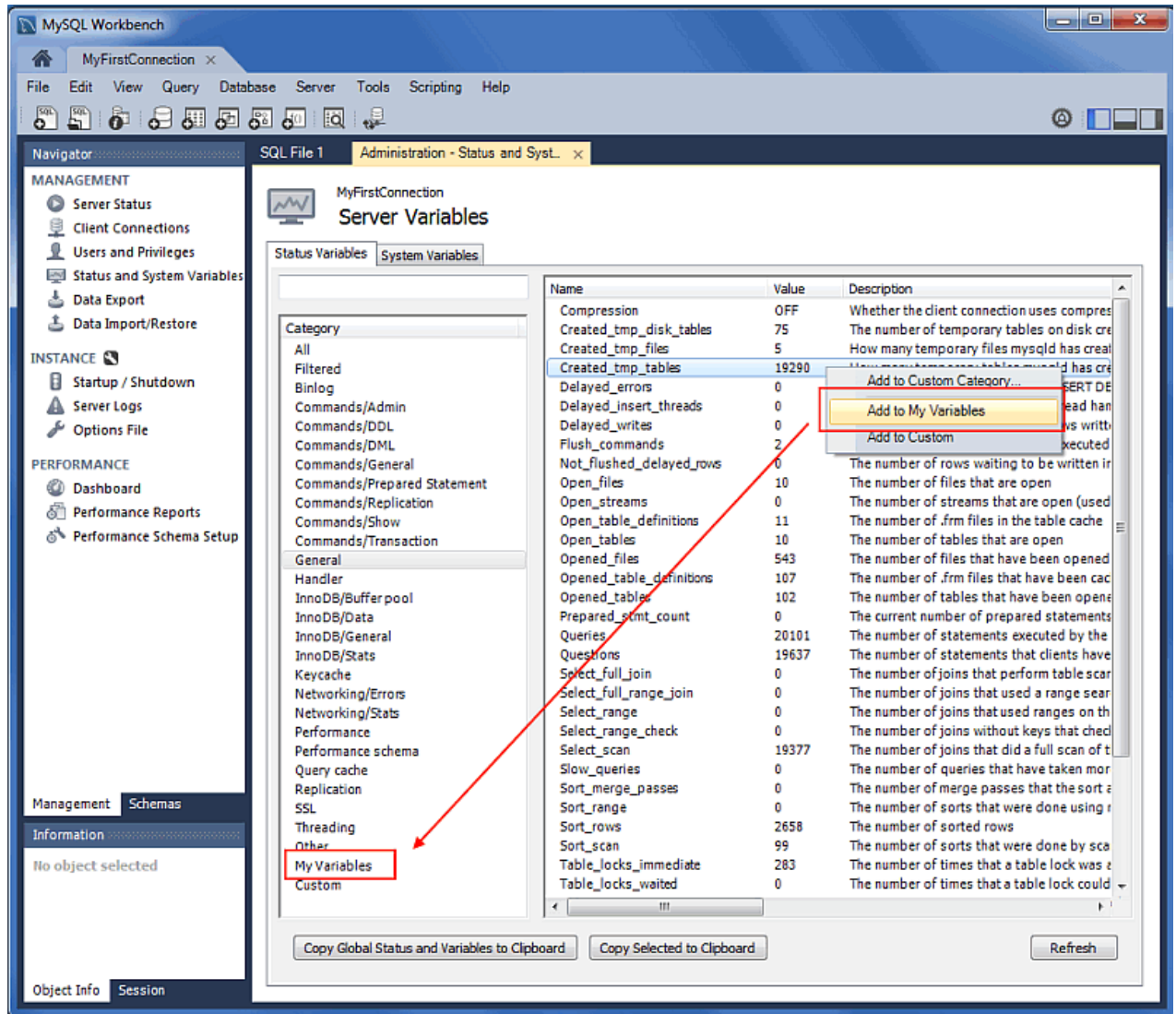
To reset a persistent global system variable, deselect the individual check box and then confirm the reset action for the individual variable in the dialog box that opens. To reset all the persistent variables at once, click **Reset persist values** and then confirm the reset action for all of the persistent variables.

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 a variable, choose the custom group (or create a new one), and then add the variable to the aforementioned group. Your custom groups are listed along with the pre-existing groups.

The following figure shows an example of a custom group titled *My Variables*, which is being added to the *Created_tmp_variables* variable.

Figure 6.10 Navigator Management: Adding A Variable To A Custom Group



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.

GUI Location	Data Set	Export Types	Import Types	Additional Details
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**, as the next figure shows.

Figure 6.11 Table Data Wizards: Open

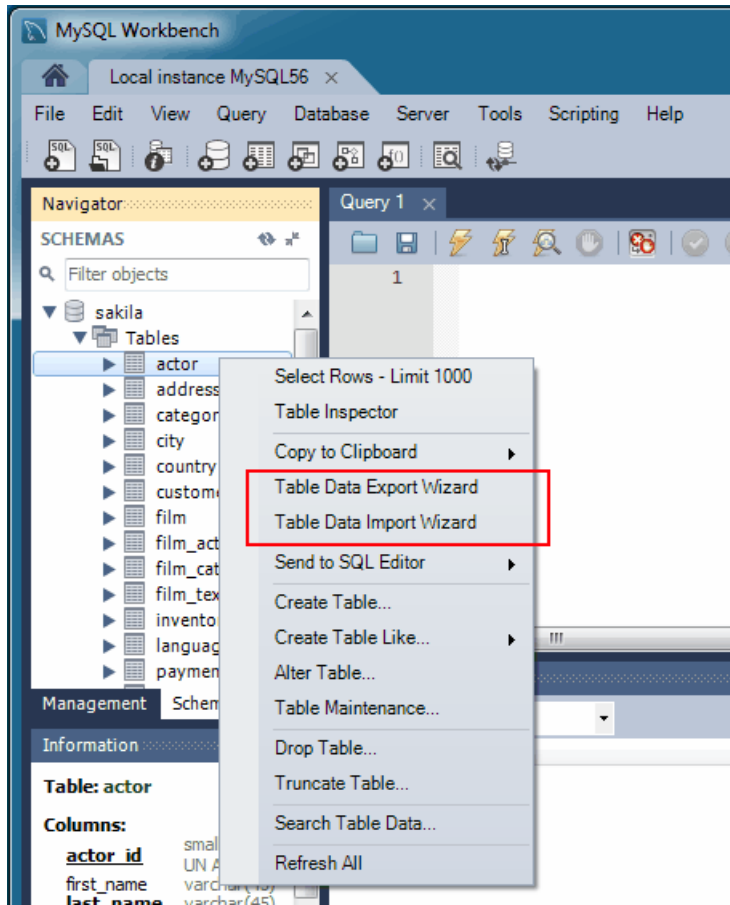


Table Data Export Wizard

Export table data to either a JSON or CSV file. The following figures show an example of an export sequence of the *sakila.actor* table to a CSV file.

Figure 6.12 Table Data Export: Source

Select data for export

Select source table for export.

sakila.actor

Select columns you'd like to export

Export	Column name
<input checked="" type="checkbox"/>	actor_id
<input checked="" type="checkbox"/>	first_name
<input checked="" type="checkbox"/>	last_name
<input checked="" type="checkbox"/>	last_update

Select / Deselect all entries

Row Offset: Count:

Advanced >> < Back Next > Cancel

Figure 6.13 Table Data Export: CSV Configuration

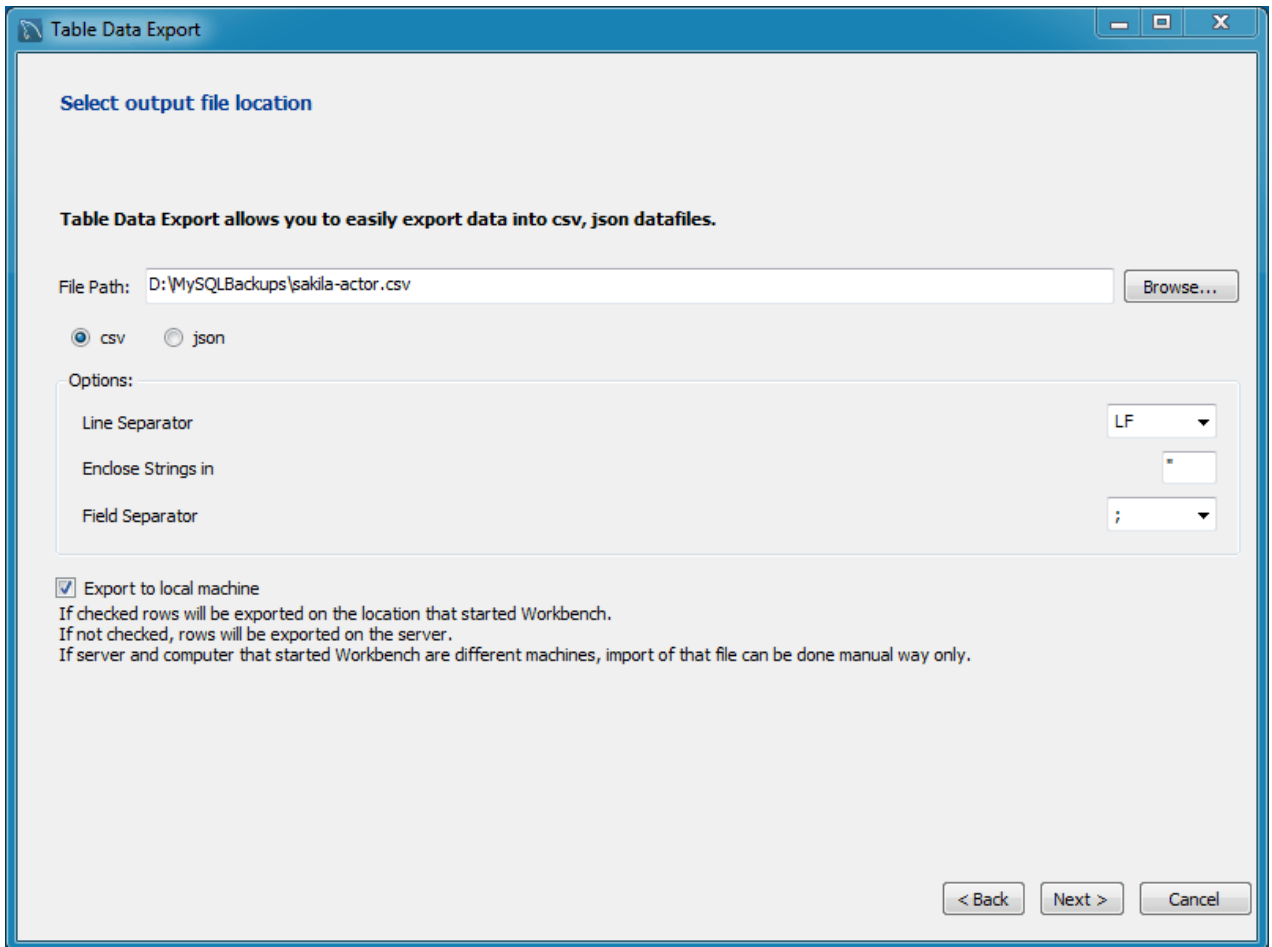


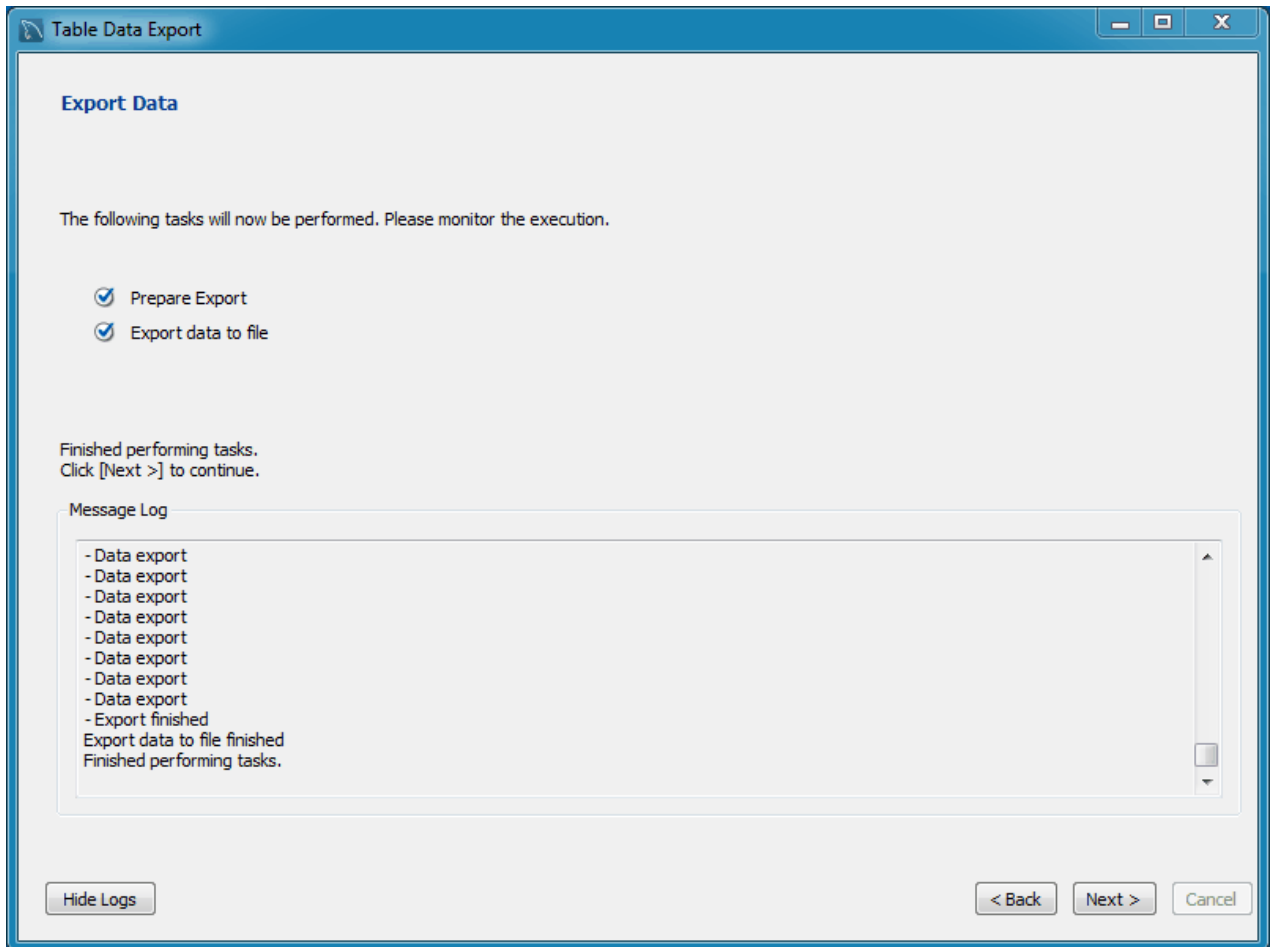
Figure 6.14 Table Data Export: Results

Table Data Import Wizard

You can import table data from either a JSON or CSV file. The following figures show an example of an import sequence of the *sakila.actor* table from a CSV file.

Figure 6.15 Table Data Import: CSV Source

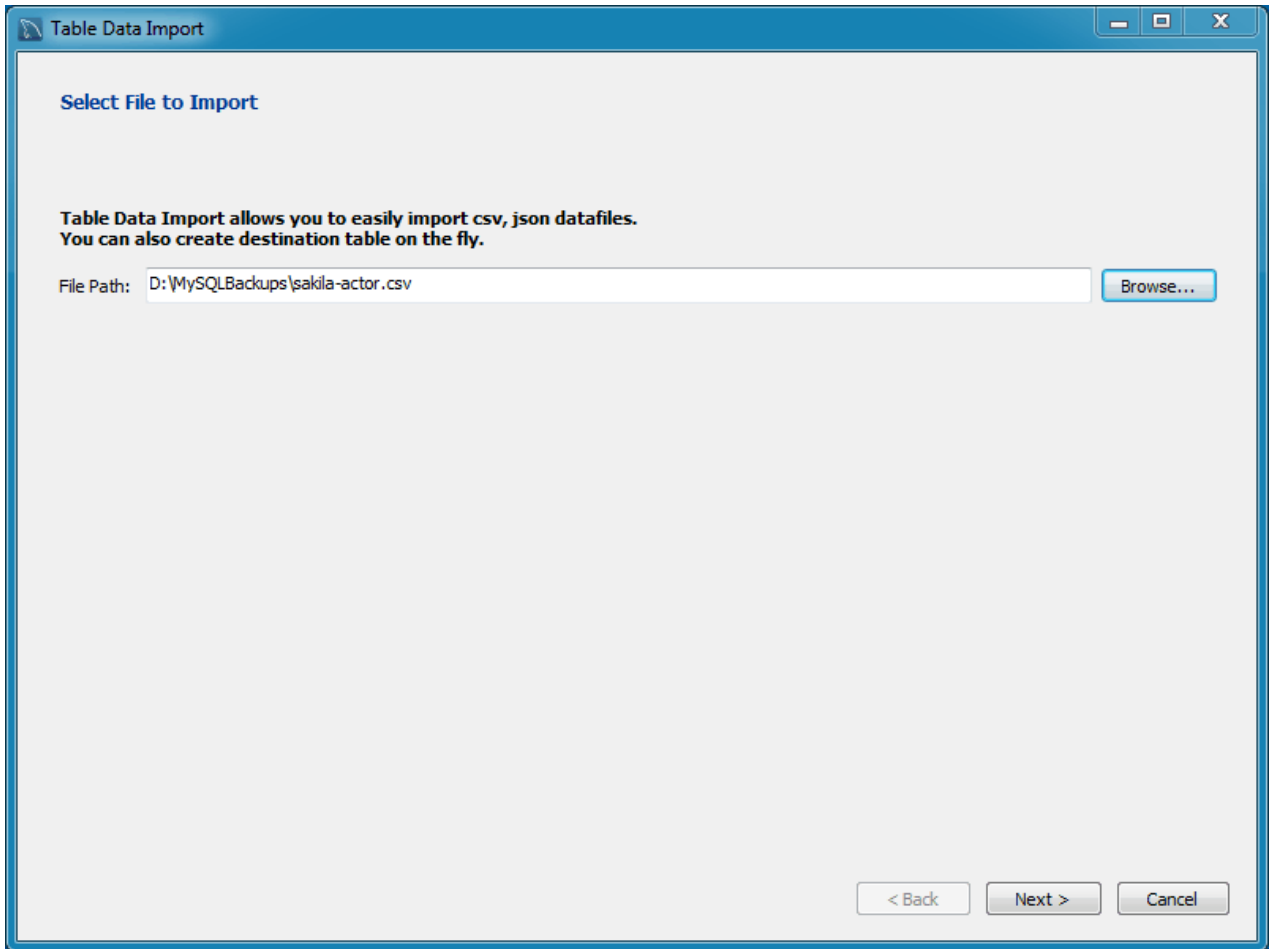


Figure 6.16 Table Data Import: Destination Table

Table Data Import

Select Destination

Select destination table and additional options.

Use existing table: sakila.actor

Create new table: sakila . sakila-actor-other

Drop table if exists

< Back Next > Cancel

Figure 6.17 Table Data Import: CSV Configuration

Table Data Import

Configure Import Settings

Detected file format: csv

Options:

Line Separator: LF

Endose Strings in: "

Field Separator: ;

Encoding: utf-8

Columns:

<input checked="" type="checkbox"/>	Source Column	Field Type
<input checked="" type="checkbox"/>	actor_id	int
<input checked="" type="checkbox"/>	first_name	text
<input checked="" type="checkbox"/>	last_name	text
<input checked="" type="checkbox"/>	last_update	text

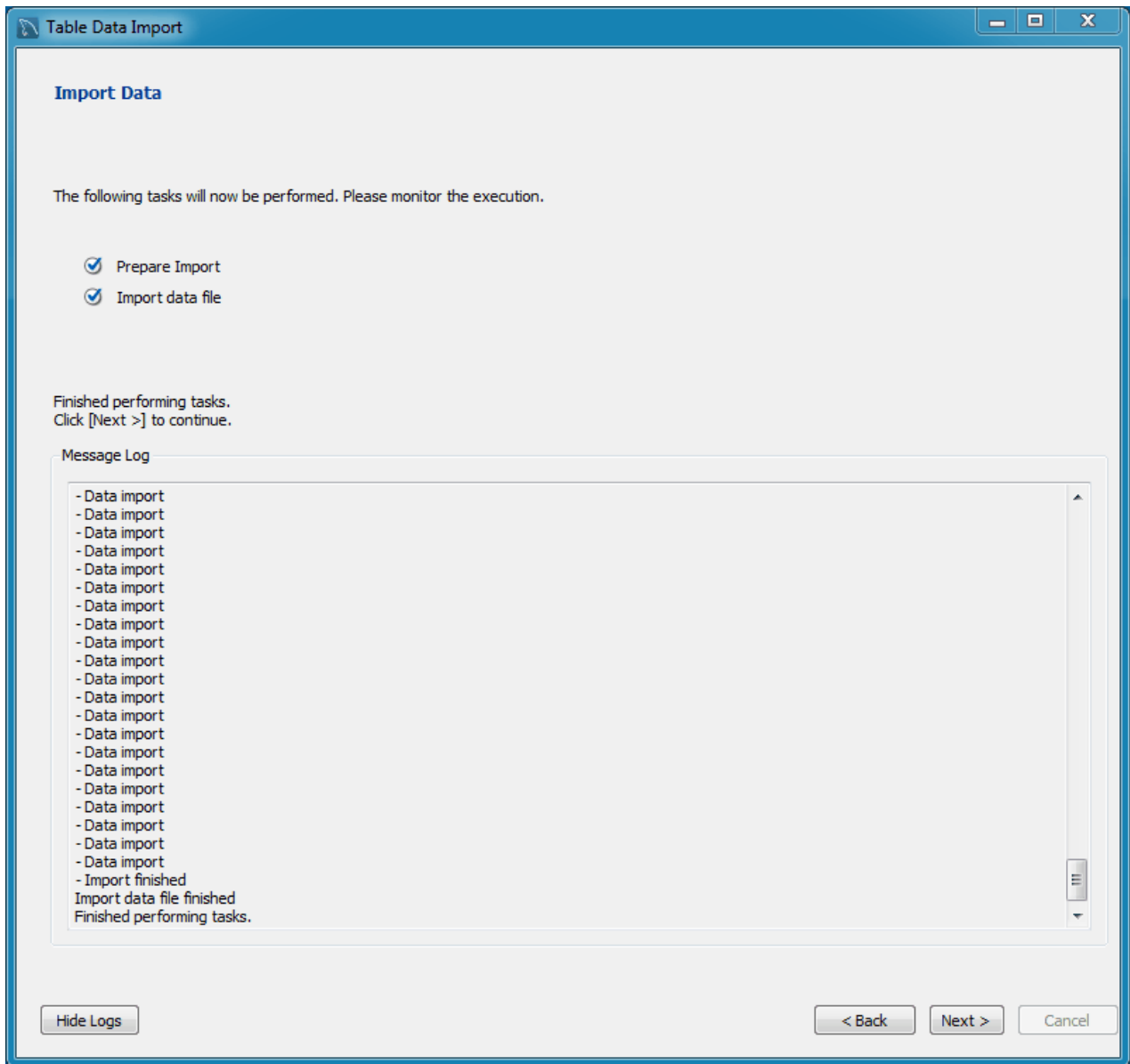
actor_id	first_name	last_name	last_update
1	PENELOPE	GUINNESS	2006-02-15...
2	NICK	WAHLBERG	2006-02-15...
3	ED	CHASE	2006-02-15...
4	JENNIFER	DAVIS	2006-02-15...
5	JOHNNY	LOLLOBRIG...	2006-02-15...

< Back Next > Cancel

**Note**

The **Encoding** field should correspond with your CSV file.

Figure 6.18 Table Data Import: Results



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 area of the sidebar, 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.

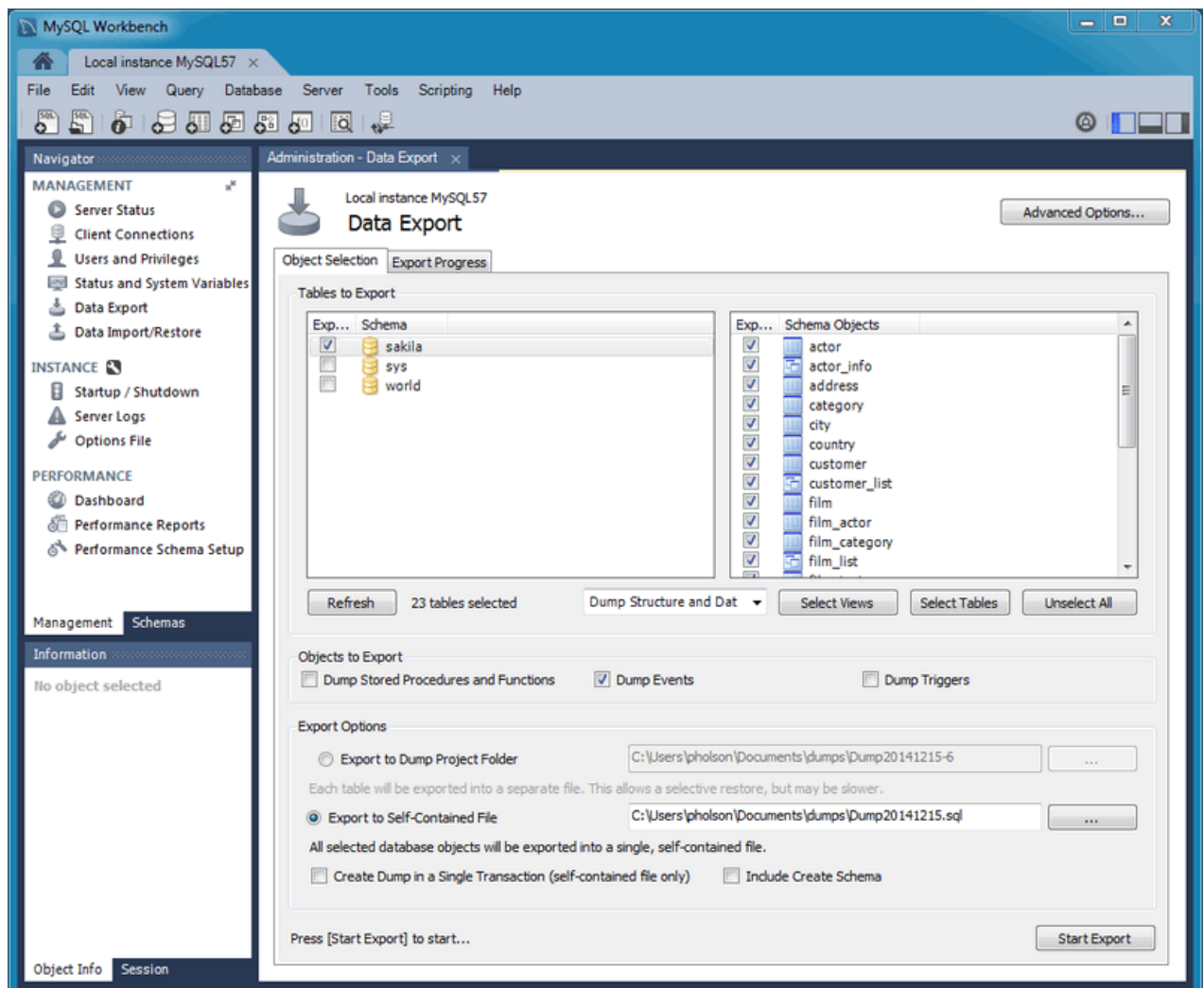
Select the schema objects to export and then configure the related options. The figure that follows shows the `sakila` database ready for export.



Note

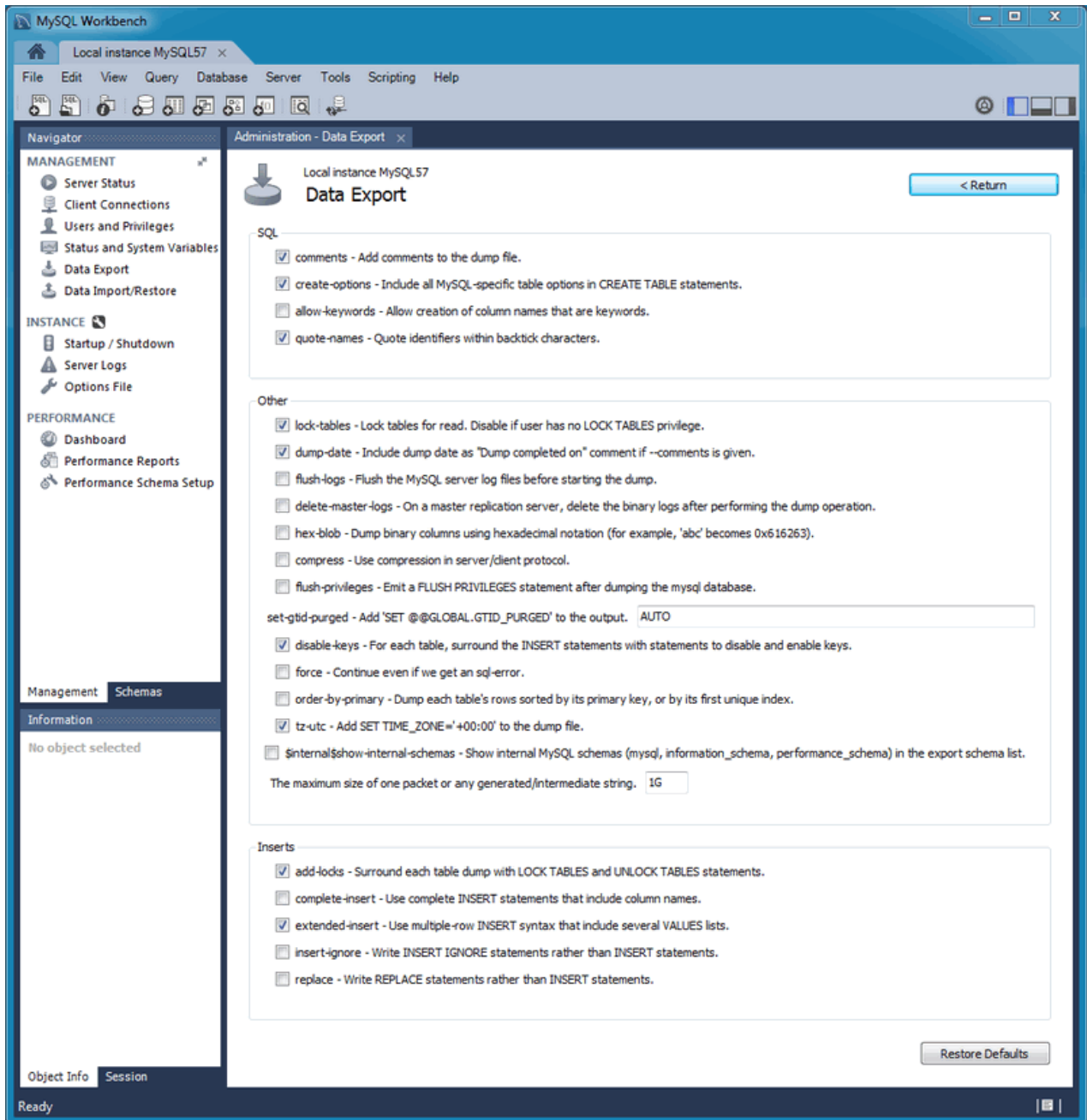
Click **Refresh** to load the current objects.

Figure 6.19 Navigator Administration: Data Export: Object Selection



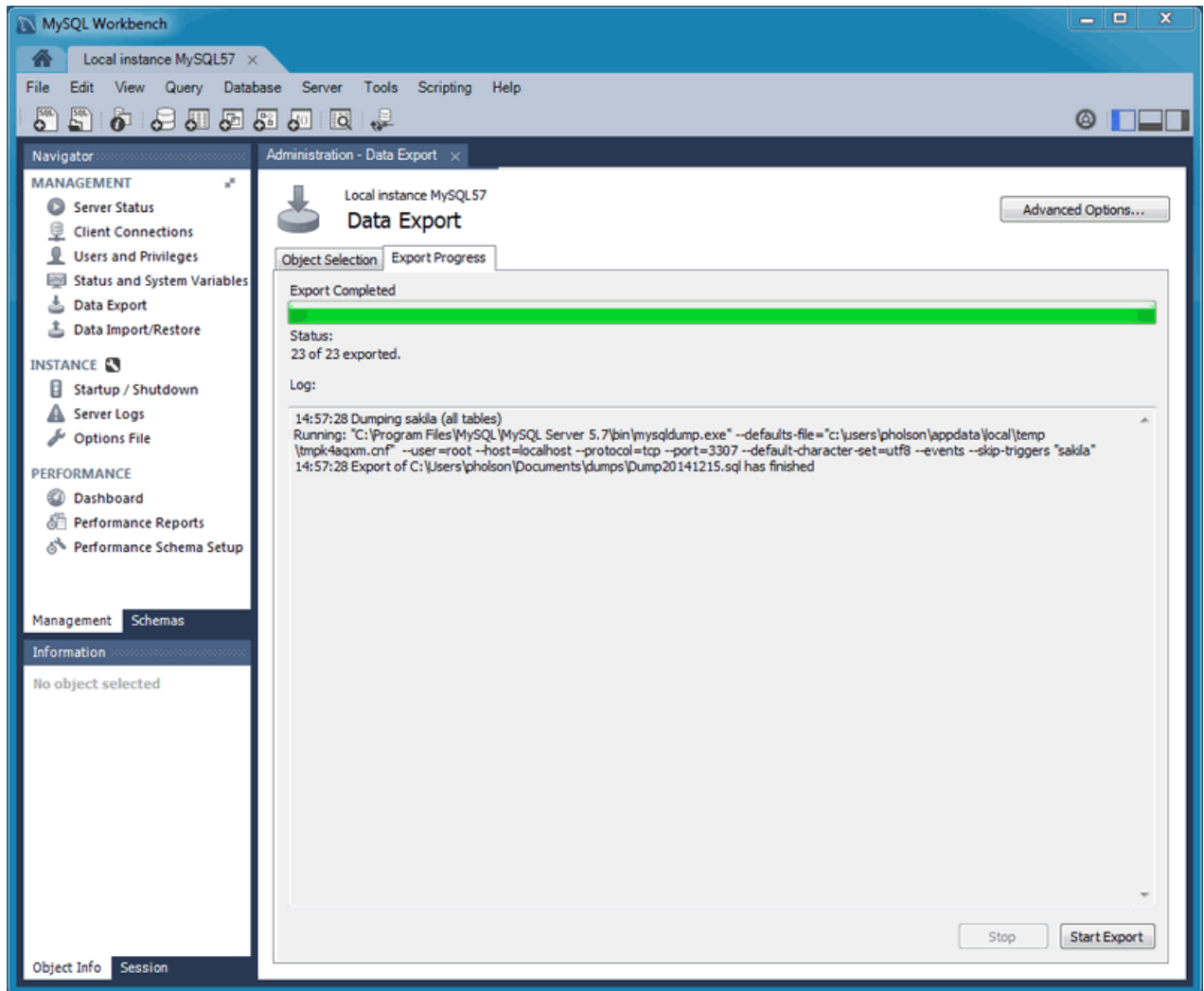
Optionally open the **Advanced Options** tab that allows you to refine the export operation. The next figure shows an example that adds table locks, uses replace instead of insert statements, quotes identifiers with backtick characters, and so on.

Figure 6.20 Navigator Administration: Data Export: Advanced Options



Click **Start Export** to begin the export process. As the next figure shows, status information indicates when the export is finished.

Figure 6.21 Navigator Administration: Data Export: Export Progress



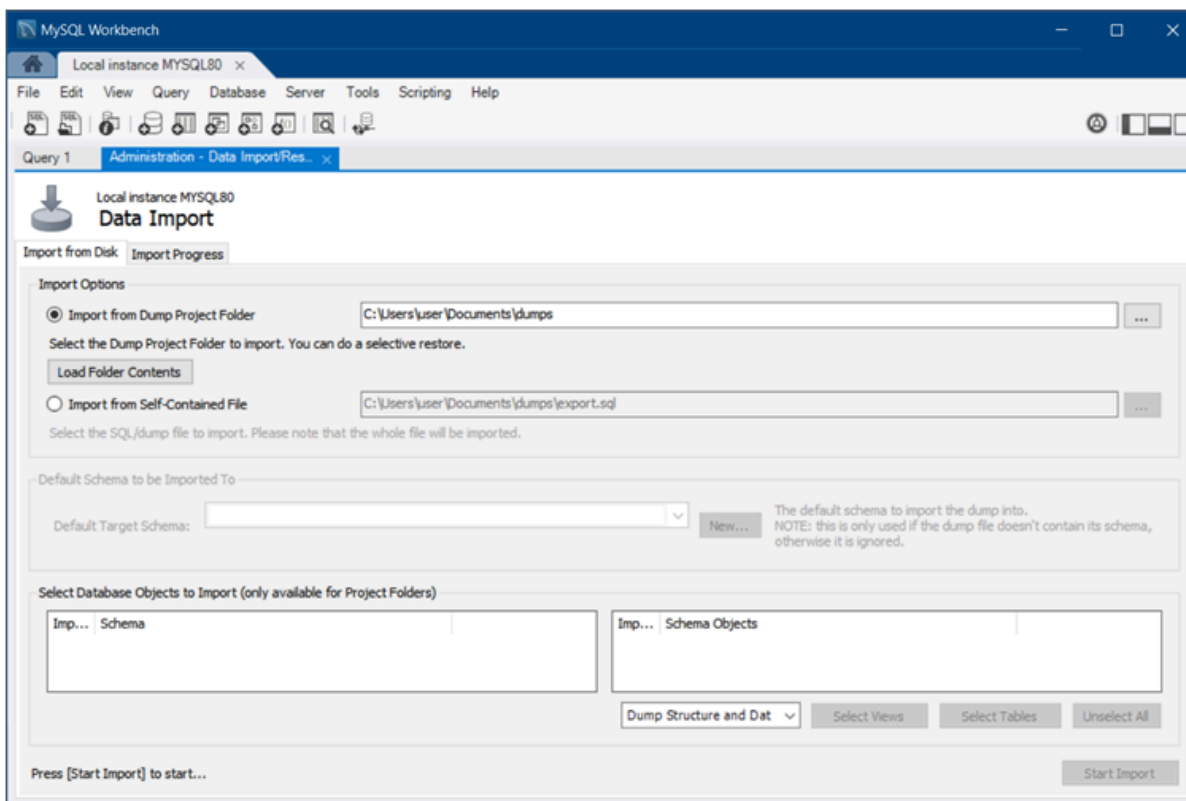
This functionality uses the `mysqldump` command.

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. The following figure shows an example of an import from a dump project folder.

Figure 6.22 Navigator Administration: Data Import: Import From Disk

**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. Use the **Import Progress** tab to monitor the progress. Status information indicates when the import is finished and displays the log.

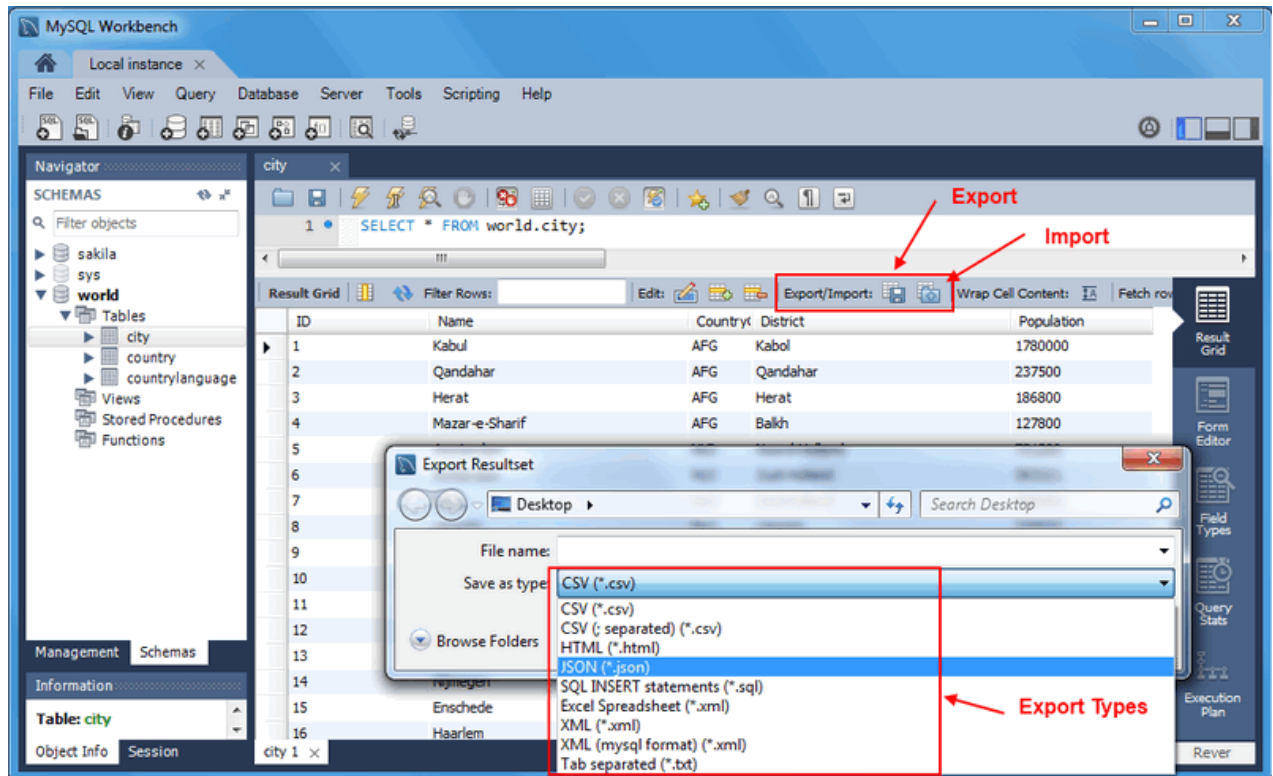
6.5.3 Result Data Export and Import

Export or Import data directly from or into the result set. The export and import operations shown in this section apply to result data only. For an overview of the various data export and import options in MySQL Workbench, see [Section 6.5, “Data Export and Import”](#).

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. The following figure shows the selection of JSON as the export type from within the Export Resultset window. By default, MySQL Workbench exports your result set using Unix-style line endings (LF), rather than Windows-style (CR/LF) or macOS-style (CR) line endings.

Figure 6.23 Exporting a Result Set



Import into a Result Set

Records from a CSV file can be imported into the result set of the visual SQL editor. The import icon opens the Table Data Import dialog from which you can select a data file and other options, such as the destination table. The wizard-like dialog also enables you to detect and modify the configuration settings of the CSV file.



Note

The result set must have a unique row identifier (such as a Primary Key or NOT NULL unique index) as otherwise values cannot be imported because the result set will be read-only.

6.6 MySQL Audit Inspector Interface

MySQL Workbench offers a graphical interface to MySQL Enterprise Audit. By default, the `audit_log` plugin is not installed.

To activate the `audit_log` plugin manually using `mysql`, load the plugin code from the library file at runtime with the `INSTALL PLUGIN` statement. This statement requires the `INSERT` privilege for the `mysql.plugin` table.

- On Windows:

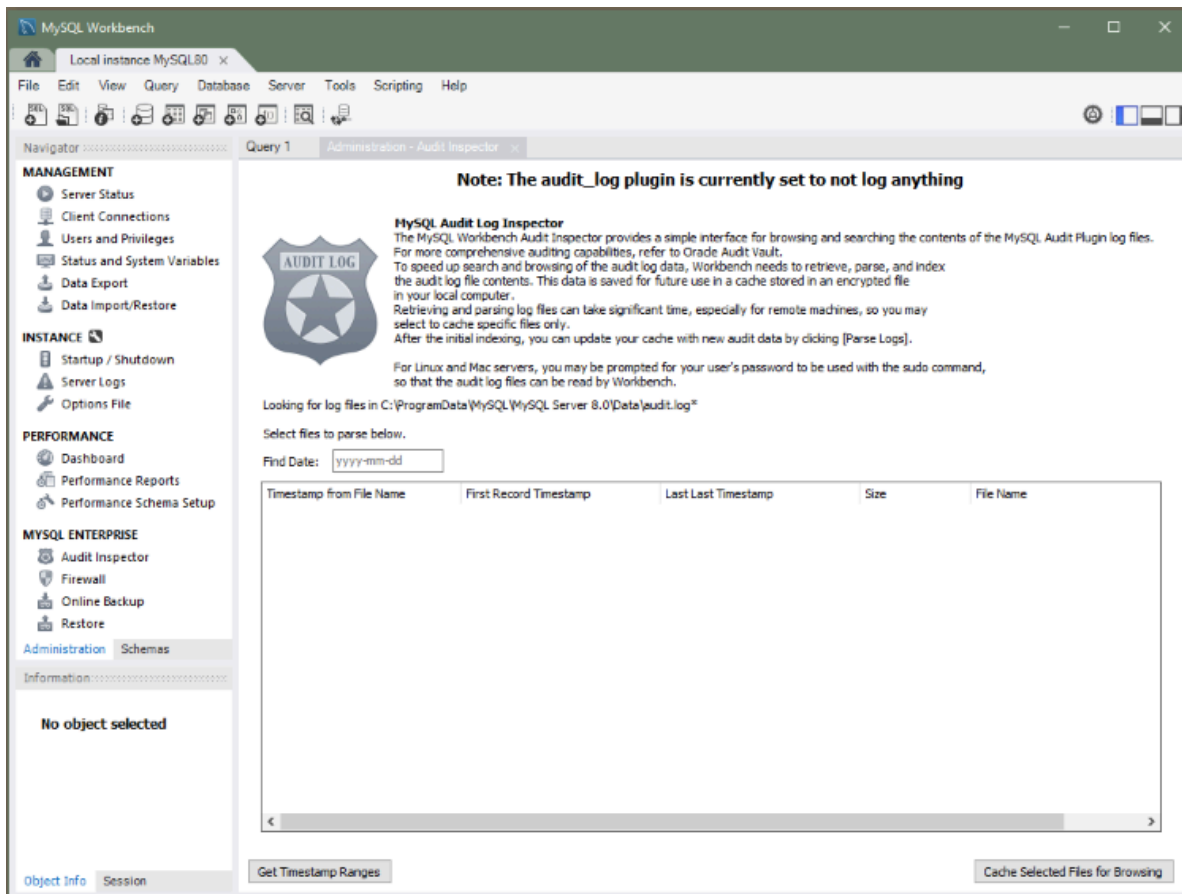
```
INSTALL PLUGIN audit_log SONAME 'audit_log.dll'
```

- On Linux and macOS:

```
INSTALL PLUGIN audit_log SONAME 'audit_log.so'
```

After the plugin is installed, start MySQL Workbench and select the **Administration** tab in the navigator. Under **MySQL Enterprise**, click **Audit Inspector**. The initial **Administration - Audit Inspector** tab looks similar to the figure that follows. At this point, no entries are listed.

Figure 6.24 Workbench: Audit Inspector: Initializing



When you first start the **Audit Inspector**, MySQL Workbench must cache the audit log for performance reasons. MySQL Workbench then parses, indexes, and retrieves values from the encrypted cached file on your local computer.

Generating the cache file can take a long time. If you press **Abort** during the caching process, MySQL Workbench saves the results that were cached at the point you pressed **Abort**.

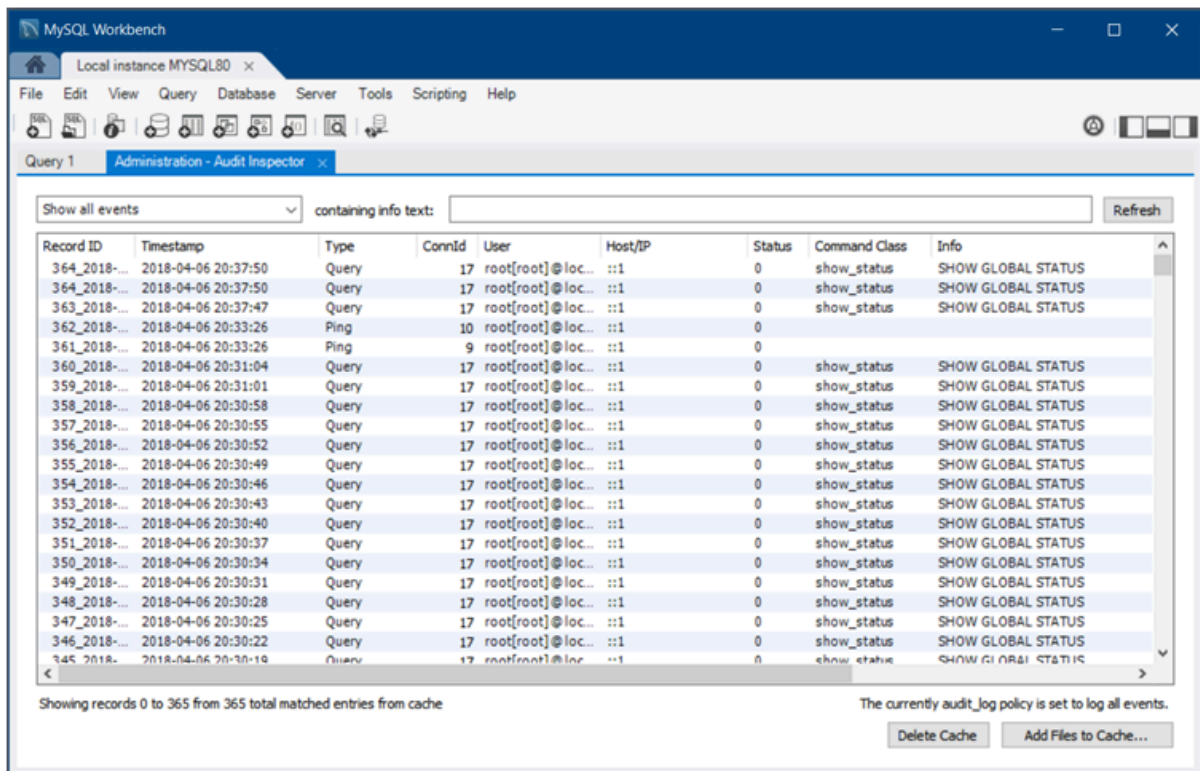


Note

MySQL Workbench will prompt for **sudo** access if the MySQL Workbench user is unable to read the audit log file.

To generate the cache file, click **Cache Selected Files for Browsing**. When prompted, set a password for the encrypted file to use when viewing this file. After caching an audit log, the **Audit Inspector** tab displays the results as shown in the following figure. Each record includes a record ID, timestamp, type, connection ID, user, host/IP, status, command class, and information.

Figure 6.25 Workbench: Audit Inspector



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.

Click **Add Files to Cache** at any time to add new files to the cache. Click **Delete Cache** to remove the cache file.

The next time you start the Audit Inspector, MySQL Workbench prompts you for the password that you set during the initial step.

6.7 MySQL Enterprise Backup Interface

Commercial releases of MySQL Workbench include a graphical interface that enables you to use MySQL Enterprise Backup (MEB) functionality to safeguard your MySQL data from unexpected loss. The specific options may vary, depending on the version of MySQL and MySQL Workbench that you have installed. For more information, see [Section 6.7.1, “General Requirements”](#).

There are two MySQL Enterprise Backup operations available from MySQL Workbench:

- **Online backup.** Establishes a backup profile to define *what* should be backed up, *where* the backup should be stored, and *when* (the frequency) MySQL should be backed up. With an active MySQL connection tab open, select **MySQL Enterprise Backup** from the **Server** menu to view the Online Backup page in the current tab.
- **Backup recovery.** 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. With an active MySQL connection tab open, select **Backup Recovery** from the **Server** menu to view the Recovery page in the current tab.

The MySQL Enterprise Backup configuration is located on the MySQL server, and not locally within MySQL Workbench. This information includes the MySQL Enterprise Backup configuration backup profiles, job scheduling, backup operations, and data. The backup and restore operations can be scheduled to executed with (or without) MySQL Workbench running.

For information comparing the different methods to import and export data using MySQL Workbench, see [Section 6.5, “Data Export and Import”](#).

6.7.1 General Requirements

MySQL Enterprise Backup (MEB) is a MySQL Enterprise feature that is separate from MySQL Workbench. For more information about its functionality, see the [MySQL Enterprise Backup documentation](#). MySQL Workbench provides an interface to MySQL Enterprise Backup, as described in this section.

In addition to having MySQL Enterprise Backup installed on the target server, 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.
- Setting an encryption password is required to perform backup and restore operations on encrypted tables (see [Options Tab](#)).
- Managing both local and remote MySQL instances is available on Linux and macOS, 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 and macOS) that is able to execute the MySQL Enterprise Backup binary.
- The `sudo` user must keep the `HOME` environment variable when executing system commands, which means adding the following entry to the `/etc/sudoers` file safely by using the `visudo` command:

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

Prerequisite Settings

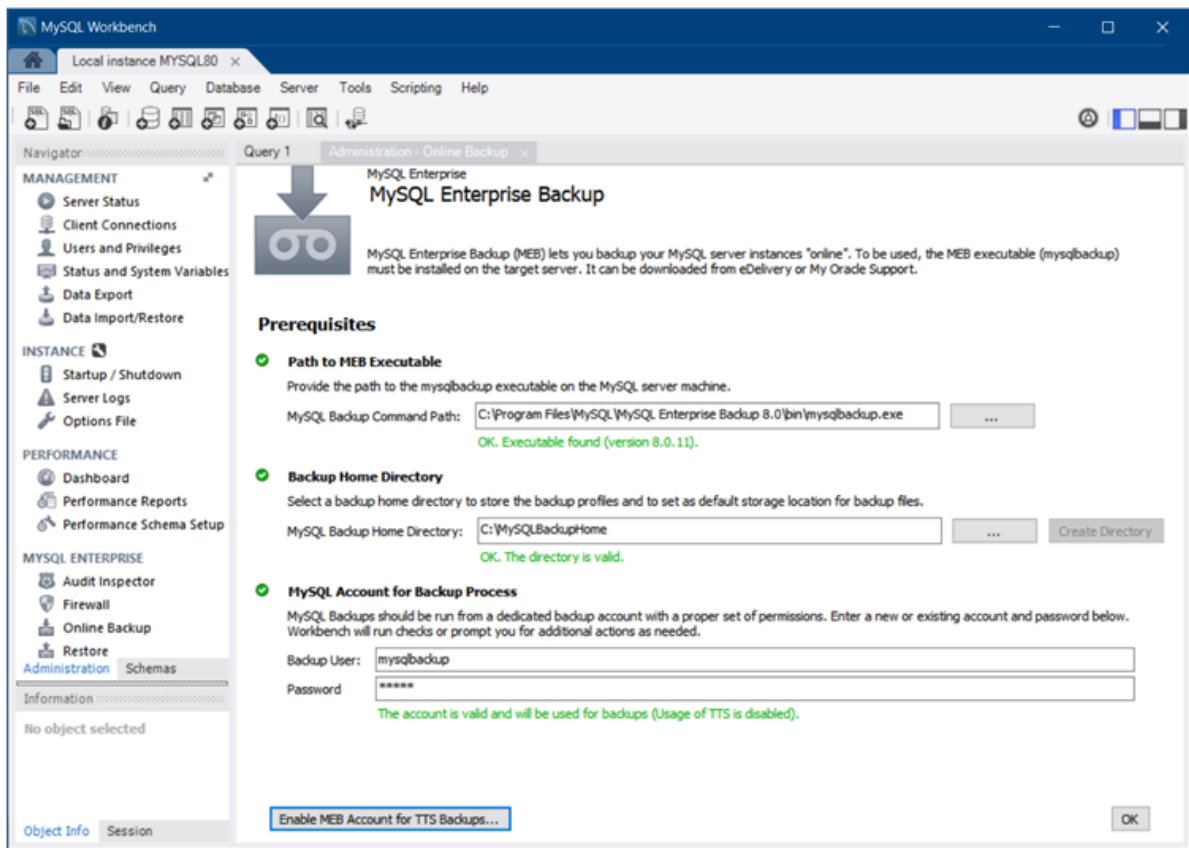
To set the following prerequisites, click **Settings** from the main page (either Online Backup or Restore):

- A path to the MySQL Enterprise Backup executable. MySQL Enterprise Backup is available with eDelivery or My Oracle Support (MOS). MySQL Workbench attempts to locate the MySQL Enterprise Backup executable based on the version MySQL server in use, so check the path and adjust it accordingly. For example:
 - For MySQL 8.0, use the MySQL Enterprise Backup version with the same version number as the server.
 - For MySQL 5.7, use MySQL Enterprise Backup 4.1.
 - For MySQL 5.6, use MySQL Enterprise Backup 3.12.
- The path to the backup home directory, where backup profiles and data is stored. This can be created from within Workbench from **Settings**.

- 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 account by adding the `RELOAD`, `SUPER`, and `REPLICATION CLIENT ON *.*` privileges.

The following figure shows an example of the prerequisites.

Figure 6.26 Workbench: MySQL Enterprise Backup Settings



If any of the requirements are not met, then an error will be generated when attempting to use MySQL Enterprise Backup features.

After Uninstalling Workbench

The following list of notes describes the behavior of MySQL Enterprise Backup operations after you remove MySQL Workbench:

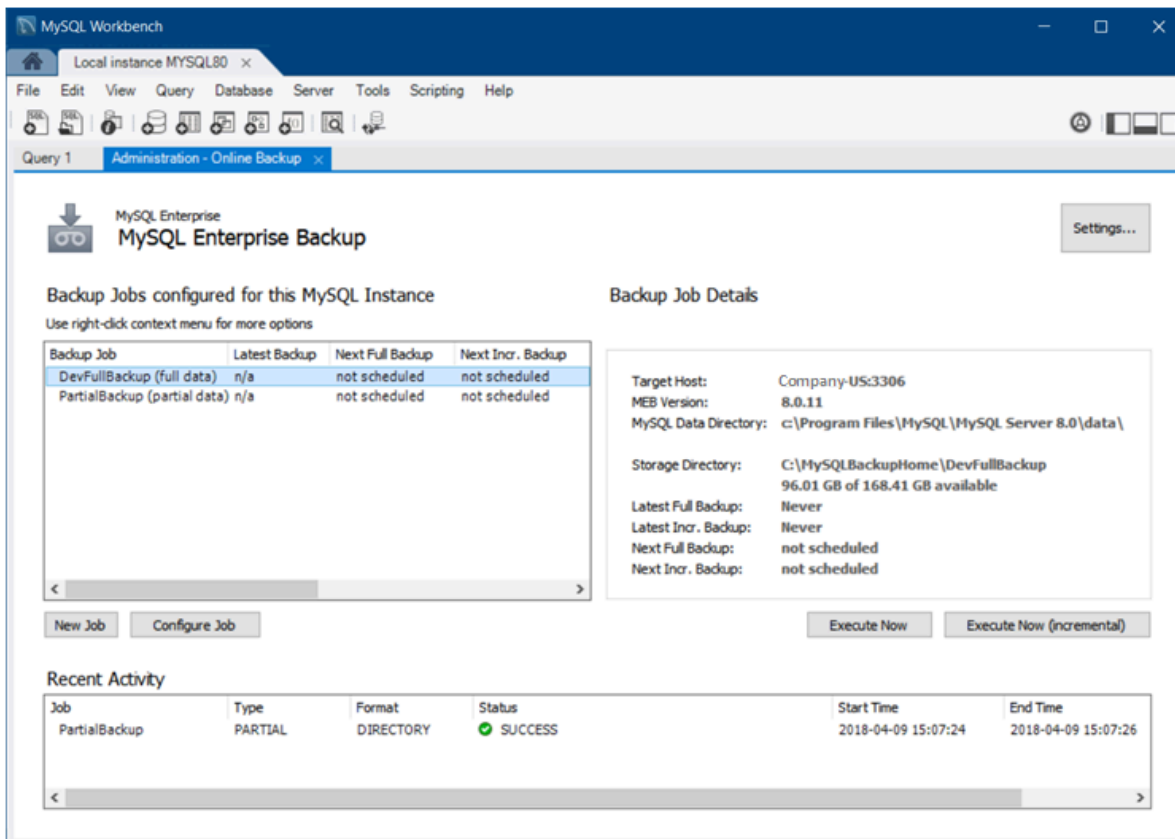
- Uninstalling Workbench does not remove the associated MySQL Enterprise Backup backup tasks. To stop the scheduled backups, edit the related "Task Scheduler" entries on Windows, or remove the associated cron jobs on Linux and macOS.
- Uninstalling MySQL Workbench does not remove the MySQL Enterprise Backup configuration file, the configuration files generated for each defined profile, or the MySQL backups.

6.7.2 Online Backup

This section describes how to configure 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. With an active MySQL connection tab open, select **MySQL Enterprise Backup** from the **Server** menu to view the main Online Backup overview page in the current tab.

The following figure shows the Online Backup page that includes a full and a partial backup job configured for the current MySQL instance.

Figure 6.27 Workbench: MySQL Enterprise Backup



Online Backup Elements

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-click a backup job to access the available actions, such as **Configure Job**, **Delete Job**, and **Execute Backup**. The context menu 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.

Configuring a Backup Job

The following information applies to the new job operation. **Configure Job** is used to modify existing jobs.

The **Backup Profile Name** option and its associated **Comments** field are used to identify the profile of the backup job. Each backup job name is shown on the main page. A new backup job separates the configuration information into four subtabs: **Schedule**, **Contents**, **Options**, and **Advanced**.

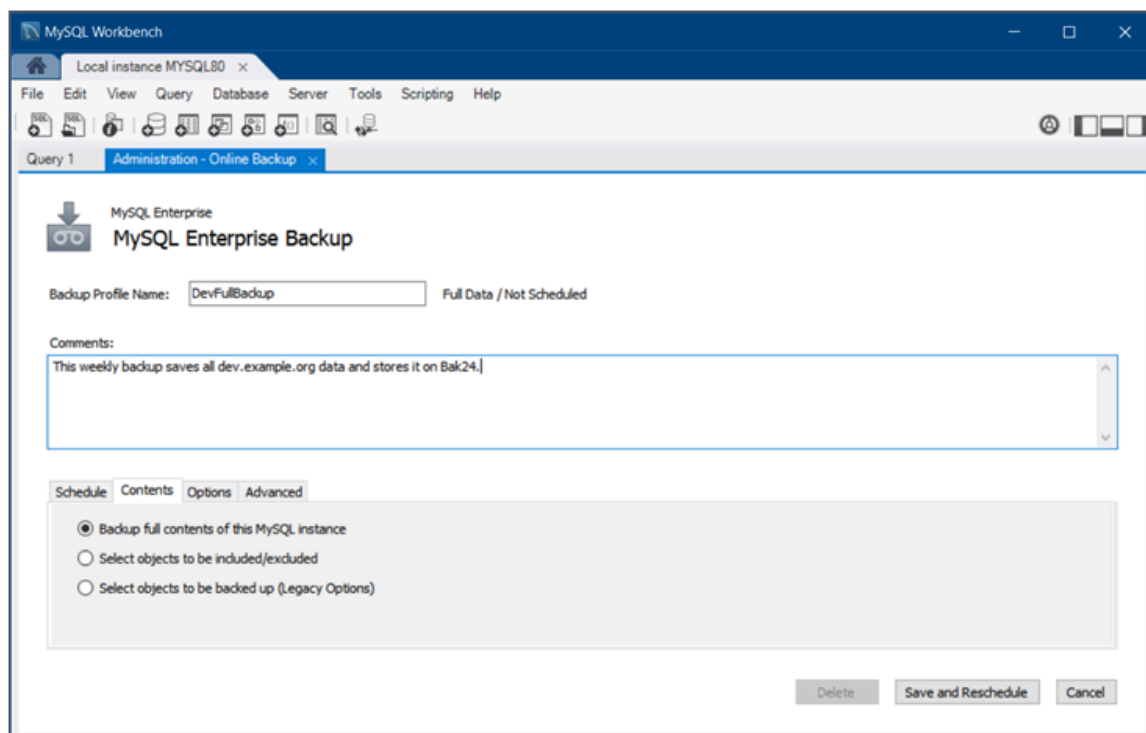
Schedule Tab. The **Schedule** subtab optionally sets a backup schedule for both full and incremental backups. The schedule uses the Windows Tasks Scheduler on Microsoft Windows, and a cron job on Linux and macOS. It is scheduled using the operating system user that is scheduling the backup, which is 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](#).

Contents Tab. The **Contents** subtab 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 (see the figure that follows).

Figure 6.28 Workbench: MySQL Enterprise Backup Configuration Showing the Contents Tab



- **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](#).

Options Tab. The **Options** subtab 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 option increases the restoration time, because the unused pages that were removed must be added back during the recovery process.

- **Incremental Backups Using Only the Redo Log (incremental-with-redo-log-only):** Specifies that an incremental backup is to be created using only the redo log.
- **Encrypt Password:** Sets a password that is required to back up and restore encrypted tables.

Advanced Tab. The **Advanced** subtab 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"](#).

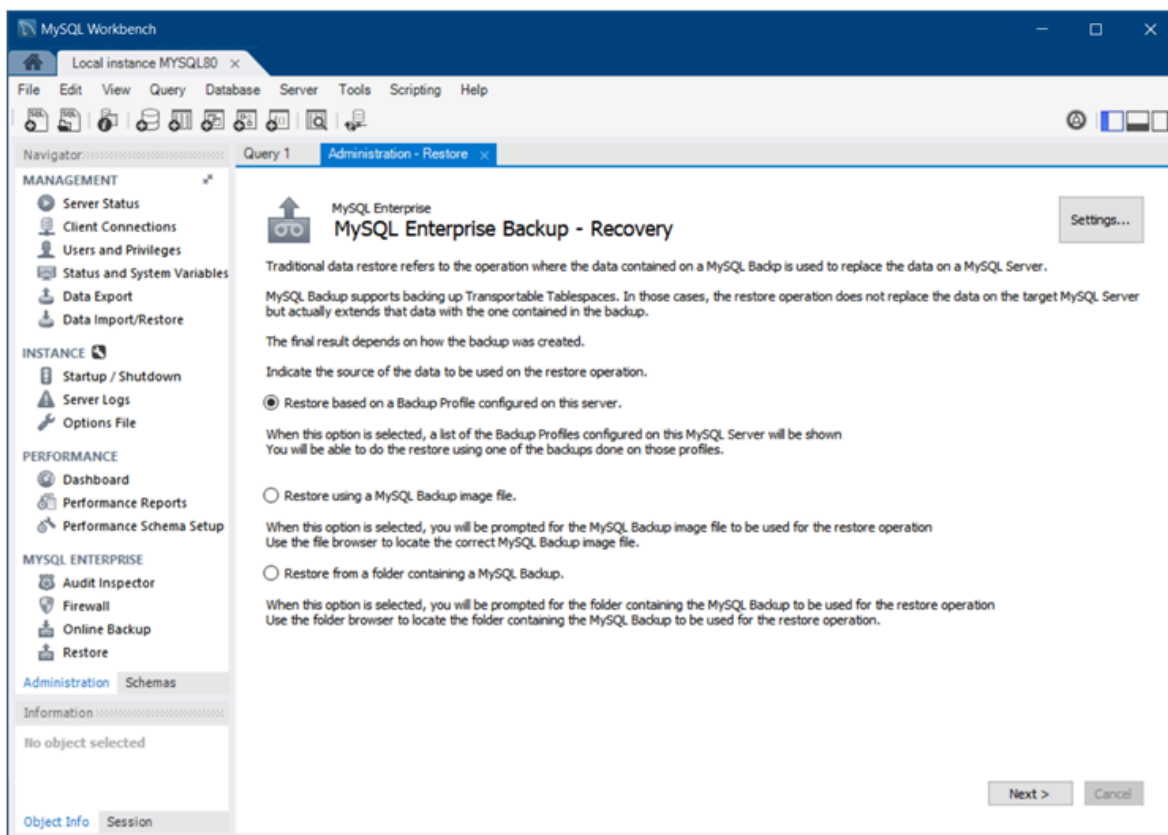
6.7.3 Backup Recovery

The backup-recovery wizard is used to restore MySQL Enterprise Backup data. For more information about creating a backup using MySQL Workbench, see [Section 6.7.2, "Online Backup"](#).

The wizard enables you to restore a backup from a folder, an image file, or from a backup profile. Before attempting to restore MySQL data from a backup to a target server, remove all of the files in the data directory of that server.

With an active MySQL connection tab open, select **Backup Recovery** from the **Server** menu to open the wizard in the current tab. As the following figure shows, you must first select the source of data to use for the recovery.

Figure 6.29 Workbench: MySQL Enterprise Backup - Recovery



Select one of the following source options and then click **Next** to continue:

- **Restore based on a Backup Profile configured on this server.**

This option enables you to choose from one of the existing MySQL Enterprise Backup profiles. For information about configuring a profile, see [Section 6.7.2, “Online Backup”](#).

- **Restore using a MySQL Backup image file.**

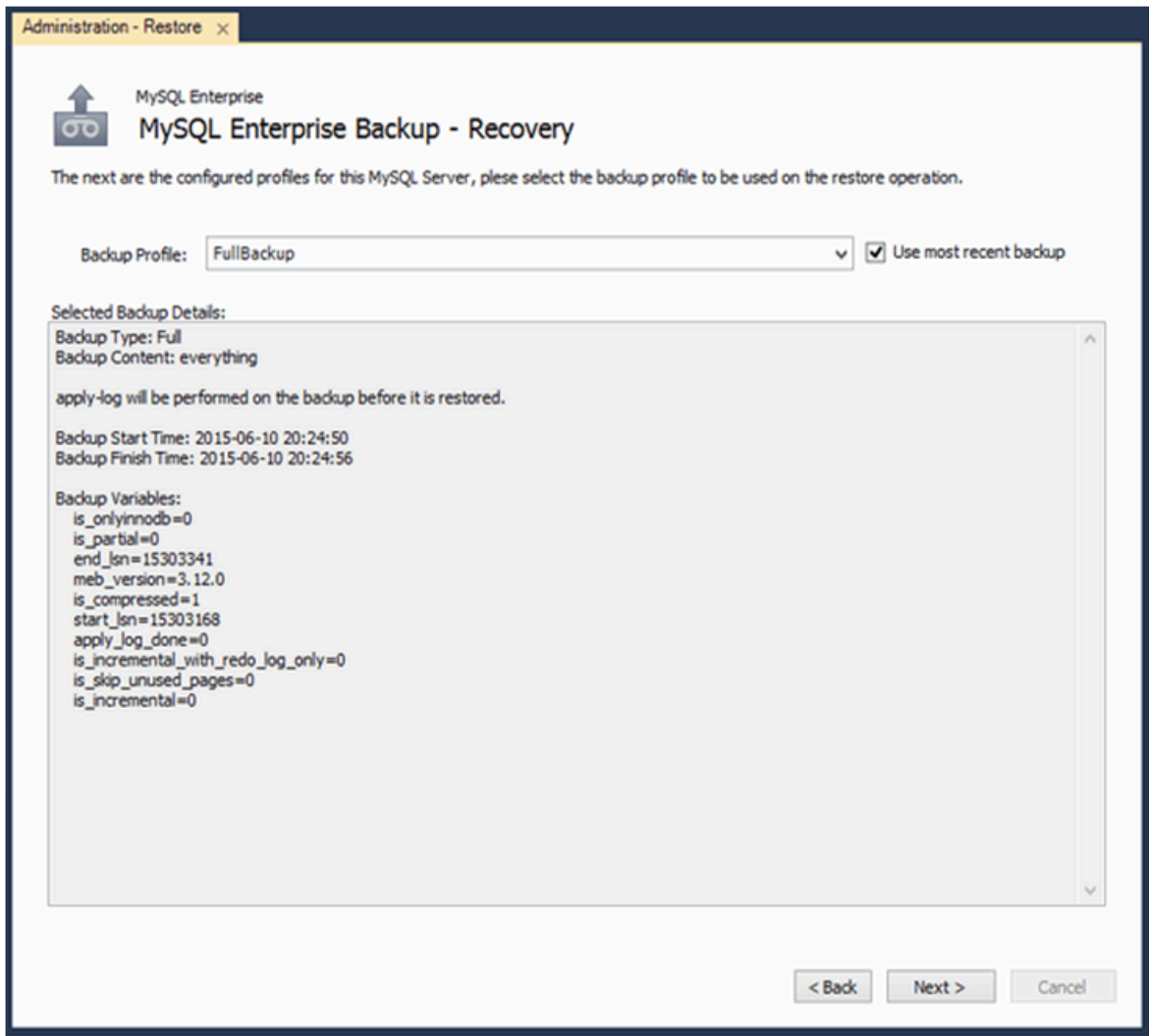
This option opens the default file browser to assist you with locating a backup image file to restore.

- **Restore from a folder containing a MySQL Backup.**

This option opens the default file browser to search for a backup folder.

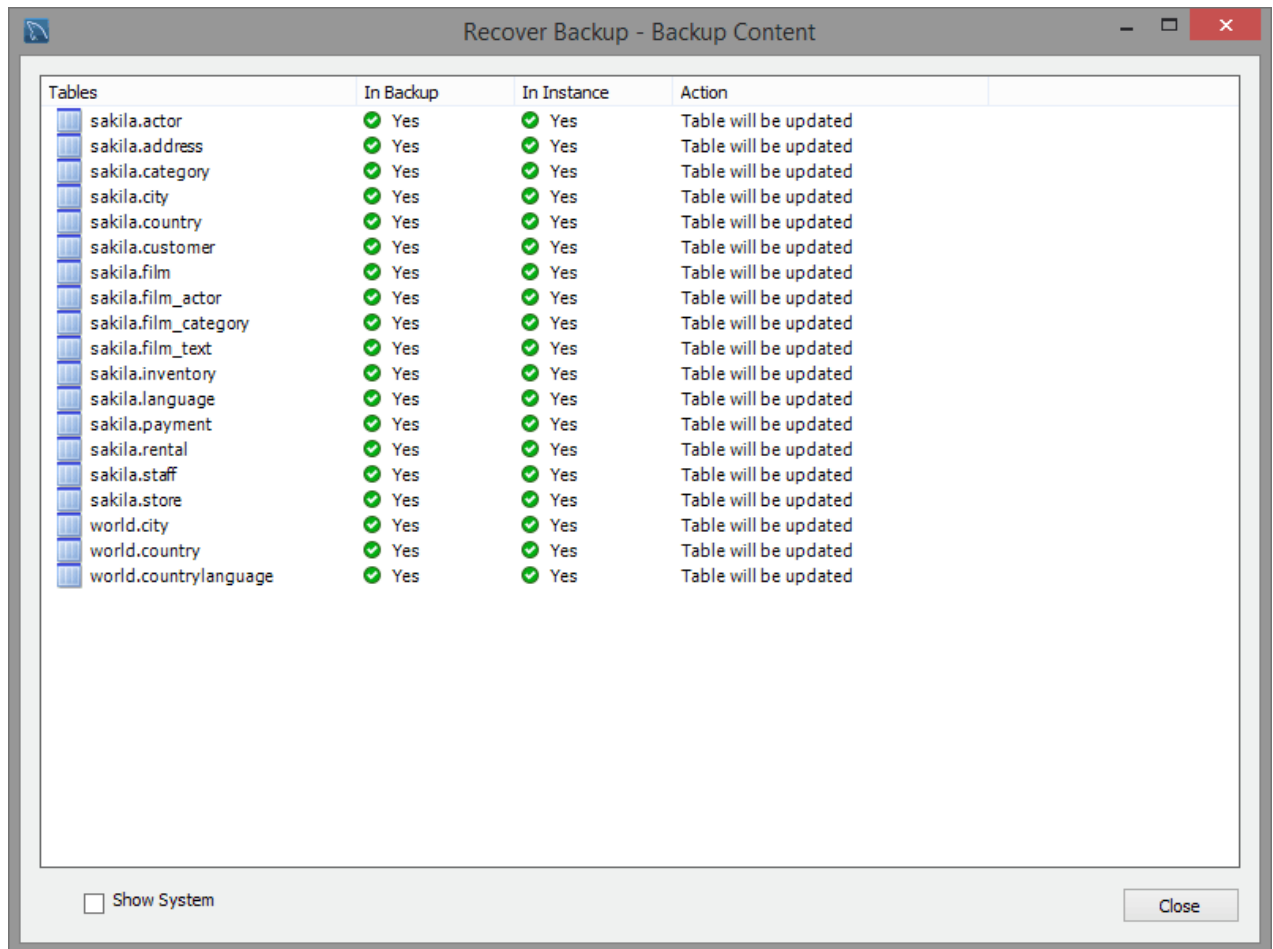
For example, if the backup profile option was selected as a data source type in the wizard, the next step provides a list of existing profiles from which to choose. The following figure shows this wizard step with the [FullBackup](#) profile selected and the **Use most recent backup** option enabled to ensure that only the latest content is included. The wizard also shows the specific details of the selected profile, such as the backup type, content, start and finish time, and the backup variables.

Figure 6.30 Workbench: MySQL Enterprise Backup - Recovery Using a Backup Profile



The next step provides the **View Backup Content** option, which lists the content to be restored. For example, the following figure shows a set of sakila database tables, whether each table is in the backup and in the MySQL instance, and the action to be performed by the wizard.

Figure 6.31 Workbench: MySQL Enterprise Backup - Recovery Showing Content Table View

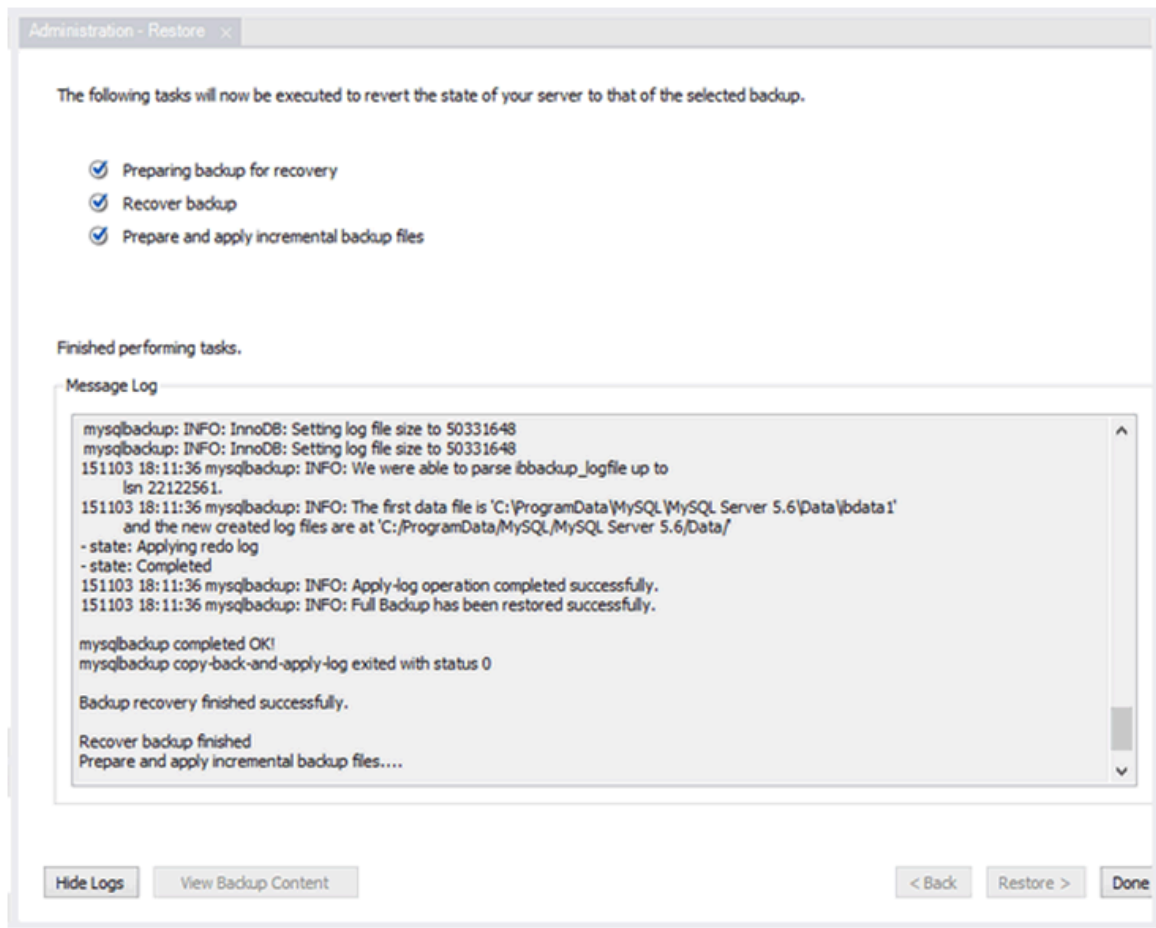


Note

The **Show System** check box toggles internal schemas from view, schemas such as the internal `performance_schema` and `mysql` tables.

After verifying the content and closing the content view, click **Restore** to execute the restoration process and to toggle the message logs as shown in the following figure.

Figure 6.32 Workbench: MySQL Enterprise Backup - Recovery Tasks



6.8 MySQL Enterprise Firewall Interface

MySQL Workbench provides a graphical interface to MySQL Enterprise Firewall. For additional information about MySQL Enterprise Firewall, see <https://dev.mysql.com/doc/en/firewall.html>.

Setup and Configuration

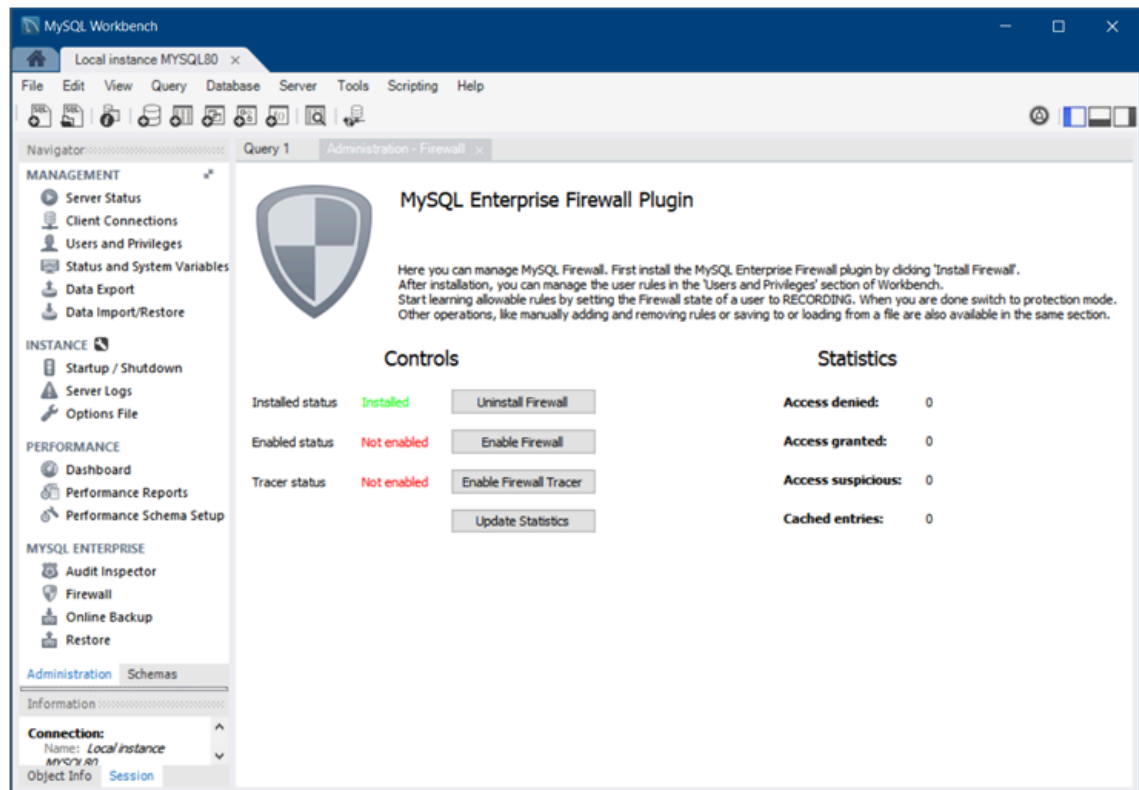
MySQL Workbench can manage the MySQL Enterprise Firewall installation and configuration by installing (or uninstalling) and enabling (or disabling) the plugin, and enabling (or disabling) Firewall Tracer.

To prepare MySQL Enterprise Firewall plugin:

1. Open a connection to MySQL Enterprise Edition.
2. From the Navigator area of the sidebar and with the **Administration** secondary tab selected, click **Firewall** (see MYSQL ENTERPRISE) to open the **Administration - Firewall** secondary tab in the workspace.

If MySQL Enterprise Firewall is not installed, click **Install Firewall**. After the plugin is installed, MySQL Workbench displays three controls: **Installed status**, **Enabled status**, and **Tracer status**. The following figure shows the MySQL Enterprise Firewall installed, but not yet enabled within MySQL Workbench.

Figure 6.33 MySQL Enterprise Firewall Installation and Configuration



3. Click **Enable Firewall** to make the plugin fully operational, and optionally click **Enable Firewall Tracer** to enable tracing. You can modify the plugin controls as follows:
 - **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.
 - **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. Specifically, select the `mysql_firewall_mode` option in the configuration option to enable it after a restart. You can edit the MySQL configuration file with an external editor or use MySQL Workbench to edit it.

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. Figure 6.34, “MySQL Enterprise Firewall Rules” shows the location of actions available within the **Firewall Rules** tab. The actions are:

- **State (mode)**: Options include **OFF** (disables firewall protection for the account), **PROTECTING** (enables the allowlist), **RECORDING** (training mode), and **RESET** (removes the rules, sets the mode to OFF mode). For additional information about the meaning of these states, see [Firewall Concepts](#).
- 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.

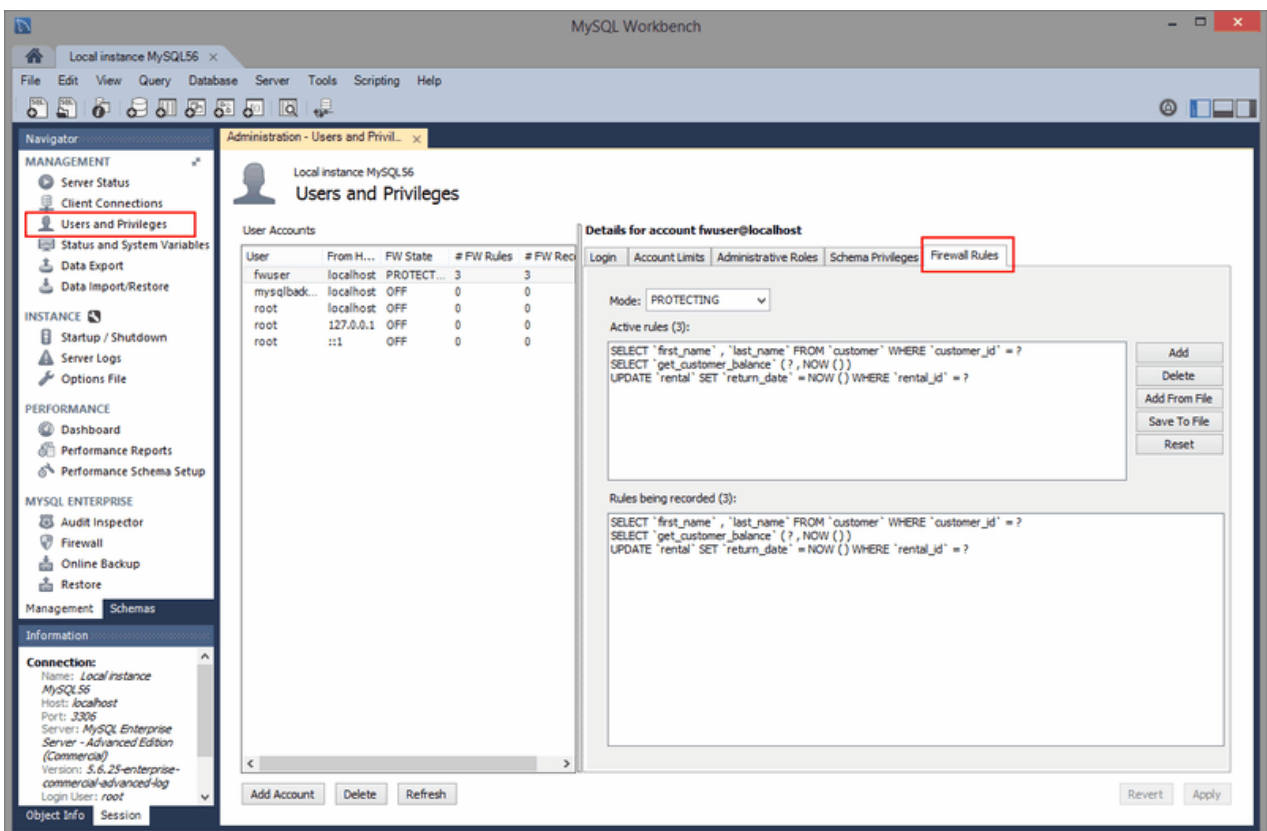
- *Active rules* are used in PROTECTIVE mode, and *Rules being recorded* are entries still being RECORDED. Switching from RECORDING to PROTECTING mode copies the recorded rules into the active rule subset.



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 records the behind-the-scenes operations performed by MySQL Workbench for the MySQL user. Also, a MySQL user in PROTECTING mode could have access to operations in MySQL Workbench that are not accessible to a typical firewalled MySQL user.

Figure 6.34 MySQL Enterprise Firewall Rules



6.9 wbcopytables Utility

`wbcopytables` is a command-line utility included in MySQL Workbench that enables you to copy table data from a supported source database server to MySQL. It is used by the MySQL 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` to insert the retrieved rows into the target MySQL server.

Table 6.2 File Location (Default)

Operating System	Location
Linux	<code>/usr/bin/wbcopytables</code>
macOS	<code>/Applications/MySQLWorkbench.app/Contents/MacOS/wbcopytables</code>
Windows	<code>C:\Program Files\MySQL\MySQL Workbench 8.0\wbcopytables.exe</code>

Connection Parameters

Option parameters for source and target connections 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 or 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`
- `--source-password`: Passes the connection password of the data source.
- `--target=MySQL_connection_string`: Specifies the target connection.
- `--passwords-from-stdin`: Passes 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.

Option parameters for source and target connections that support SSH tunneling to copy data are:

- `--source-ssh-port=ssh port`
- `--source-ssh-host=ssh host`
- `--source-ssh-user=ssh user`
- `--source-ssh-password=ssh password`
- `--target-ssh-port=ssh port`
- `--target-ssh-host=ssh host`
- `--target-ssh-user=ssh user`
- `--target-ssh-password=ssh password`

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 or catalog to which the table belongs. 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 argument 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 argument 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 Range_End
```

The generated expression is:

```
key_column >= range_start AND key_column <= range_end
```

If you specify `-1` for `Range_End`, then the expression is:

```
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 performs a `COUNT(*)` of the `SELECT` generated by the `--table` option that was used. The target schema and table can be omitted in this case.
- `--truncate-target`: Executes 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`: Bypasses the trigger disabling step.

Chapter 7 Performance Tools

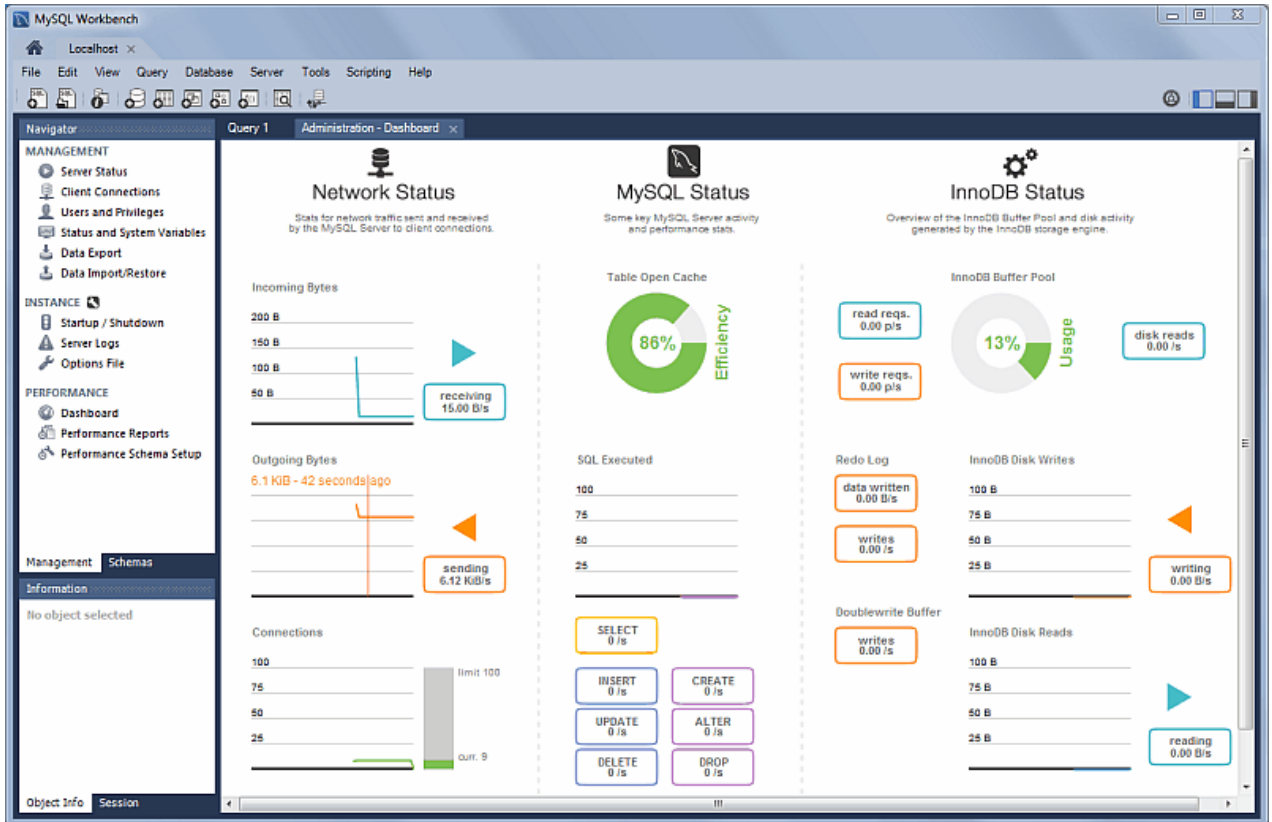
Table of Contents

- 7.1 Performance Dashboard 185
- 7.2 Performance Schema Reports 186
- 7.3 Query Statistics 191
- 7.4 Visual Explain Plan 192
- 7.5 Tutorial: Using Explain to Improve Query Performance 195

7.1 Performance Dashboard

View server performance statistics in a graphical dashboard. To display the dashboard, open a query tab and then click **Dashboard** from the Performance area of the Navigator sidebar with the **Management** tab selected. The following figure shows the layout of the information within the **Administration - Dashboard** tab.

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](#).

Installation and Configuration

A GUI for configuring and fine tuning the Performance Schema instrumentation (see the figure that follows). Initially, this loads an **Easy Setup** tab that is enough for most users. To enable all available Performance Schema instruments, pause your pointer device over **Fully Enabled** and click the circle on the slide bar.

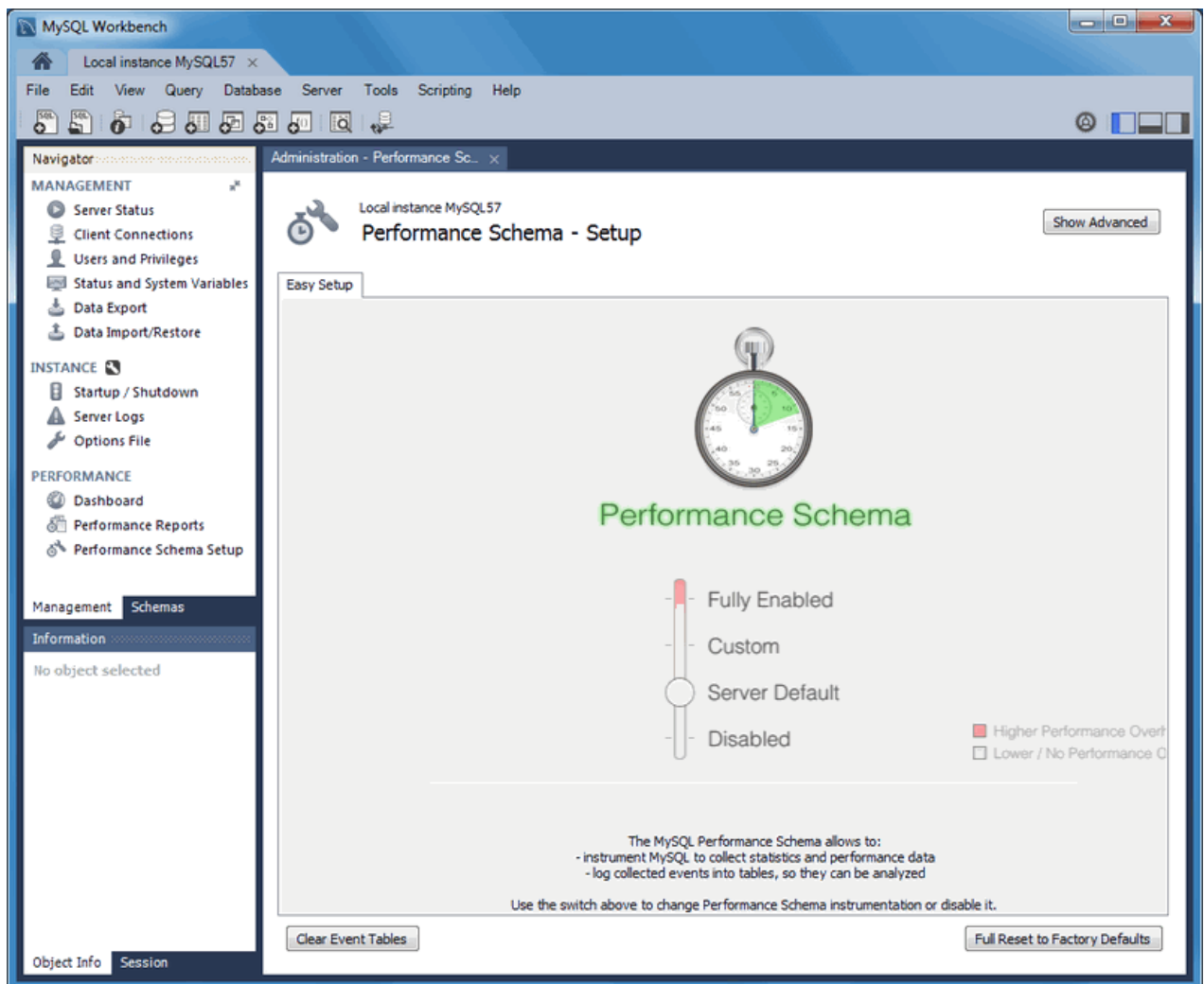
The SYS schema is bundled with MySQL Server 5.7 and above, and MySQL Workbench uses that version. However, for MySQL Server 5.6, Workbench installs its own bundled version of the SYS schema.



Note

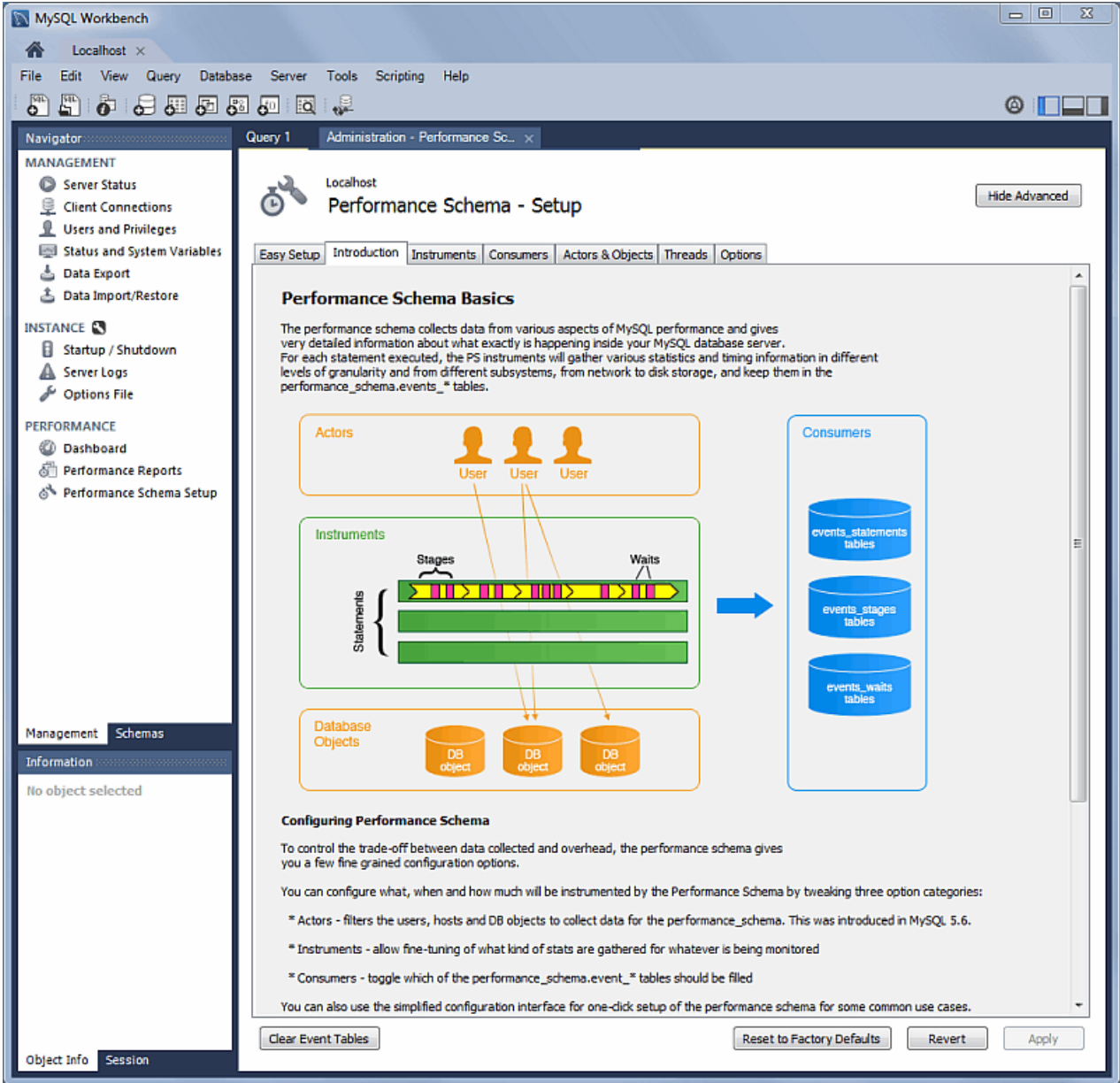
The size of the saved digested query is determined by the MySQL server.

Figure 7.2 Performance Schema Setup: Easy Setup



Clicking **Show Advanced** provides methods to fine tune the Performance Schema instrumentation. The next figure shows the tabs related to advanced instrumentation and the **Introduction** tab selected.

Figure 7.3 Performance Schema Setup: Introduction



Performance Report Controls

Performance report data can be viewed and exported using the following controls (see the figure that follows):

- **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 clipboard.

- **Refresh:** Refreshes (reloads) the performance report.

Performance Report Descriptions

Figure 7.4 Performance Reports: Statement Analysis

The screenshot shows the MySQL Workbench interface with the Performance Reports window open. The 'Statement Analysis' report is selected, displaying a table of queries with their execution statistics. The table columns are: Query, Full Table Scan, Executions, Errors (#), Warnings (#), Total Time (ms), Max Time (us), and Avg Time (us). The first few rows of the table are:

Query	Full Table Scan	Exec...	Errors (#)	Warnings (#)	Total Time (...)	Max Time (us)	Avg Time (us)
SHOW GLOBAL STATUS	*	326	0	0	356225.70	2065.90	1092.70
CALL `sys`.`ps_truncate_all_tables` (?)		1	0	0	270782.70	270782.70	270782.70
SELECT (`cat`.`name` COLLATE `utf8_tol...`)		1	0	0	89757.80	89757.80	89757.80
CALL `sys`.`ps_setup_reset_to_default` (?)		1	0	0	51245.30	51245.30	51245.30
SELECT `performance_schema`.`events_st...`	*	1	0	0	17928.90	17928.90	17928.90
SELECT SUM (`performance_schema`.`me...`)	*	1	0	0	16738.90	16738.90	16738.90
SELECT IF (`performance_schema`.`eve...`)	*	1	0	0	13832.20	13832.20	13832.20
SHOW SESSION VARIABLES LIKE ?	*	2	0	0	8418.50	4382.30	4209.20
SELECT `substring_index` (`performance_sc...`)	*	1	0	0	5435.50	5435.50	5435.50
SELECT `performance_schema`.`file_summ...`	*	3	0	0	5180.40	3361.50	1726.80
SHOW VARIABLES LIKE ?	*	1	0	0	3087.00	3087.00	3087.00
SELECT `substring_index` (`performance_sc...`)	*	1	0	0	2504.80	2504.80	2504.80
SHOW SESSION STATUS LIKE ?	*	1	0	0	1765.40	1765.40	1765.40
SELECT * FROM `performance_schema`.`s...`	*	1	0	0	1614.10	1614.10	1614.10
SELECT SUM (IF (`ENABLED` = ? AND (NA...`)	*	1	0	2	1394.10	1394.10	1394.10
SELECT `cs`.`name` AS `CHARACTER_SET...`	*	1	0	0	918.30	918.30	918.30
SELECT `THREAD_ID`, NAME, TYPE, `PRO...`	*	1	0	0	542.00	542.00	542.00
SELECT COUNT (*) FROM `performance_sc...`	*	2	0	0	507.40	283.50	253.70
SELECT ? AS `sys_version`, `version` () AS...`	*	2	0	0	479.80	269.60	239.90
SELECT NAME, `timer_name` FROM `perfor...`	*	1	1	0	435.40	435.40	435.40
SELECT SUM (IF (`ENABLED` = ? AND NAM...`)	*	1	0	0	359.30	359.30	359.30
SHOW GRANTS	*	2	0	0	346.50	207.20	173.20
SELECT * FROM `performance_schema`.`p...`	*	1	0	0	328.90	328.90	328.90
SELECT @@`performance_schema`	*	2	0	0	317.20	185.70	158.60

Individual reports are shown in the following groups:

Memory Usage

- **Total Memory** – Shows total memory allocated.
- **Top Memory by Event** – Shows events consuming the most memory.
- **Top Memory by User** – Shows users consuming the most memory.
- **Top Memory by Host** – Shows hosts consuming the most memory.
- **Top Memory by Thread** – Shows threads consuming the most memory.

Hot Spots for I/O

- **Top File I/O Activity Report** – Shows the files with the most I/O usage in bytes.
- **Top I/O by File by Time** – Shows the highest I/O usage by file and latency.
- **Top I/O by Event Category** – Shows the highest I/O data usage by event categories.
- **Top I/O in Time by Event Categories** – Shows the highest I/O time consumers by event categories.

- [Top I/O by User/Thread](#) – Shows the top I/O time consumers by user and thread.

High Cost SQL Statements

- [Statement Analysis](#) – Lists statements with various aggregated statistics.
- [Statements in Highest 5 percent by Runtime](#) – Lists all statements in which the average runtime (in microseconds) is in the highest five percent.
- [Using Temp Tables](#) – Lists all statements that use temporary tables (access the highest percentage of disk temporary tables, then memory temporary tables).
- [With Sorting](#) – Lists all normalized statements that have done sorts (access in the following priority order: `sort_merge_passes`, `sort_scans`, and `sort_rows`).
- [Full Table Scans](#) – Lists statements that have performed a full table scan. Access query performance and the `WHERE` clause (or clauses). If no index is used, consider adding indexes for large tables.
- [Errors or Warnings](#) – Lists statements that have raised errors or warnings.

Database Schema Statistics

- [Schema Object Overview \(High Overhead\)](#) – Shows the count by object for each schema. Note that for instances with a large number of objects, this report may require extended time to execute.
- [Schema Index Statistics](#) – Shows the general statistics related to indexes.
- [Schema Table Statistics](#) – Shows the general statistics related to tables.
- [Schema Table Statistics \(with InnoDB buffer\)](#) – Shows schema tables with InnoDB buffer statistics.
- [Tables with Full Table Scans](#) – Finds tables that are being accessed by full table scans, ordering by the number of rows scanned (descending).
- [Unused Indexes](#) – Shows the list of indexes that were never used since the server started or since P_S data collection started.

Wait Event Times (Expert)

- [Global Waits by Time](#) – Lists the top global wait events by their total time, ignoring idle (this may not be very large).
- [Waits by User by Time](#) – Lists the top wait events by user and by their total time, ignoring idle (this may not be very large).
- [Wait Classes by Time](#) – Lists the top wait classes by total time, ignoring idle (this may not be very large).
- [Waits Classes by Average Time](#) – Lists the top wait classes by average time, ignoring idle (this may not be very large).

InnoDB Statistics

- [InnoDB Buffer Stats by Schema](#) – Summarizes the output of the `INFORMATION_SCHEMA.INNODB_BUFFER_PAGE` table, aggregating by schema.

- [InnoDB Buffer Stats by Table](#) – Summarizes the output of the `INFORMATION_SCHEMA.INNODB_BUFFER_PAGE` table, aggregating by schema and table name.

User Resource Use

- [Overview](#) – Shows the resource use summary for each user.
- [I/O Statistics](#) – Shows the I/O use for each user.
- [Statement Statistics](#) – Shows the statement execution statistics for each user.

7.3 Query Statistics

The **Query Stats** SQL editor results tab (see the next two figures) uses Performance Schema data to gather key statistics collected for executed query, such as timing, temporary tables, indexes, joins, and more.

Requirements

- **Query, Collect Performance Schema Stats** enabled.
- The `performance_schema` enabled with statement instrumentation.

Figure 7.5 SQL Editor: Query Stats

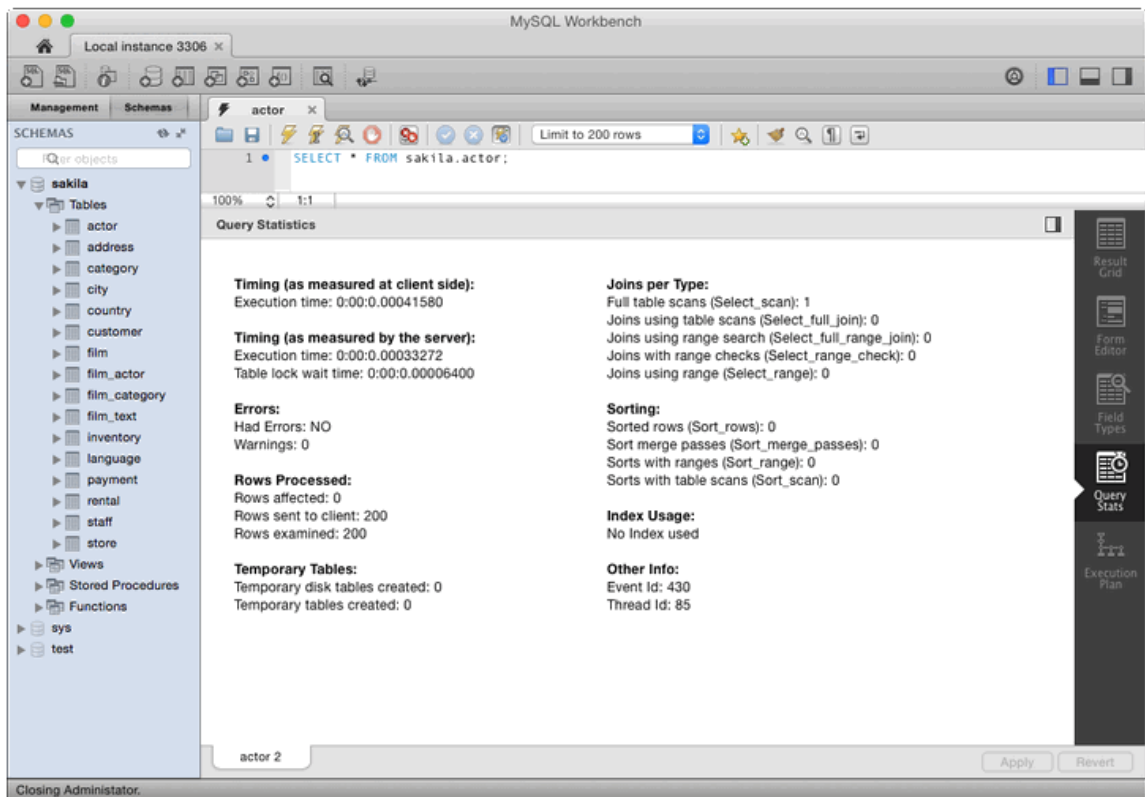
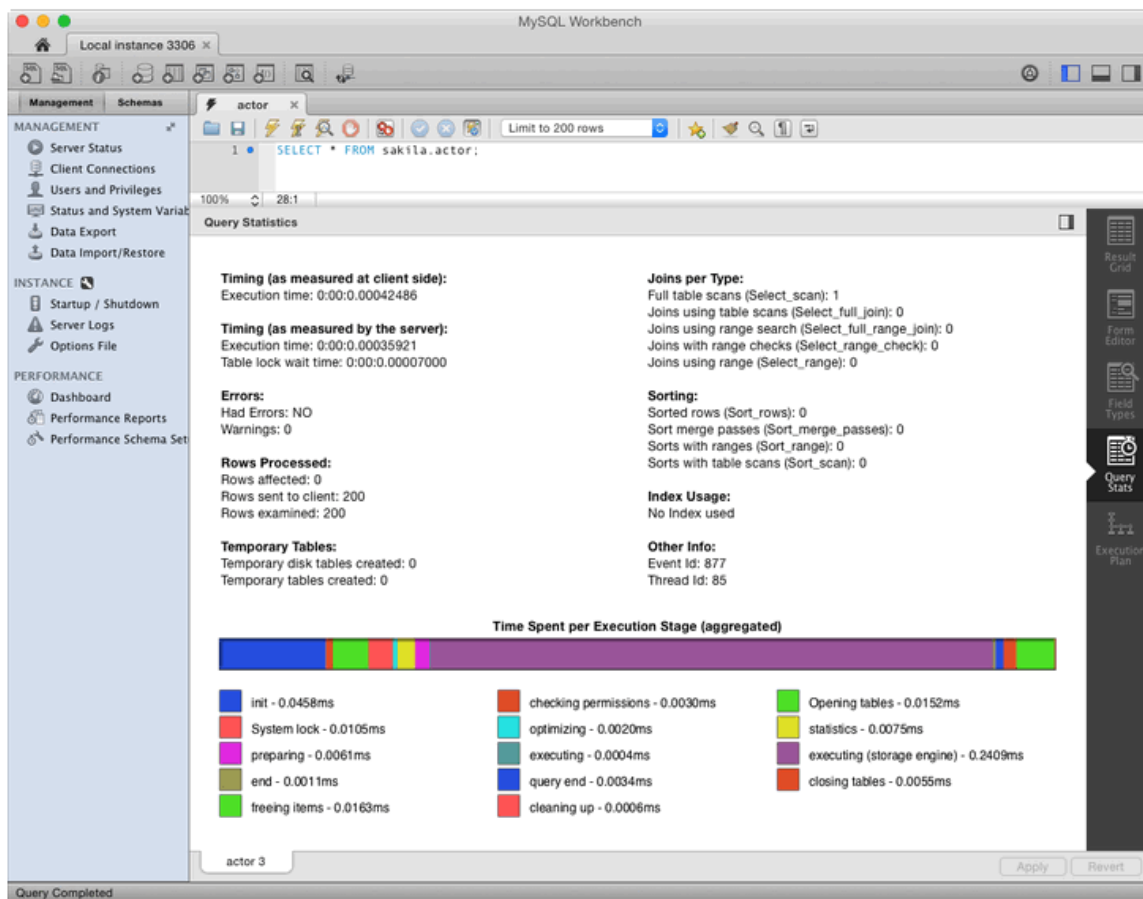


Figure 7.6 SQL Editor: Query Stats with Performance Schema Graphs



7.4 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. MySQL Workbench provides all of the [EXPLAIN](#) formats for executed queries including the raw extended JSON, traditional format, and visual query plan.

Visual Explain Usage

To view a visual explain execution plan, execute your query from the SQL editor and then select **Execution Plan** within the query results tab. The execution plan defaults to [Visual Explain](#), but it also includes a [Tabular Explain](#) view that is similar to what you see when executing [EXPLAIN](#) in the MySQL client. For information about how MySQL executes statements, see [Optimizing Queries with EXPLAIN](#).

Visual Explain Conventions

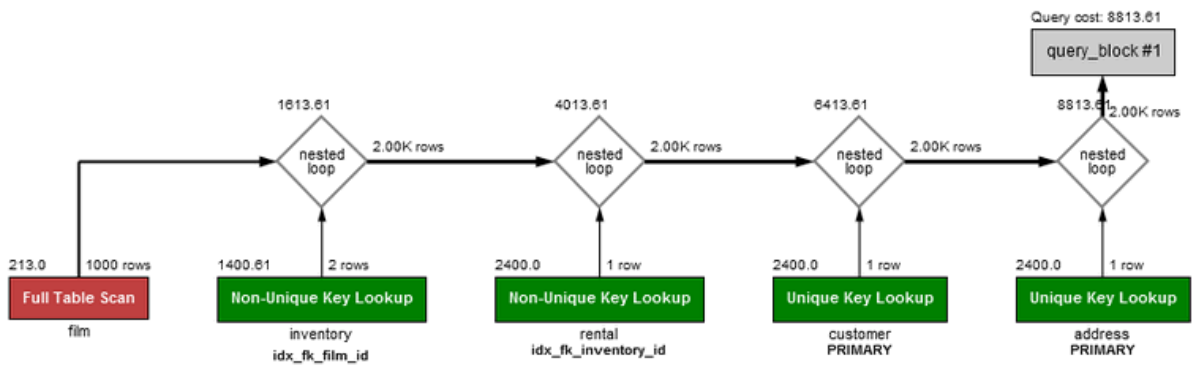
The order of execution in a visual explain diagram is bottom to top and left to right. The diagram examples that follow provide an overview of the graphic, textual, and informational conventions used to represent aspects of the visual explain plans. For specific information, see:

- [Graphic Conventions](#)
- [Textual and Informational Conventions](#)

The visual explain diagram in the first figure shows a visual representation of the following query.

```
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;
```

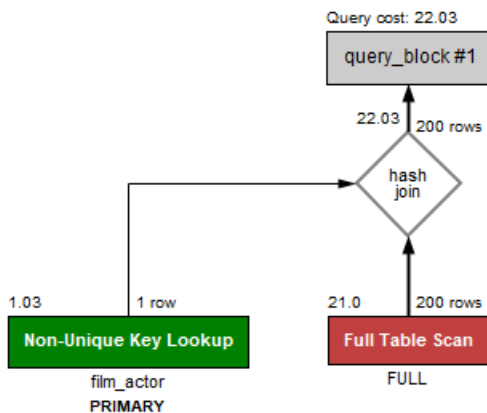
Figure 7.7 A Visual Explain Example



The next visual explain diagram shows the visual representation of a query that contains a hash join. Prior to MySQL Workbench 8.0.22, a hash join was represented by a `block nested loop` diamond for queries executed by MySQL 8.0.19 (or earlier).

```
SELECT first_name, last_name
FROM actor
FULL JOIN film_actor
WHERE '' = film_actor.actor_id;
```

Figure 7.8 A Visual Explain Example with a Hash Join



Graphic Conventions

- Standard Boxes: tables
- Rounded boxes: operations such as GROUP and SORT
- Framed boxes: subqueries
- Diamonds: joins

Textual and Informational 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 higher)
- Number to the right of nested loop (or hash join) diamonds: number of rows produced by the JOIN
- Number above the diamonds: relative cost of the JOIN (requires MySQL 5.7 or higher)

The following table shows the associated colors and descriptions used in the visual explain diagram. For more information about cost estimates, see [The Optimizer Cost Model](#).

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
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

System Name	Color	Text on Visual Diagram	Tooltip Related Information
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

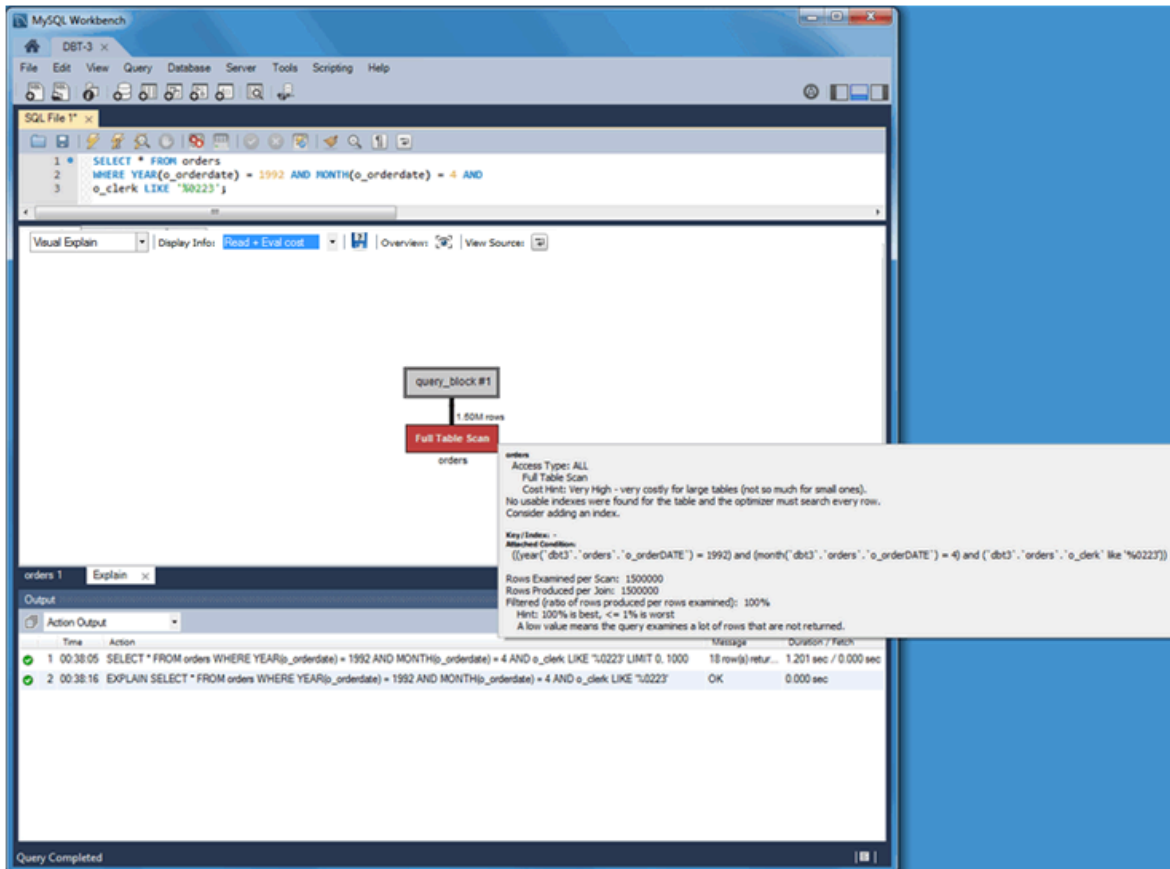
7.5 Tutorial: Using Explain to Improve Query Performance

This tutorial describes how to use Explain reports to locate and fix problematic (slow) queries. It uses the [DBT-3 database](#) and begins with the following simple query example.

```
SELECT * FROM orders
WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4
AND o_clerk LIKE '%0223';
```

As shown in the figure that follows, the query example was first executed in the Visual SQL editor. Next, an Explain report was generated by clicking **Explain Current Statement** from the **Query** menu. The initial report shows a Visual Explain image with information that appears when you move your pointer device over the `orders` table in full table scan.

Figure 7.9 DBT-3 Explain Tutorial: Visual Explain with Full Table Scan



Optionally, you can switch to Tabular Explain as the next figure shows. Use the drop-down list to switch between the visual and tabular representations.

Figure 7.10 DBT-3 Explain Tutorial: Tabular Explain with Full Table Scan

	id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
▶	1	SIMPLE	orders	ALL	NULL	NULL	NULL	NULL	1500000	Using where

Questions about the query:

- Why did this query generate a full table scan?
- Why is the indexed `o_orderdate` column missing as a possible key?

Looking more closely, also notice that the indexed column is being used in an expression as "`WHERE YEAR(o_orderdate) = 1992 AND MONTH(o_orderdate) = 4`", so the index is not used. To use the existing index, you can adjust the query as follows.

```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE '%0223';
```

The updated query example results in a Visual Explain image in which `Index Range Scan` replaces the `Full Table Scan` generated by the last query example. The next two figures show the visual and tabular representations of the modified query example.

Figure 7.11 DBT-3 Explain Tutorial: Visual Explain with Index Range Scan

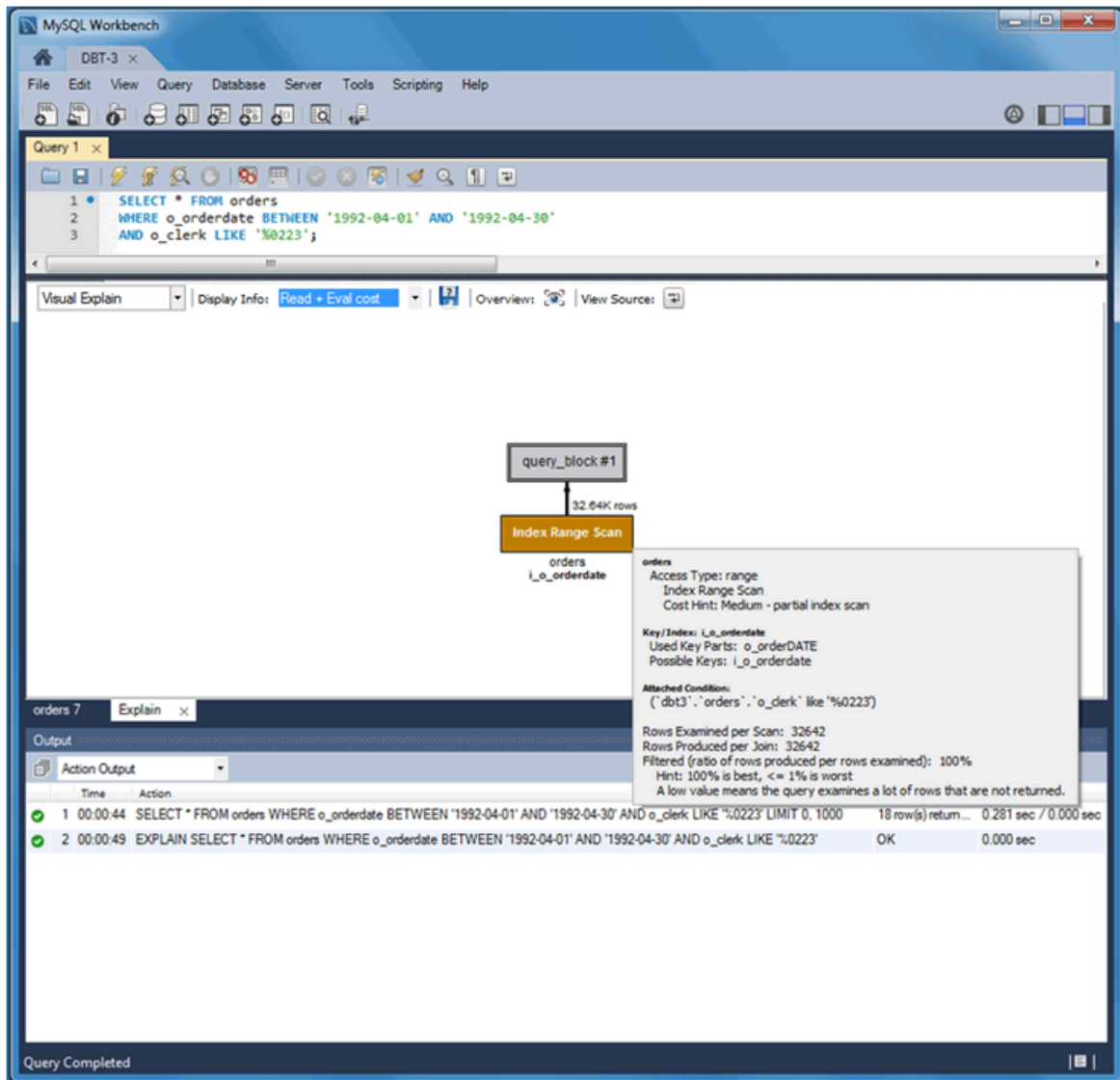


Figure 7.12 DBT-3 Explain Tutorial: Tabular Explain with Index Range Scan

	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 the focus can shift to the `o_clerk` column. The next query example (and Tabular Explain figure) adds the following index that should improve performance.

```
CREATE INDEX i_o_clerk ON orders(o_clerk);
```

Figure 7.13 DBT-3 Explain Tutorial: Tabular Explain with Index Range Scan and 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

The new index is not being considered as a possible key because the query is searching the suffix of the `o_clerk` column and indexes do not work with suffixes (although they do work with prefixes). Instead, this simple example could use the entire clerk ID. Adjusting the query as follows shows better results.

```
SELECT * FROM orders
WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
AND o_clerk LIKE 'Clerk#000000223';
```

The figures that follow represent the effect of the updated query example in Visual Explain and Tabular Explain respectively.

Figure 7.14 DBT-3 Explain Tutorial: Visual Explain with Index Range Scan and Full ID

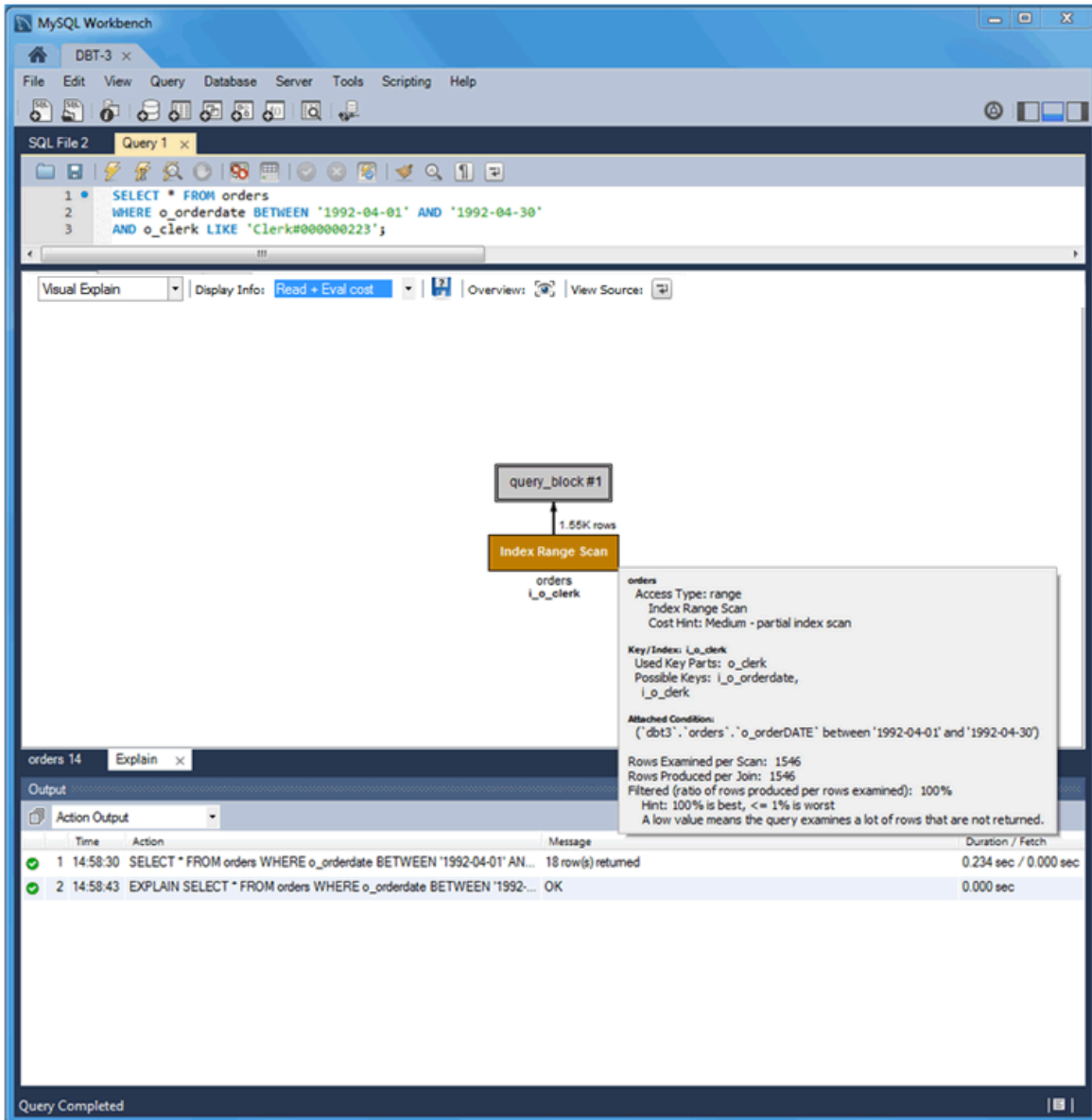


Figure 7.15 DBT-3 Explain Tutorial: Tabular Explain with Index Range Scan and Full ID

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 the 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 the query again, consider that a multiple-column index can meet the conditions of the `WHERE` clause that is based on both the `o_orderdate` and `o_clerk` columns as the next statement shows.

```
CREATE INDEX io_clerk_date ON orders(o_clerk, o_orderdate)
```



Note

`o_clerk` appears as the first column in the index because `o_orderdate` uses a range.

Now, executing the adjusted query produces even better results. An estimated 18 rows are both scanned and returned, and the execution time of the query example is 0.234 seconds as the next Visual Explain and Tabular Explain figures show.

Figure 7.16 DBT-3 Explain Tutorial: Visual Explain with Multiple-Column Index Range Scan

The screenshot shows MySQL Workbench with a query window containing the following SQL:

```
1 SELECT * FROM orders
2 WHERE o_orderdate BETWEEN '1992-04-01' AND '1992-04-30'
3 AND o_clerk LIKE 'Clerk#000000223';
```

The Visual Explain view shows a diagram with a box labeled "Index Range Scan" pointing to "query_block #1" with "18 rows". A tooltip for the "Index Range Scan" provides the following details:

- 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.

The Action Output window shows the following results:

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 7.17 DBT-3 Explain Tutorial: Tabular Explain with Multiple-Column Index Range Scan

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

The table that follows summarize the results of the modifications made to the query during this tutorial.

Table 7.2 DBT-3 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

8.1 Visual SQL Editor	201
8.1.1 SQL Query Tab	203
8.1.2 SQL Query Toolbar	204
8.1.3 Query and Edit Menus	206
8.1.4 Result Grid	207
8.1.5 SQL Additions - Snippets Tab	211
8.1.6 SQL Additions - Context Help Tab	213
8.1.7 Output Panel	214
8.1.8 Table Data Search Tab	216
8.1.9 Export or Import a Table	217
8.1.10 MySQL Table Editor	218
8.1.11 Code Generation Overview	227
8.2 Object Management	232
8.2.1 Object Browser and Editor Navigator	232
8.2.2 Session and Object Information Panel	235
8.2.3 Schema and Table Inspector	235

A set of visual tools to create, edit, and manage SQL queries, database connections, and objects.

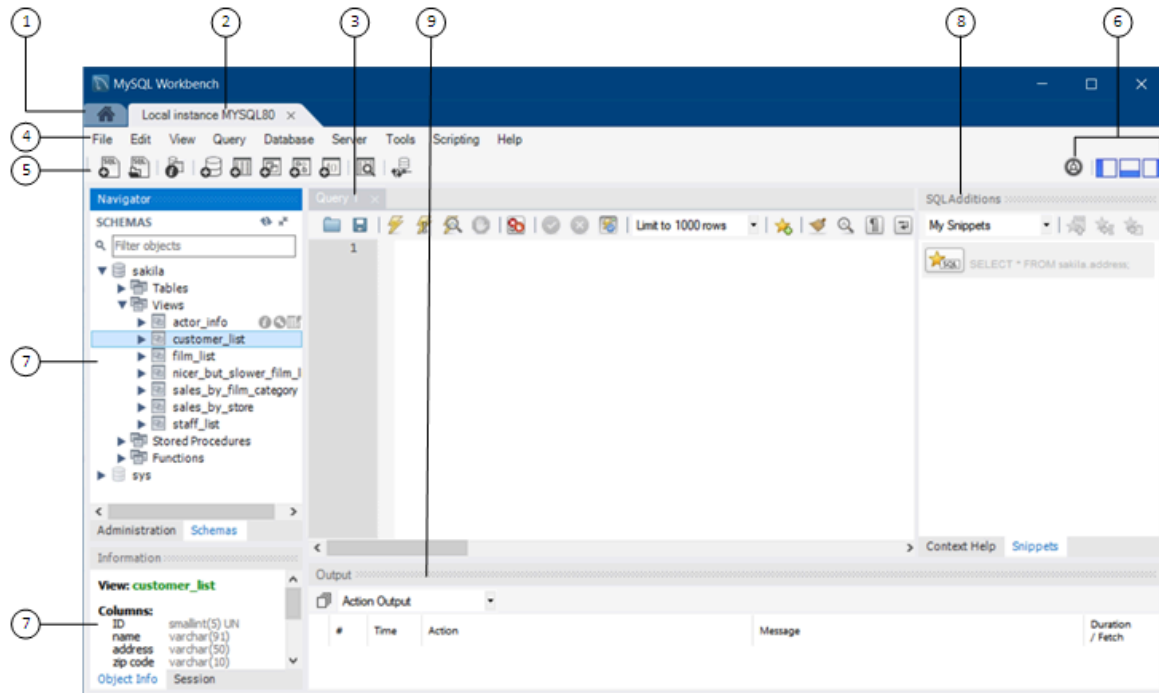
8.1 Visual SQL Editor

The visual SQL editor consists of a set of specialized editors (query, schema, table, and so on) and three panels: sidebar, secondary sidebar, and output area. Each editor opens in a separate secondary tab within an active MySQL connection tab. Each panel can be hidden or shown. Together the editors and panels enable you to:

- Build, edit, and run queries
- Create and edit data
- View and export results
- Perform basic RDBMS administrative tasks

Color syntax highlighting, context help, and code completion assist you in writing and debugging SQL statements. The integrated EXPLAIN plans provide data to help optimize the your queries. The following figure shows the main elements of the visual editor.

Figure 8.1 Visual SQL Editor




Description of the Visual SQL Editor Elements

1. Home screen tab. The Home screen tab provides quick access to connections, models, and the MySQL Migration wizard. Unlike the other main tabs, the Home screen tab does not close. For additional information, see [Chapter 4, Home Screen Tab](#).
2. Connection tab. Each connection made to the MySQL server is represented by a separate connection tab. A server can be active or inactive when the connection tab for it is opened. For assistance in creating and managing MySQL connections, see [Chapter 5, Connections in MySQL Workbench](#).
3. SQL query tab. The SQL query tab is a secondary tab that opens by default when you make a connection to a MySQL server. Each query tab is uniquely identified by an incrementing number: [query 1](#), [query 2](#), and so on. To close an open tab, click the **x** on the tab. For a closer look at query editing in MySQL Workbench, (see [Section 8.1.1, "SQL Query Tab"](#)).

All SQL query tabs provide an area to edit queries. You can open other specialized editors within tabs in this same central area. For example, you can edit schemas, tables, columns, and so on. Administration tabs also open in this area.

4. Main menu bar. The menu bar has the following menus: **File**, **Edit**, **View**, **Query**, **Database**, **Server**, **Tools**, **Scripting**, and **Help**. The actions available to you depend on which tab is selected when you click a menu. For a description of frequently used menus, see [Section 8.1.3, "Query and Edit Menus"](#).
5. Main toolbar. The quick actions in this toolbar are (ordered from left to right):
 - Create a new SQL tab for executing queries
 - Open an SQL script file in a new query tab
 - Open Inspector for the selected object
 - Create a new schema in the connected server

- Create a new table in the active schema in connected server
 - Create a new view in the active schema in the connected server
 - Create a new stored procedure in the active schema in the connected server
 - Create a new function in the active schema in the connected server
 - Search table data for text in objects selected in the sidebar schema tree
 - Reconnect to DBMS
6. Shortcut actions. Provides the following shortcuts (ordered from left to right):
- Show preferences dialog (see [Section 3.2, “Workbench Preferences”](#))
 - Hide or show the sidebar panel
 - Hide or show the output area panel
 - Hide or show the secondary sidebar panel
7. Sidebar panel. The sidebar has two main labels: Navigator and Information. The labels are omitted on some hosts.
- The Navigator has two subtabs: **Administration** (previously named **Management**) and **Schemas**. You can merge (or separate) the content of the two tabs into a single list by clicking merge ().
- The Information area provides the **Object Info** and **Session** subtabs, which include read-only information about a selected object and about the active connection.
8. Secondary sidebar panel (SQL Additions). The SQL Additions area provides the following subtabs:
- **Context Help** (see [Section 8.1.6, “SQL Additions - Context Help Tab”](#))
 - **Snippets** (see [Section 8.1.5, “SQL Additions - Snippets Tab”](#))
9. Output area panel. The output panel can display a summary of the executed queries in the following forms: Action Output, Text Output, or History Output. For additional information, see [Section 8.1.7, “Output Panel”](#).

The next sections describe how to use the visual SQL editor.

8.1.1 SQL Query Tab

The SQL query secondary tab opens by default when you make a connection to a server from the Home screen. It includes a query editor area and a toolbar. You can enter SQL statements directly into the query editor area. 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.


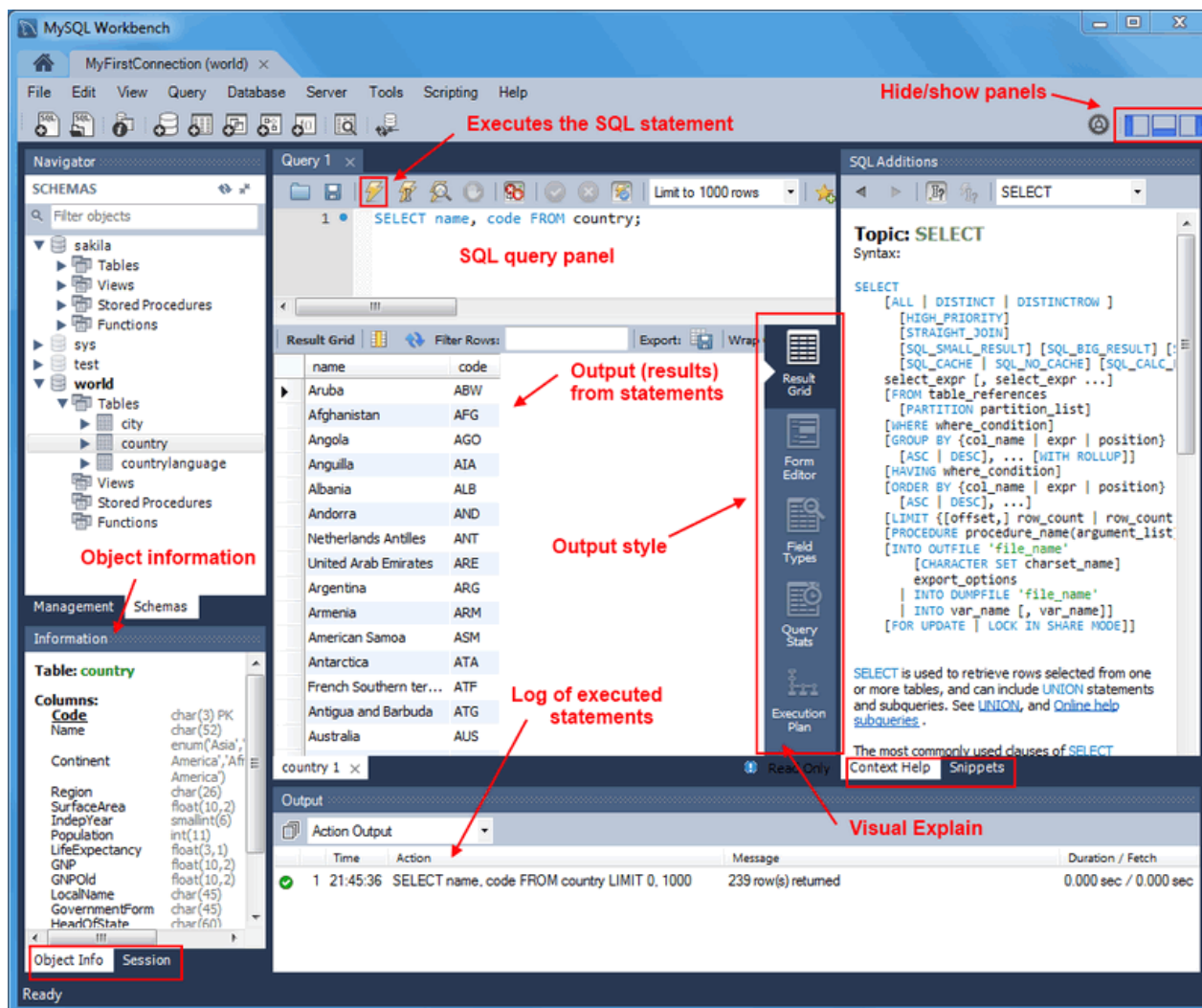
To save a snippet of code entered into the query editor, click **Save SQL to Snippets List** () from the SQL query toolbar, enter a name (optional), and click **OK**. The following figure shows the main elements of a query tab.

Figure 8.2 SQL Editor - SQL Query Tab



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 Schema Palette. The item name will be inserted into the SQL Query panel.

The SQL editor has several configurable panels and tabs, as shown in the previous figure.

8.1.2 SQL Query Toolbar

The SQL query toolbar provides actions that enable you to create and manage queries. The following figure shows the set buttons in the toolbar, located within the SQL query tab.

Figure 8.3 SQL Query Toolbar



SQL query tools (from left to right) are:

- **Open a script file in this editor:** Loads content from a saved SQL script into the SQL editor.
- **Save the script to a file:** Enables you to save the current content of the SQL editor to a file.
- **Execute the selected portion of the script or everything, if there is no selection:** Provides a simple way to execute the entire query or a subset of the query.
- **Execute the statement under the keyboard cursor:** Uses the position the keyboard cursor to identify and execute the query.
- **Execute the EXPLAIN command on the statement under the keyboard cursor:** Uses the position the keyboard cursor to identify the query and then executes `EXPLAIN`. A result grid tab is also displayed when executing an `EXPLAIN` statement.

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. The connection to the database server is not restarted and all open transactions 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 you click the button 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** pane. You can also set this behavior from the **SQL Execution** user preferences panel.
- **Commit the current transaction:** All query tabs in the same connection share the same transactions. To have independent transactions, you must open a new connection.
- **Rollback the current transaction:** All query tabs in the same connection share the same transactions. To have independent transactions, you must open a new connection.
- **Toggle autocommit mode:** When enabled, each statement is committed immediately. All query tabs in the same connection share the same transactions. To have independent transactions, you must open a new connection.

Autocommit is enabled by default. To disable the default behavior, see the **SQL Execution** section of the MySQL Workbench Preferences dialog.

- **Set limit for the number of rows returned by queries:** MySQL Workbench automatically adds the `LIMIT` clause with the configured number of rows to `SELECT` queries. The default value is 1000.

The default value (1000) can be changed from the **SQL Execution** section of the MySQL Workbench Preferences dialog.

- **Save current statement or selection to the snippet list:** For more information about the snippet list, see [Section 8.1.5, “SQL Additions - Snippets Tab”](#).
- **Beautify/reformat the SQL script:** By default, SQL keywords are changed to uppercase. This functionality can be changed from the **SQL Execution** section of the MySQL Workbench Preferences dialog.

- **Show the Find panel for the editor:** Click **Done** to close the panel.
- **Toggle display of invisible characters:** When selected, displays invisible characters, such as newlines, spaces, and tabs. A new line is represented as **[LF]**, a space as a single dot (**.**), and a tab as a right arrow.
- **Toggle wrapping of long lines:** When selected, wraps long lines in the SQL editor to eliminate the need to scroll. Deselecting this feature for long files is recommended.

8.1.3 Query and Edit Menus

When an SQL query 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 7.4, “Visual Explain Plan”](#) and [Section 7.5, “Tutorial: Using 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 provides 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 Result Grid

The results area of the screen shows the results from executed statements. If the script contains multiple statements, a result subtab will be generated for each statement that returned results. The following figure shows a single subtab and highlights the main features of the result grid.



Note

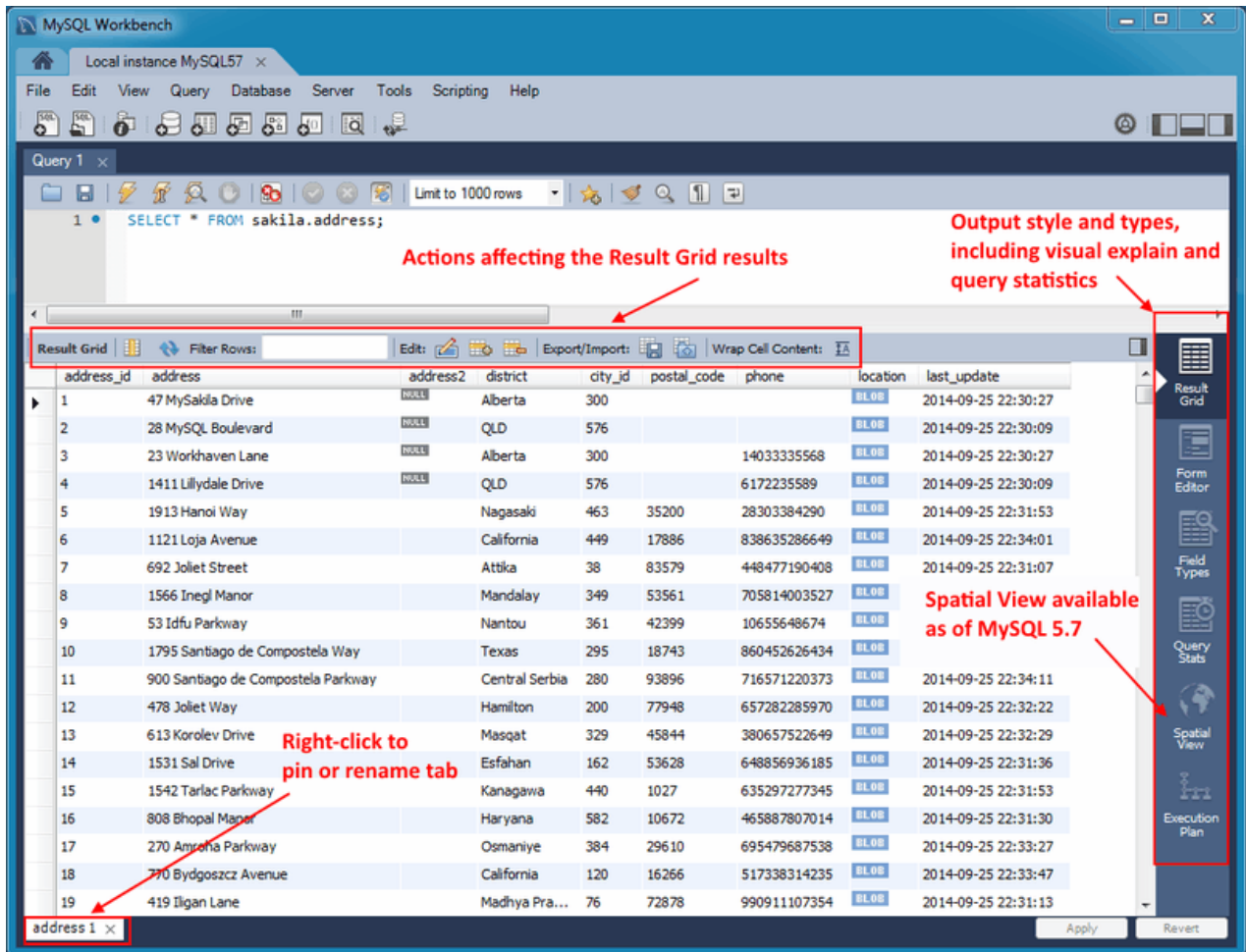
MySQL Workbench handles quoting and escaping for strings entered into the result grid, so adding quotes and proper escaping here is optional.



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



Result Grid Toolbar

Elements of the result grid toolbar include:

- **Reset:** Resets all sorted columns.
- **Refresh:** Refreshes all data by re-executing the original statement.
- **Filter Rows:** performs a search of all cells (not case-sensitive). 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** to 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.

Result Grid Tab Menu

Right-click a result grid subtab to open the context menu, which appears in the figure that follows.

Figure 8.5 SQL Editor - Result Grid Context Menu



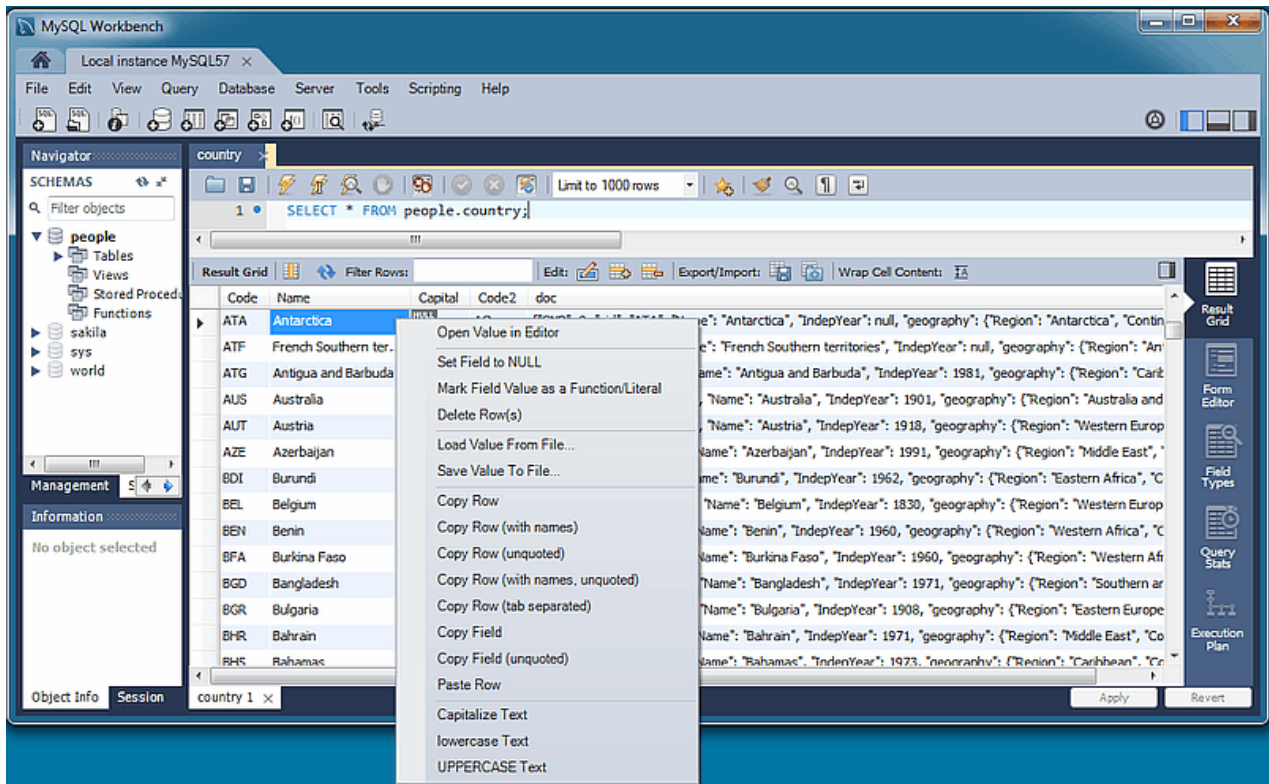
The result grid context menu includes:

- **Rename Tab:** Customize the name (title) of this tab.
- **Pin Tab:** Pin the results tab to the result 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.

Result Grid Field Menu

Right-click any field in the result grid to open the context menu for that field type. An example field with an open menu is shown in the figure that follows.

Figure 8.6 SQL Editor - Result Grid Field Menu



The result grid field-context menu includes:

- **Open Value in Editor:** Opens a new editor window that specializes in editing Binary and JSON data, but can edit text.
- **Show point in browser:** Evaluates whether the field is a geometry point and then opens the point in a browser using openstreetmap.org by default. An alternative online service can be configured (see [Section 3.2.7, "Other Preferences"](#)). This option is available for columns with valid point-location types only.
- **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 the value of a field 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.

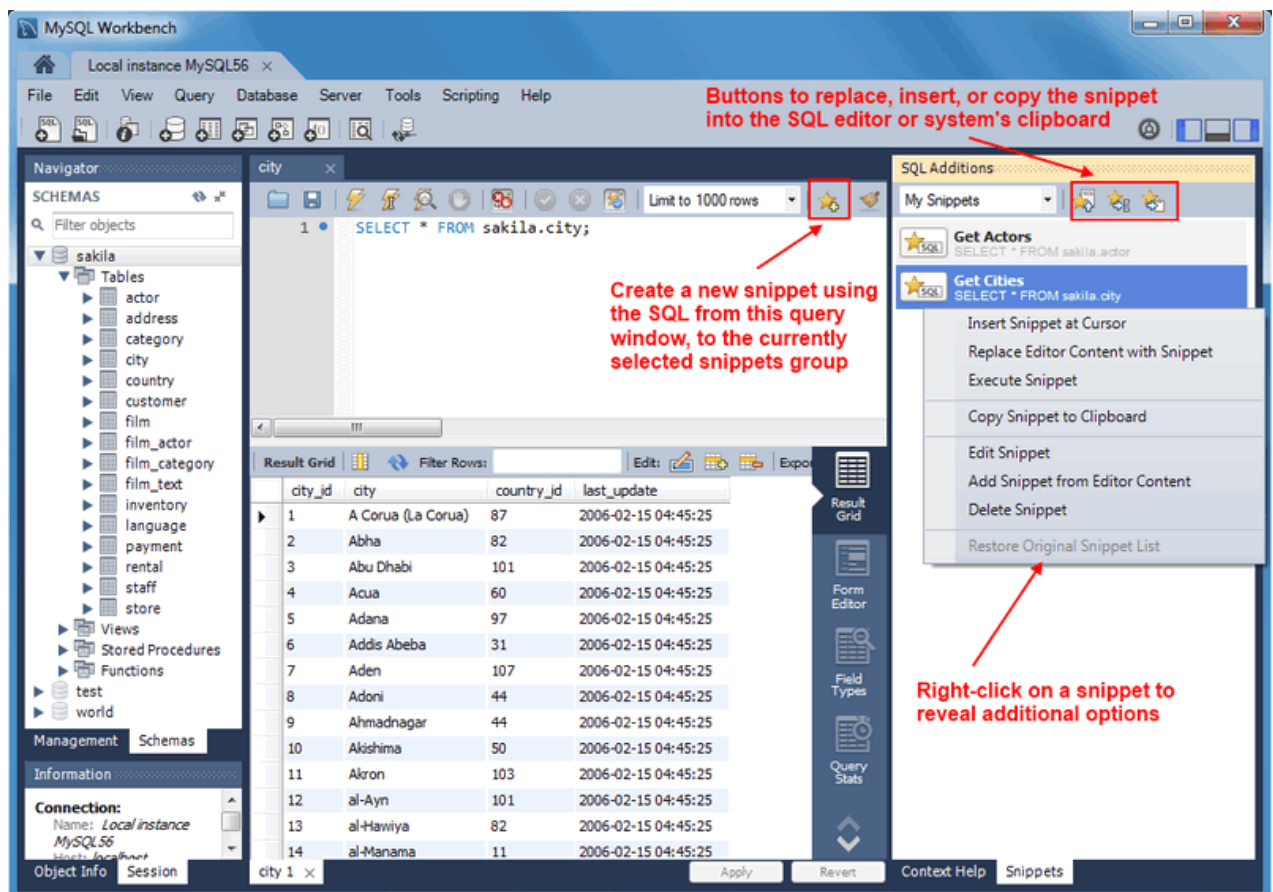
8.1.5 SQL Additions - Snippets Tab

The Snippets secondary tab includes built-in, local, and shared custom snippets. The **My Snippets** option 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. The next figure shows the location of the main action icons to use with snippets.

Figure 8.7 SQL Snippets: Usage



Local Snippets (My Snippets)

Local snippets are stored in the MySQL Workbench directory. By default, the **My Snippets** SQL snippets are stored as indicated in the following table.

Table 8.1 Default Local Snippet File Location

Operating System	File Path
Windows	%AppData%\MySQL\Workbench\User Snippets.txt
macOS	~username/Library/Application Support/MySQL/Workbench/snippets/User Snippets.txt
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


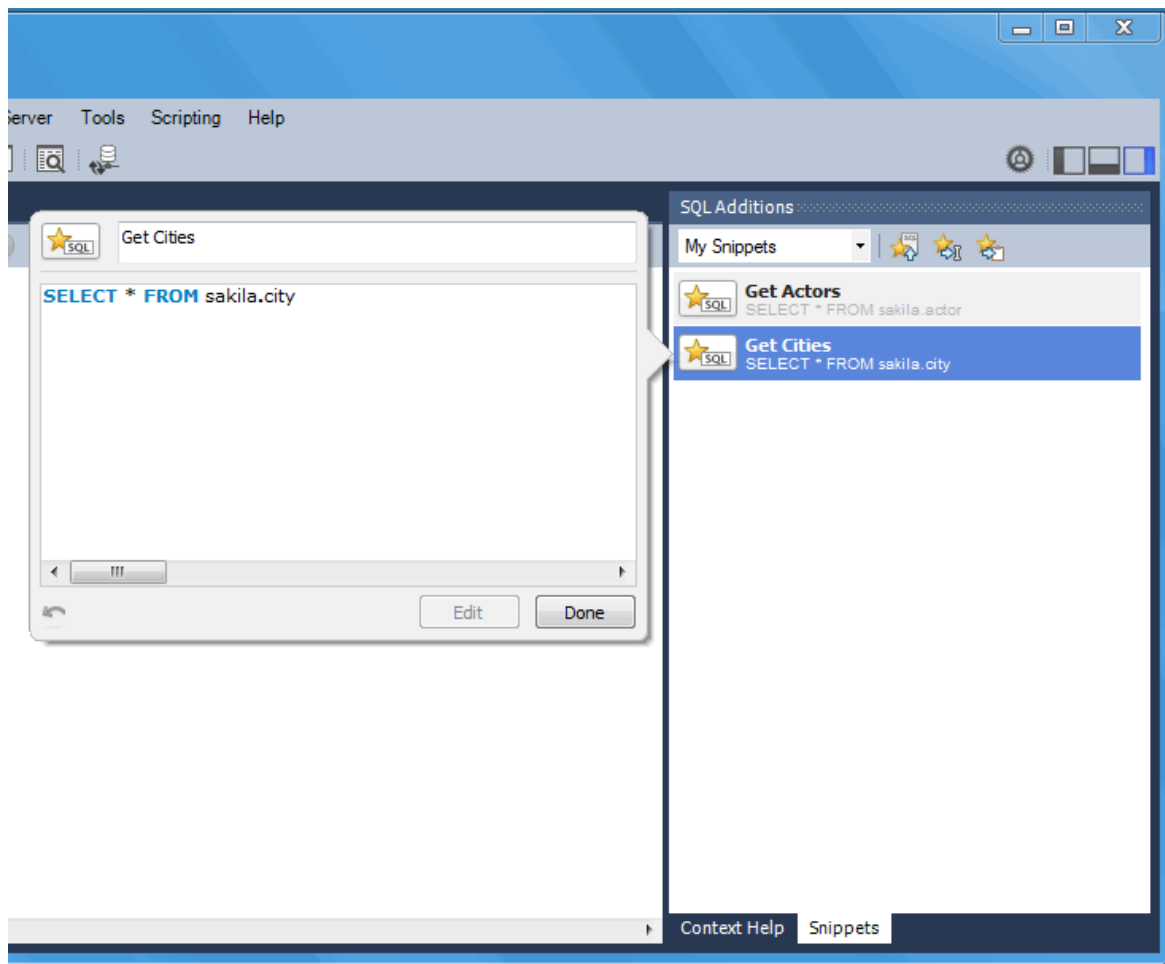
To save a snippet, choose the Snippets Insert icon () or right-click in the snippet window and choose **Add Snippet from Editor Content** from the context-menu. Double-click a snippet to open it, and choose the snippet editor to edit its body or title. The example in the following figure shows two snippets with only the first having defined a name.


Figure 8.8 SQL Snippets: Editor




8.1.6 SQL Additions - Context Help Tab


Within the visual SQL Editor, you can select a keyword or function in your query to open a help topic describing its syntax and usage. The level of information is equivalent to typing `help keyword` at the `mysql` prompt and the content is specific to the release series (5.6, 5.7, or 8.0) of the current MySQL connection. Context help requires no MySQL connection to use because the context help is stored locally.

To view context help, first open the **SQL Additions** panel by clicking **View, Panels**, and then **Show**

Secondary Sidebar from the menu. Alternatively, you can click the panel shortcut () from the toolbar. This panel has two bottom tabs; select **Context Help**.

You have several options for displaying context help: automatic, manual, and most frequent list. Initially, automatic context help is disabled.

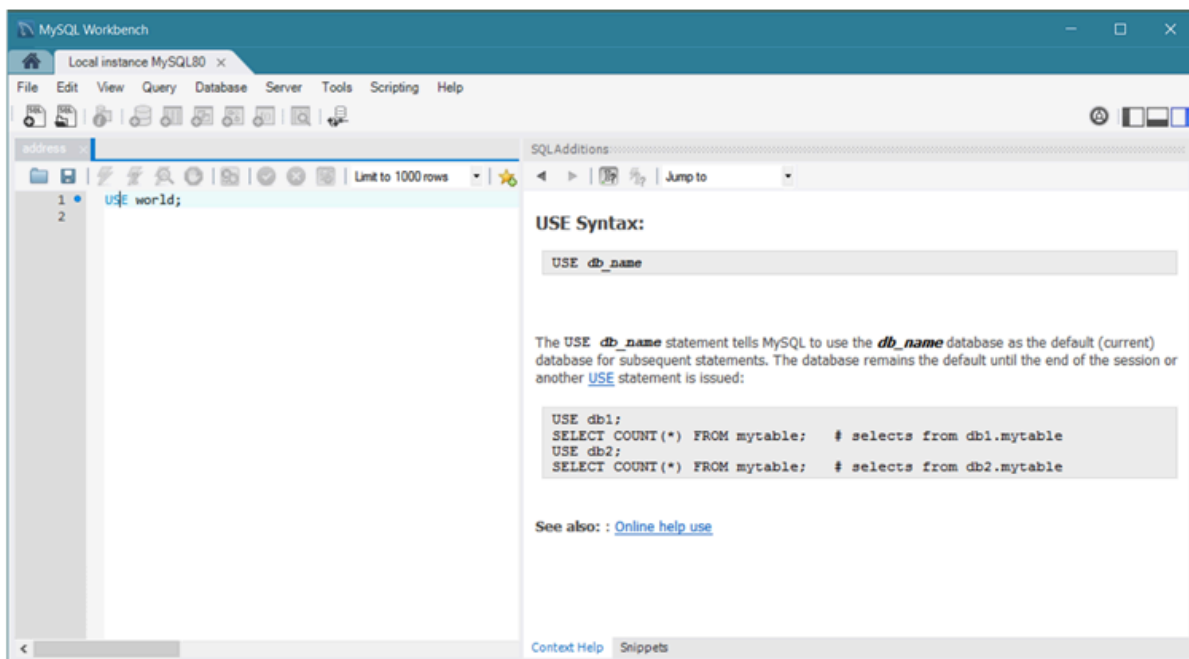
Automatic context help. To enable automatic context help, click the automatic context button () from the SQL Additions toolbar. As you click on or near different SQL keywords and functions in the editor, the context help shifts to the specific topic.

Manual context help. Manual context help is available when automatic context help is disabled. Click on or near the SQL keyword in the editor and then click the manual context button () from the SQL Additions toolbar to view each new topic.

Most frequent list. Use the drop-down list to select a help topic for display. The set of SQL keywords in this list is predefined. Most frequent keywords, when selected in the SQL editor, also display in the list box. Less common keywords display [Jump To](#) in the list box when selected from the editor in either automatic or manual mode.

The SQL Additions toolbar also includes forward and reverse arrows that enable you to browse recent content help topics. The following figure shows an example in which automatic context help is enabled and open. Note that when automatic context help is enabled, the manual context button is disabled.

Figure 8.9 SQL Editor: Context Help



8.1.7 Output 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. The following figure shows this output, which can be useful to troubleshoot a script.

Figure 8.10 SQL Editor: Output: Action Output

The screenshot shows the MySQL Workbench interface. The main window displays the 'actor' table from the 'sakila' database. The Output Panel is visible at the bottom, showing a list of executed SQL statements. The 'Action Output' scheme is selected in the dropdown menu. The list of statements includes successful queries and one error message: 'Error Code: 1146. Table 'worlds.country' doesn't exist'.

	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

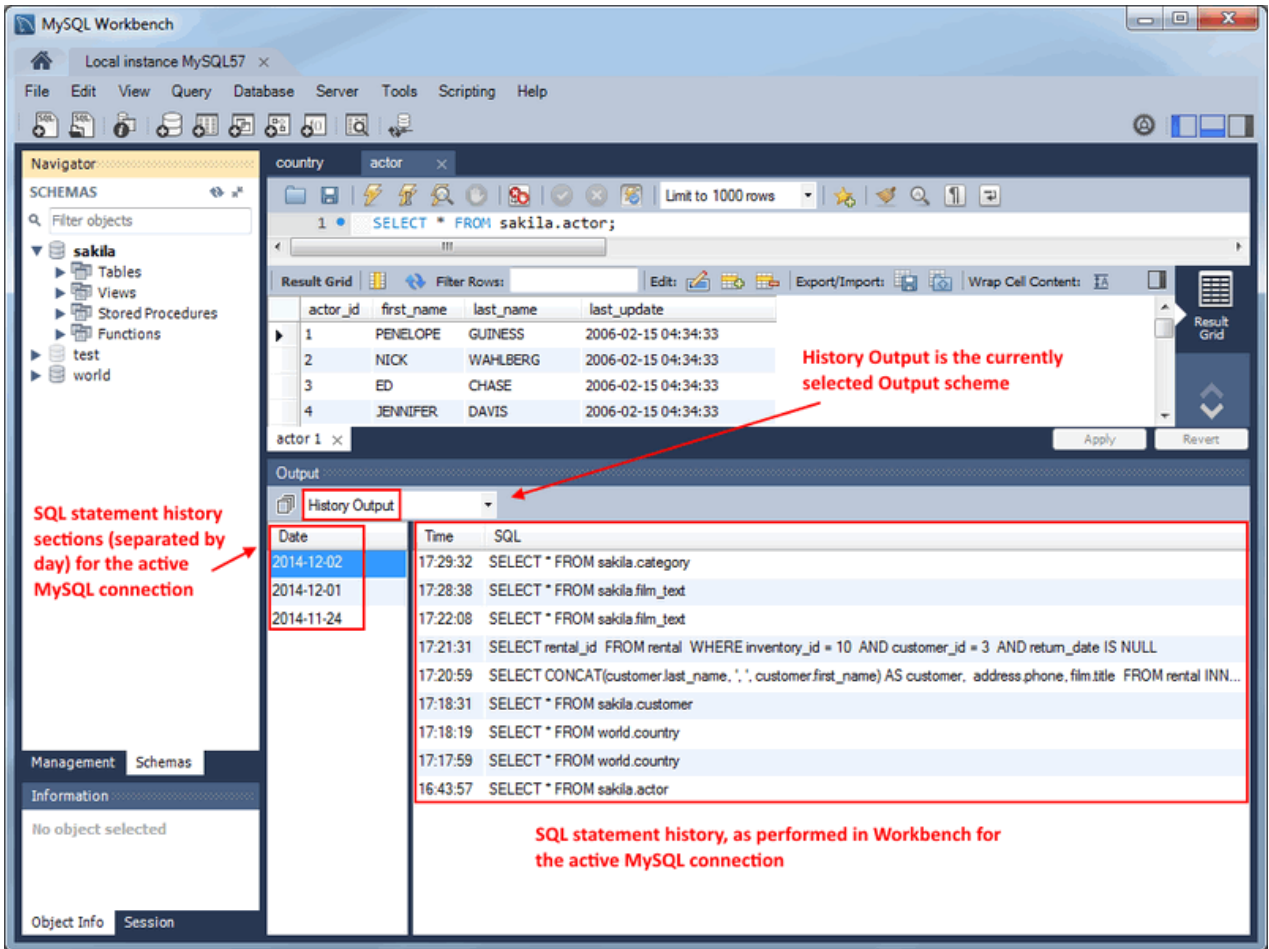
Action Output is the currently selected Output scheme

Error status icon

Action Output: A summary of executed MySQL statements in the current Workbench session, including errors and general status information.

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, as the following figure shows.

Figure 8.11 SQL Editor: History Output

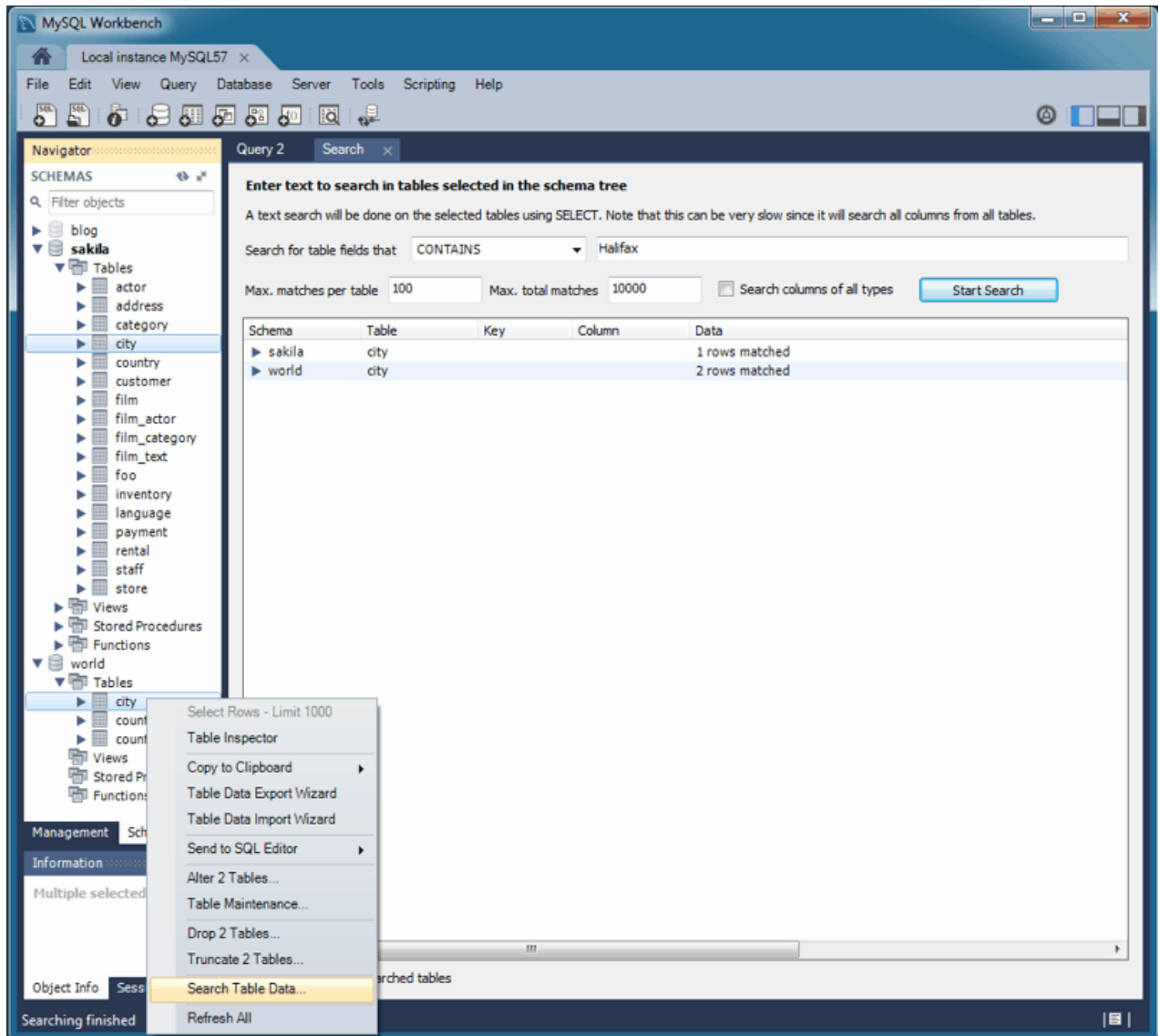


8.1.8 Table Data Search Tab

Find data across a MySQL connection by using the text search feature on any number of tables and schemas.

From the schema tree, select the tables, schemas, or both to search and then right-click the highlighted items and click **Search Data Table** from the context menu. The following figure shows the options available in the open **Search** secondary tab.

Figure 8.12 Table Search Example: Multiple Tables and Schemas



The search options include:

- **Search for table fields that:** "CONTAINS", "Search using =", "Search using LIKE", "Search using REGEXP". These search options are not case-sensitive.
- **Max. matches per table:** [100]
- **Max. total matches:** [1000]
- **Search columns of all types** check box: 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 or Import a Table

Export or import tables using a wizard to assist you.

**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 MySQL Table Editor

The MySQL Table Editor is used to create and modify tables. You can add or modify the columns or indexes of a table, change the engine, add foreign keys, or alter the table name.

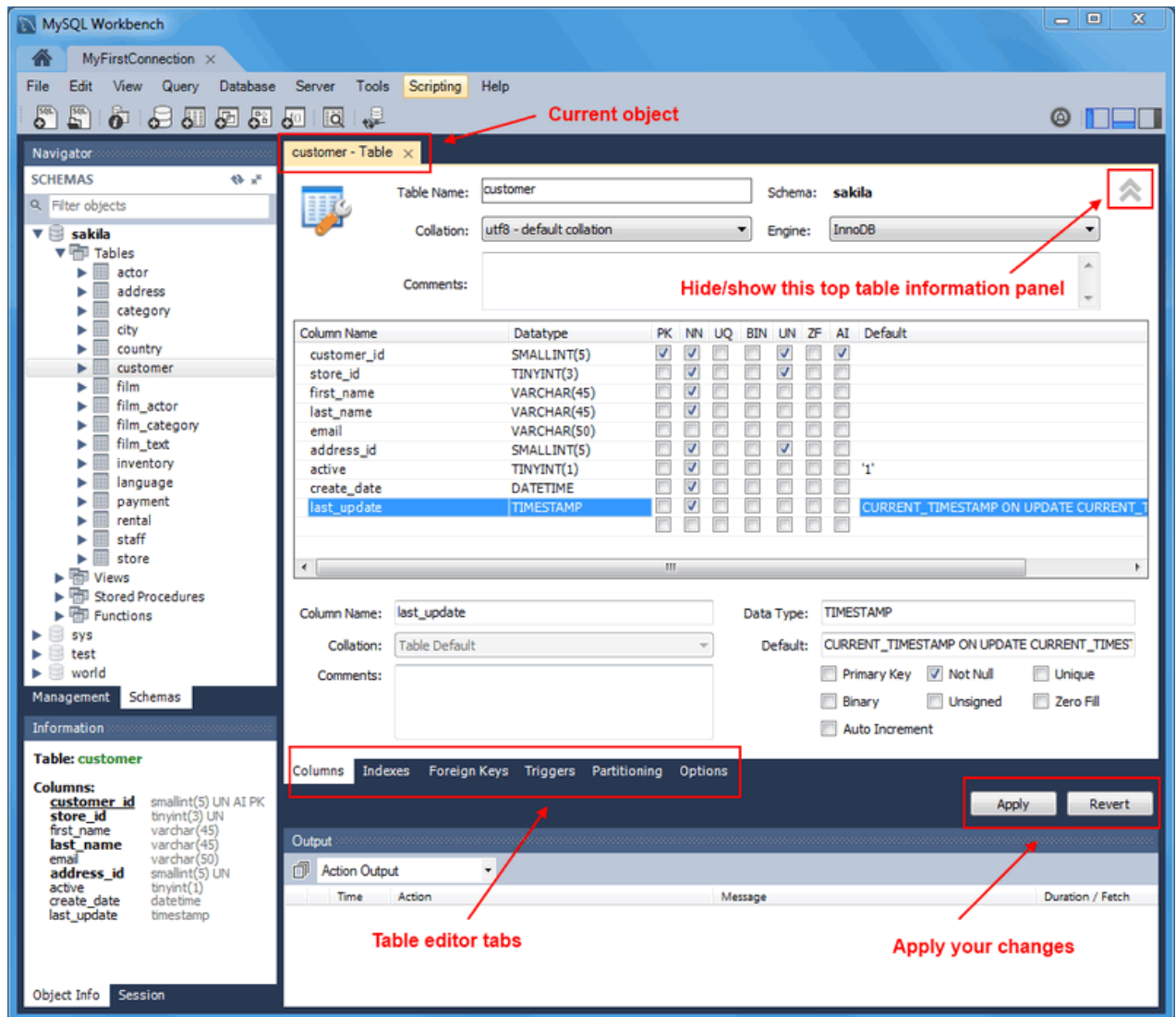
To access the MySQL Table Editor, right-click a table name in the Navigator area of the sidebar with the **Schemas** secondary tab selected and click **Alter Table**. This action opens a new secondary tab within the main **SQL Editor** window. You can also access the MySQL Table Editor from an EER Diagram by double-clicking on a table object.

8.1.10.1 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 secondary 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 for the `customer` table with the **Columns** subtab selected is shown in following figure.

Figure 8.13 The Table Editor



The MySQL Table Editor provides a work space that has subtabs used to perform these actions:

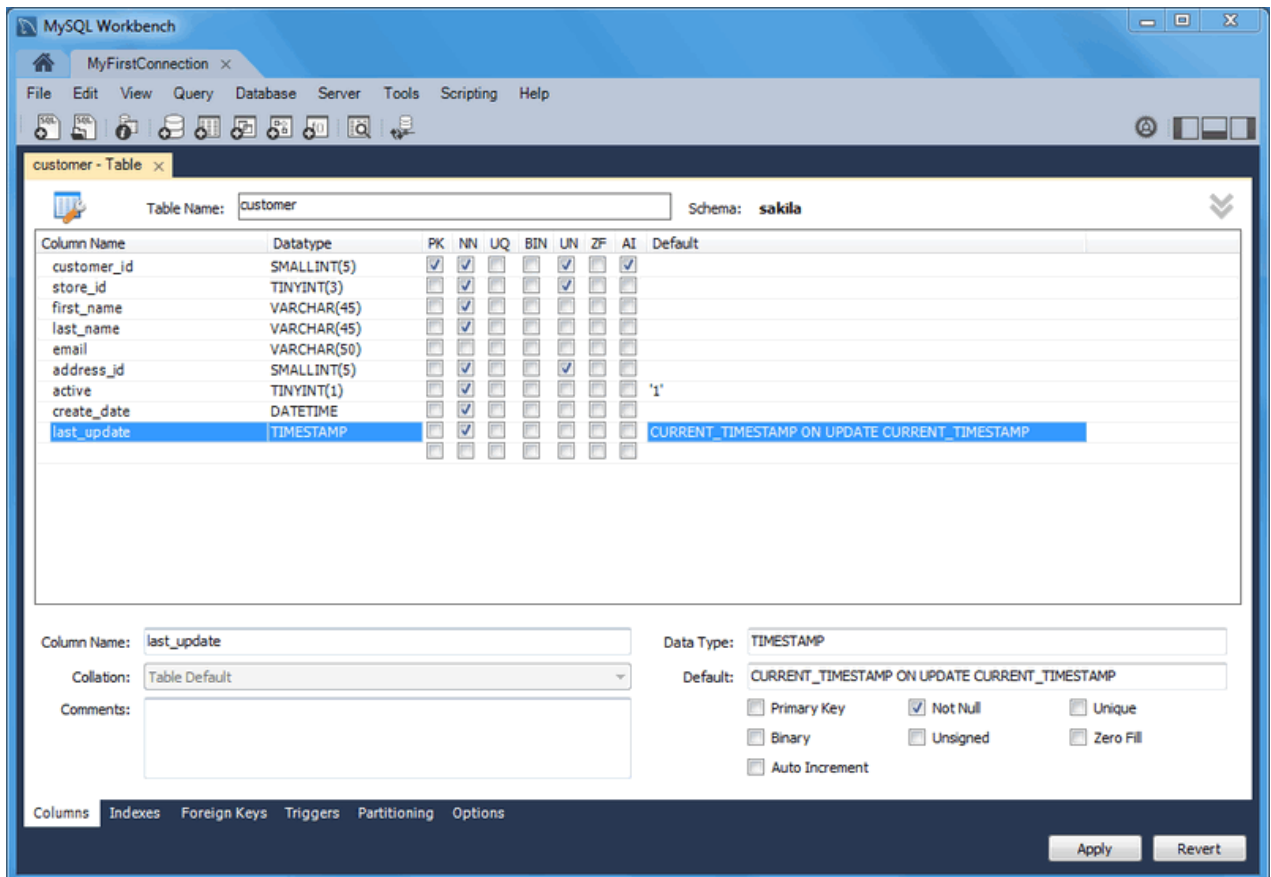
- **Columns:** Add or modify columns.
- **Indexes:** Add or modify indexes.
- **Foreign Keys:** Add or modify foreign keys.
- **Triggers:** Add or modify triggers.
- **Partitioning:** Manage partitioning of a table.
- **Options:** Add or modify other options, divided in categories named general, row, storage, and merge.

8.1.10.2 Columns Tab

Use the **Columns** subtab to display and edit all the column information for a table. With this subtab, you can add, drop, and alter columns.

You can also use the **Columns** subtab to change column properties such as name, data type, and default value. The following figure shows an example of the **Columns** subtab.

Figure 8.14 The Columns Tab



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** subtab.
- **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 that follow. For a description of each item, see [CREATE TABLE](#).

- **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.



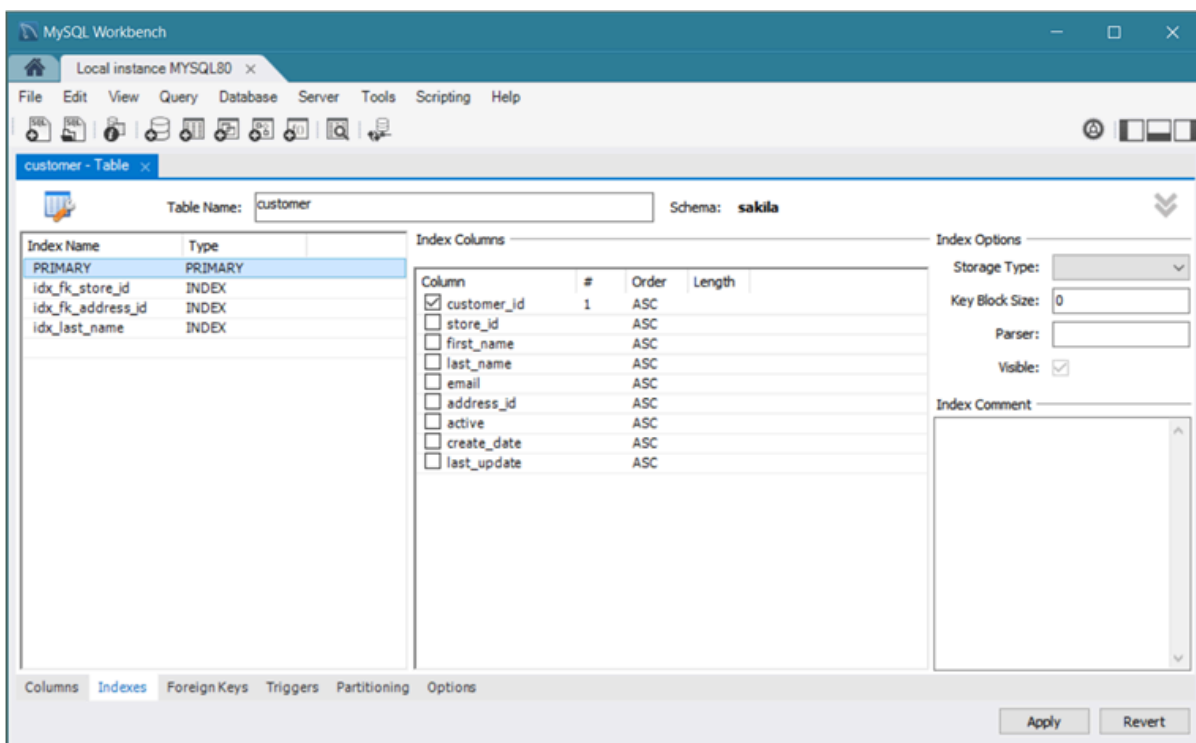
Note

ENUM, BIT, and SET must contain at least one value when entering these data types into MySQL Workbench.

8.1.10.3 Indexes Tab

The **Indexes** subtab contains all of the index information for your table. Use this subtab to add, drop, and modify indexes. The following figure shows an example of the layout with the [PRIMARY](#) index of the [customer](#) table selected and both the index columns and index options shown.

Figure 8.15 The Indexes Tab



All indexes for a table are listed by index name. Click an index name to display the **Index Columns** section with information about the selected index. In addition, you can configure the storage type, key block size, parser, and the visibility of the index. Index comments, when added, apply to the selected index only.

The actions available from this subtab include:

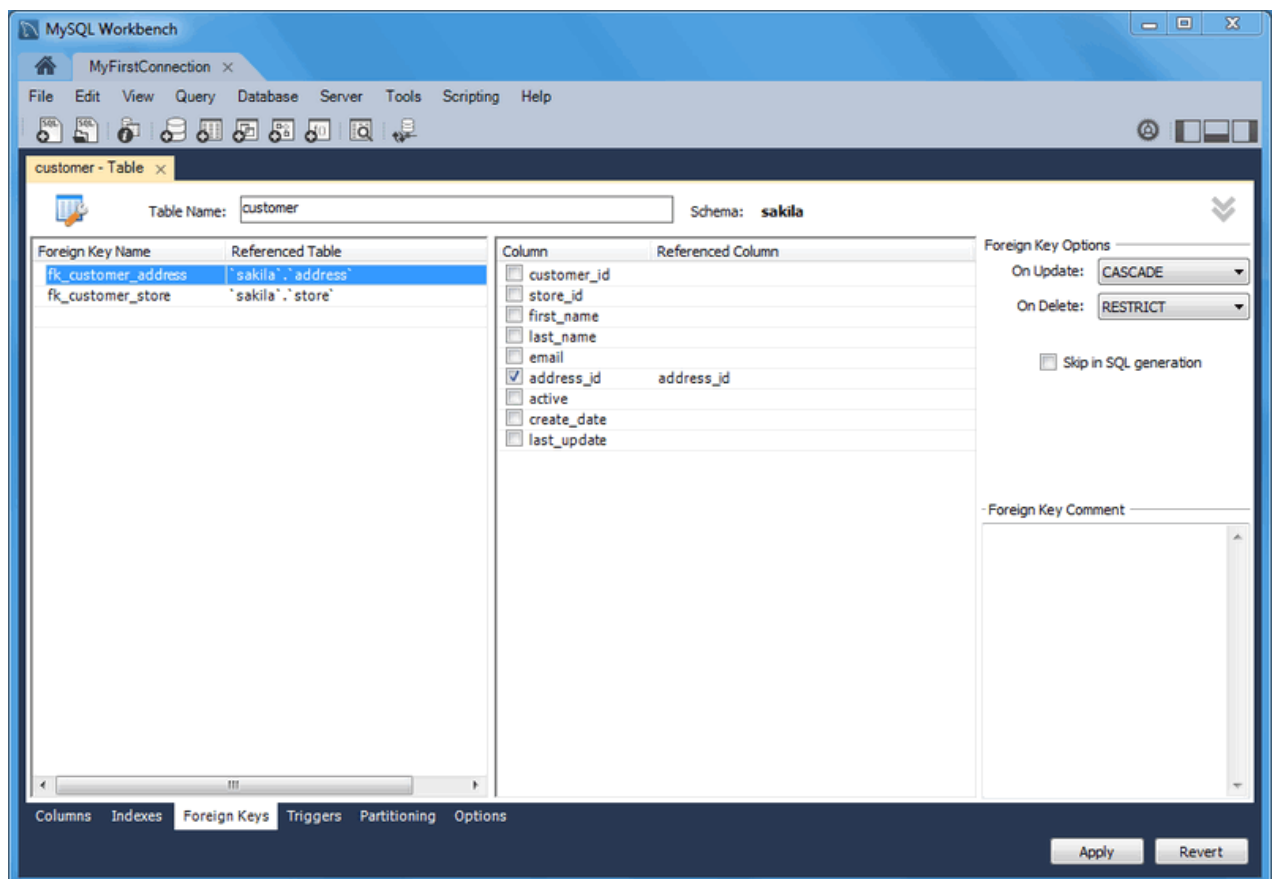
- To add an index, click the last row in the index list. Enter a name for the index and select the index type from the list. Select the column or columns that you wish to index by checking the column name in the **Index Columns** list. You can remove a column from the index by removing the check mark from the appropriate column.
- To specify the order of a column within an index, select [ASC](#) or [DESC](#) from 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 render a secondary index (an index other than the primary key or a unique column) invisible to the optimizer, deselect the **Visible** option. By default, all indexes are visible. This feature must be supported by the active server; otherwise, the **Visible** check box is grayed out. For a description of the usage patterns for this option, see [Invisible Indexes](#).
- To drop an index, right-click the index name and then click the **Delete Selected** menu item.

8.1.10.4 Foreign Keys Tab

The **Foreign Keys** subtab is organized in much the same fashion as the **Indexes** subtab and adding or editing a foreign key is similar to adding or editing an index. The following figure shows an example of the **Foreign Keys** tab.

Figure 8.16 The Foreign Keys Tab



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.

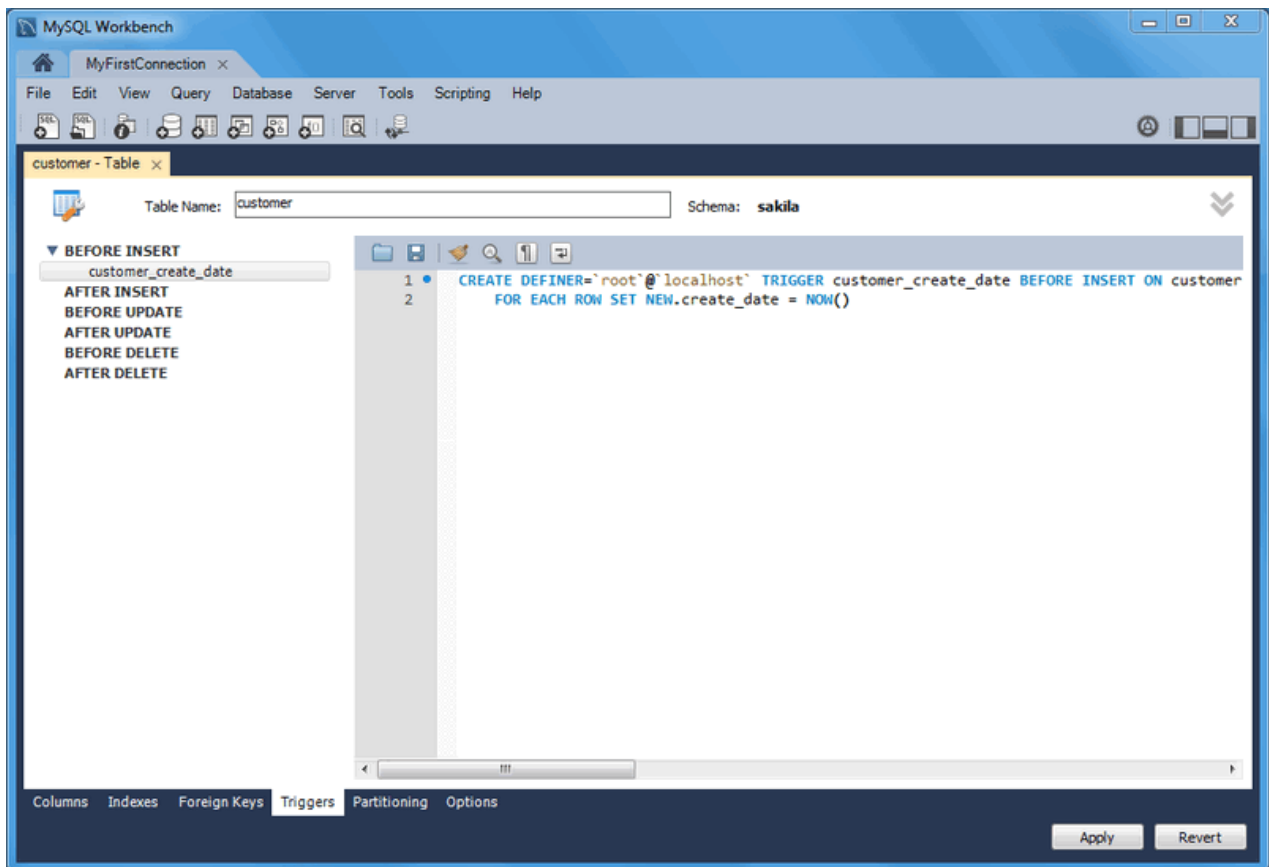
To modify properties of a foreign key, select it and make the desired changes.

8.1.10.5 Triggers Tab

The **Triggers** subtab opens a workspace that enables you to create new triggers or edit existing triggers. All triggers are organized within a tree structure by section, such as **BEFORE INSERT** and **AFTER INSERT**.

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. The next figure shows an example of the `customer_create_date` trigger.

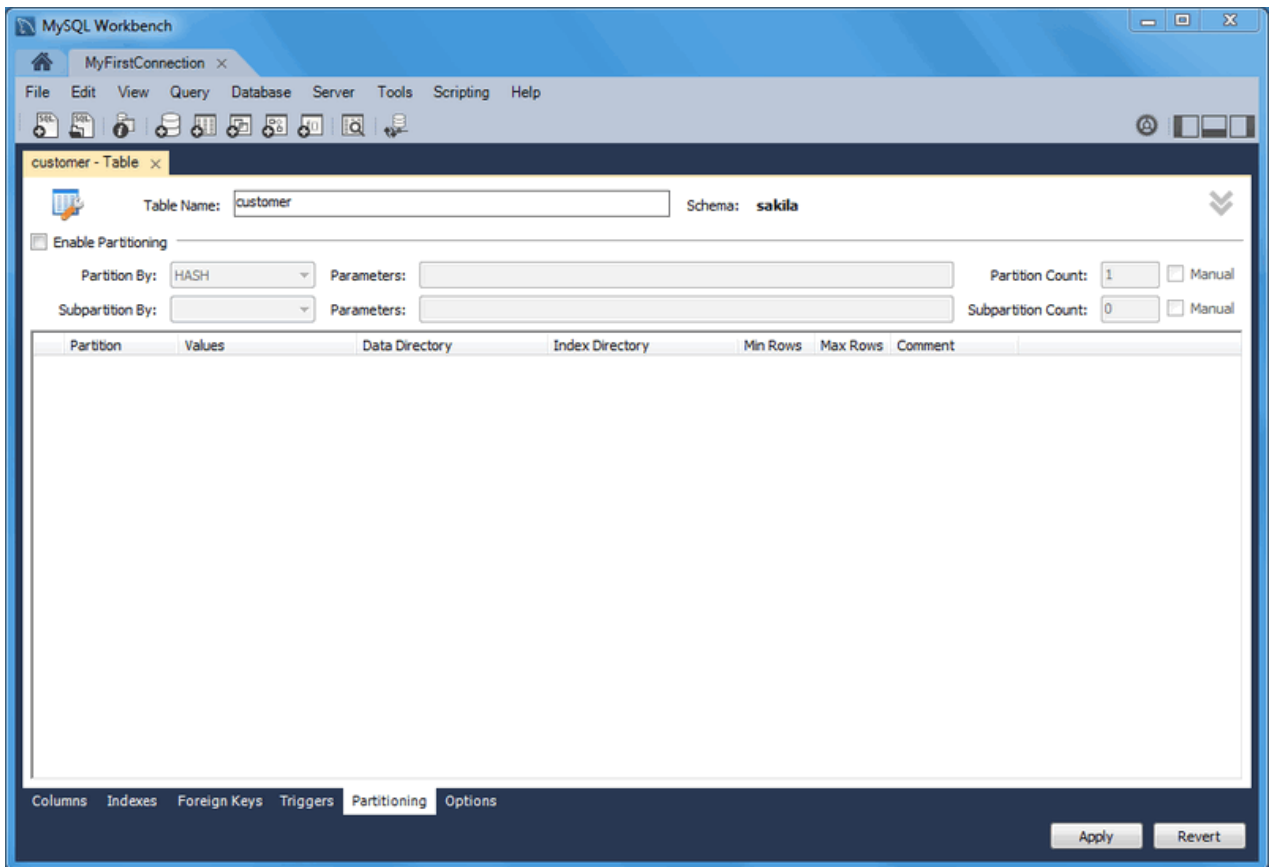
Figure 8.17 The Triggers Tab



8.1.10.6 Partitioning Tab

To enable partitioning for your table, check the **Enable Partitioning** check box. This enables the partitioning options (shown in the figure that follows).

Figure 8.18 The Partitioning Tab



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](#).

8.1.10.7 Options Tab

The **Options** subtab enables you to set several types of options. The next figure shows an example of this tab.

Figure 8.19 The Options Tab

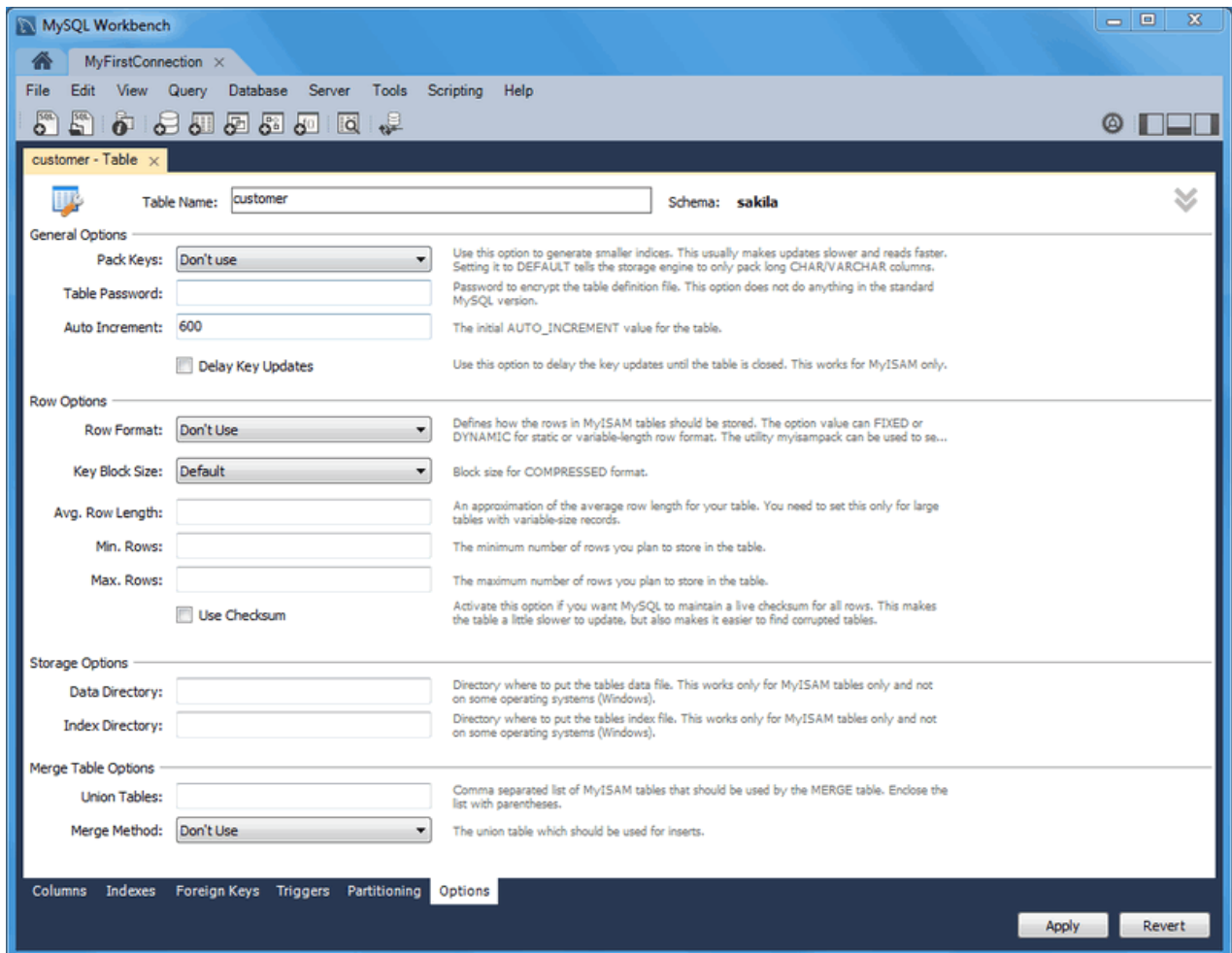


Table options are grouped into the following sections:

- **General Options**
- **Row Options**
- **Storage Options**
- **Merge Table Options**

The following sections describe 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 Statement](#) 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](#).

8.1.11 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.

8.1.11.1 Generating SQL Statements

MySQL Workbench can be used to generate SQL, most typically as either [INSERT](#) statements or [SELECT](#) statements.

The following common methods are 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', 'GUINNESS', '2006-02-15 04:34:33'
```

Copy Rows (with names, unquoted)

```
# actor_id, first_name, last_name, last_update
1, PENELOPE, GUINNESS, 2006-02-15 04:34:33
```

Copy Row (tab separated)

```
1 PENELOPE GUINNESS 2006-02-15 04:34:33
```

Copy Field

```
'GUINNESS'
```

8.1.11.2 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.

The example scenario that follows demonstrates 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 and PHP](#).

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 area of the sidebar contains options to manage the active MySQL connection. It also lists the schemas on the server for that connection. To access the Navigator area, open an existing connection (or create a new connection) from the home screen. If the panel is not visible, click **View, Panels**, and then **Show Sidebar**.

Navigator Schemas Tab

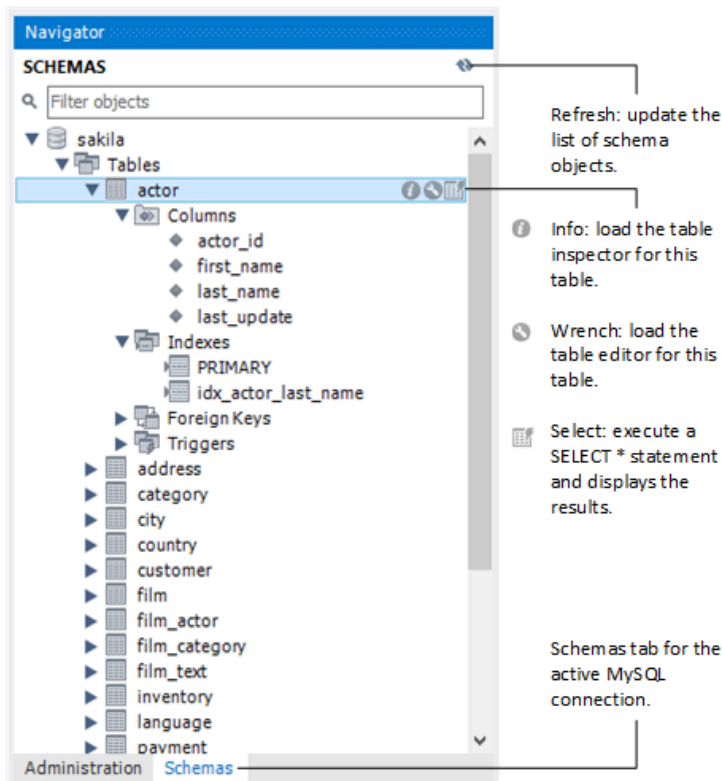
The **Schemas** tab shows available schema on the currently connected server. These items can be explored to show tables, views, and routines within the schema as the following figure shows.



Note

Internal schemas, such as `performance_schema`, `information_schema`, and `mysql`, are hidden by default. Select the **Show Metadata and Internal Schemas** preference (see [Preferences: SQL Editor: Main](#)) to list them in the object browser. Schemas beginning with the period character (.) are also controlled by this setting.

Figure 8.20 SQL Editor - Navigator Schemas Tab



Schema context menu. Right-click any schema object to show the context menu. Right-clicking on a schema provides similar options to the table context menu (see [Table, view, and column context](#)

menus), but the operations refer to the schema. For example, the **Table Maintenance** item in the table context menu opens the **Schema Inspector**, which is a schema context menu item, but it is populated with information about the selected table.

- **Load Spatial Data:** Imports a shapefile (.shp) containing spatial data to load into MySQL. A new table with the imported fields is created in the selected schema, unless you select the append or update (overwrite) option. Another option creates a spatial index. If enabled, the import operation makes a spatial index around the geometry column. You can import spatial data with or without an EPSG format conversion.
- **Set as Default Schema:** Sets the selected schema as the default schema. 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.

- **Filter to This Schema:** Enables you to target specific schemas in the list.
- **Schema Inspector:** Displays information about the selected schema. For additional information, see [Schema Inspector](#).
- **Table Data Import Wizard:** Opens the wizard.
- **Copy to Clipboard:** Copies the schema name or a `CREATE` statement to the clipboard.
- **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 Schema:** Launches a dialog to enable you to create a new schema.
- **Alter Schema:** Launches a dialog to enable you to change the name or character/collation of an existing schema.
- **Drop Schema:** Drops the schema. All data is lost if this operation is carried out.
- **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 objects in the schema tree by resynchronizing with the server.

Double-clicking a table, view, or column name in the schema 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.

Table, view, and column context menus. The schema navigator also features a context menu, which can be displayed by right-clicking a table, view, or column object. For example, right-clicking a table displays the following menu items:

- **Select Rows - Limit 200:** Pulls up to 200 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.

- **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}>;`.
- **Join Select Tables:** Joins the selected tables.
- **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 Row References:** Copies a `SELECT` statement that selects all objects that reference the row (directly or indirectly).

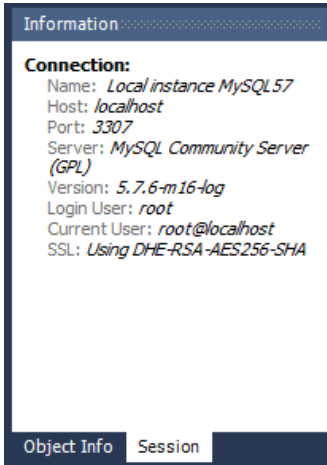
Use **Delete with References** to generate a `DELETE` statement for this operation.
- **Table Data Export Wizard:** Opens the table export wizard to export the table's data to JSON or customized CSV.
- **Table Data Import Wizard:** Opens the table import wizard to import JSON or CSV formatted data to the selected or new table.
- **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 objects in the schema tree by resynchronizing with the server.

8.2.2 Session and Object Information Panel

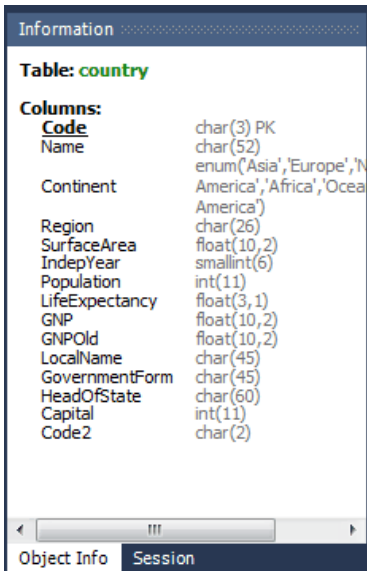
The **Session** tab of the Information panel summarizes the current connection to the server, as the following figure shows.

Figure 8.21 SQL Editor - Connection Information Palette



The **Object Info** tab of the Information panel summarizes information about a specific object, such as a table. The following figure shows an example of object information.

Figure 8.22 SQL Editor - Object Info



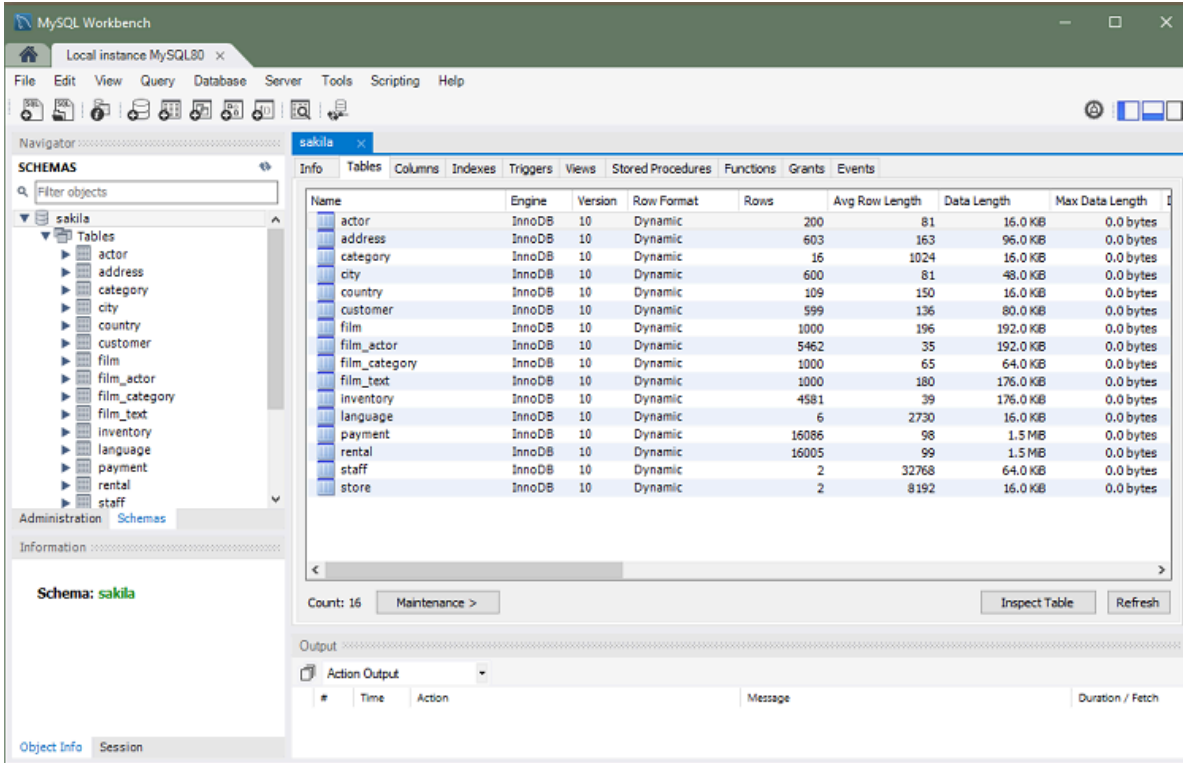
8.2.3 Schema and Table Inspector

The Schema Inspector and Table Inspector each include 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 (shown in the figure that follows). It allows you to perform maintenance tasks on tables such as ANALYZE, OPTIMIZE, CHECK, and CHECKSUM TABLE. To access the inspector, right-click a schema and select the **Schema Inspector**.

Figure 8.23 Schema Inspector



Each tab lists topic-oriented information in categories, 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 (shown in the figure that follows).

Figure 8.24 Schema Inspector: Table Maintenance

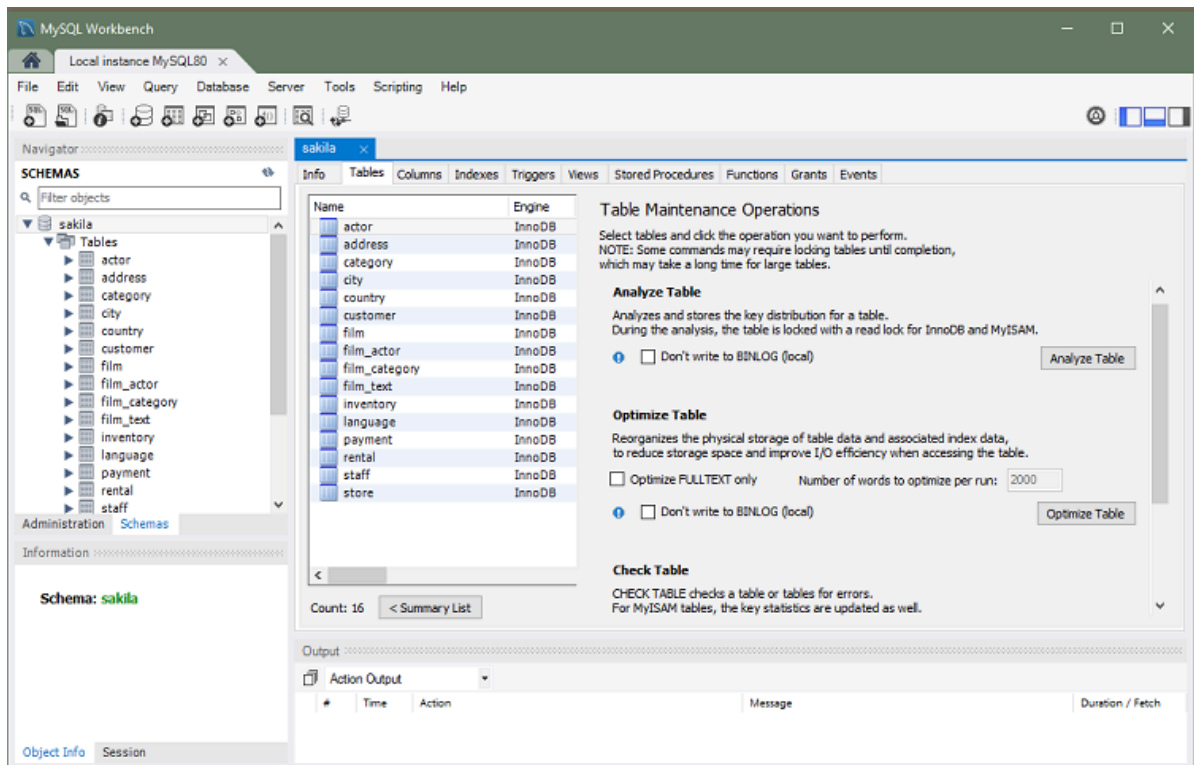
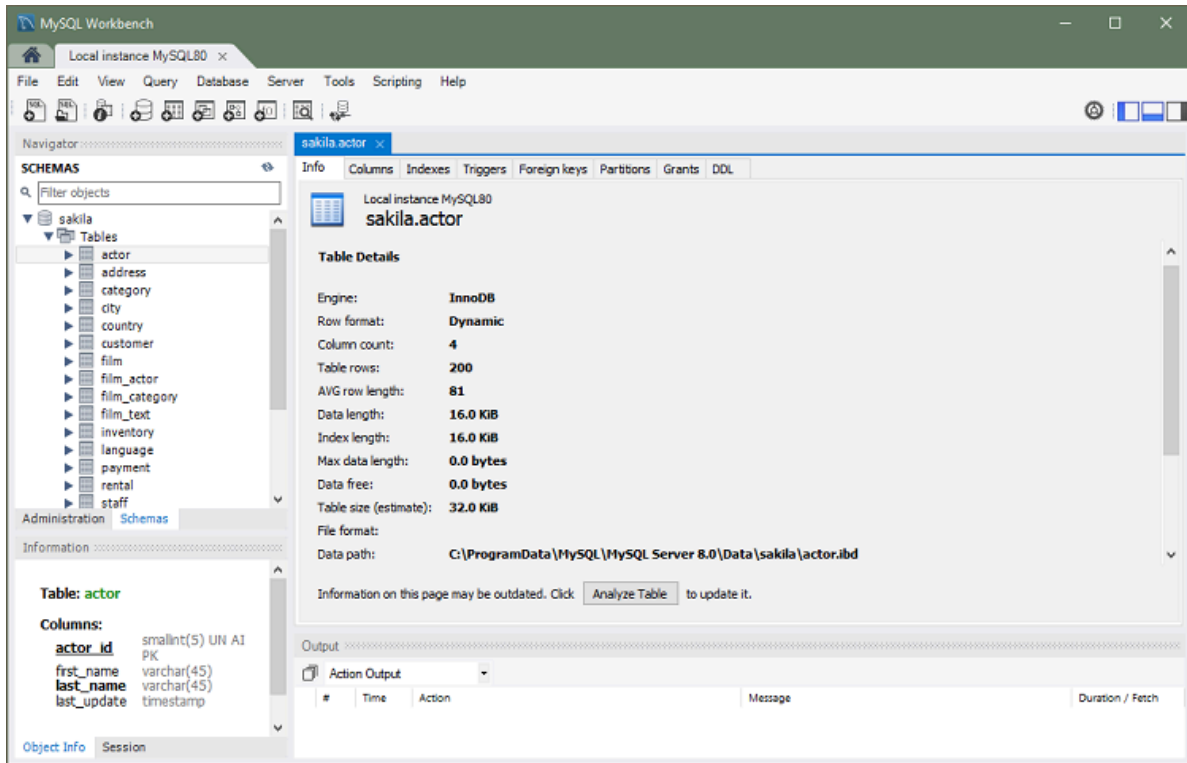


Table Inspector

You can use the Table Inspector to view table information, which is similar to the Schema Inspector. The Table Inspector includes an easy-to-use interface for analyzing and creating indexes for your tables. To open, right-click a table in the object browser of the Navigator pane and choose **Table Inspector** from the context menu.

The **Table Inspector** shows information related to the table. The next figure shows an example using the `sakila.actor` table.

Figure 8.25 Table Inspector: Info Tab



Chapter 9 Database Design and Modeling

Table of Contents

9.1 Modeling Interface	240
9.1.1 Model Editor	241
9.1.2 EER Diagram Editor	256
9.1.3 Creating Tables	265
9.1.4 Creating Foreign Key Relationships	267
9.1.5 Creating Views	270
9.1.6 Creating Routines and Routine Groups	272
9.1.7 Creating Layers	276
9.1.8 Creating Notes	278
9.1.9 Creating Text Objects	279
9.1.10 Creating Images	280
9.2 Additional Modeling Tools	281
9.2.1 Printing Diagrams	281
9.2.2 DBDoc Model Reporting	281
9.2.3 Schema Validation Plugins	285
9.3 Modeling Tutorials	287
9.3.1 Creating a Model	287
9.3.2 Basic Modeling	294
9.3.3 Importing a Data Definition SQL Script	296
9.3.4 Using the Default Schema	298
9.3.5 Documenting the sakila Database	301
9.4 Forward and Reverse Engineering	303
9.4.1 Forward Engineering	303
9.4.2 Reverse Engineering	313
9.5 Schema Synchronization and Comparison	323
9.5.1 Database Synchronization	323
9.5.2 Compare and Report Differences in Catalogs	329
9.6 Table Templates	331
9.7 Customizing DBDoc Model Reporting Templates	334
9.7.1 Supported Template Markers	337
9.7.2 Creating a Custom Template	341

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 EER diagrams and physical MySQL databases.

MySQL Workbench provides extensive capabilities for creating and manipulating database models, including these:

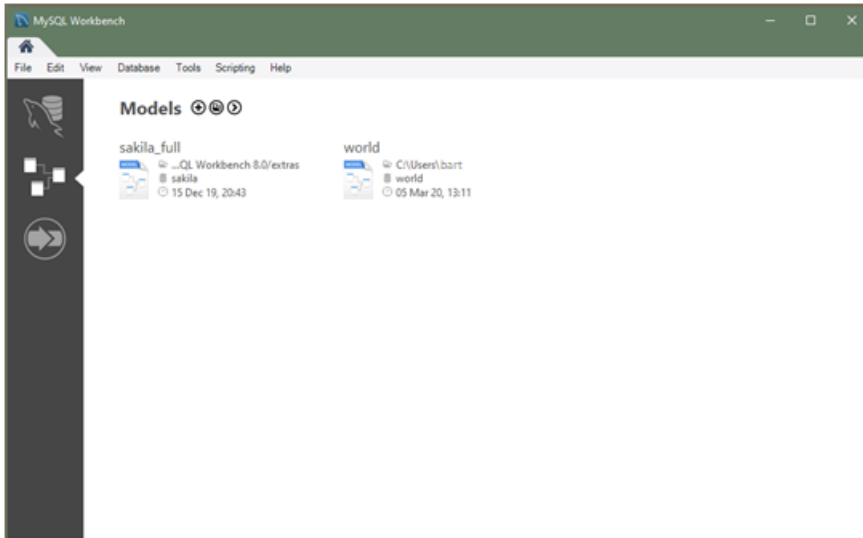
- Create and manipulate a model graphically.
- Reverse engineer a live database to a model.
- Forward engineer a model to a script or live database.
- Create and edit tables and insert data.

This is not an exhaustive list. The following sections describe these and additional data-modeling capabilities.

9.1 Modeling Interface

MySQL Workbench represents each active data model as an icon in the models view of the home screen tab. The following figure shows the `sakila_full` and `world` database models. Both models derive from MySQL database samples (see <https://dev.mysql.com/doc/index-other.html>), which you can download and use to explore the MySQL Workbench modeling interface.

Figure 9.1 Data Models on the Home Screen



The `sakila` database sample includes a model file (`sakila.mwb`) in the product package. After you set up the database using the instructions provided on the download page, MySQL Workbench adds the **sakila_full** icon to the models view automatically. For all of the other database samples, such as `world` or `employee data`, you must create the MySQL Workbench (`.mwb`) file first before you can view the database objects in a model or add an EER diagram for it.

To create a MySQL Workbench model file for MySQL database samples:


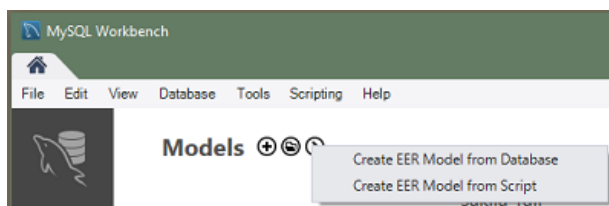
1. Download the MySQL database you intend to use as a model. Each database has a setup guide (see <https://dev.mysql.com/doc/index-other.html>).
2. Start MySQL Workbench and select the models view () from the side panel of the home screen tab.
3. Click the arrow icon and select **Create EER Model from Database** as shown in the figure that follows.

Figure 9.2 Create EER Model from Database



A wizard-like dialog presents the following steps: Connection Options, Connect to DBMS, Select Schemas, Retrieve Objects, Select Objects, Reverse Engineer, and Results. Click **Show Logs** to assist with the operations.

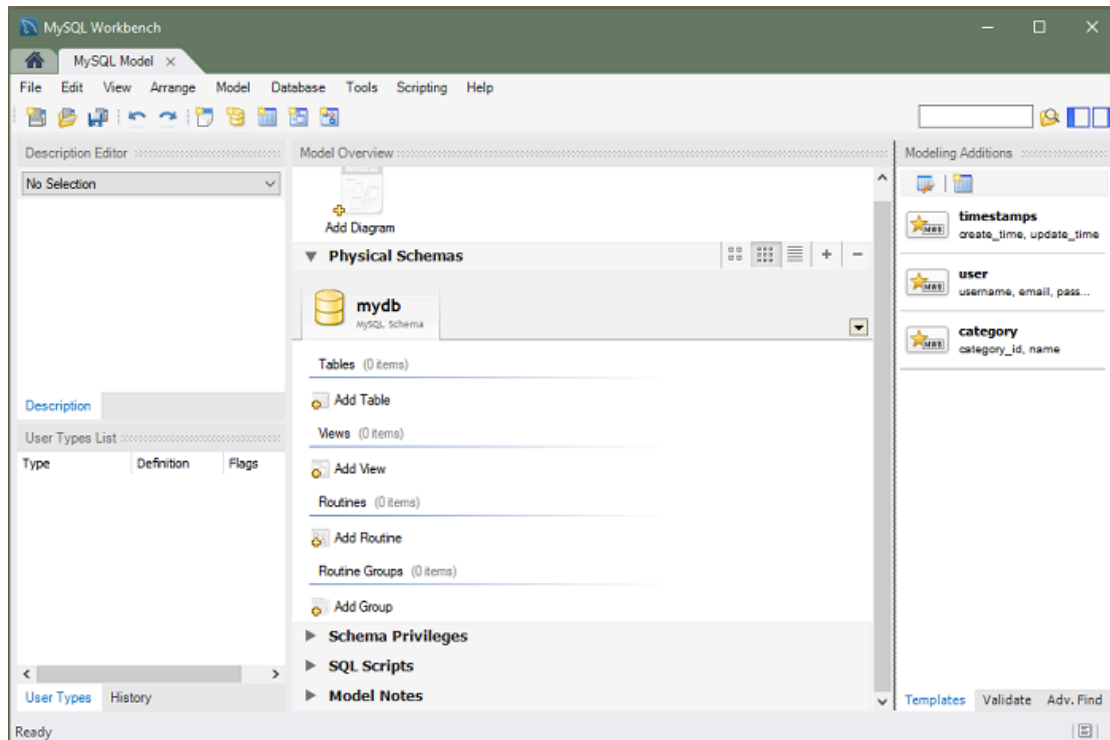
4. Modify the default values as needed, select a schema (and objects), and then click **Finish** to close the wizard. MySQL Workbench adds an icon for each model you create.

Modeling concepts and interface elements are described in the next sections.

9.1.1 Model Editor

When the Model Editor is executed from the home screen, MySQL Workbench displays the **MySQL Model** tab. The MySQL Model page has three main panels, as shown in the following figure.

Figure 9.3 The MySQL Model Page



The Description Editor and User Types List (with the **User Types** and **History** secondary tabs) are contained within the sidebar.

The **Model Overview** panel includes the following sections:

- Enhanced Entity-Relationship (EER) Diagrams
- Physical Schemas
- 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 context 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. The following submenu items apply to the active model tab, EER diagram tab, or both:

- **New Model:** Opens a new **MySQL Model** tab and closes the tabs of a previously opened model and diagram. The new MySQL Model tab includes an initial schema named `mydb` by default. You can rename or remove (and replace) the initial `mydb` schema. For additional information, see [Section 9.1.1.4, “The Physical Schemas Panel”](#).
- **Open Model:** Opens a file-selection window with the default file type set to MySQL Workbench (`.mwb` file extension). To display a list of recently opened `.mwb` files, select **Open Recent**.
- **Include Model:** Adds the database objects defined within an existing model file to the active MySQL model and to its diagram, if applicable. This operation also adds a separate diagram icon containing the included objects only to the active model.
- **Open Recent:** Displays the file path of each model file opened previously. Selecting a file from the list closes the tabs of an open model.
- **Close Tab:** If selected when the **MySQL Model** tab is shown, the action closes both the **MySQL Model** and **EER Diagram** tabs. However, if the **EER Diagram** tab is shown, this action closes the **EER Diagram** tab only. To reopen an **EER Diagram** tab, double-click the **EER Diagram** icon in the **Model Overview** section of the **MySQL Model** tab.
- **Save Model** or **Save Model As:** 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 `.mwb` extension.
- **Import:** Imports 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 initial schema, named `mydb` by default.
 - If the script creates a database, a new tab bearing the database name is added to the **Physical Schemas** 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”](#).

- **Export:** Generates 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”](#). Use the **Export** submenu items to export an EER diagram as a PNG, SVG, PDF, or Postscript file. For an example of a PNG file, see [Figure 9.35, “The sakila Database EER Diagram”](#).
- **Page Setup:** Enables you to set the paper size, orientation, and margins for printing purposes. This item is enabled only if the **EER Diagrams** tab is selected.

- **Print Preview:** Opens a print preview window for the active EER diagram. This item is enabled only if the **EER Diagrams** tab is selected. For more information, see [Section 9.2.1, “Printing Diagrams”](#).
- **Print:** Opens print window for the active EER diagram. This item is enabled only if the **EER Diagrams** tab is selected. For more information, see [Section 9.2.1, “Printing Diagrams”](#).
- **Print to File:** Prints the diagram (or diagrams) associated with the active model as a PDF or Postscript file. If your model has multiple diagrams, you can deselect one or more to exclude them from the file, but you must include at least one diagram in the file.
- **Document Properties:** Sets the following properties of your project:
 - **Name:** The model document 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.
- **Exit:** Prompts you to save the current changes and then closes MySQL Workbench.

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.

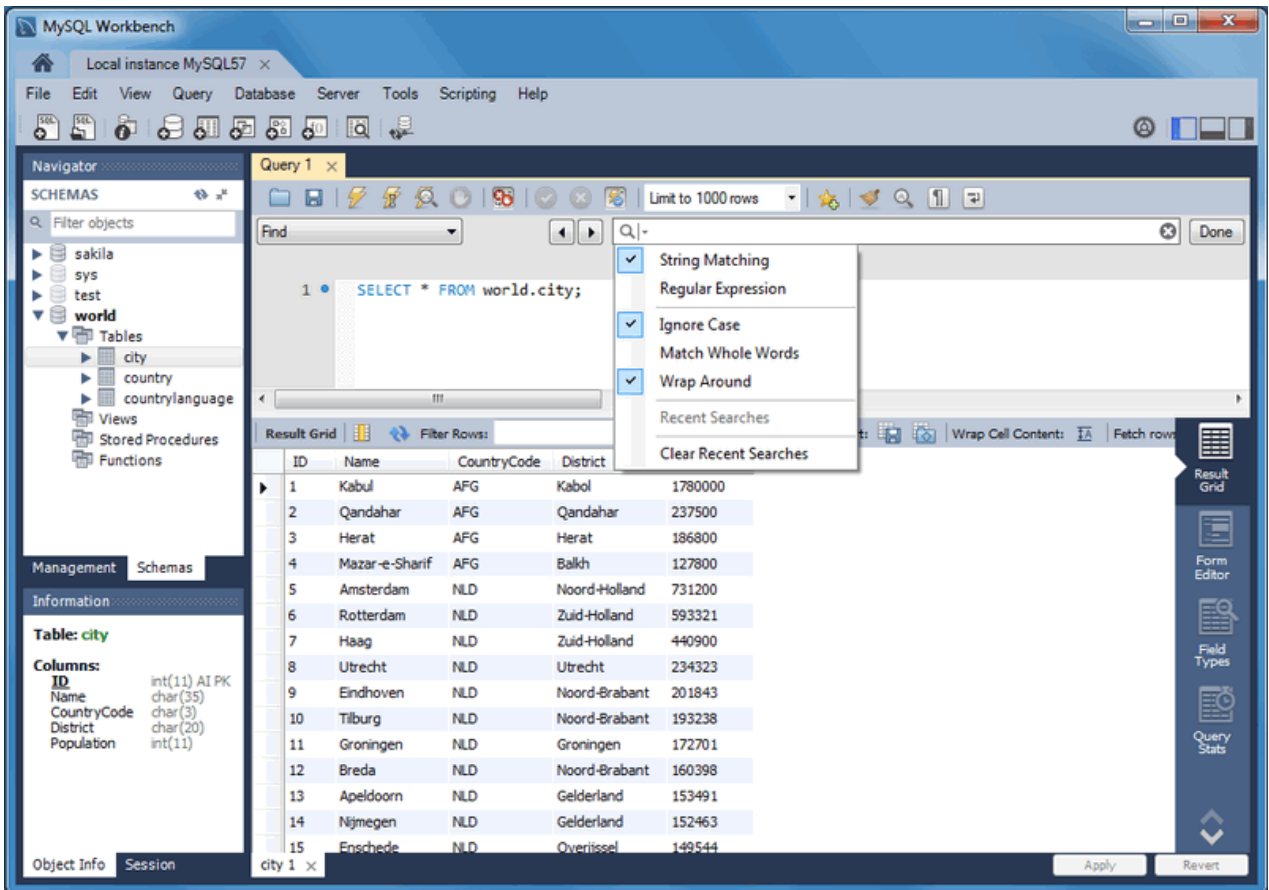
When multiple objects have been selected using one of these menu items, you can navigate between selected items by choosing the **Go to Next Selected** or **Go to previous Selected** menu item.

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 in the following figure.

Figure 9.4 The Find Panel with Find & Replace



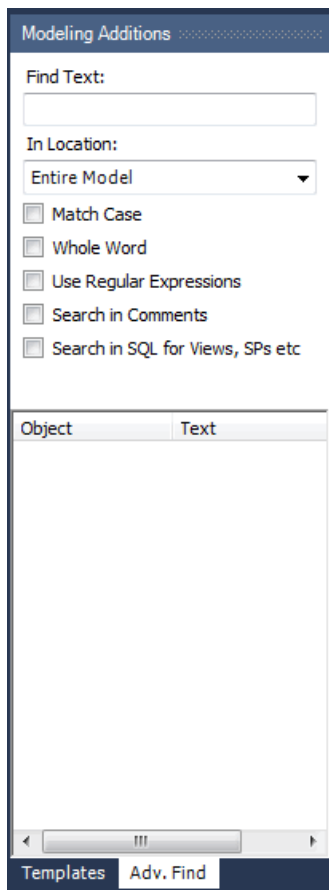
Find Options

The **Find** dialog options are described in the following list:

- **String Matching** (default) or **Regular Expression**: Search by matching a string, or a PCRE regular expression.
- **Ignore Case**: A search method that is not case-sensitive. 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, as indicated in the figure that follows.

Figure 9.5 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 Previous 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 9.1.2.3, “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 \(MySQL Workbench Commercial\)](#). MySQL Workbench Commercial only.

- **Validation:** Checks the validity of the model using ANSI standards. For information about this menu item, see [The Validation Submenus \(MySQL Workbench Commercial\)](#). MySQL Workbench Commercial only.
- **Validation (MySQL):** Checks the validity of the model using MySQL standards. For information about this menu item, see [The Validation Submenus \(MySQL Workbench Commercial\)](#). MySQL Workbench Commercial 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 (MySQL Workbench Commercial)

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 MySQL Workbench Commercial.

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 (MySQL Workbench Commercial)

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 MySQL Workbench Commercial.

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 **Object Notation** submenu apply to both a model and an EER diagram.

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 to both a model and an EER diagram.

The **Relationship Notation** submenu has these items:

- **Crow's Foot (IE):** The default modeling style. For an example, see [Figure 9.30, “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”](#). MySQL Workbench 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 8.1.11.2, “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 website 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 9.1.2.2, “The Vertical Toolbar”](#).

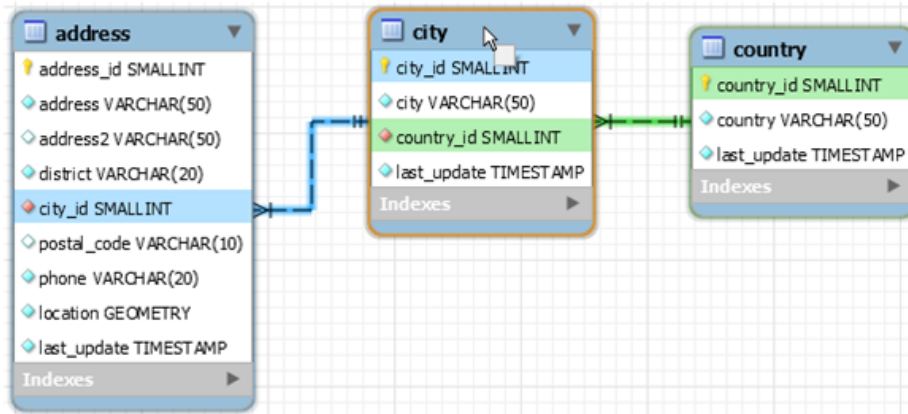
If the [Table](#) tool is active, schemata lists, engine types, and collations appear on the toolbar. The table properties can be modified using the Properties Editor.

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

Enhanced Entity-Relationship (EER) diagrams are an essential part of the modeling interface in MySQL Workbench. EER diagrams provide a visual representation of the relationships among the tables in your model. Revisions made with the Model Editor are shown in the associated diagram. Similarly, the changes you make within a diagram register seamlessly in the Model Editor. MySQL Workbench applies standard concepts and symbols to show table relationships, as indicated in the following figure. To get started, review the diagram editor's [color key](#) and [vertical toolbar](#).

Figure 9.6 Enhanced Entity-Relationship Diagram



From the Model Editor, click [Add Diagram](#) to create a new EER diagram. When you add a diagram, a new tab appears below the toolbar. Use this tab to navigate to the newly created EER diagram. For more information about working with EER diagrams in MySQL Workbench, see [Section 9.1.2, “EER Diagram Editor”](#).

9.1.1.4 The Physical Schemas Panel

The [Physical Schemas](#) panel of the [MySQL Model](#) page shows the active schemas and the objects that they contain. Expand and contract the [Physical Schemas](#) panel by clicking the arrow for the panel.

MySQL Workbench provides the details of each physical schema in a separate tab. A new model includes an initial schema named `mydb` by default. You can start working with this schema (add or edit table, view, and group objects) or create additional schemas in the model. When you save a new model, MySQL Workbench creates an icon in the models view of the home screen tab. Each model icon shows the model name, schema name, and the location of the model file (models use the `.mwb` extension). A model with multiple schemas shows the name of the last schema that you added to the model.

To add a new schema, click **+** on the [Physical Schemas](#) panel. To remove a schema, select its tab and click **-**. The other three buttons on the panel control how database object information is displayed, specifically:

- 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 Schemas](#) 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 Schemas](#) section. For information about adding objects to an EER diagram canvas, see [Section 9.1.2, “EER Diagram Editor”](#).

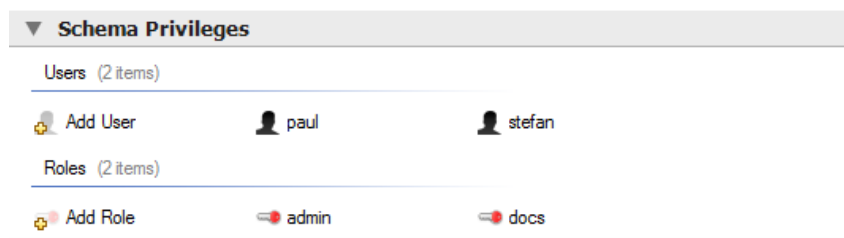
9.1.1.5 The Schema Privileges Panel

The [Schema Privileges](#) panel has the following sections, used to create users for your schemas and to define roles:

- Users
- Roles

The following figure displays the [Schema Privileges](#) section of the **MySQL Model** tab.

Figure 9.7 Roles and Privileges



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 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 **<**. Roles may be revoked by moving them in the opposite direction.

Right-clicking a user opens a context menu. The items in the menu function as described in [Adding Roles and Object Privileges](#).

Adding Roles and Object Privileges

To add a role, double-click the **Add Role** icon. This action creates a role with the default name `role1` in the **Roles** area of the **Schema Privileges** panel and opens the role editor. The editor has two tabs located at the bottom of the editor: **Role** and **Privileges**.

To open a context menu with the following actions, right-click an existing role:

- **Cut 'role_name'**: Cuts the role.
- **Copy 'role_name'**: Copies the role.

- **Paste:** Pastes a role from the clipboard. Dimmed if the clipboard is empty.
- **Edit 'role_name':** Opens the role editor.
- **Edit 'role_name' in New Tab:** Currently not implemented.
- **Copy SQL to Clipboard:** Currently not implemented.
- **Delete 'role_name':** Removes the role.
- **Remove Figure 'role_name':** Currently not implemented.

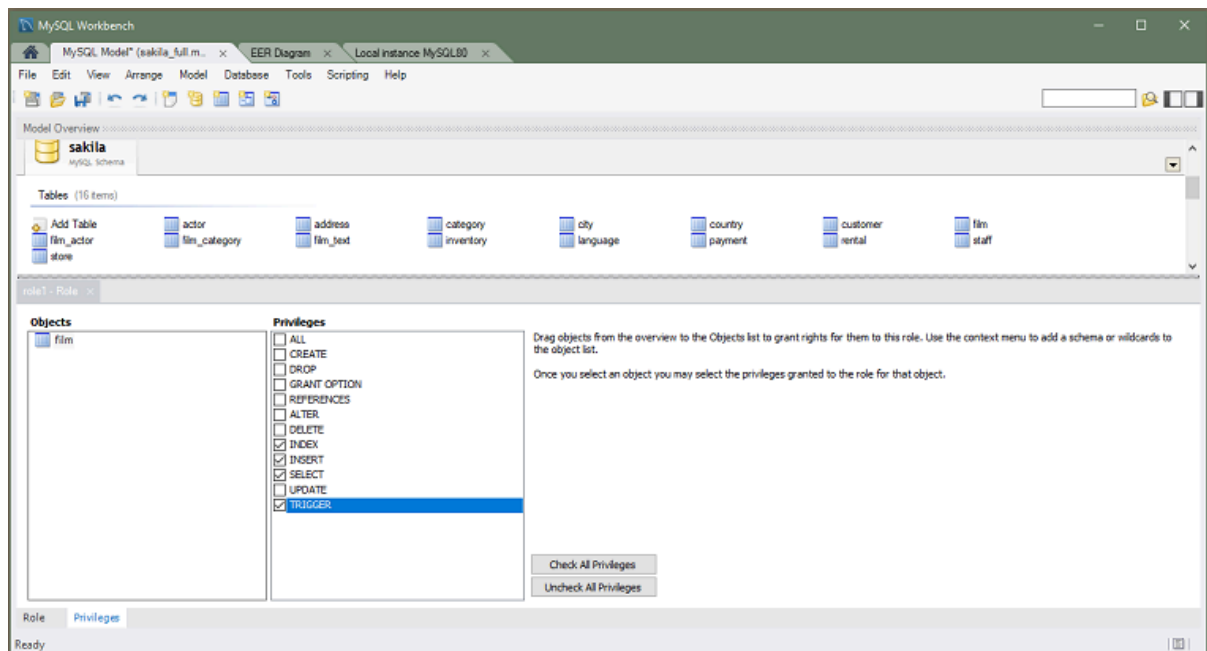
Role tab. The **Role** tab of the role editor includes the **Name** field with a default value consisting of `role + x` (`x` is an incremental number starting at 1). Each role also appears in the **Roles** group, which represents the hierarchical relationship among the defined roles. The **Parent** drop-down list enables you to share the privileges of an existing role with descendent roles.

To rename a role, double-click the role name to reopen the role editor. Then edit the text in the **Name** field.

Privileges tab. A newly created role might include the privileges of a parent role. However, if no parent role was provided, the **Objects** and **Privileges** groups are empty. To add an object and define privileges for it, drag the object icon from the **Schema Privileges** panel to the **Objects** group and then select the object to show a list of valid privileges. Grant an individual privilege by selecting the check box for it or click **Check all Privileges** to select all of the rights.

The following figure shows the `film` table object from the `sakila` schema with `INDEX`, `INSERT`, `SELECT`, and `TRIGGER` selected for the `role1` role.

Figure 9.8 Role Editor



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. The following figure shows the options available to customize the placement of attached SQL Scripts.

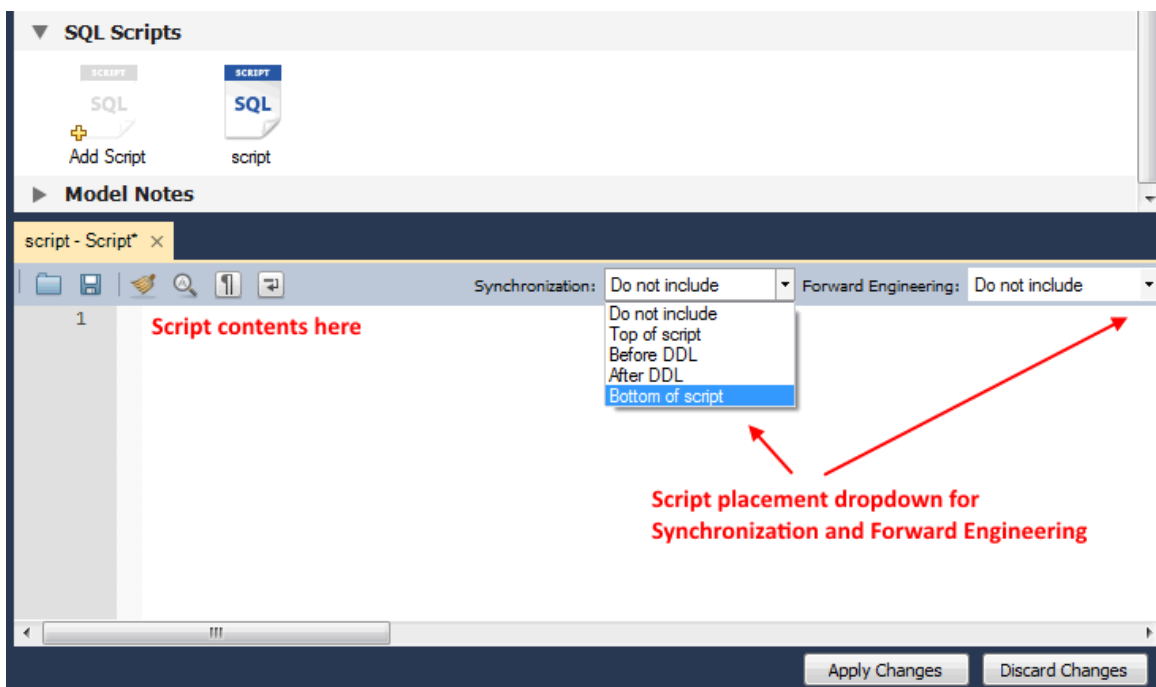
If you created your project from an SQL script and plan to create an **ALTER** script, you may want to add the original script here, because it will be needed to create an **ALTER** script. For more information, see [Altering a Schema](#).



Note

The ability to use the attachments when performing forward engineering and synchronization was added in MySQL Workbench 6.2.0.

Figure 9.9 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.

9.1.2 EER Diagram Editor

Enhanced Entity-Relationship (EER) diagrams are created by double-clicking [Add Diagram](#) in the Model Editor. You may create any number of EER diagrams just as you may create any number of physical schemas (databases). Each EER diagram opens in 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 the canvas.



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 Color Key for EER Diagrams

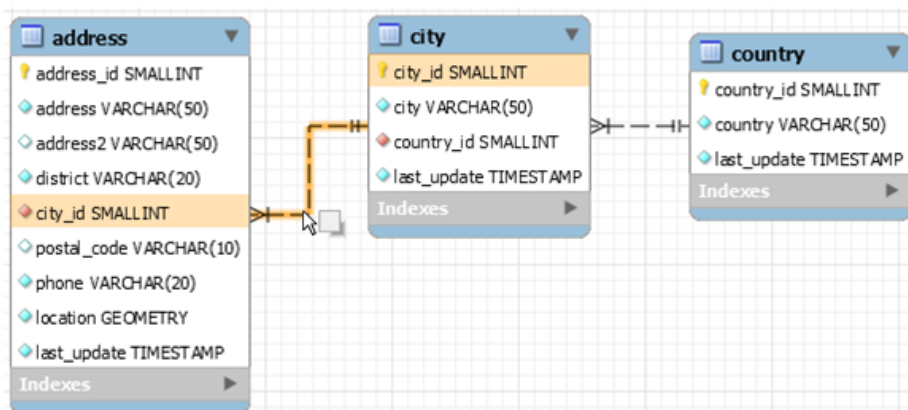
Color connection lines within EER diagrams provide quick visual information about how the tables in your model relate. Existing connection lines are highlighted when you use a pointer device to hover on different database objects. The following table defines the meaning of each color and indicates how you can open the underlying table information from your diagram.

Table 9.1 MySQL Workbench Color Key for EER Diagrams

Color	Hover On	Meaning	Related Action
Golden yellow	Connection line	Highlights the fields represented by the connection between two tables (see Figure 9.10, “Golden Yellow Highlight”).	Double-click to open the Relationship tab.
Green	Table name	Shows all of the outgoing relationships (foreign keys) defined on the table in focus, which reference a different table (see Figure 9.11, “Green and Blue Highlight”).	Double-click to open the Table tab.
Blue	Table name	Shows all of the incoming relationships (foreign keys) defined in another table, which end on the table in focus (see Figure 9.11, “Green and Blue Highlight”).	Double-click to open the Table tab.

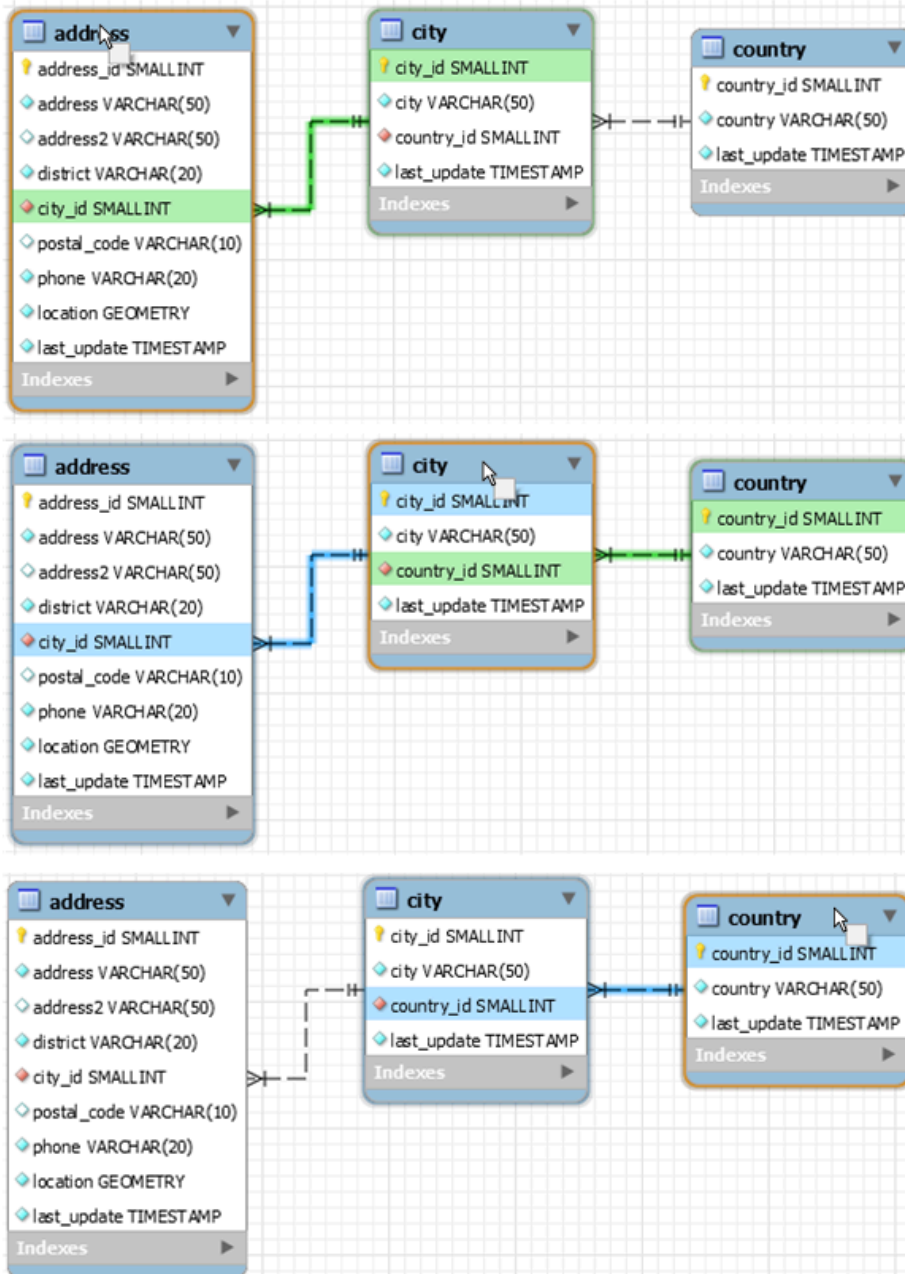
Hover on a connection between tables to highlight the line and the related fields. In the following figure, the connection between `address` and `city` tables is highlighted, together with the `address.city_id` and `city.city_id` fields. The small box near the pointer indicates that you can open a tab containing relationship information.

Figure 9.10 Golden Yellow Highlight



The presence of green and blue highlighting is determined by the table in focus. Line color changes dynamically when you move the focus from one table to another. As the next figure shows, hovering on the `address`, `city`, and `country` table names sequentially causes the connection line (or lines) and related fields to change color. For the meaning of each color, see [Table 9.1, “MySQL Workbench Color Key for EER Diagrams”](#). The small box near the pointer indicates that you can open a tab containing table information.

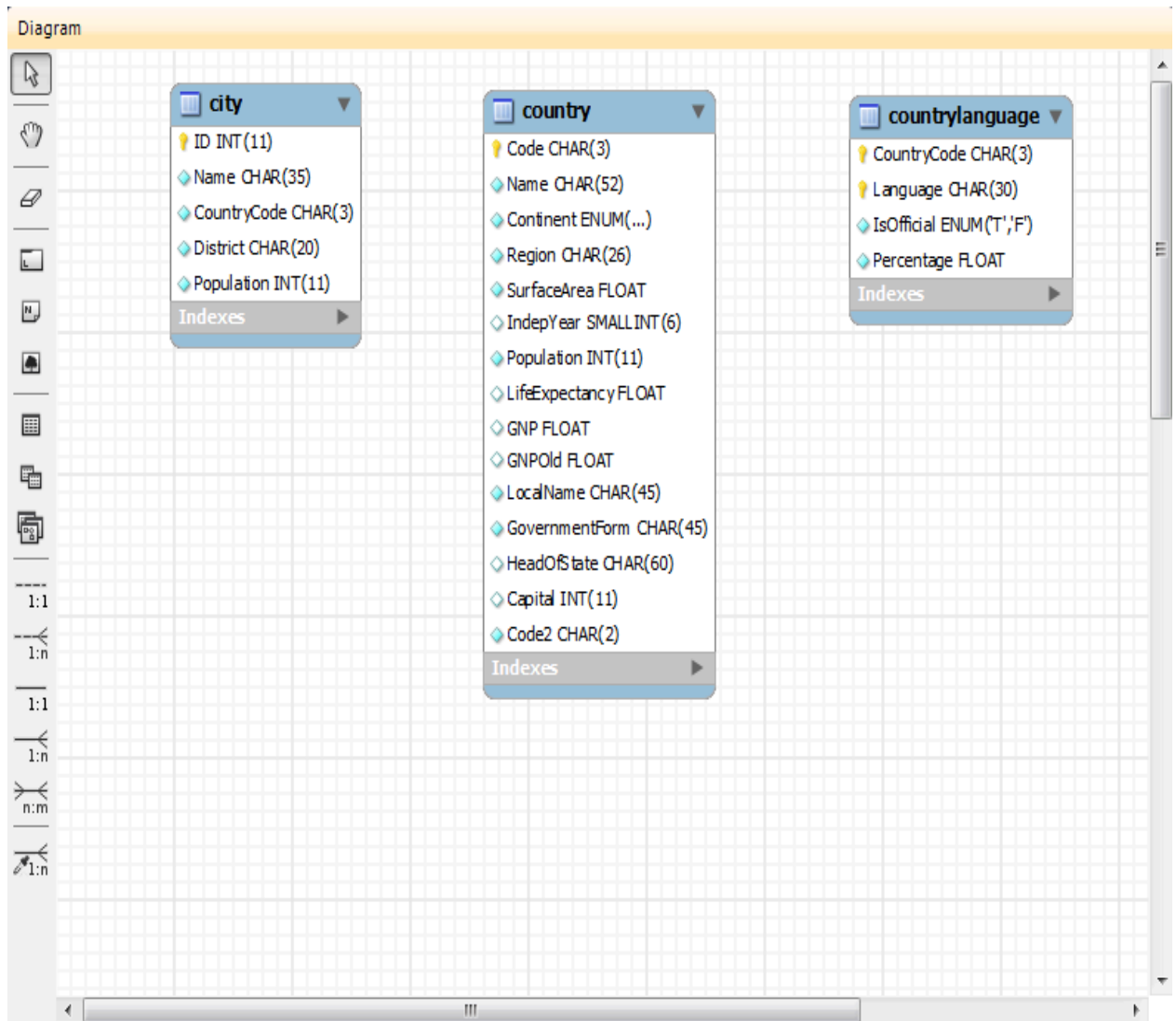
Figure 9.11 Green and Blue Highlight



9.1.2.2 The Vertical Toolbar

The vertical toolbar appears on the left sidebar (see the following figure) when an EER diagram tab is selected. The tools on this toolbar assist in creating EER diagrams.

Figure 9.12 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 9.1.2.3, “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 8.1.10, “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.

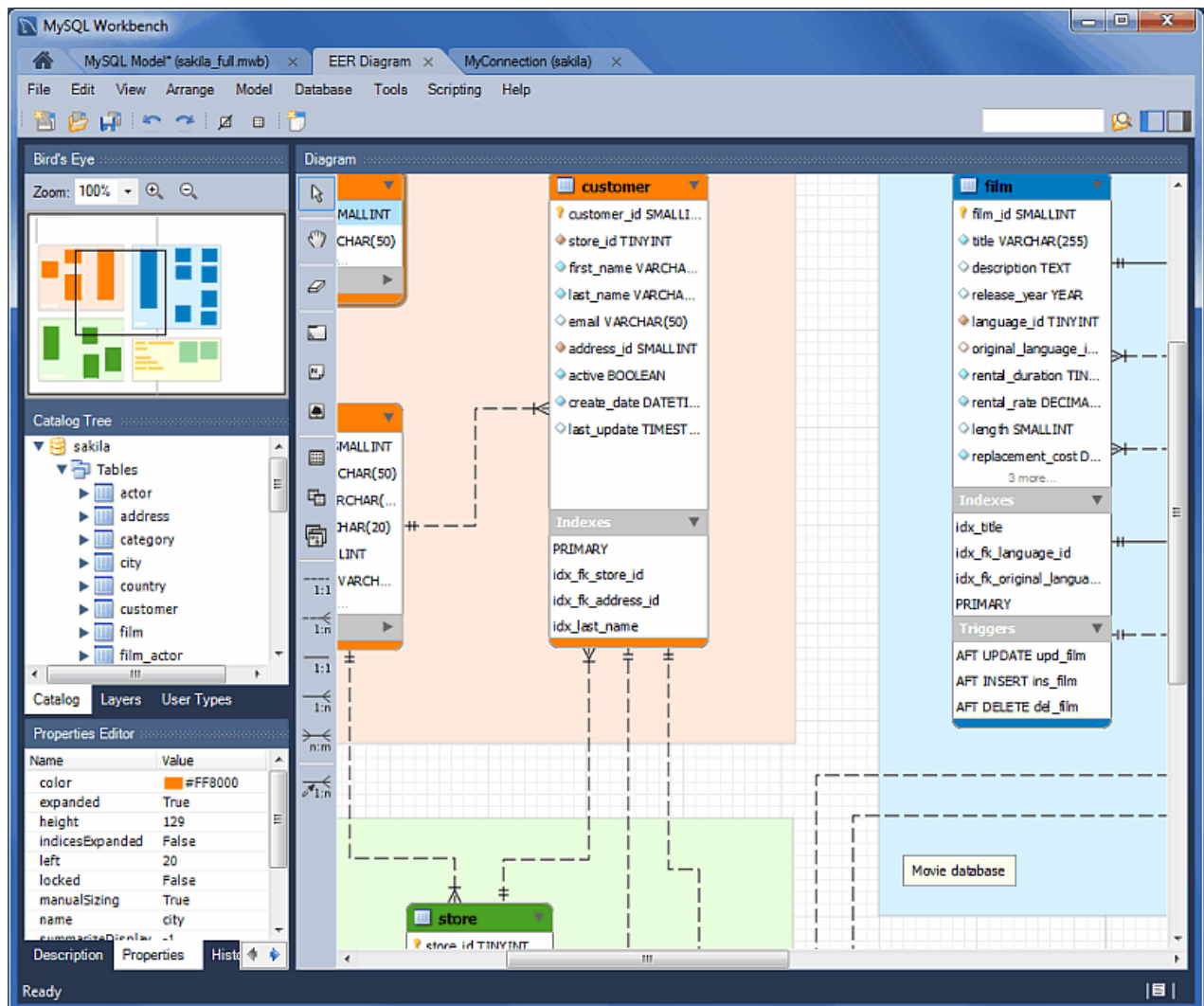
For more information about relationships, see [Section 9.1.4, “Creating Foreign Key Relationships”](#).

9.1.2.3 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.

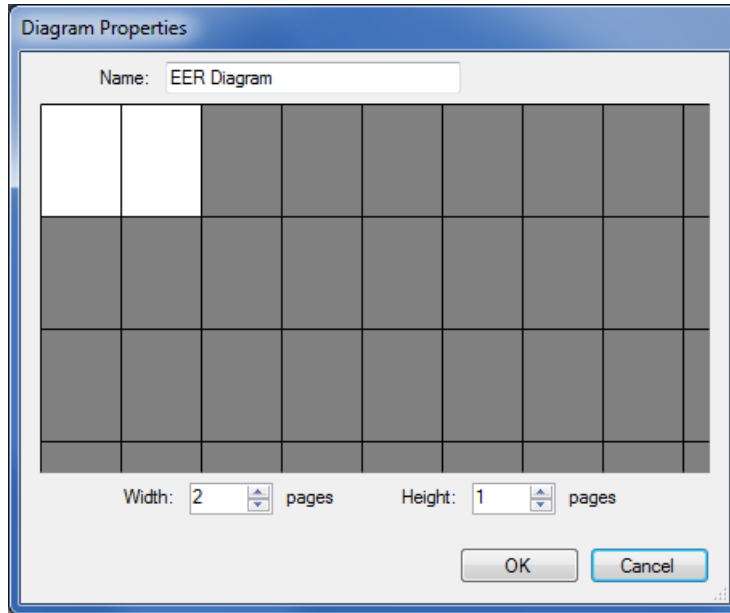
The Model Navigator shows the total area of an EER diagram (see the figure that follows). A black rectangular outline indicates the view port onto the visible area of the canvas. To change the view port of an EER diagram, left-click this black outline and drag it to the desired location. You can zoom 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 9.13 The Model Navigator: Example



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. The following figure shows an example of the Diagram Properties dialog.

Figure 9.14 The Model Navigator Palette



9.1.2.4 The Catalog Tree Palette

The [Catalog Tree](#) palette shows all the schemas that are present in the [Physical Schemas](#) 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.

9.1.2.5 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`.

9.1.2.6 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"](#).

9.1.3 Creating Tables

9.1.3.1 Adding Tables to the Physical Schemas

Double-clicking the **Add table** icon in the **Physical Schemas** section of the **Model Overview** tab 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 8.1.10, “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_name'**: Changes the docked table editor to the selected table.
- **Edit 'table_name' 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.

- **Remove Figure 'table_name'**: If applicable, remove the figure associated with the table.

If the table editor is not open, the **Edit 'table_name'** item opens it. If it is already open, the selected table replaces the previous one. **Edit 'table_name' in New Tab** opens an additional table editor tab.

Any tables added to the **Physical Schemas** section also show up in the **Catalog Tree** palette within the **EER Diagram** tab. 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

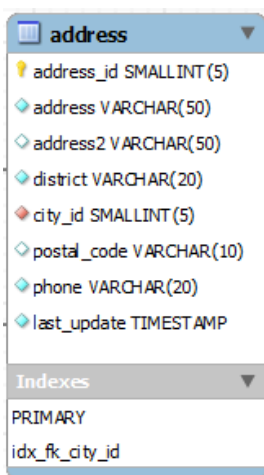
Tables can also be added to an EER Diagram using the `table` tool on the vertical toolbar. Make sure that the **EER Diagram** tab is selected, then right-click the table icon on the vertical toolbar. The table icon is the rectangular tabular icon.

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 schemas 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 action 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 9.15 A Table on an EER Diagram



As shown in the preceding figure, 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 Schemas”](#). 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 Constraint Differences](#).

MySQL Workbench enables you to add a foreign key from within the table editor or from within an EER diagram by using the relationship tools on the vertical toolbar. This section describes how to add a foreign key using the foreign key tools. To add a foreign key using the table editor, see [Section 8.1.10.4, “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.

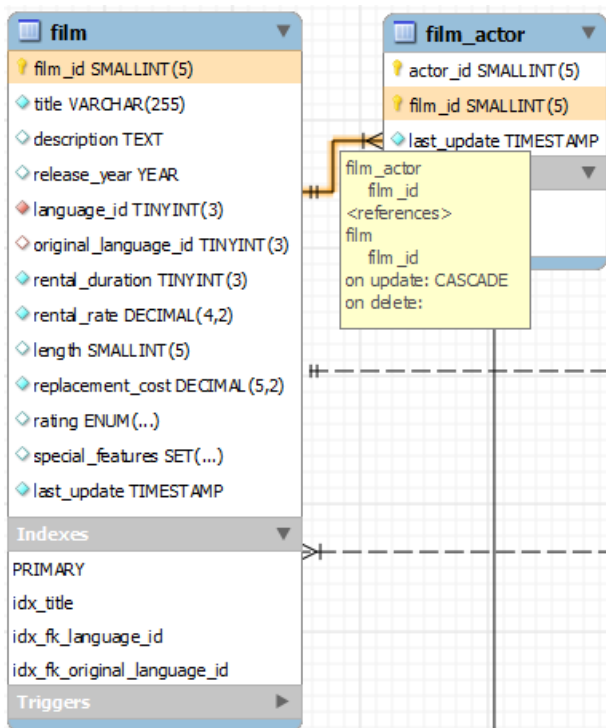
When the many-to-many tool is active, double-clicking a table creates an associative table with a many-to-many relationship. For this tool to function there must be a primary key defined in the initial table.

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.

Pausing your mouse pointer 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 pausing over a relationship for a second, a yellow box is displayed that provides additional information.

Figure 9.16 The Relationship Connector



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 9.16, "The Relationship Connector"](#) is the default, crow's foot. You can change this if you are using a Commercial Edition 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**.

Relationship Tab

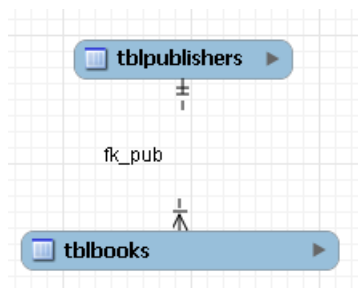
In the **Relationship** tab, you can set the caption of a relationship using the **Caption** field. This name displays on the canvas and is also the name used for the constraint itself. The default value for this name

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 shown in the following figure.

Figure 9.17 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](#).

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 Schemas` section of the `MySQL Model` page or from the EER Diagram.

9.1.5.1 Adding Views to the Physical Schemas

To add a view, double-clicking the `Add View` icon in the `Physical Schemas` 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 schemas. **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 Schemas](#) 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 schemas 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 Schemas”](#). 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 Schemas](#) 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 9.1.2.6, “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 and Object Privileges](#).

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 9.1.2.6, "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.

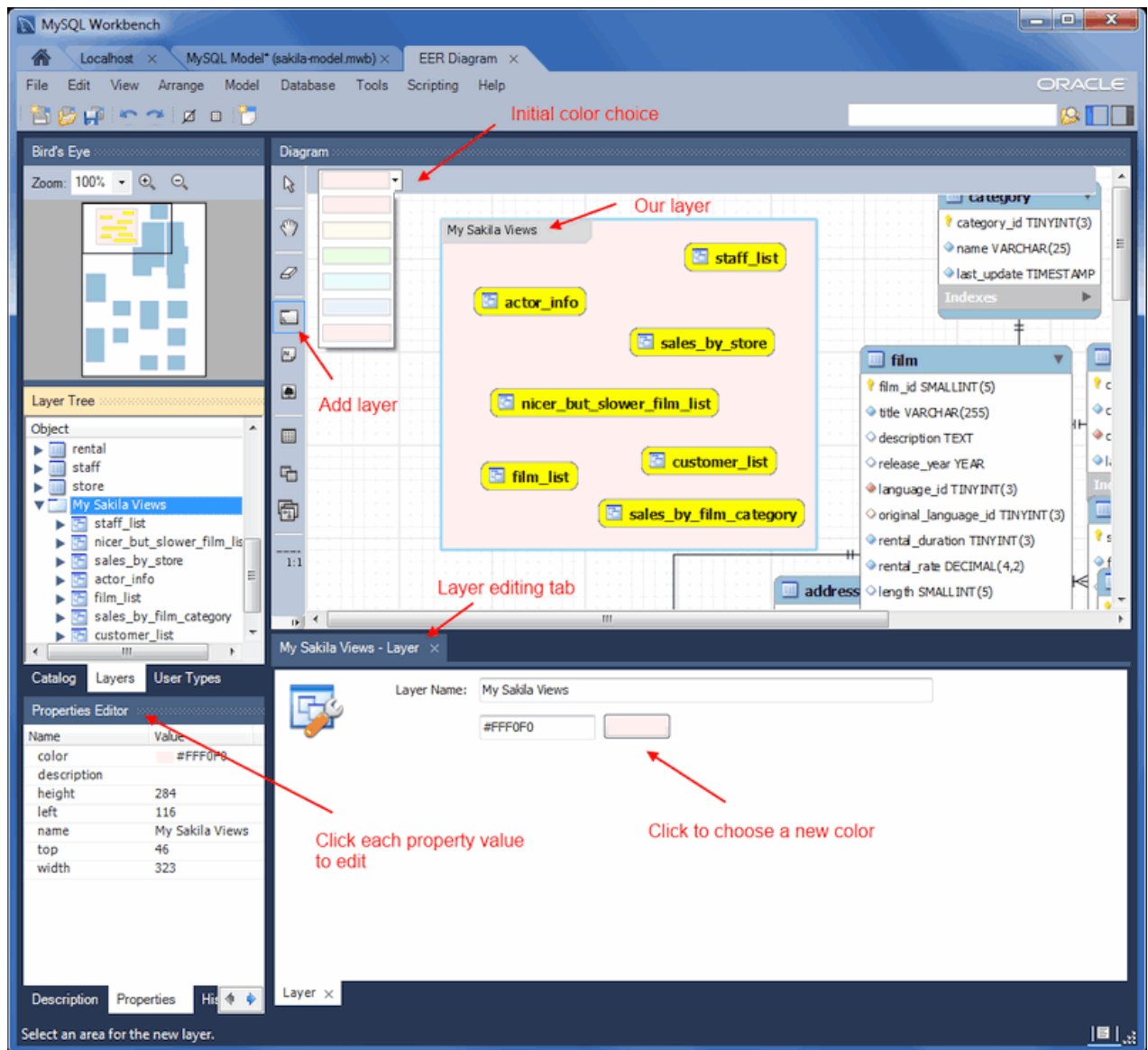
Clicking the mouse on this icon changes the mouse pointer to a layer pointer. You can also change the mouse pointer to a layer pointer by pressing the **L** key.

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 figure shows a layer named "My Sakila Views" with several views:

Figure 9.18 The Layer Object



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 9.1.2.6, “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 9.1.2.6, “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 9.1.2.6, “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 \(MySQL Workbench Commercial\)](#).

9.2.1.1 Printing Options

The printing menu items are 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 multipage 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

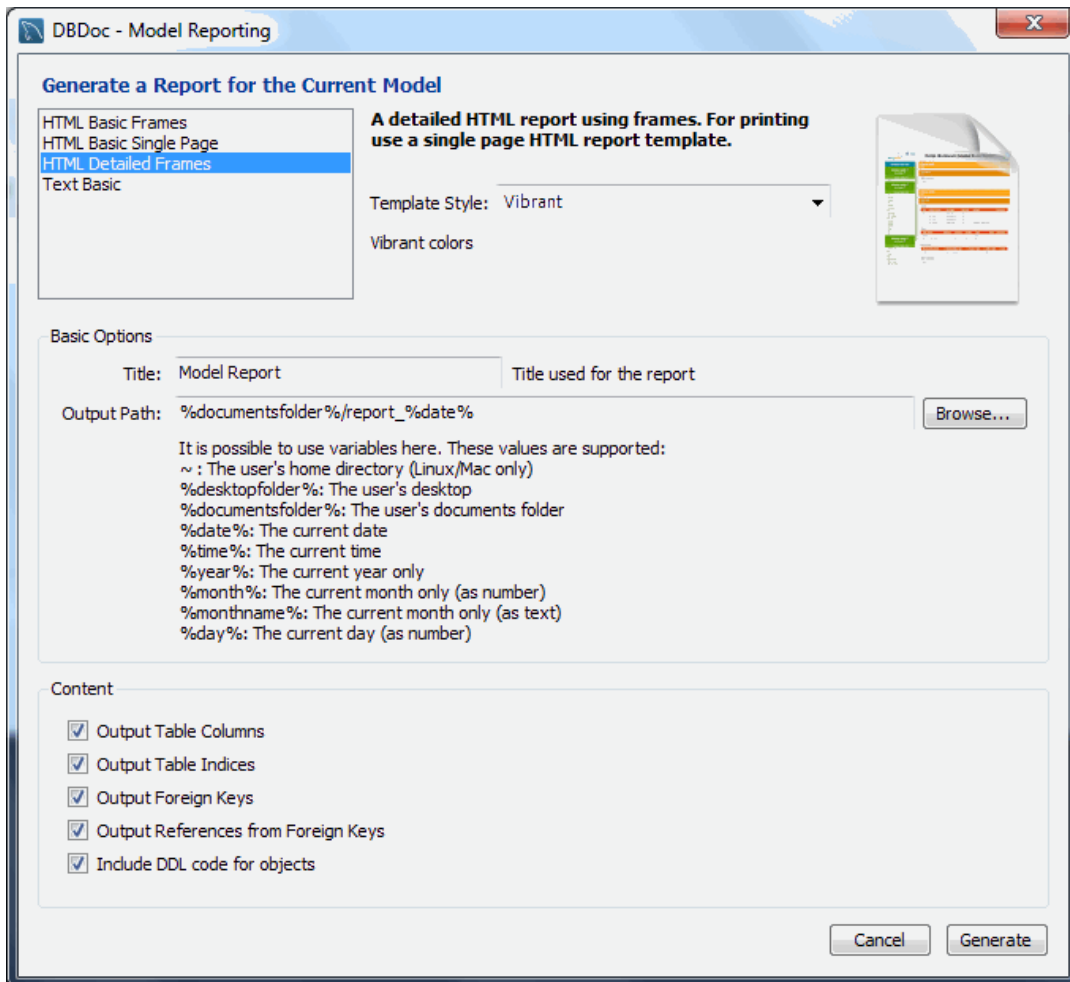
The DBDoc - Model Reporting dialog is found by opening a model, selecting **Model** from the menu, and then **DBDoc - Model Reporting**.

**Note**

This functionality is only available in the MySQL Workbench Commercial Editions.

Use the DBDoc- Model Reporting dialog to set the options for creating documentation of your database models. For example, the following figure shows the dialog with [HTML Detailed Frames](#) selected and the template style set to [Vibrant](#).

Figure 9.19 The DBDoc Model Reporting Options



You can choose from four available templates to generate a report:

- **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 home directory of the user. Available on Linux and macOS versions only.
- **%desktopfolder%**: The desktop of the user.

- `%documentsfolder%`: The Documents folders of the user. The following table shows typical values for common platforms.

Platform	Typical Default Documents Folder
Windows	<code>users\user_name\My Documents</code>
Linux	<code>Documents</code>
macOS	<code>rs/user_name/Documents</code>

- `%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 selected:

- **Output Table Columns**: Display all the columns.
- **Output Table Indices**: Display all the indexes.
- **Output Foreign Keys**: Display all the foreign keys.
- **Output References from Foreign Keys**: 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:

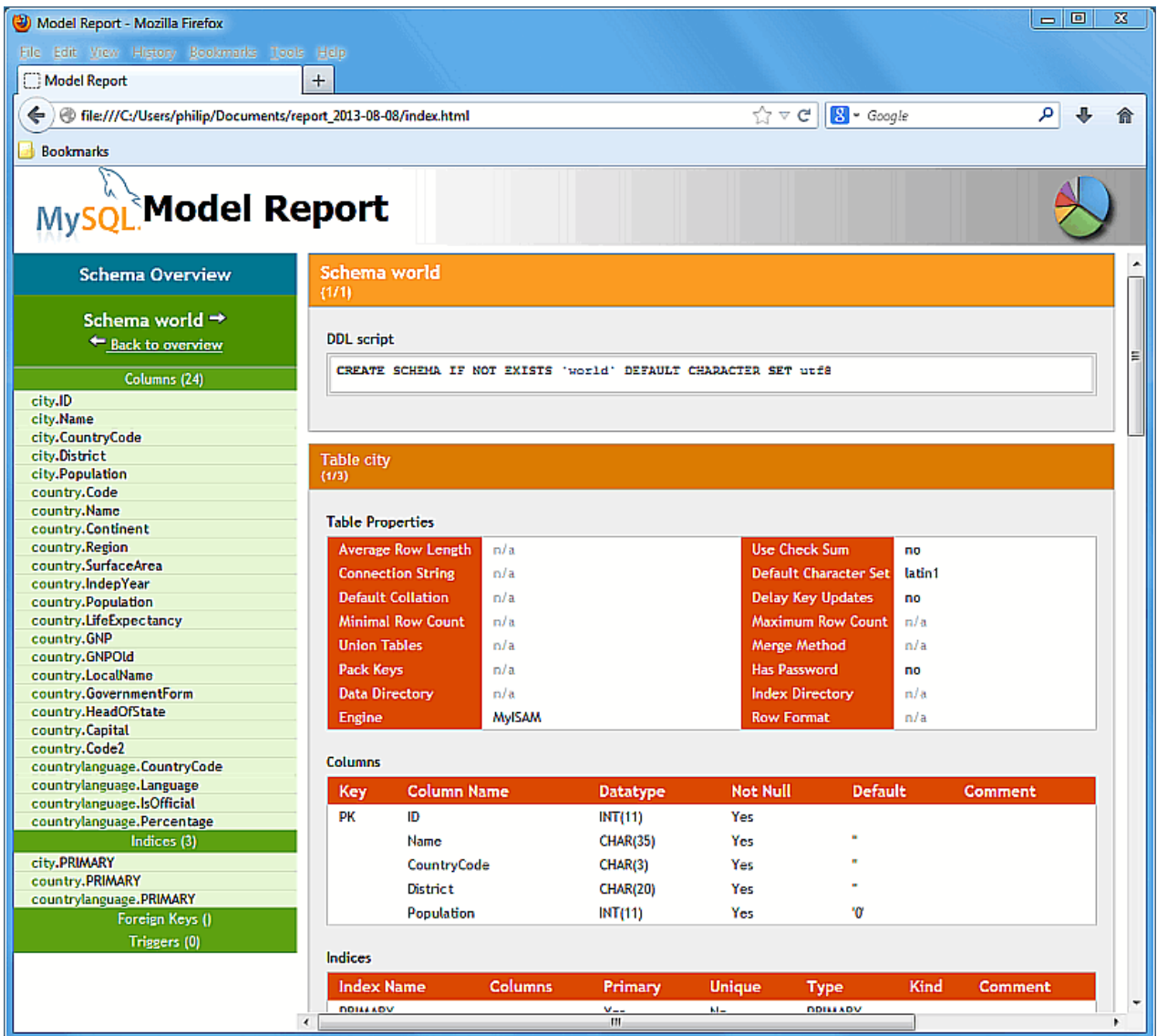
- `basic.css`: The style sheet for the `overview.html` page. This is used if the **vibrant** style is chosen.
- `coated.css`: The CSS file used if the **Coated** style option was chosen.
- `index.html`: The main page.
- `overview.html`: Overview information for the report such as report title, project 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 figure shows an output example of the **HTML Detailed Frames** report for the `world` schema.

Figure 9.20 The DBDoc Model Report



For more information about creating custom templates, see [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.



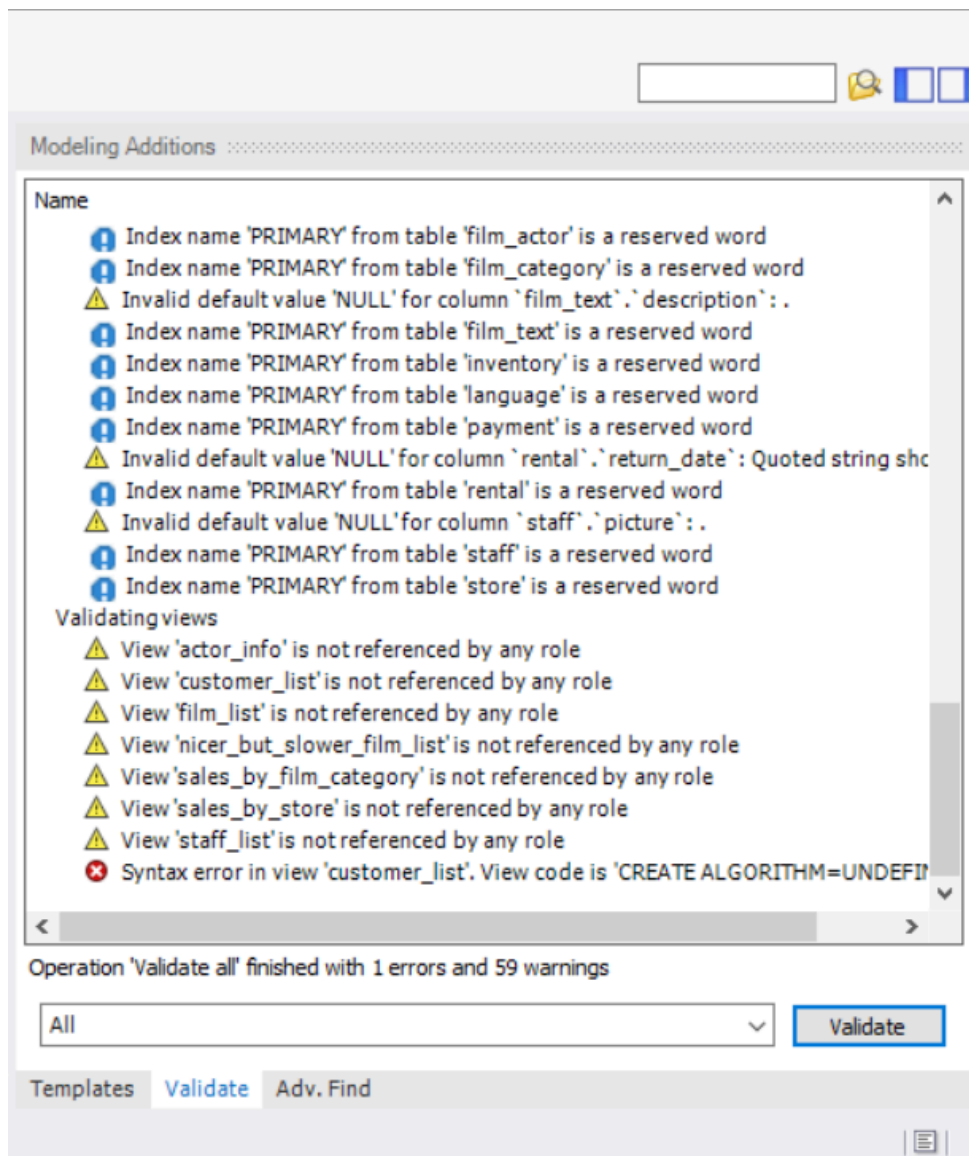
Note

This functionality is only available in MySQL Workbench Commercial.

The validation (MySQL) plugins are accessed from the **Model** menu within an open [MySQL Model](#) or [EER Diagram](#) tab. Beneath the menu item are a number of specific validation tests. Running any one of these

tests (or **Validate All**) displays validation output in the **Modeling Additions** area of the window (click from the toolbar to open or close the panel). Alternatively, you can run the same validation tests from the **Validate** tab within the panel as the following figure shows.

Figure 9.21 Modeling: Validate Tab



Information, warning, and error messages include an icon to show the severity of each issue visually. In addition, the output is organized by category: Validating routine groups, Validating routines, Validating tables, and Validating views. Changes made to **Model Options** (see **Model**) may alter the output of the individual validation tests. To copy one or more messages, highlight the output and select **Copy** from the context menu.

The following list names the validation type and gives examples of specific violations:

- **Consistency Validation**
 - Use of the same column with columns of differing data types
- **Duplicated Identifiers Validation**
 - Duplicate object names
 - Duplicate role or user names
 - Duplicate index or routine names
- **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
- **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
- **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
- **Syntax Violation**
 - A routine, trigger, or view with incorrect SQL syntax
 - A reserved keyword used as an identifier

- Use of an invalid character
- **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

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

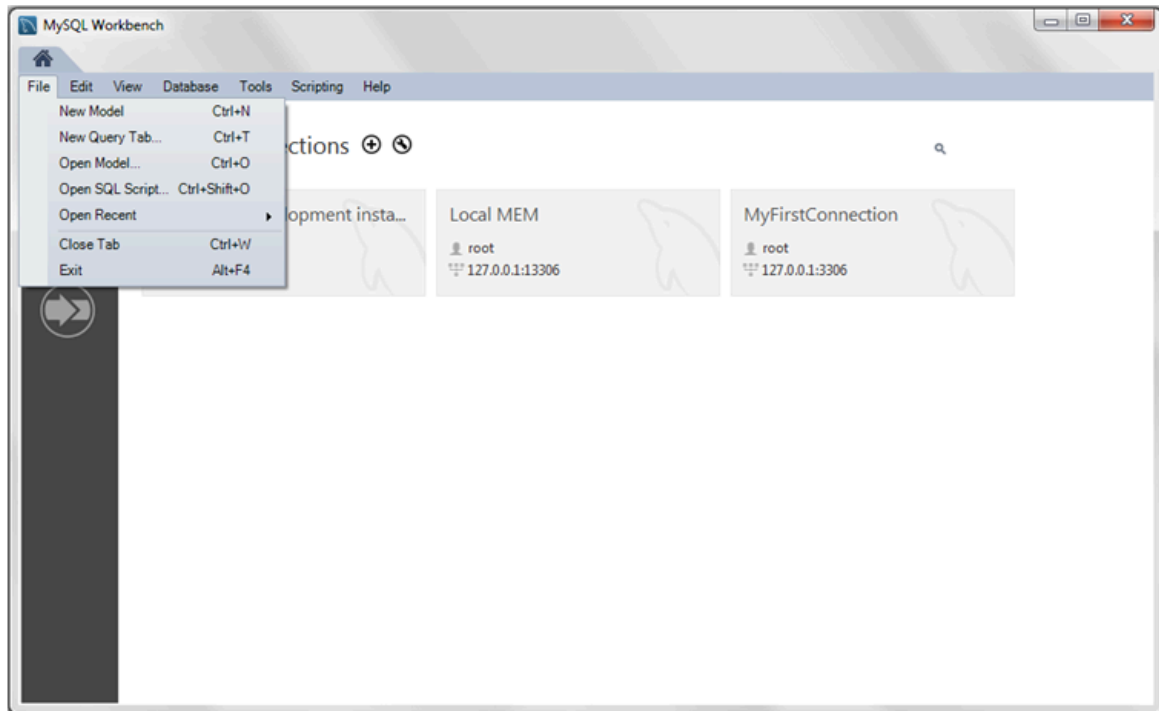
This tutorial describes how to create a new database model and how to forward-engineer a model to a live MySQL server.



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 models view from the sidebar and then click (+) next to **Models**. Alternatively, you can click **File** and then **New Model** from the menu (shown in the figure that follows).

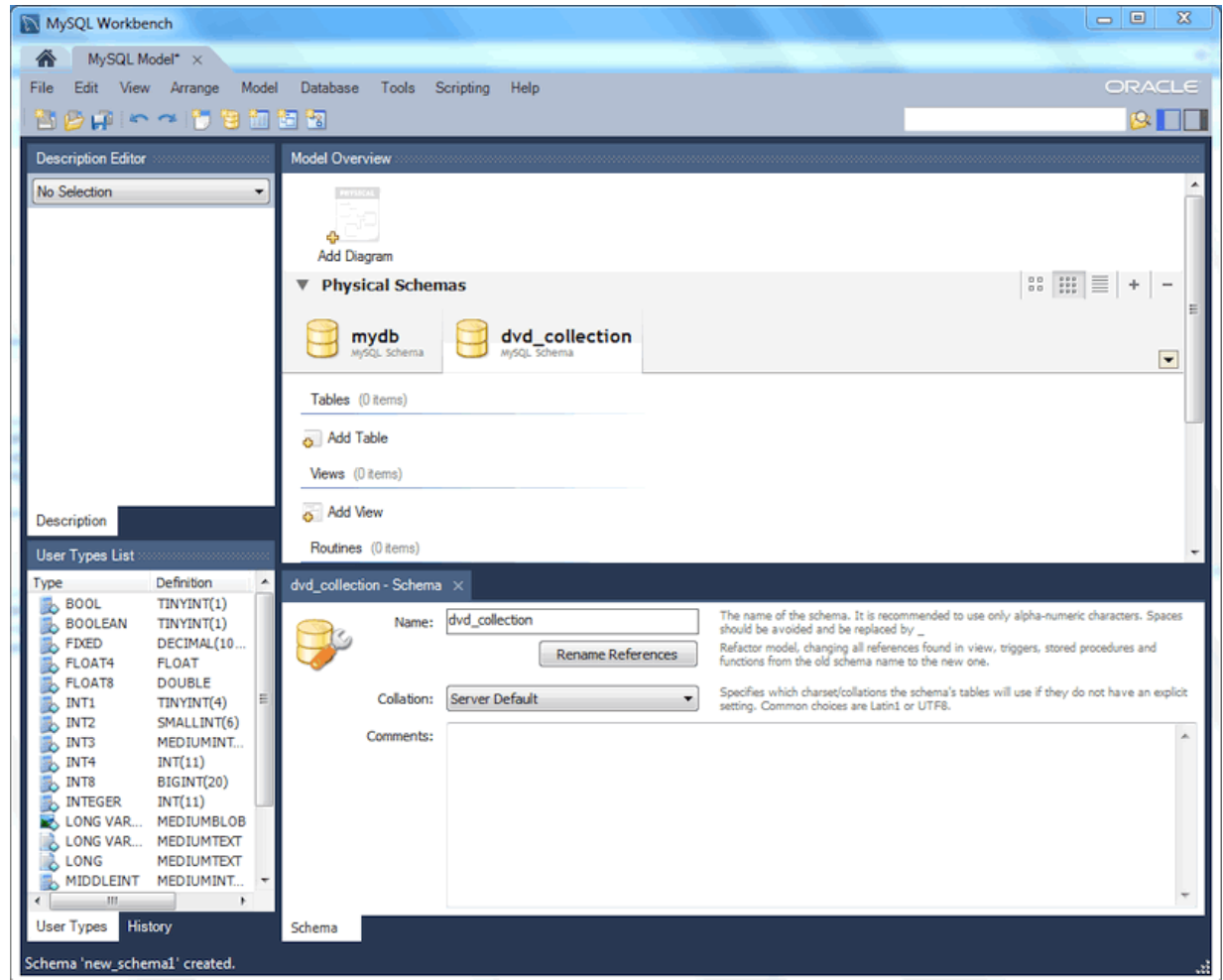
Figure 9.22 Getting Started Tutorial - Home Screen

A model can contain multiple schemas. Note that when you create a new model, it contains the `mydb` schema by default. You can change the name of this default schema as needed or you can delete it.

2. Click the **+** button on the right side of the **Physical Schemas** toolbar to add a new schema. The default schema name is `new_schema1`, which you can now change to `dvd_collection` by modifying its

Name field. Confirm this change in the **Physical Schemas** panel shown in the next figure. Now you are ready to add a table.

Figure 9.23 Getting Started Tutorial - New Schema

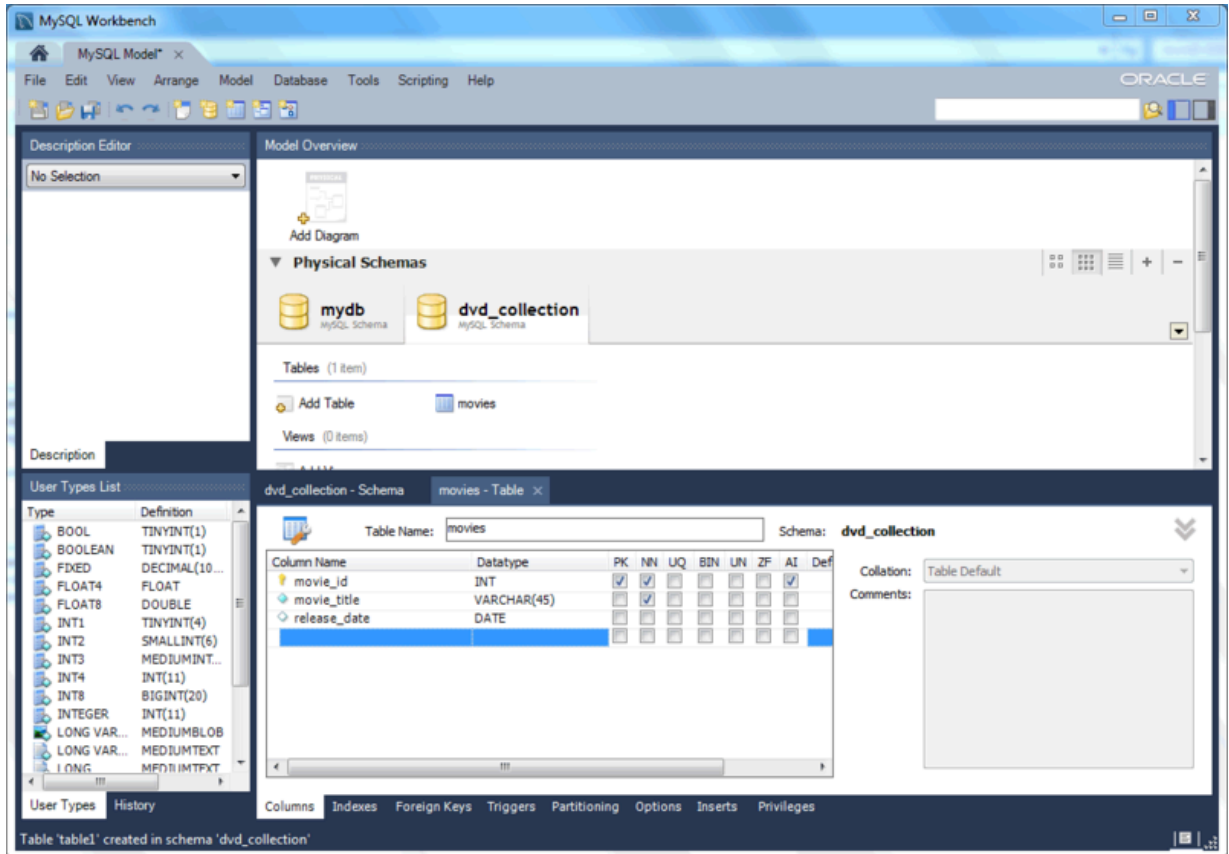


3. Double-click **Add Table** in the **Physical Schemas** section.
4. This automatically loads the table editor with the default table name `table1`. Edit the **Table Name** field to change the table name from `table1` to `movies`.
5. Next, add columns to your table. Double-click a **Column Name** cell 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 the two additional columns described in the following table. The figure that appears after the table shows all three columns in the `movies` table.

Column Name	Data Type	Column Properties
<code>movie_title</code>	<code>VARCHAR(45)</code>	NN

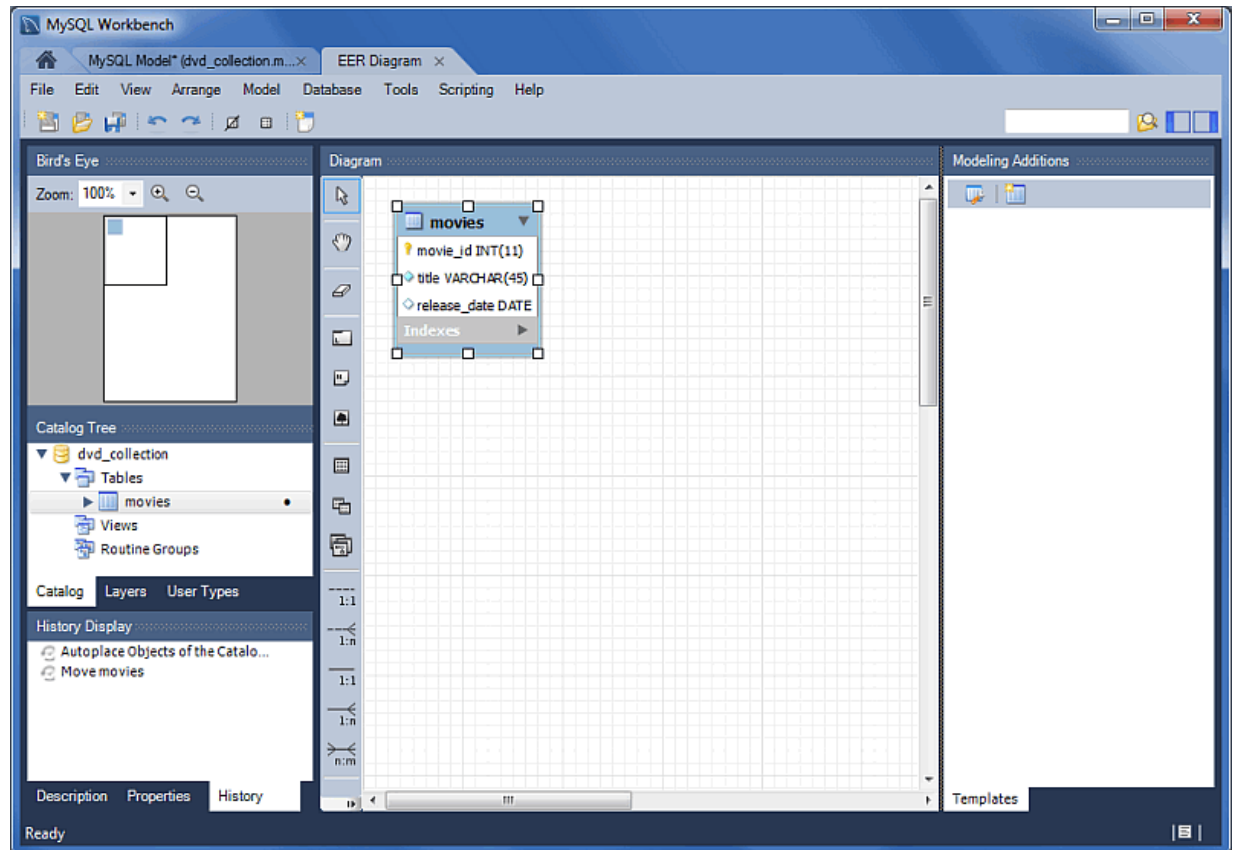
Column Name	Data Type	Column Properties
release_date	DATE (YYYY-MM-DD)	None

Figure 9.24 Getting Started Tutorial - Editing table columns



- For a visual representation (EER diagram) of this schema, select **Model** and then **Create Diagram from Catalog Objects** to create the EER Diagram for the model. The next figure shows the a new tab titled **EER Diagram**, which displays diagram representation of the movies table and columns.

Figure 9.25 Getting Started Tutorial - EER Diagram



- 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.

- Save the model by choosing **File** and then **Save Model** from the menu, or click the **Save Model to Current File** icon on the menu toolbar. For this tutorial, type `Home_Media` and then click **Save**.

Before synchronizing your new model with the live MySQL server, confirm that you already created a MySQL connection. This tutorial assumes you have created a connection already. If not, see [Section 5.2, “Creating A New MySQL Connection \(Tutorial\)”](#) and use that tutorial to create a MySQL connection named **MyFirstConnection**, although an alternative connection can also work.

Now forward-engineer your model to the live MySQL server as follows:

- Select **Database** and then **Forward Engineer** from the menu to open the Forward Engineer to Database wizard.

- The Connection Options step selects the MySQL connection and optionally sets additional options for the selected MySQL connection. Make any necessary connection changes and then click **Next**.

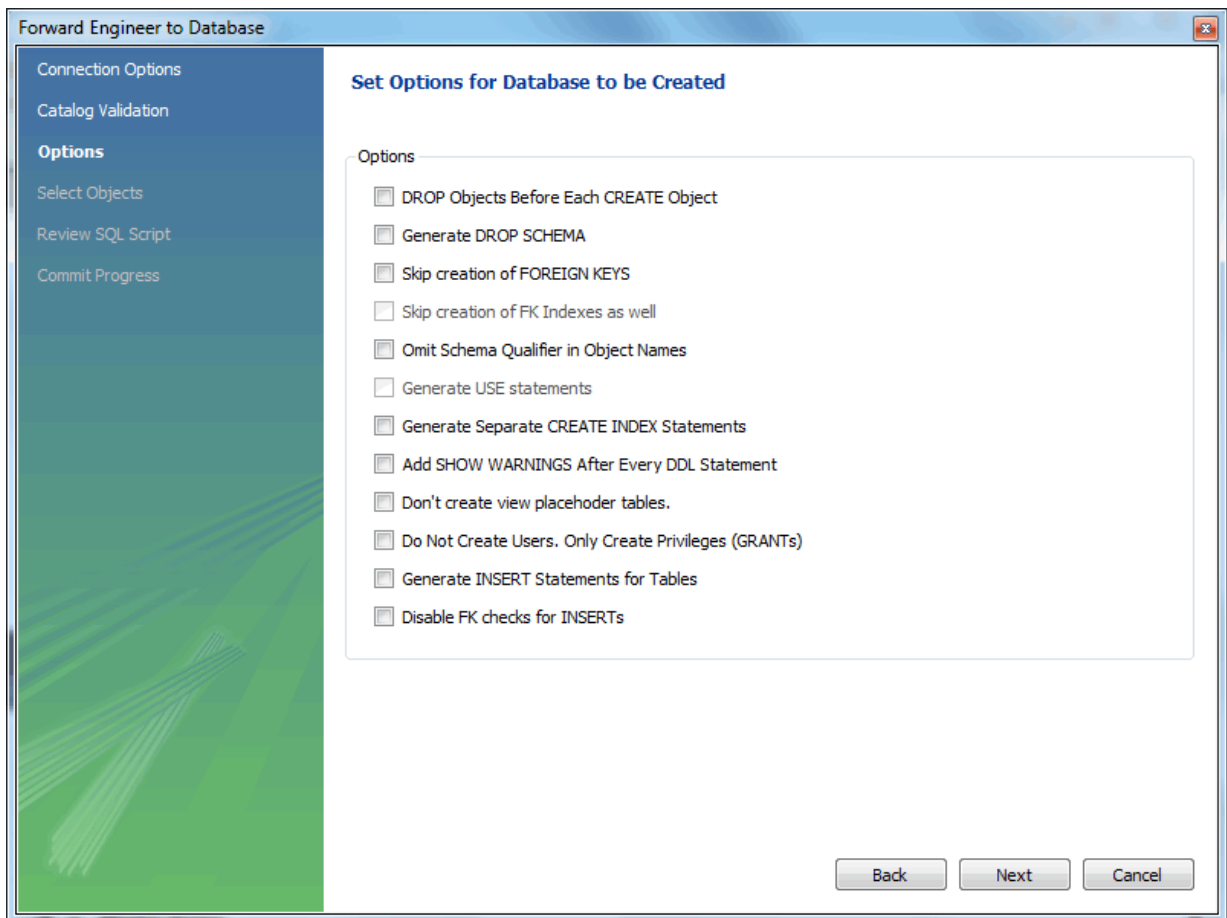


Note

You may decided to choose a different MySQL connection here, but this tutorial uses **MyFirstConnection**.

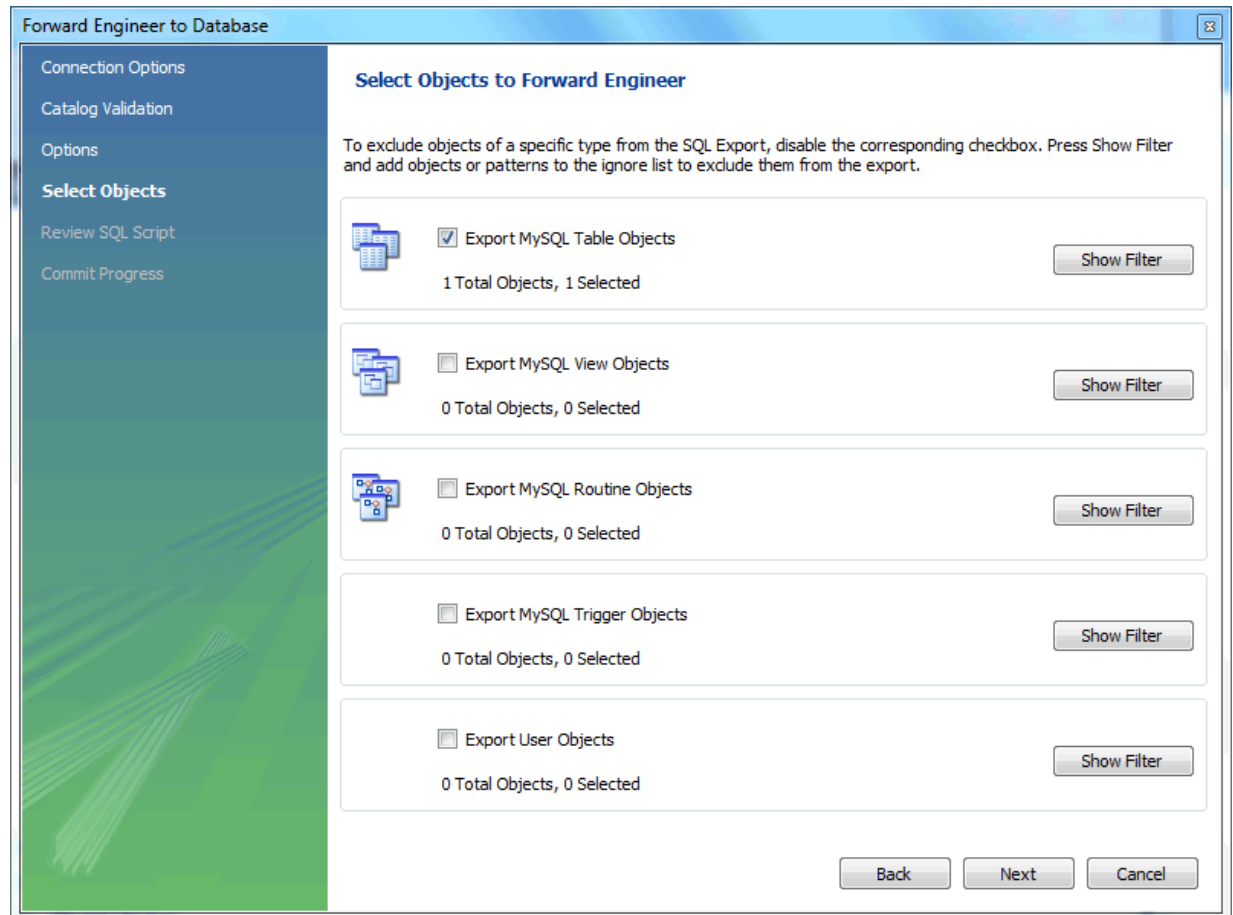
- The Options step lists optional advanced options (as shown in the figure that follows). For this tutorial, you can ignore these options and click **Next**.

Figure 9.26 Getting Started Tutorial - Options



- Select an object to export to the live MySQL server. In this case, there is only one table (`dvd_collection.movie`). Select the `Export MySQL Table Objects` check box (as the figure that shows) and then click **Next**.

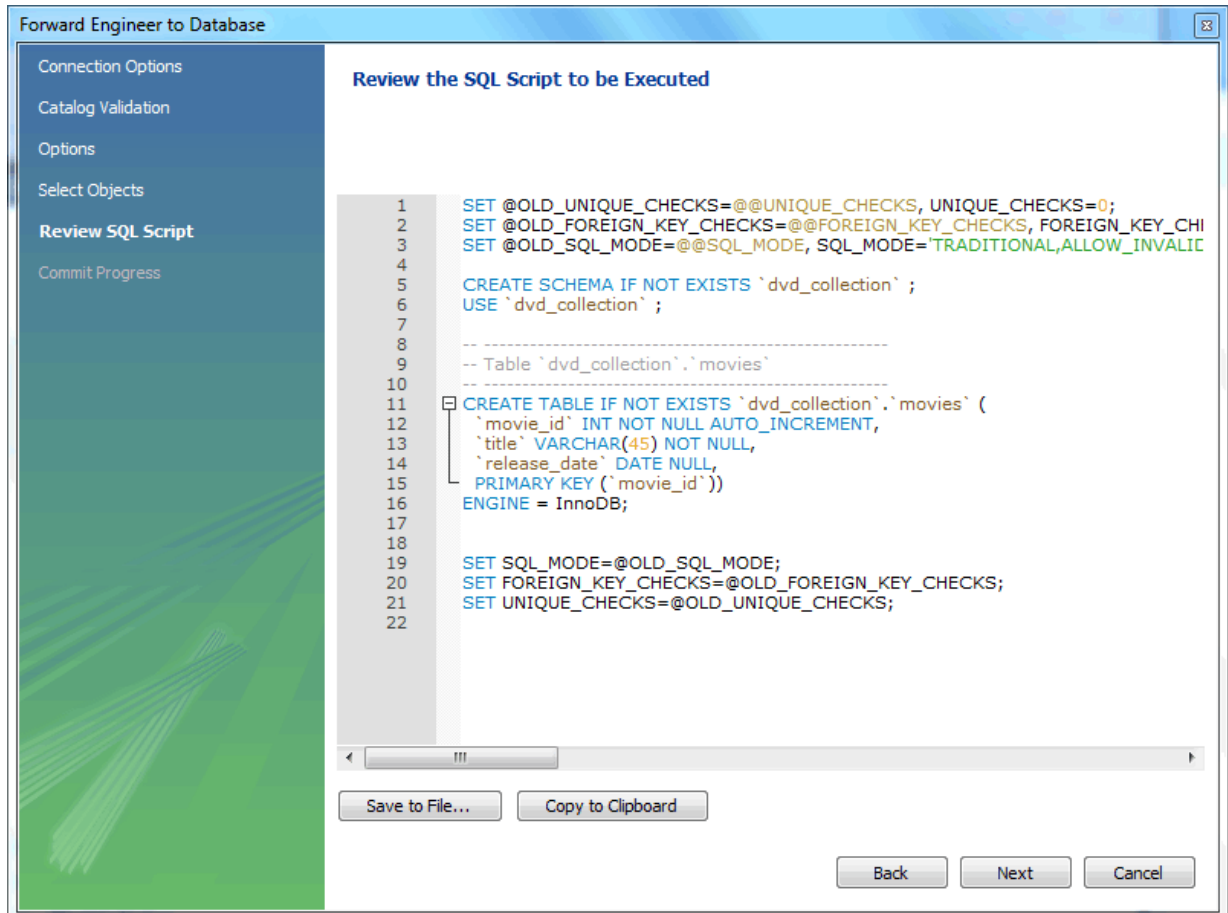
Figure 9.27 Getting Started Tutorial - Select Objects



- The Review SQL Script step 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 9.28 Getting Started Tutorial - Review SQL Script

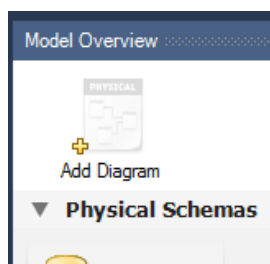


- The Commit Progress step confirms that each task was executed. Click **Show Logs** to view the logs. If no errors are present, click **Close** to close the wizard.
- The new `dvd_collection` database is now present on the MySQL server. Confirm this by opening the MySQL connection and viewing the schema list, or by executing `SHOW DATABASES` from the MySQL Command Line Client (`mysql`).
- Click the **Save Model to Current File** icon on the menu toolbar to save the model.

For additional information about data modeling, see [Chapter 9, Database Design and Modeling](#).

9.3.2 Basic Modeling

On the `MySQL Model` screen, double-click the **Add Diagram** icon (shown in the following figure). This action creates and opens a new `EER Diagram` canvas.

Figure 9.29 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 model 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 <https://dev.mysql.com/doc/index-other.html> page.

After downloading the file, extract it to a convenient location. Open MySQL Workbench, select the models view from the sidebar in the home screen, click (**>**) next to **Models**, and then click **Reverse Engineer MySQL Create Script**. 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 Schemas** section by double-clicking the arrow on the left of the **Physical Schemas** title bar. Select the tab labeled `sakila`.

You may also wish to remove the default schema tab, `mydb`. Select this tab, then click **-** on the upper right in the **Physical Schemas** panel.

To view all the objects in the `sakila` schema, you may need to expand the **Physical Schemas** panel. Move the mouse pointer anywhere over the gray area that defines the lower edge of the **Physical Schemas** panel. 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.

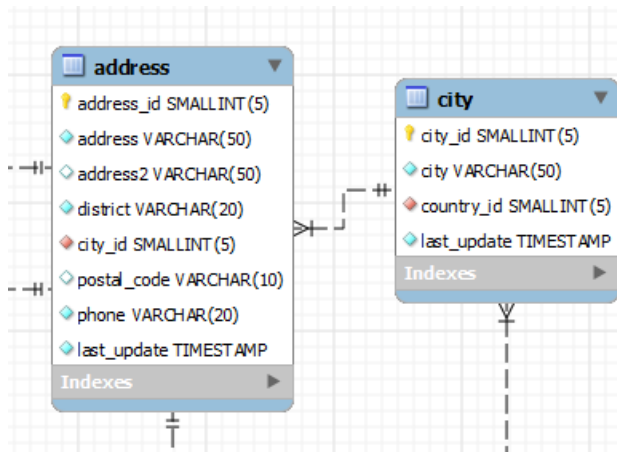
For a complete description of importing a MySQL create script, see [Section 9.4.2.1, "Reverse Engineering Using a Create Script"](#).

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 schemas that appear in the **MySQL Model** tab. Find the `sakila` schema and expand the view of its objects by clicking **+** 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, as the following figure shows.

Figure 9.30 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 9.35, “The sakila Database EER Diagram”](#).)

Click the **Properties** tab of the panel on the lower left and then click one of the tables on the canvas. This action displays the properties of the table in the `Properties` window, as the next figure shows. 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 9.31 Viewing The Properties

Properties Editor	
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

Description Properties History

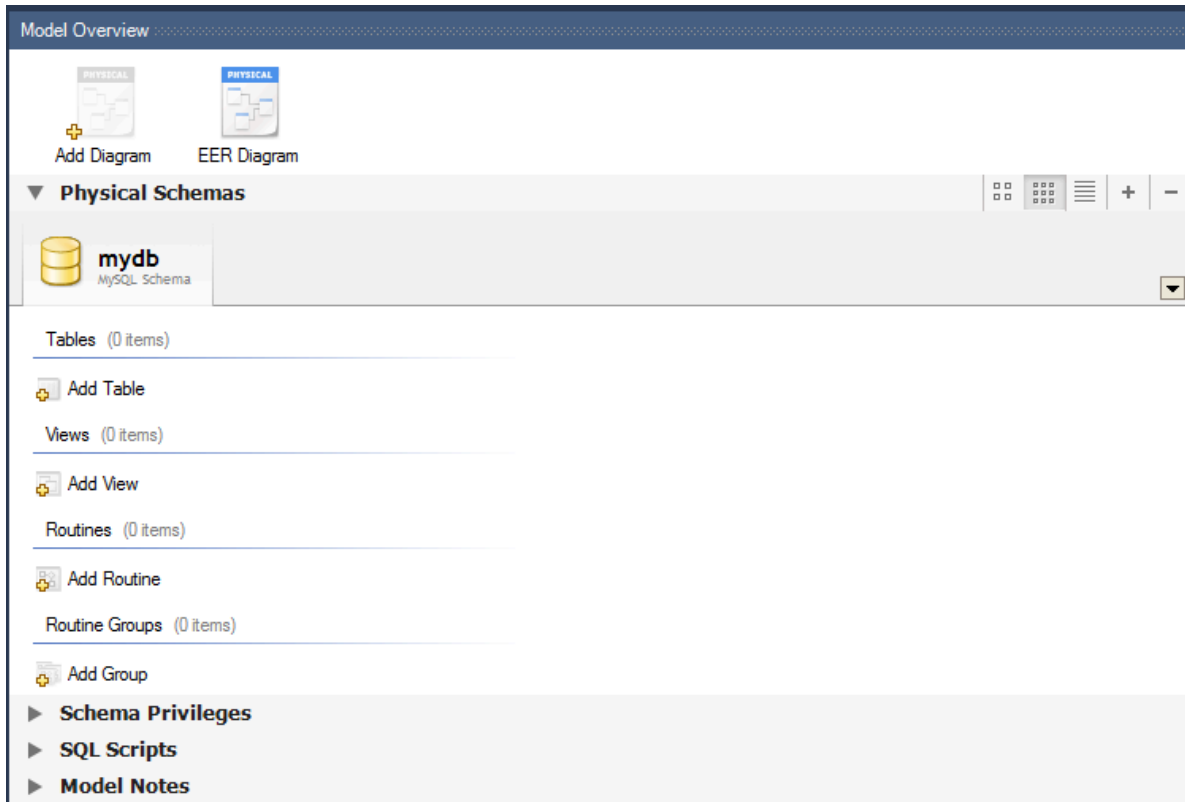
Changing the color of a table is a good way to identify a table quickly—something that becomes more important as the number of tables increases. Changing the color of a table is also an easy way to identify a table in the `Model Navigator` panel. This panel, the uppermost panel on the left side of the page, gives a bird's eye view of the entire EER canvas.

Save your changes to a MySQL Workbench model 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 Schemas** section of MySQL Workbench as the following figure shows. You can begin designing a database by using this default schema.

Figure 9.32 The Default Schema



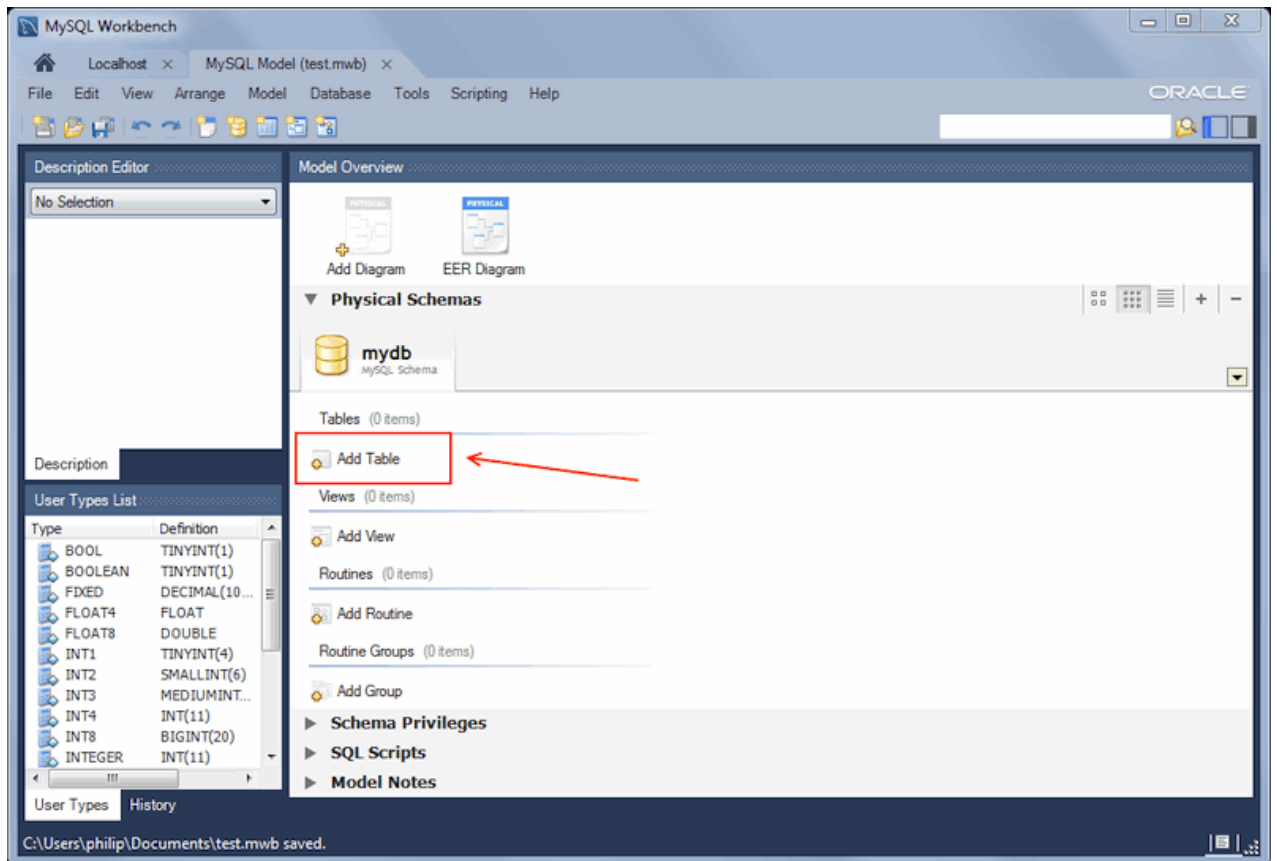
To change the name of the default schema, double-click the schema tab. This opens a schema editor window docked at the bottom of the application. To undock or redock this window, double-click anywhere in the editor title bar.

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

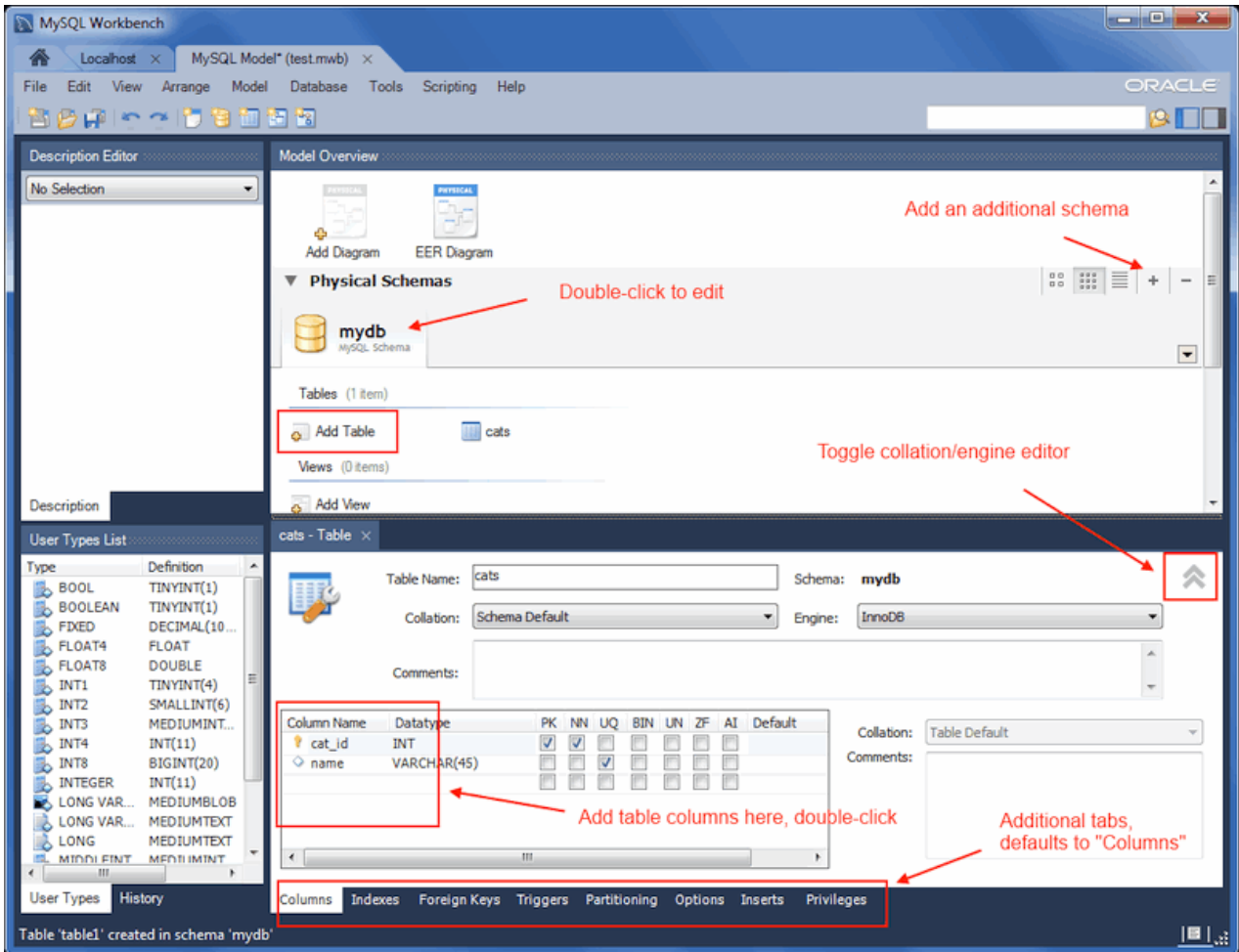
Create a new table by double-clicking the **Add Table** icon in the **Physical Schemas** panel, as the next figure shows. This action opens the table editor docked at the bottom of the application. You can undock or dock this editor in exactly the same way as the schema editor window.

Figure 9.33 Model: Creating A New Table



Initially, the table name defaults to 'table1' in the table editor. The following figure describes the available actions.

Figure 9.34 Model: Editing Table Values



In the previous example, 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 model 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 demonstrates the capabilities of MySQL Workbench as a documentation tool by using the [sakila](#) database, which is a database sample provided by MySQL. You can find this database sample, and others, by visiting the <https://dev.mysql.com/doc/index-other.html> page.

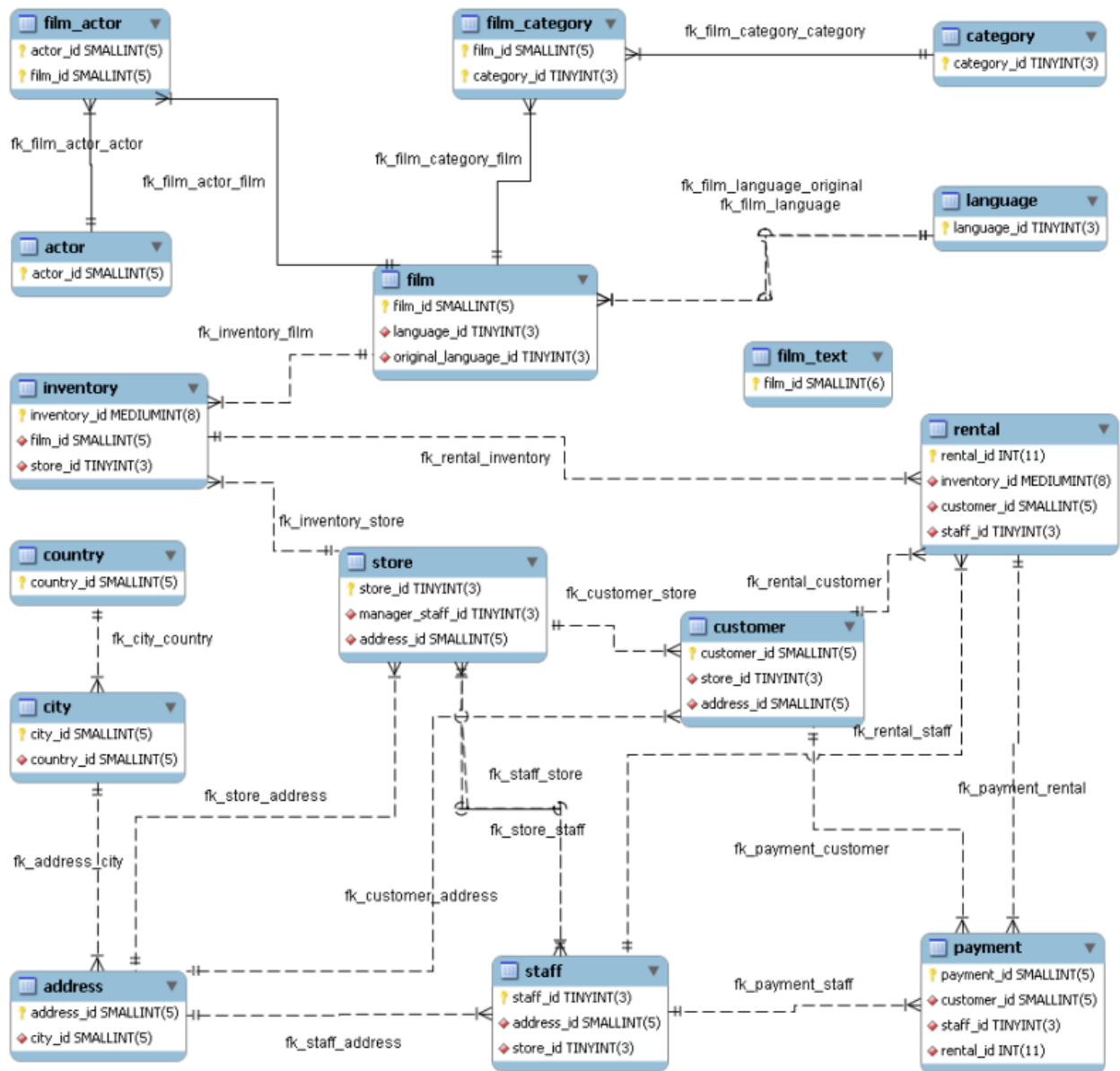
An EER diagram provides a quick overview and understanding of a database. Rather than reading through table definition statements, a quick glance at an EER diagram indicates how tables are related.

You can also see how tables are related; what the foreign keys are and what the nature of the relationship is.

A PNG File of the sakila Database

Find following an EER diagram showing the tables in the [sakila](#) database. The following figure shows the output that was created using the **File, Export, Export as PNG** menu item.

Figure 9.35 The sakila Database EER Diagram



The object notation style used in Figure 9.35, “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

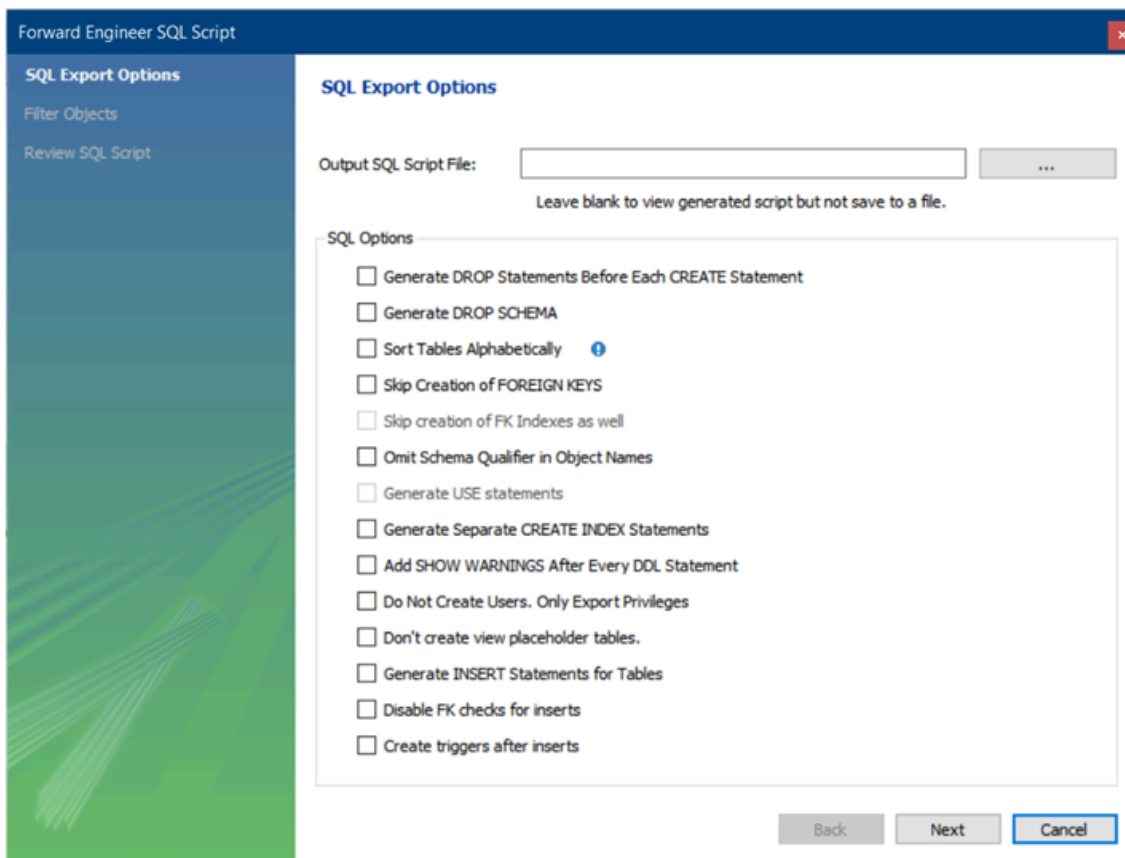
Forward engineering enables you to create a script of your database model. You may export a script to alter an existing database or create a new database. The script to create a database is similar to the one created using the `mysqldump db_name` command.

Choosing to create a database yields additional options.

Creating a Schema

With the **MySQL Model** (*model-name.mwb*) tab open, click **File, Export**, and then **Forward Engineer SQL CREATE Script** to start the Forward Engineer SQL Script wizard. The following figure shows the first page of the wizard.

Figure 9.36 SQL Export Options



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**

- **Sort Tables Alphabetically**

When this option is unchecked, tables are sorted according to foreign-key references.

- **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**

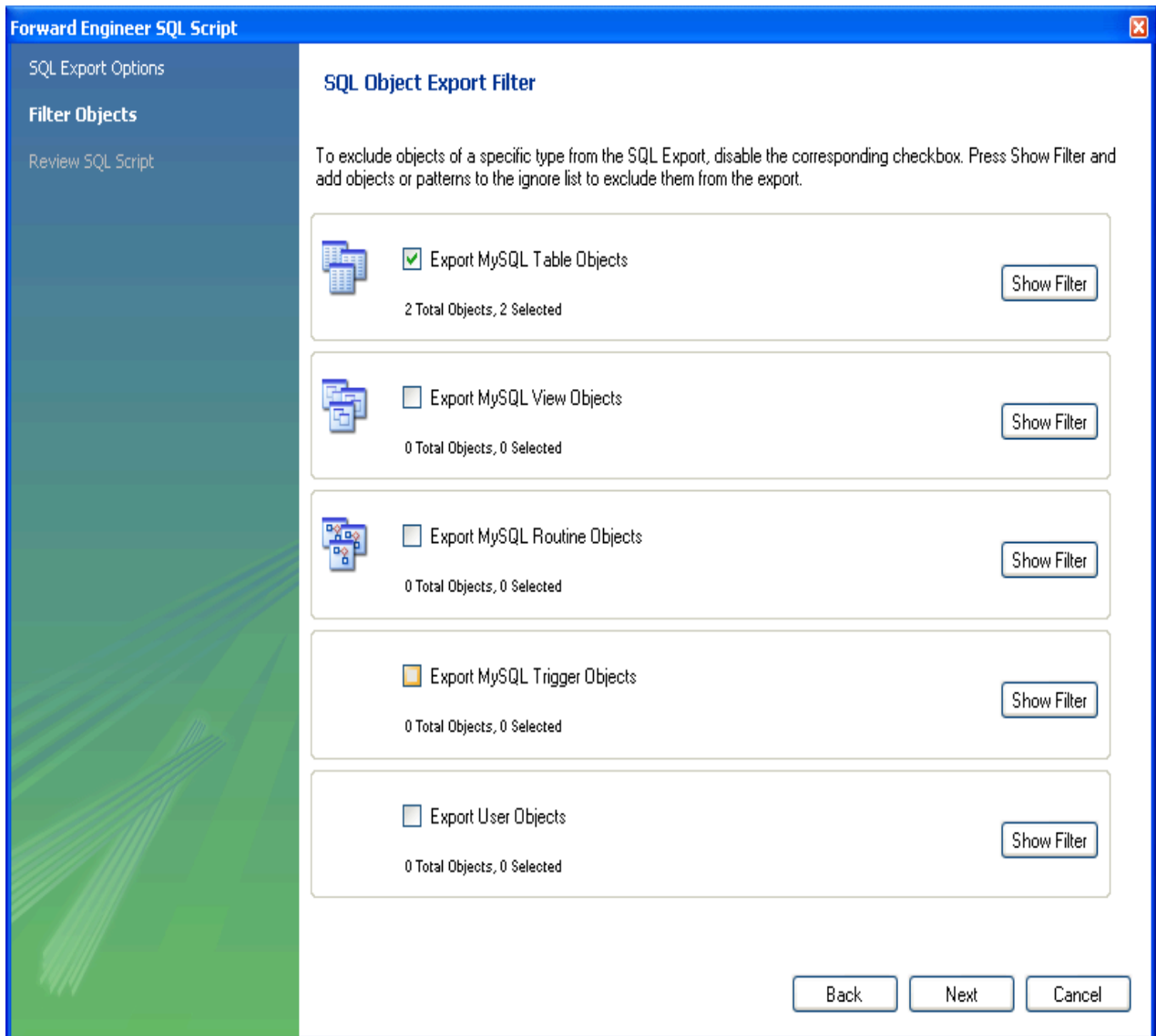
Select this option if you have added any rows to a table. For more information about inserting rows, see [Section 8.1.1, "SQL Query Tab"](#).

- **Disable FK checks for inserts**

- **Create triggers after inserts**

Clicking **Next** opens the **SQL Object Export Filter** page (see the figure that follows), which enables you to select the objects for export.

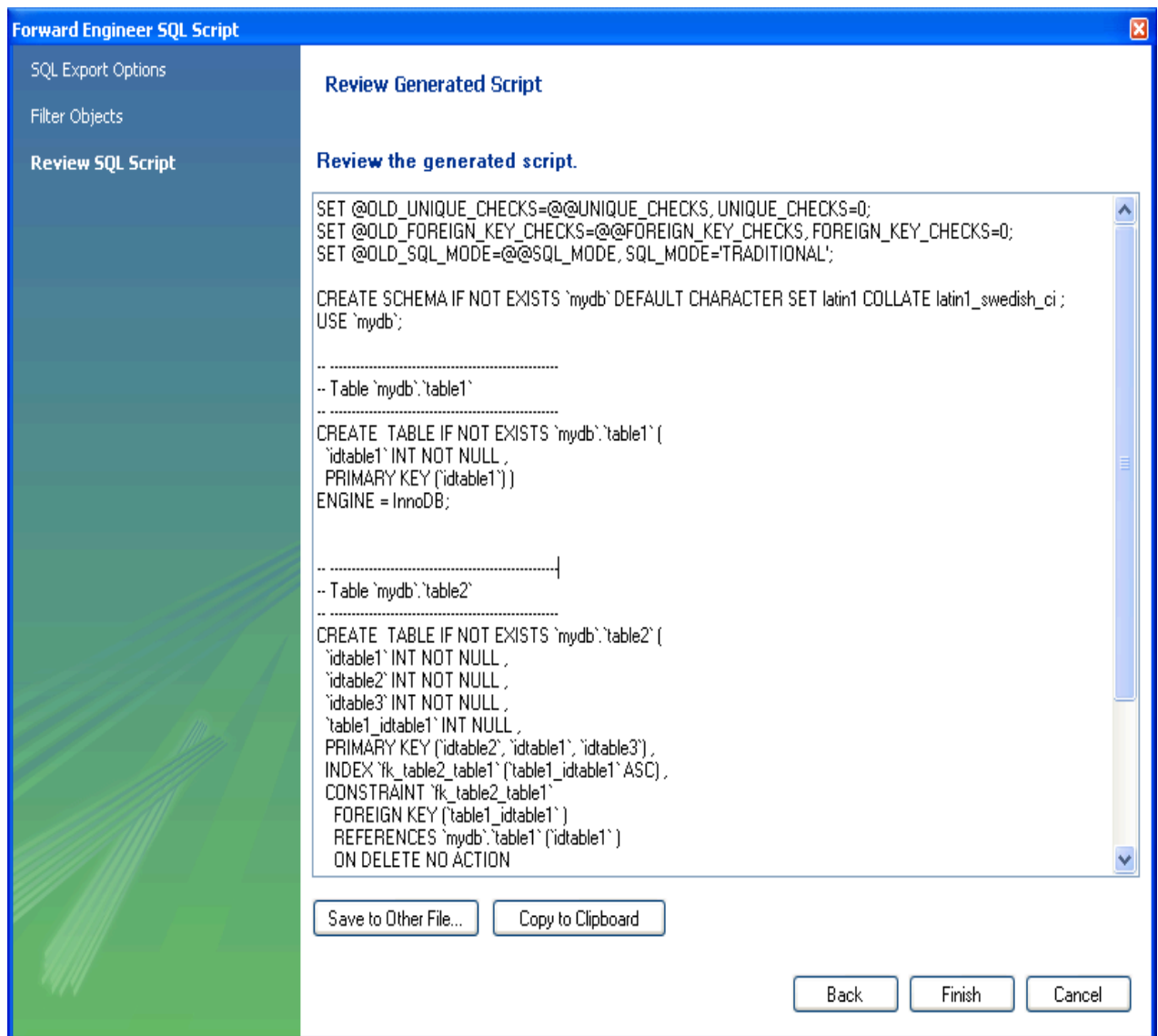
Figure 9.37 SQL Object Export Filter



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. The following figure shows an example script to review.

Figure 9.38 Review Generated Script



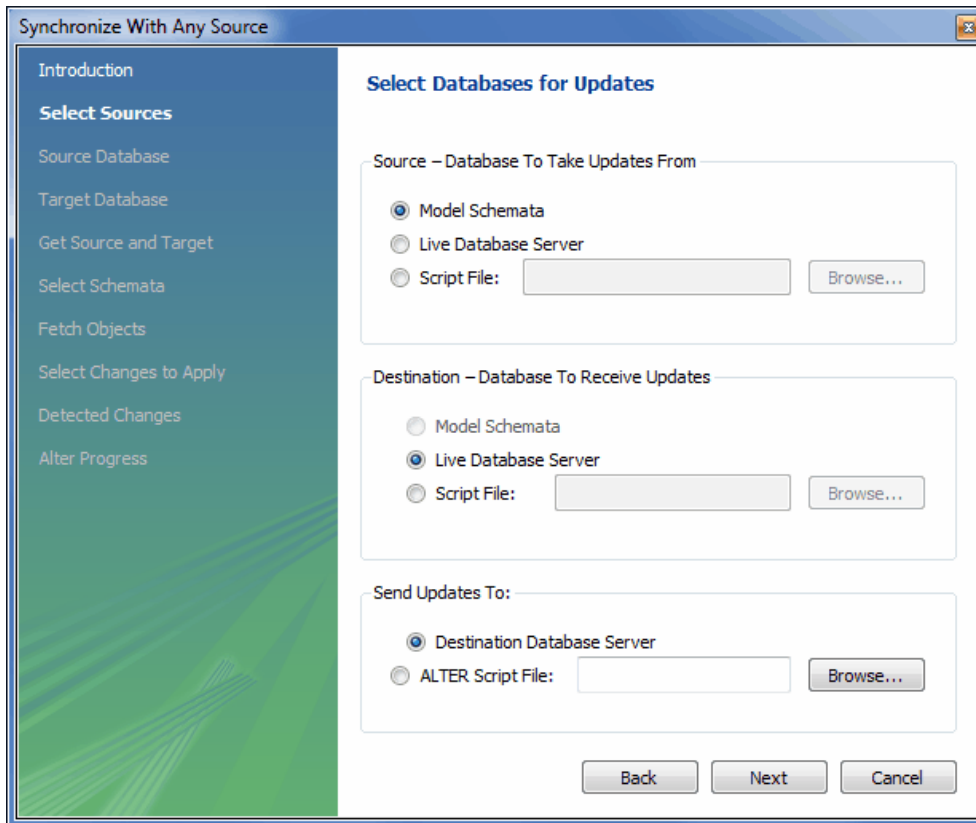
You may return to the previous page using the **Back** button.

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 for setting the source and destinations of the updates.

Figure 9.39 Synchronize With Any Source: Options

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. As the following figure shows, this page enables you to use a previously stored connection or to enter the connection parameters.

Figure 9.40 Set Parameters for Connecting to a DBMS

The screenshot shows the 'Forward Engineer to Database' wizard window. The left sidebar contains a list of steps: 'Connection Options' (selected), 'Catalog Validation', 'Options', 'Select Objects', 'Review SQL Script', and 'Commit Progress'. The main area is titled 'Set Parameters for Connecting to a DBMS'. It features a 'Stored Connection' dropdown set to 'Localhost' and a 'Connection Method' dropdown set to 'Standard (TCP/IP)'. Below these are three tabs: 'Parameters' (selected), 'SSL', and 'Advanced'. The 'Parameters' tab contains several input fields: 'Hostname' (localhost), 'Port' (3306), 'Username' (root), and 'Default Schema' (empty). Each field has a descriptive tooltip. The 'Password' field has 'Store in Vault ...' and 'Clear' buttons. At the bottom right, there are 'Back', 'Next', and 'Cancel' buttons.

Click **Next** after setting the connection parameters. The next page of the wizard displays Catalog Validation, as the following figure shows. Validation is available only in the Commercial Edition.

Figure 9.41 Catalog Validation

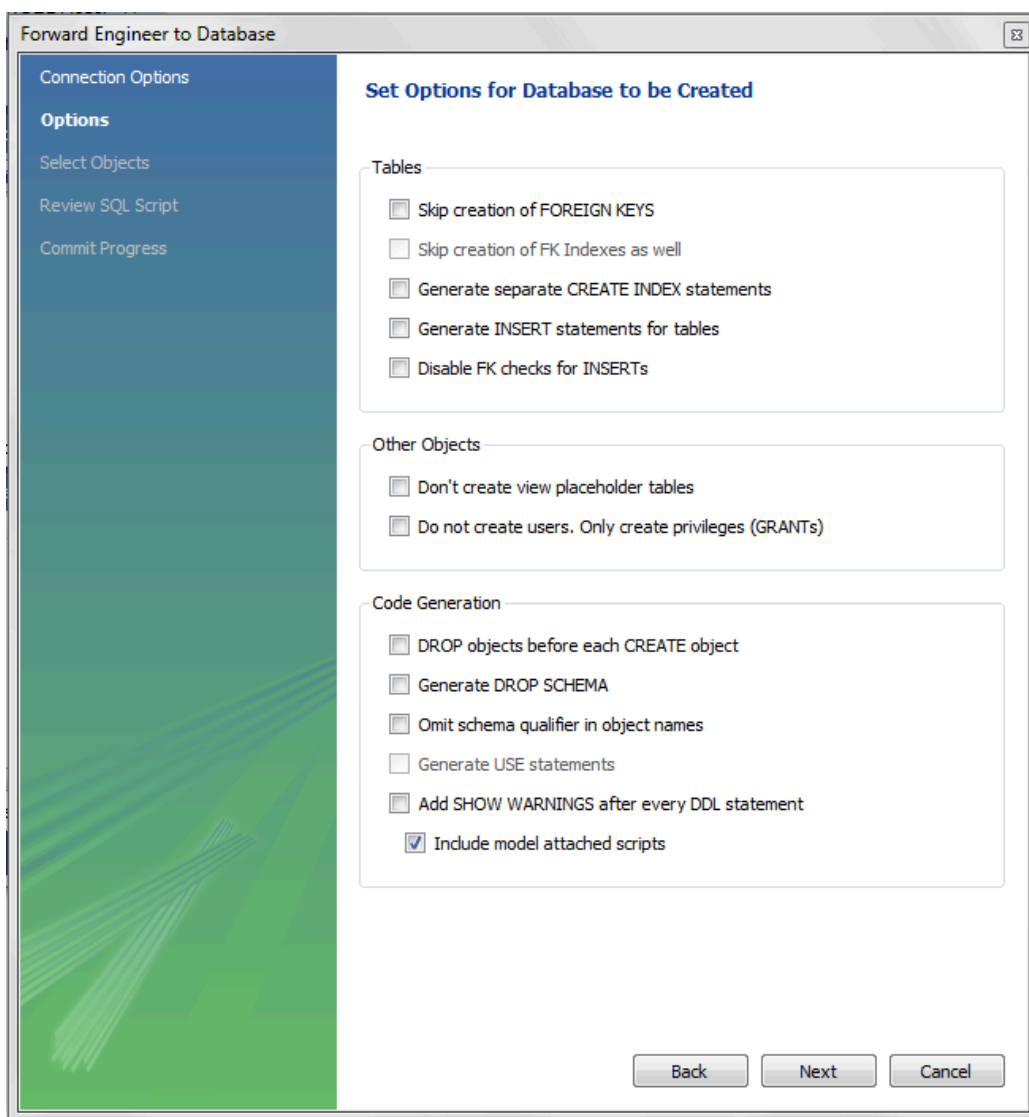
The screenshot shows the 'Forward Engineer to Database' wizard window at the 'Catalog Validation' step. The left sidebar now highlights 'Catalog Validation'. The main area is titled 'Catalog Validation' and contains the text: 'The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.' Below this, it says 'Validation Finished Successfully'. At the bottom, there are 'Show Logs', 'Run Validations', 'Back', 'Next', and 'Cancel' buttons.

Click **Run Validations** to validate the catalog.

Click **Next** to continue.

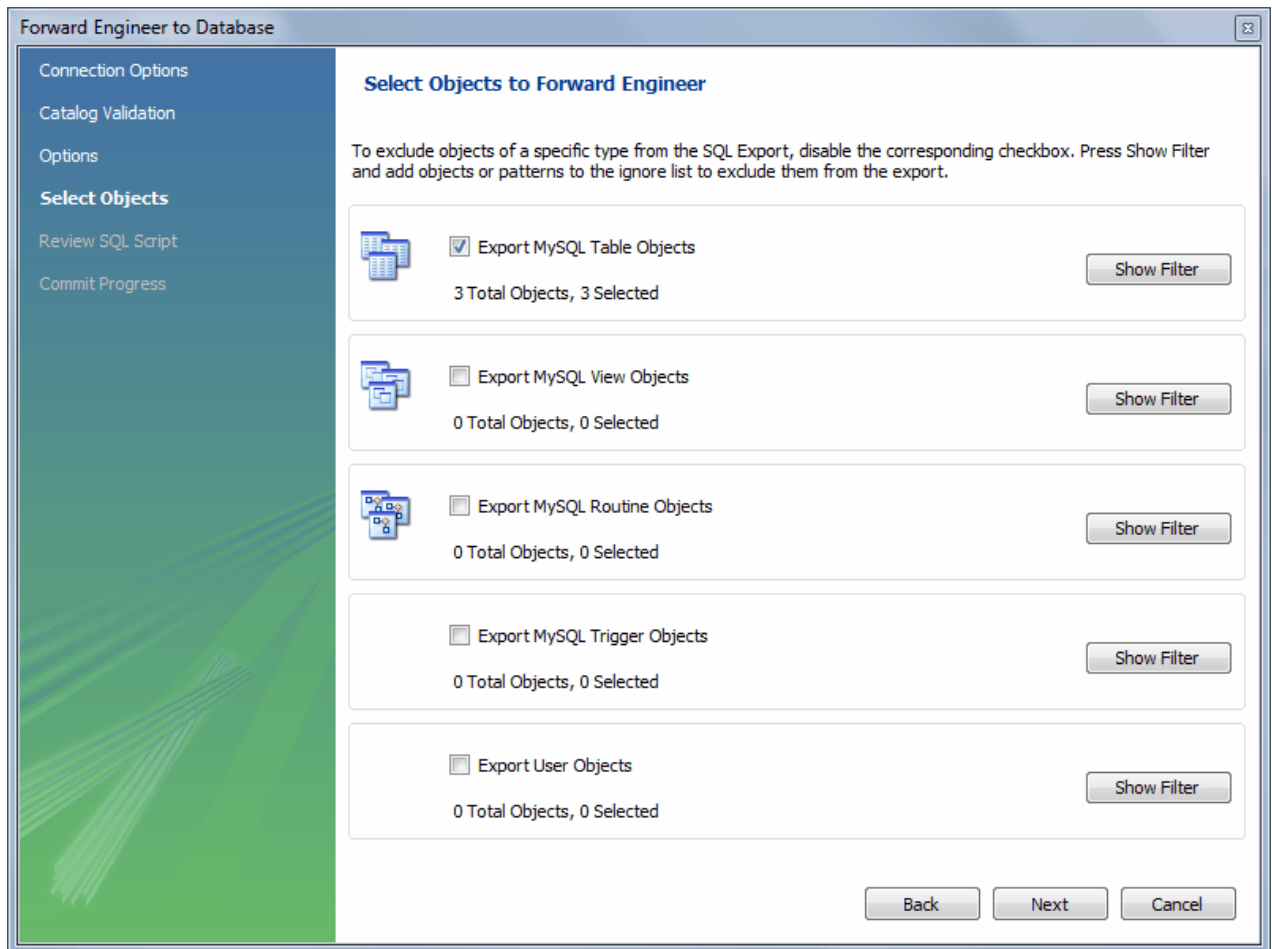
The next page enables you to set options for the database to be created (see the following figure). These options are as described in [Creating a Schema](#).

Figure 9.42 Options



Select the required options and then click **Next**.

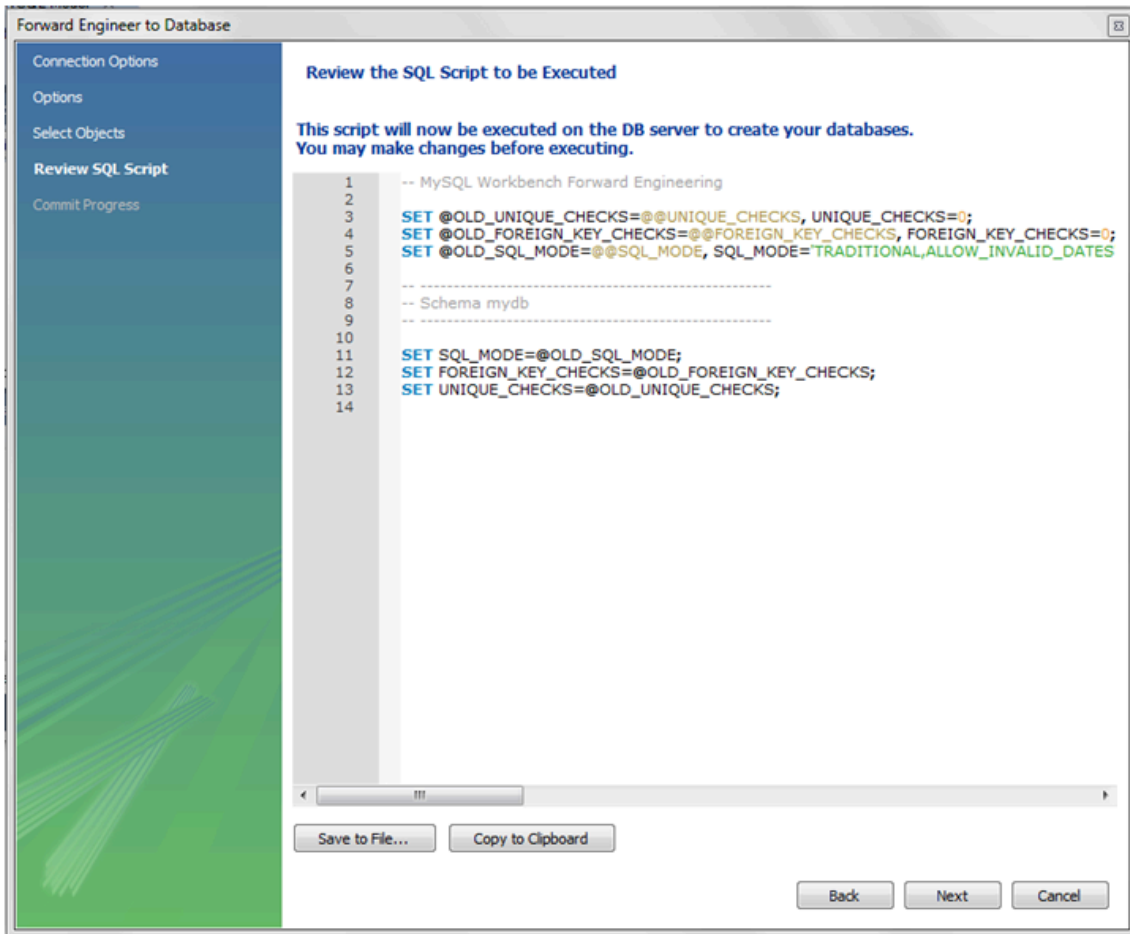
As the following figure shows, the Select Objects page enables you to select the objects to forward engineer: Table objects, view objects, routine objects, trigger objects, and user objects.

Figure 9.43 Select Objects to Forward Engineer

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.

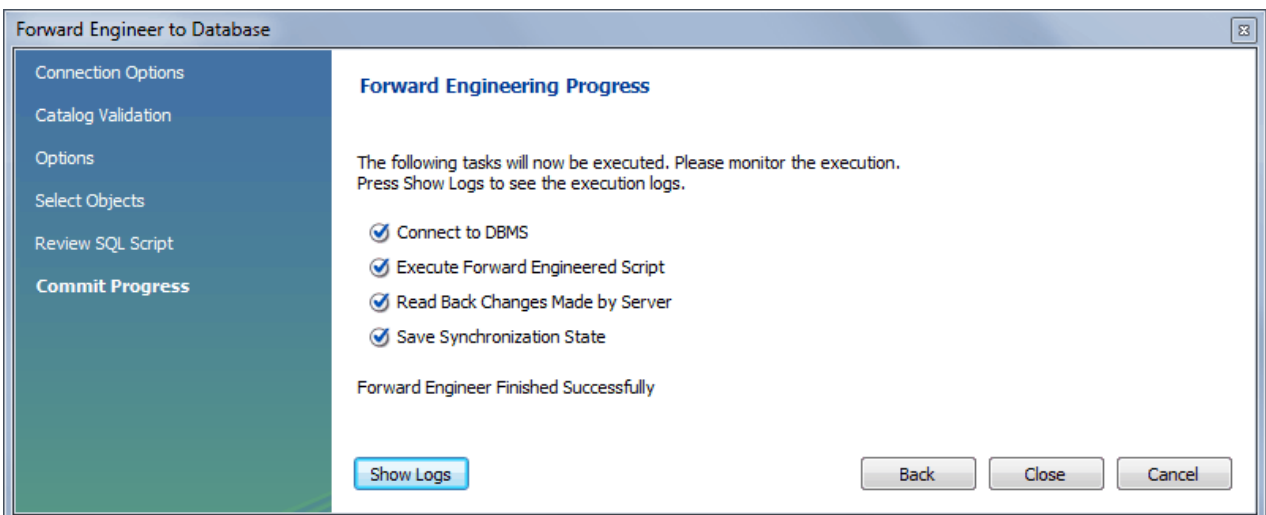
Figure 9.44 Review Script



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 9.45 Forward Engineering Progress



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.

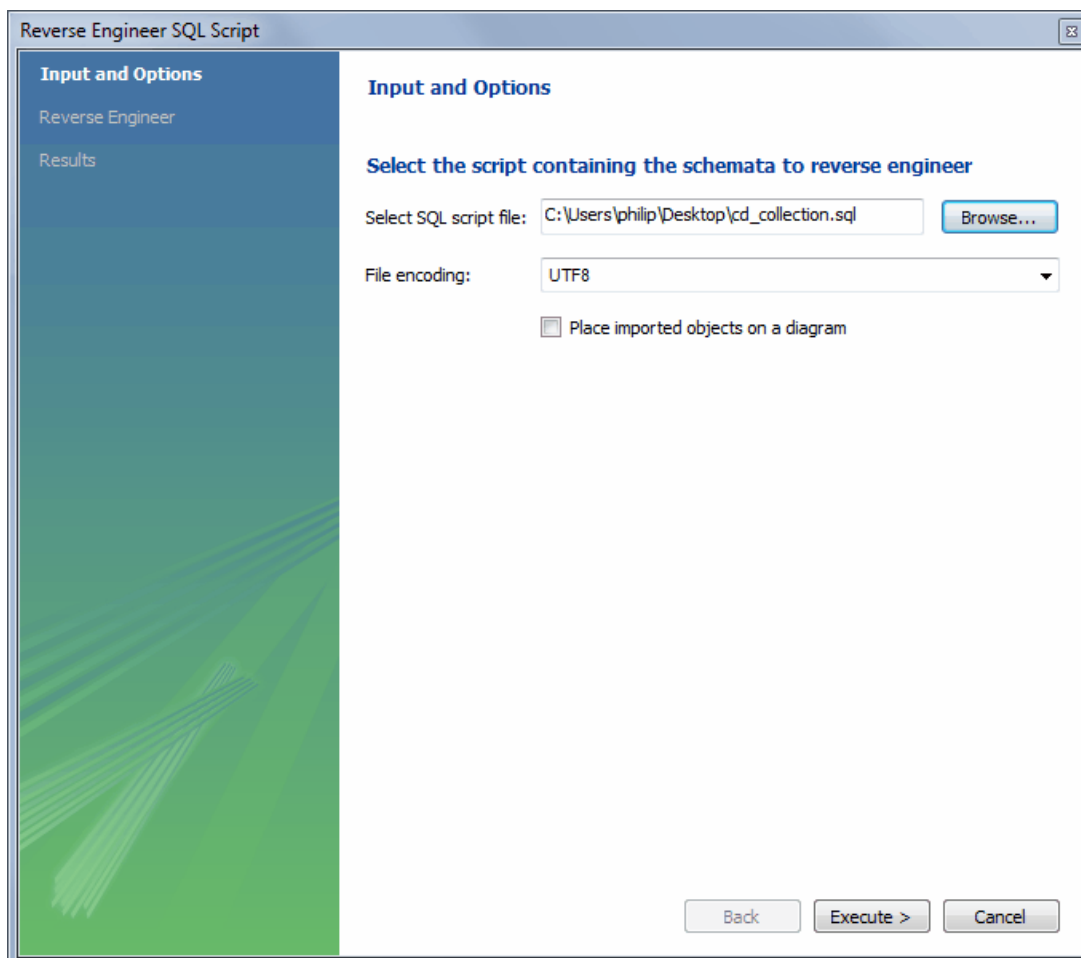
9.4.2.1 Reverse Engineering Using a Create Script

To reverse-engineer a database using a create script, do one of the following:

- On the home screen, select the model view from the sidebar, click (➤) next to **Models**, and then click **Reverse Engineer MySQL Create Script**.
- With a model selected and its model tab open click **File, Import**, and then **Reverse Engineer MySQL Create Script** from the menu.

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. The following figure shows an example of the input and options available for this action.

Figure 9.46 Reverse Engineer SQL Script: Input



- **Select SQL script file:** Open a file 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.



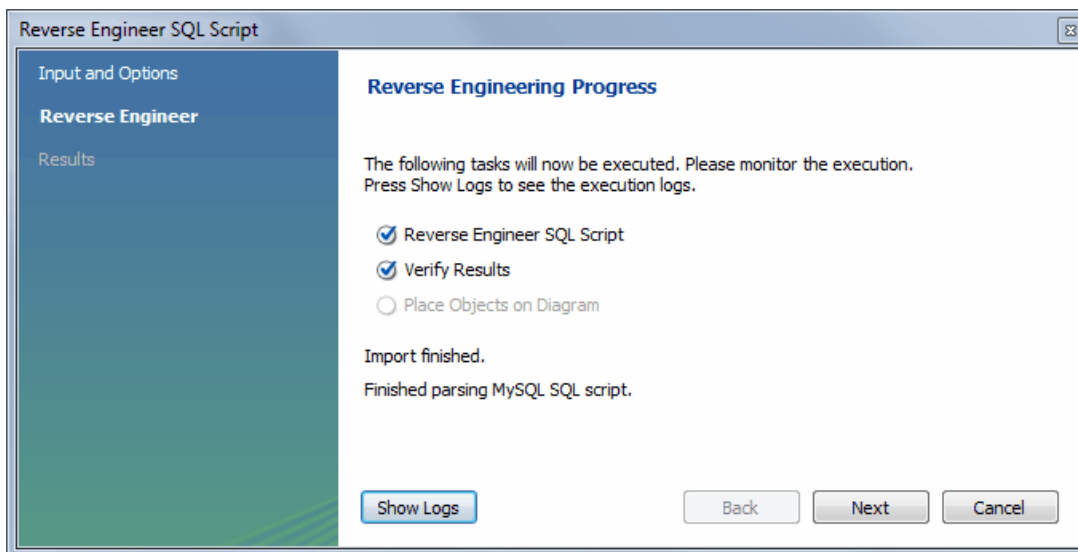
Note

Importing a large number (1000) 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 1000+ objects using the EER diagram catalog viewer.

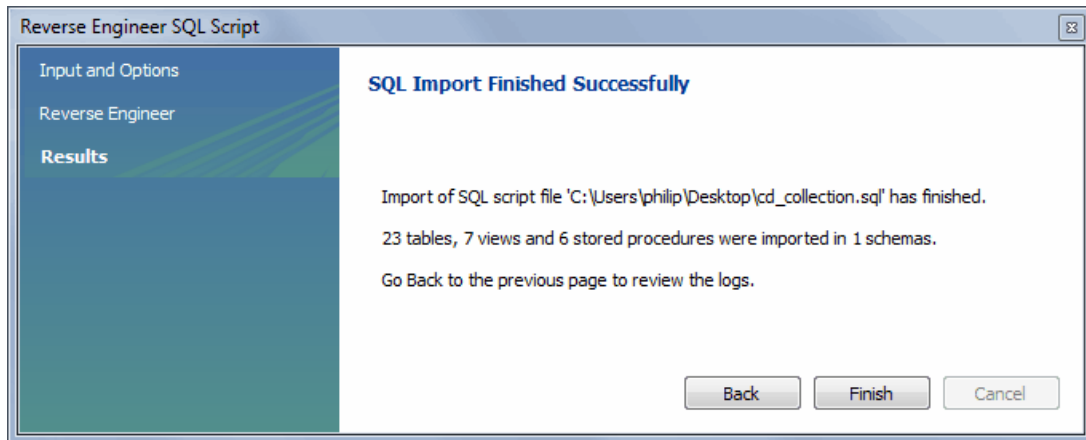
If your script creates a database, MySQL Workbench creates a new **Physical Schemas** area within the open **MySQL Model** tab.

Click **Execute** to reverse-engineer the SQL script, verify its results, and optionally place the objects in a new EER diagram. The next figure shows an example of the finished operation.

Figure 9.47 Reverse Engineer SQL Script: Execution



Click **Next** to view a summary of the results and then **Finish** to close the wizard.

Figure 9.48 Reverse Engineer SQL Script: Results

Before exiting MySQL Workbench, save the schema. Click **File** and then **Save** from the menu to save the reverse-engineered database as a MySQL Workbench file with the `.mwb` extension.

For a tutorial on reverse engineering the `sakila` database, see [Section 9.3.3, “Importing a Data Definition SQL Script”](#).

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, click **Database** and then **Reverse Engineer** from the menu. The figure that follows shows an example of the Reverse Engineer Database wizard.

Figure 9.49 Reverse Engineer Database Wizard

Reverse Engineer Database

Connection Options

- Connect to DBMS
- Select Schemas
- Retrieve Objects
- Select Objects
- Reverse Engineer
- Results

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

Parameters SSL Advanced

Hostname: localhost Port: 3306 Name or IP address of the server host. - TCP/IP p

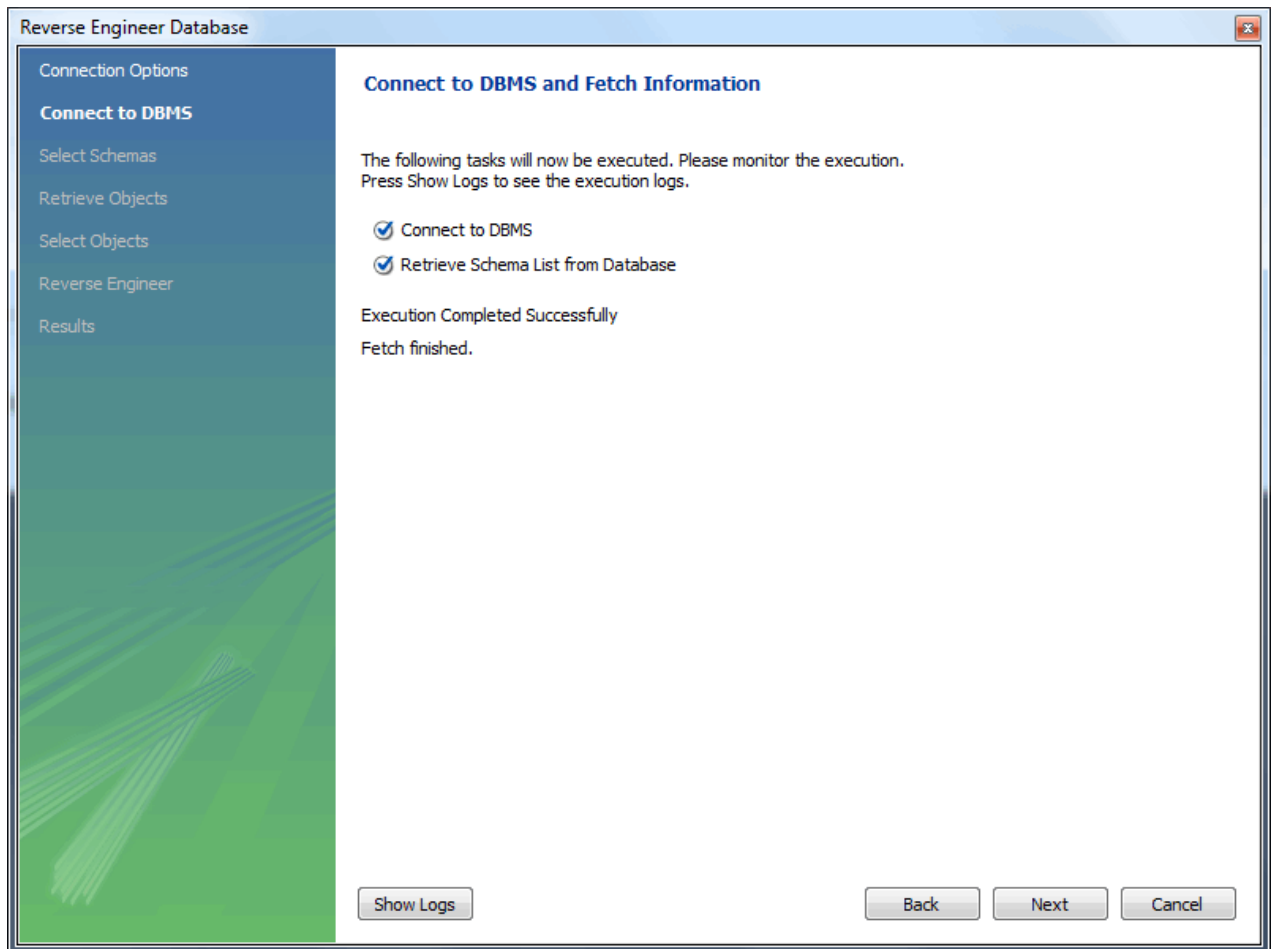
Username: root Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later if it's

Back Next Cancel

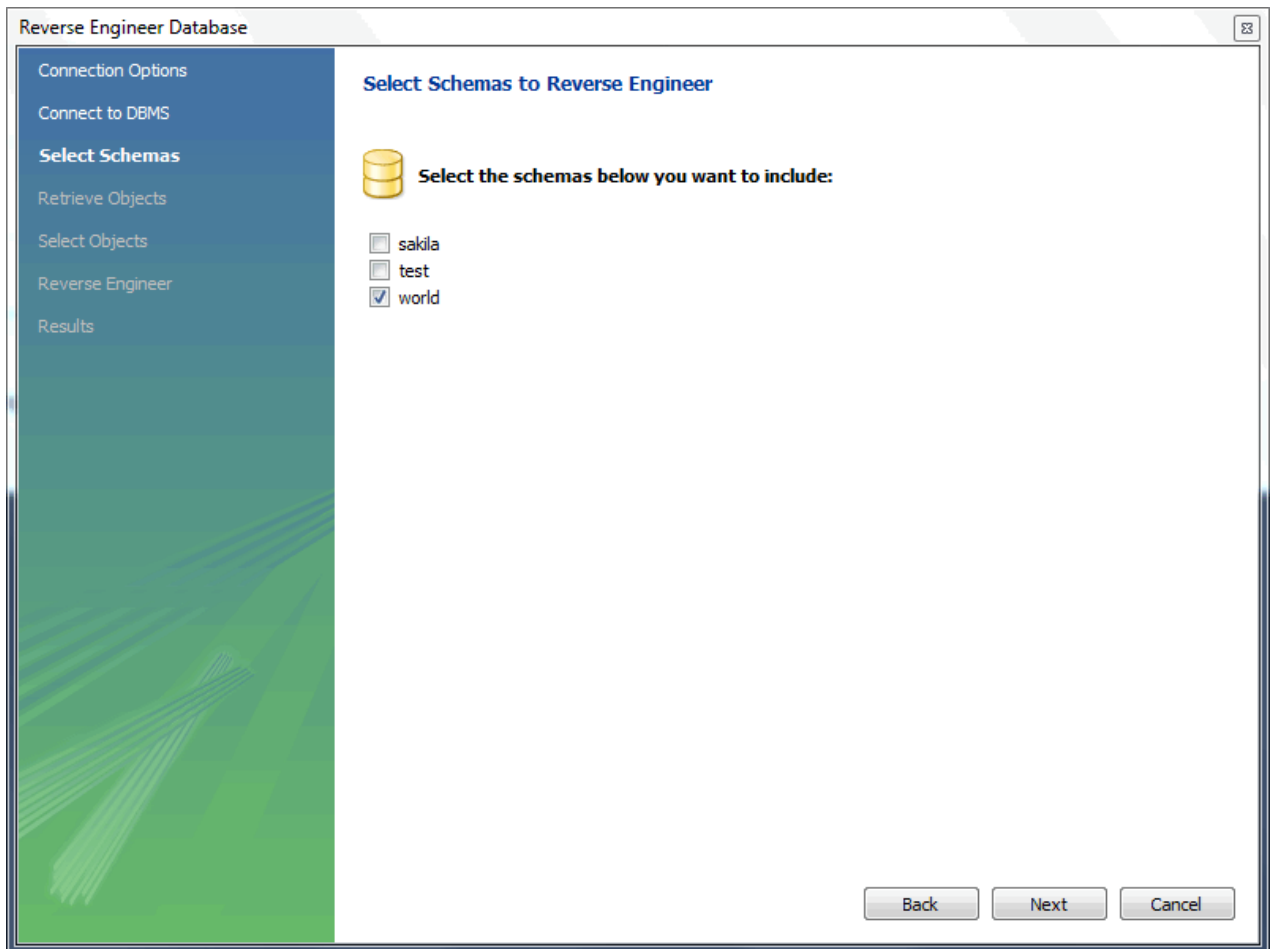
The first step of the wizard enables you to set up a connection to the live database you intend to reverse-engineer. You can set up a new connection or select a previously created stored connection. Typical information required for the connection includes host name, user name and password.

After this information has been entered, or you have selected a stored connection, click the **Next** button to proceed to the next step (shown in the figure that follows).

Figure 9.50 Connect to DBMS

Review the displayed information to make sure that the connection did not generate errors, then click **Next**.

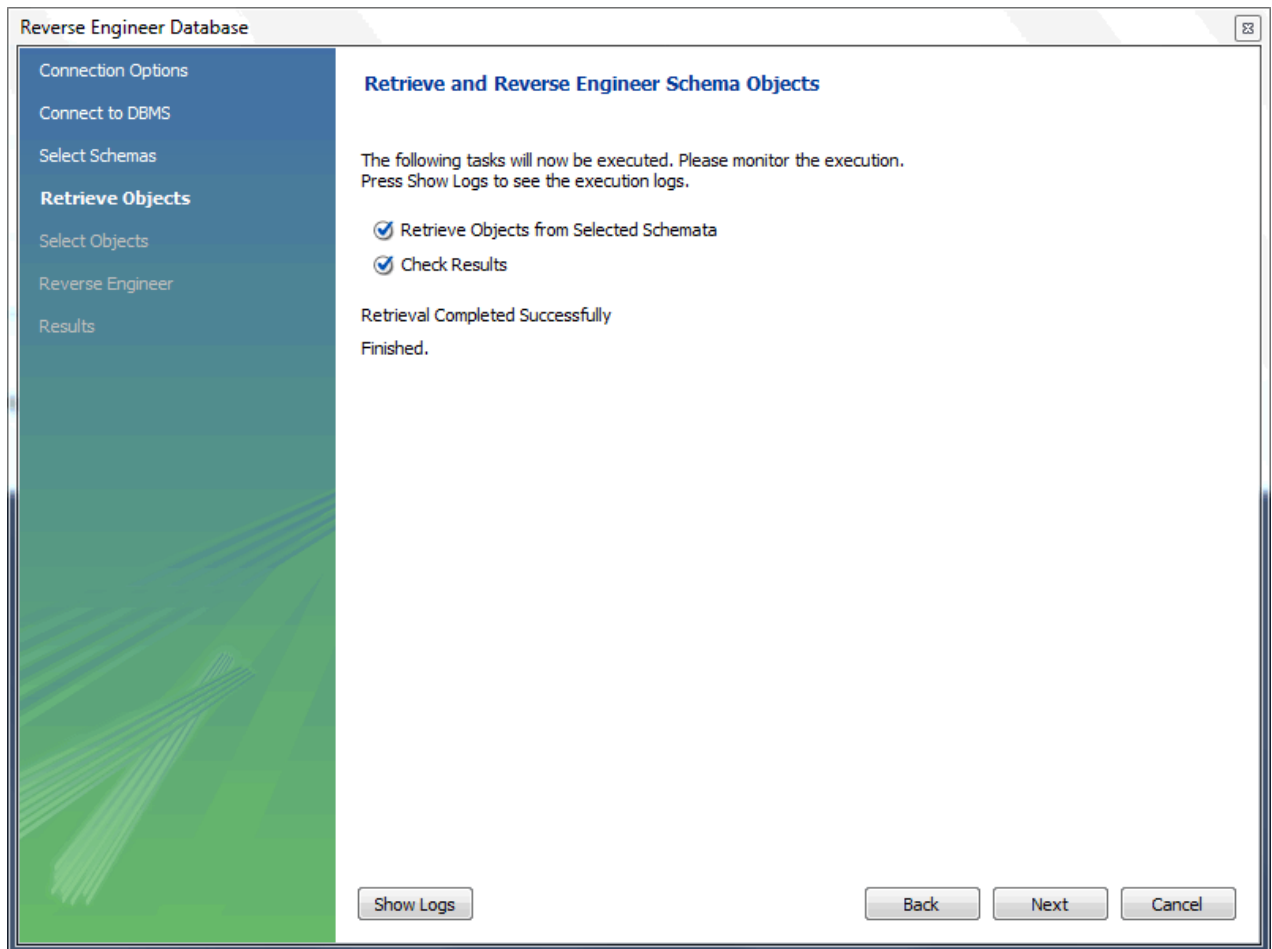
The next step displays the schemas available on the server. Select the check box of each schemas you intend to process. In the following figure, the `world` schema is selected.

Figure 9.51 Select Schemas

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 9.52 Retrieve Objects



Review the results before clicking **Next** to continue.

The next step opens the Select Objects to Reverse Engineer page. It has a section for each object type present in the schema that you can import (tables, views, routines, and so forth). All object types are selected by default. The **Place imported objects on a diagram** option is also selected by default.

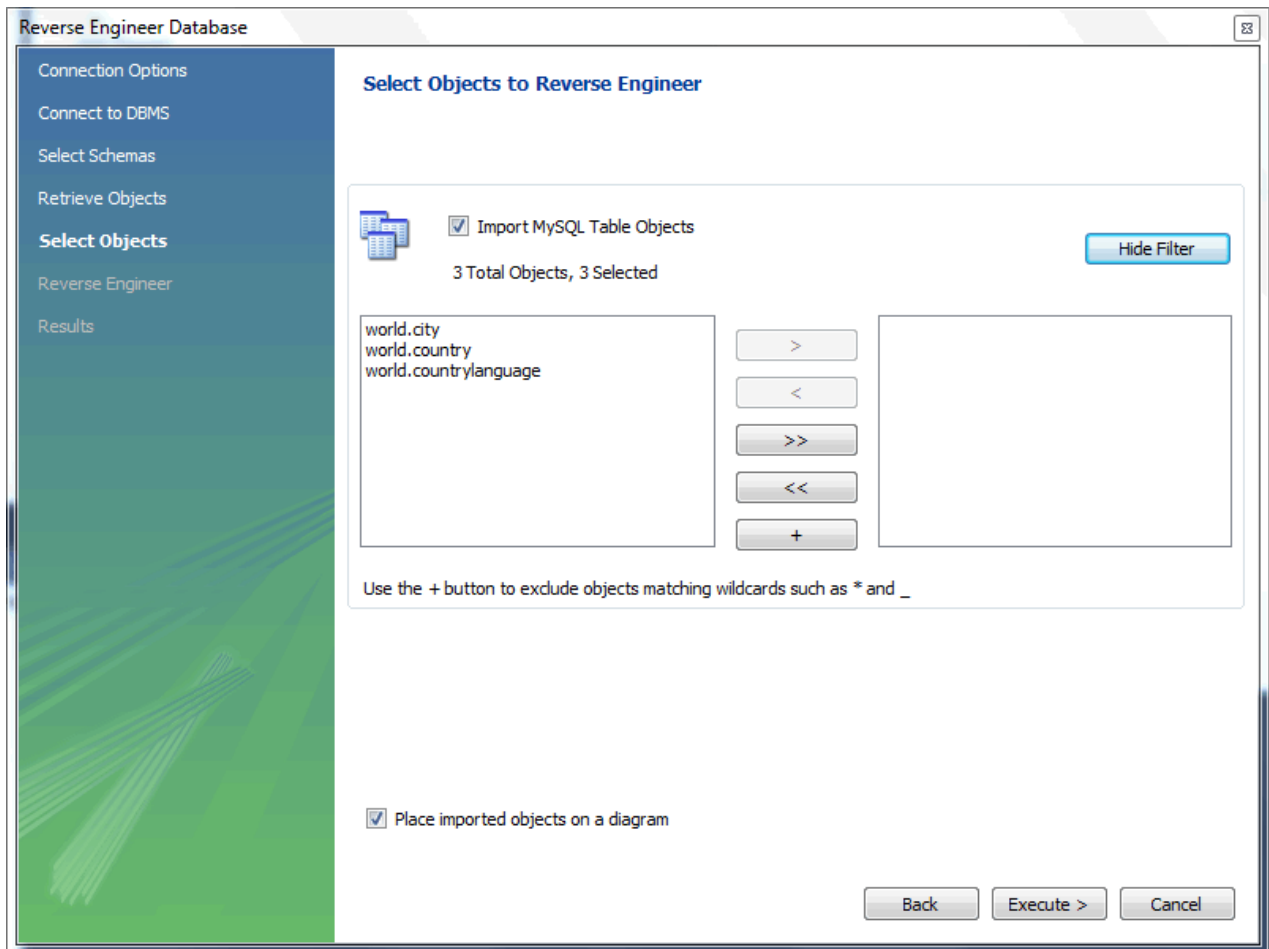


Note

Importing 250 or more 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 250+ objects using the EER diagram catalog viewer.

This step is of special interest if you do not intend 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. The following figure shows an example of the table object section with the filter open.

Figure 9.53 Select Objects

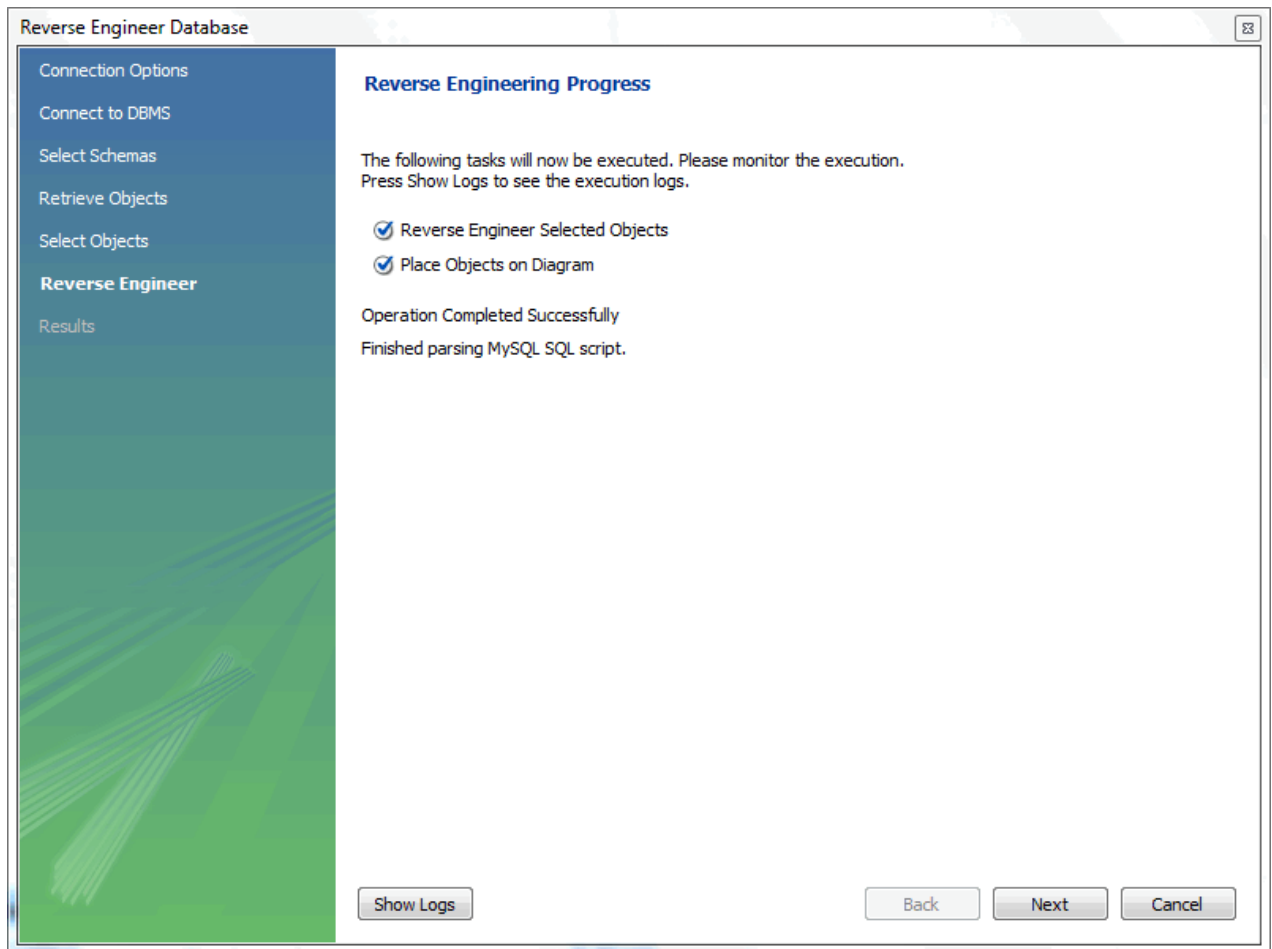


The options enable you to select specific tables for import. Having selected the desired tables, you can hide the filter by clicking **Hide Filter**.

The other sections, such as **MySQL Routine Objects**, have similar filters available.

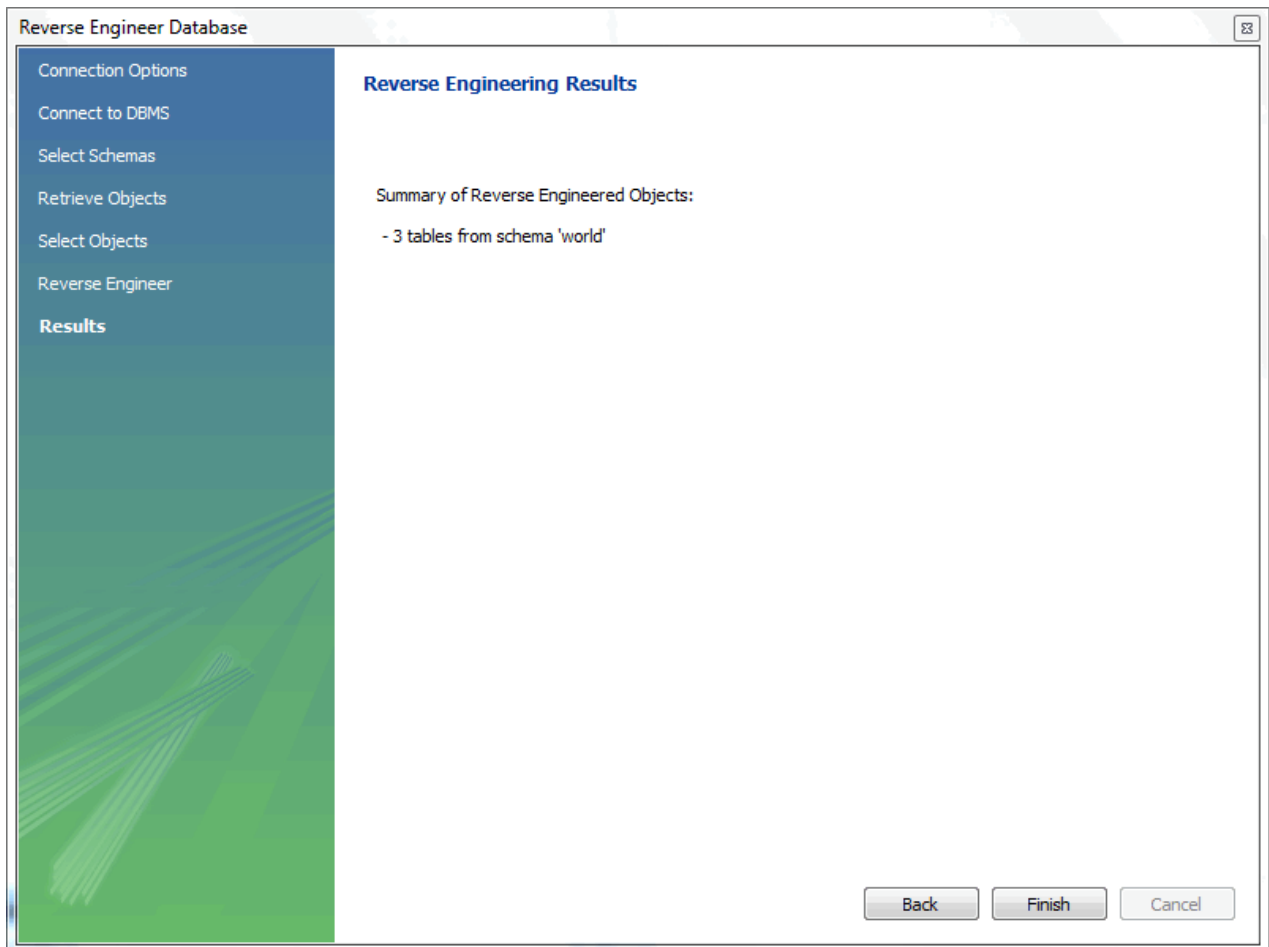
Click **Execute** to continue to the next step.

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 **Show Logs** to see the nature of the errors. The next figure shows an example of the operational progress, which completed successfully.

Figure 9.54 Reverse Engineer Progress

Click **Next** to continue to the next step.

The final step of the wizard provides a summary of the reverse-engineered objects, as the following figure shows.

Figure 9.55 Results

Click **Finish** to close the wizard.

Before closing MySQL Workbench, save the schema. Click **File** and then **Save** from the menu to save the reverse-engineered database as a MySQL Workbench file with the `.mwb` extension.

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, click **Show Logs**. This action will create a panel containing a list of messages, including any error messages that may have been generated. Click **Hide Logs** to close the panel.

If you plan 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 Schemas** area 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. To reuse a schema name, you must 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 or deselect individual objects and modify their direction during the synchronization. For example, you can 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.

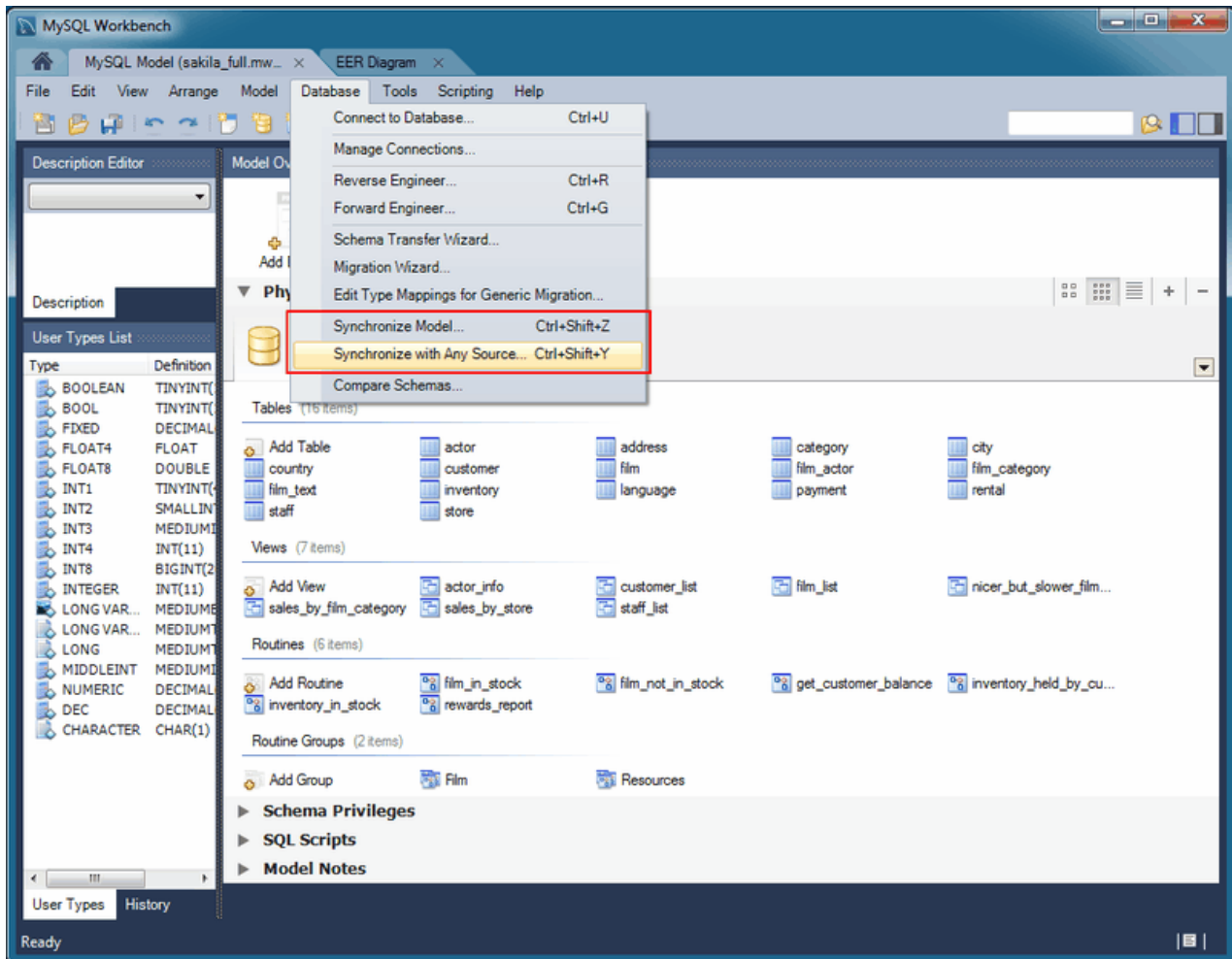


Note

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 5.6. See also [Section 3.2.4, “Modeling Preferences”](#).

To start, select **Synchronize With Any Source** from the **Database** navigation menu, as the following figure shows. 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 9.56 Start the Synchronization Wizard



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.

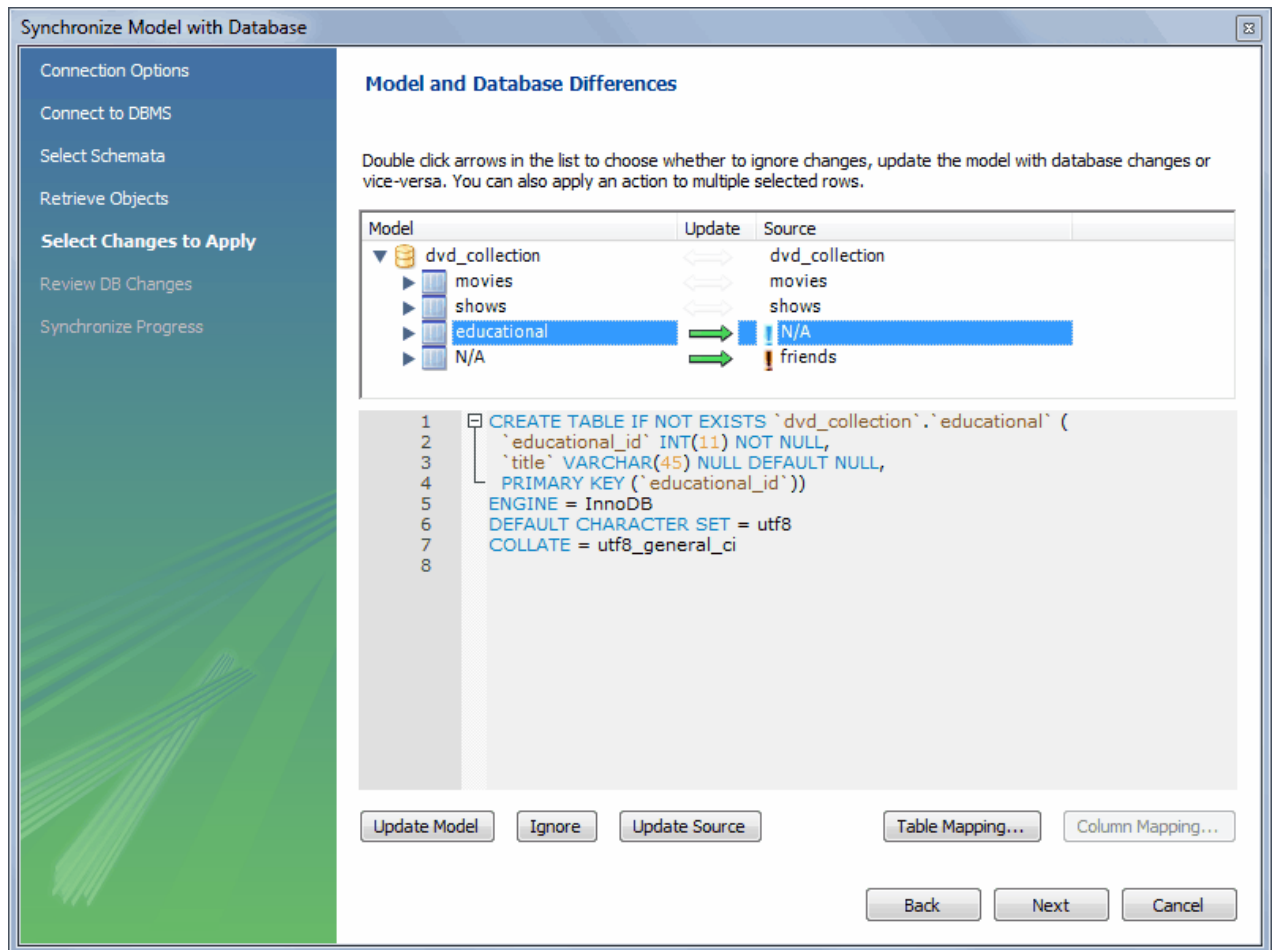
- Instead of executing the synchronization, you may generate an **ALTER Script File** to later perform the appropriate updates.
- Fine-tune how the synchronization will be performed by choosing the direction of each individual object or by configuring particular objects to be ignored.

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 that follow apply to both, unless stated otherwise.

Synchronize Model (with Database)

To start the wizard, open a model and select **Database, Synchronize Model** from the main menu. Follow the sequence of steps until you reach the **Select Changes to Apply** step, as the next figure shows.

Figure 9.57 Model and Database Differences

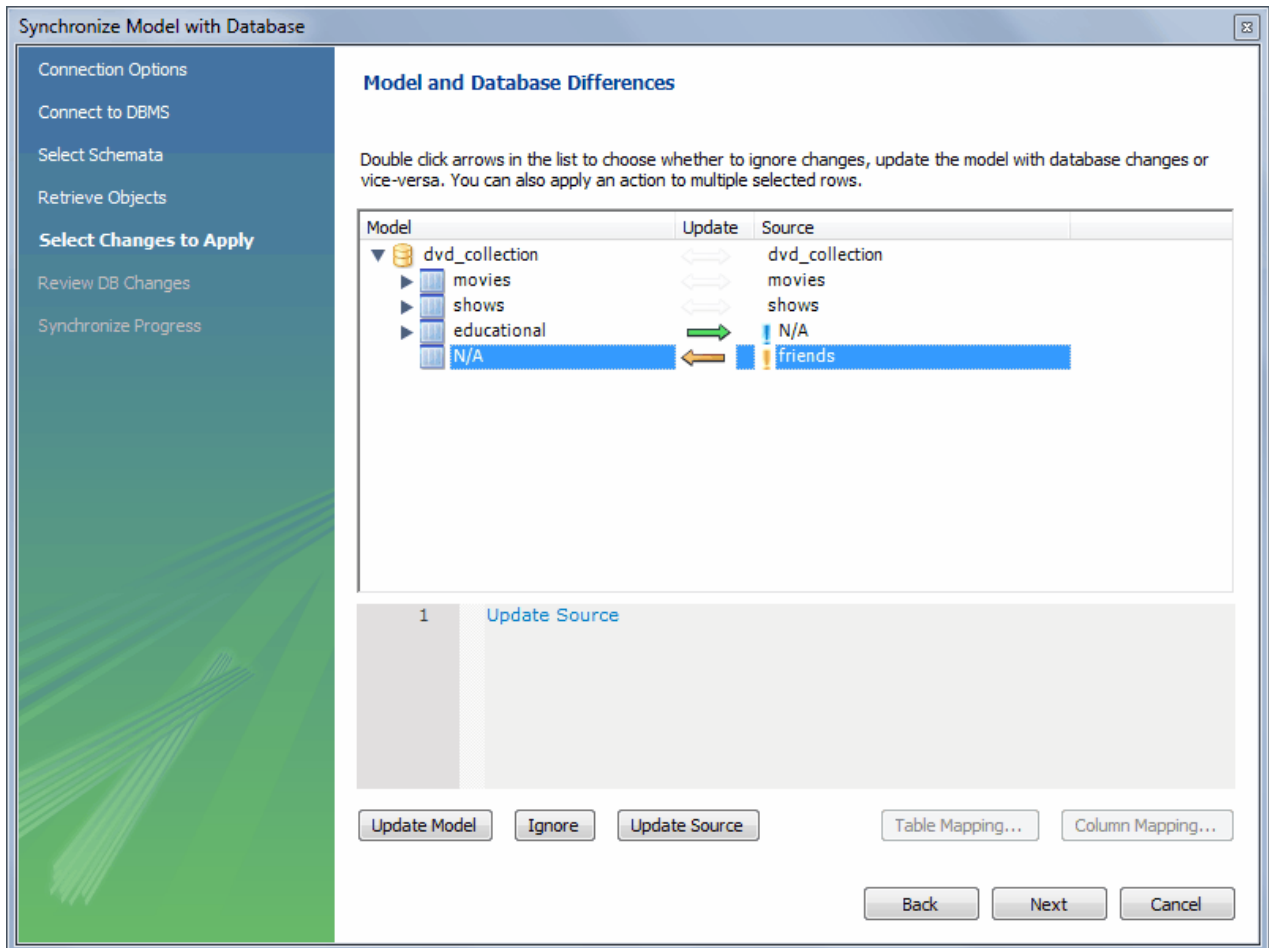


In the preceding example, the live database and model both have `movies` `shows` tables. In the MySQL Workbench, an additional table, `educational`, has been created in the model, but it lacks an equivalent in the live database. Further, `friends` exists in the live database, but it is not in the model. 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 previous figure.

The next figure shows an example of how the direction of synchronization can be changed.

Figure 9.58 Controlling Synchronization Direction

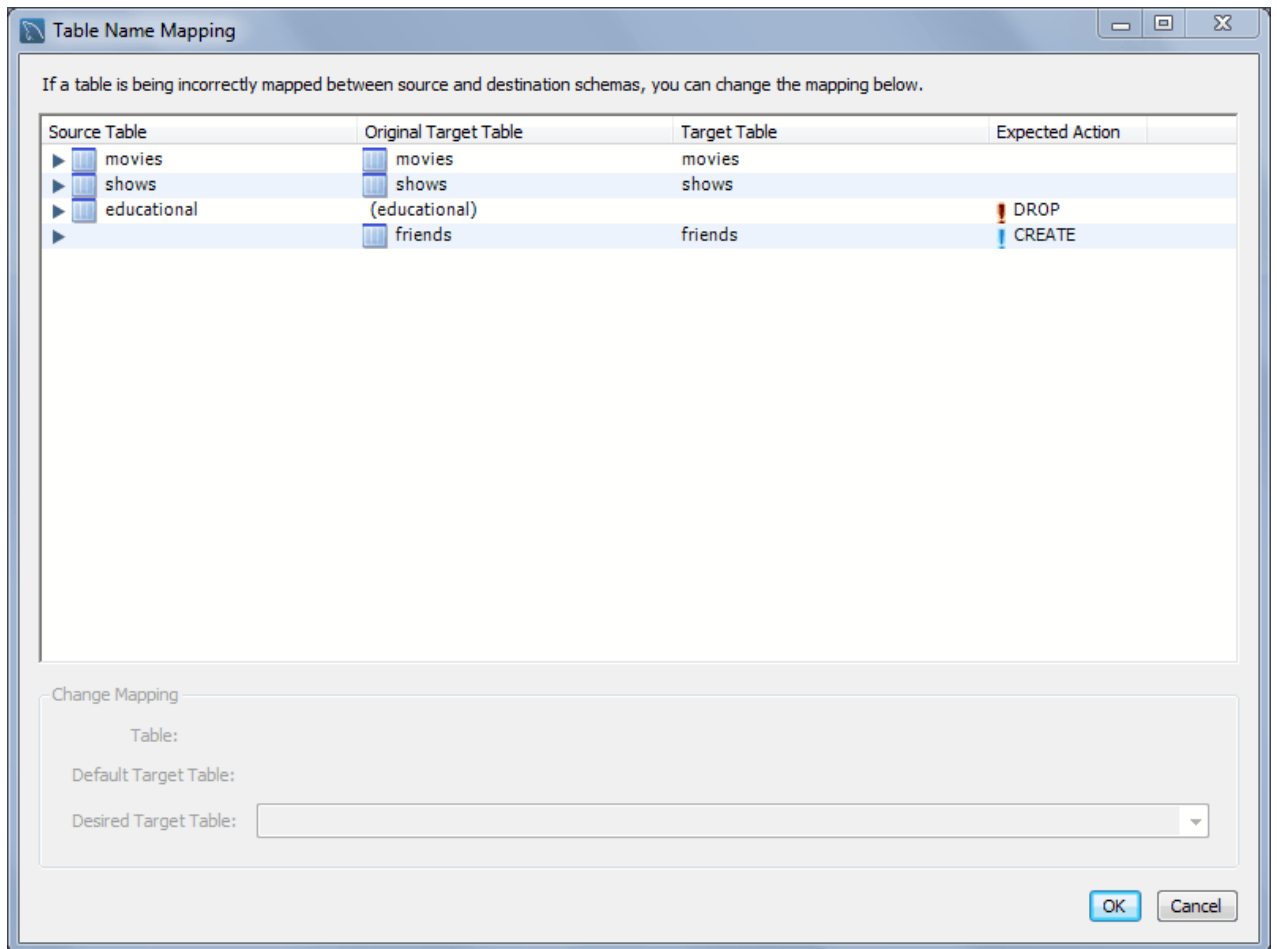


In this case, the synchronization direction has been changed so that rather than the default action of `friends` being dropped from the live database, it will be incorporated into the MySQL Workbench model. As before, `educational` table will be added to the live (source) database.

The three actions available actions are:

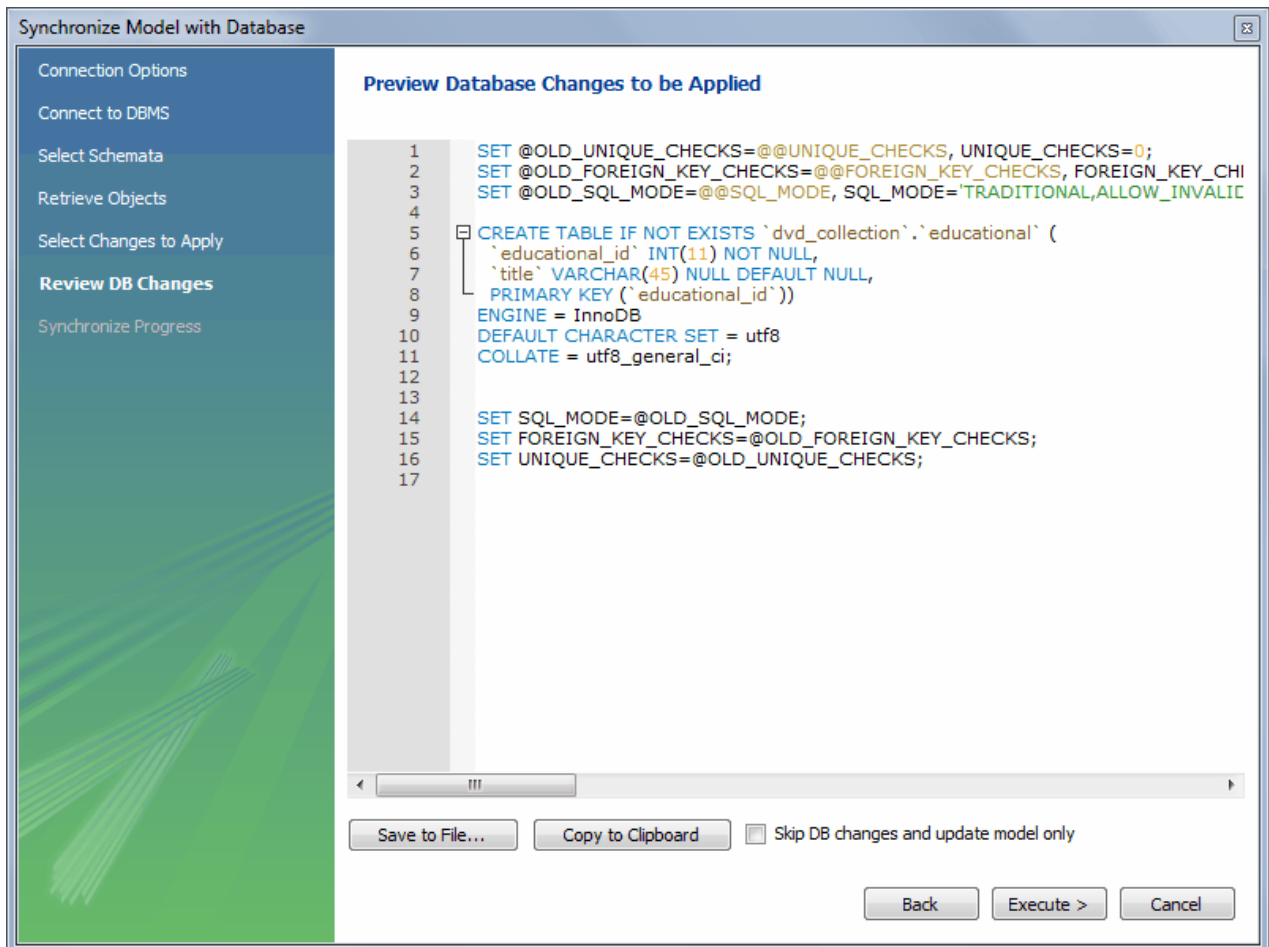
- **Update Model:** Causes the selected changes to be applied to the model, from the live database.
- **Ignore:** Causes the changes to be ignored. No synchronization will take place for those changes. This is designated with a double arrow that is crossed out.
- **Update Source:** Causes the changes to be applied only to the live database.

Clicking **Table Mapping** offers additional mapping options, as the following figure shows.

Figure 9.59 Table Mapping

Clicking **Next** reveals the SQL statement to perform the configured model and live database (source) synchronization. The following figure shows an example preview.

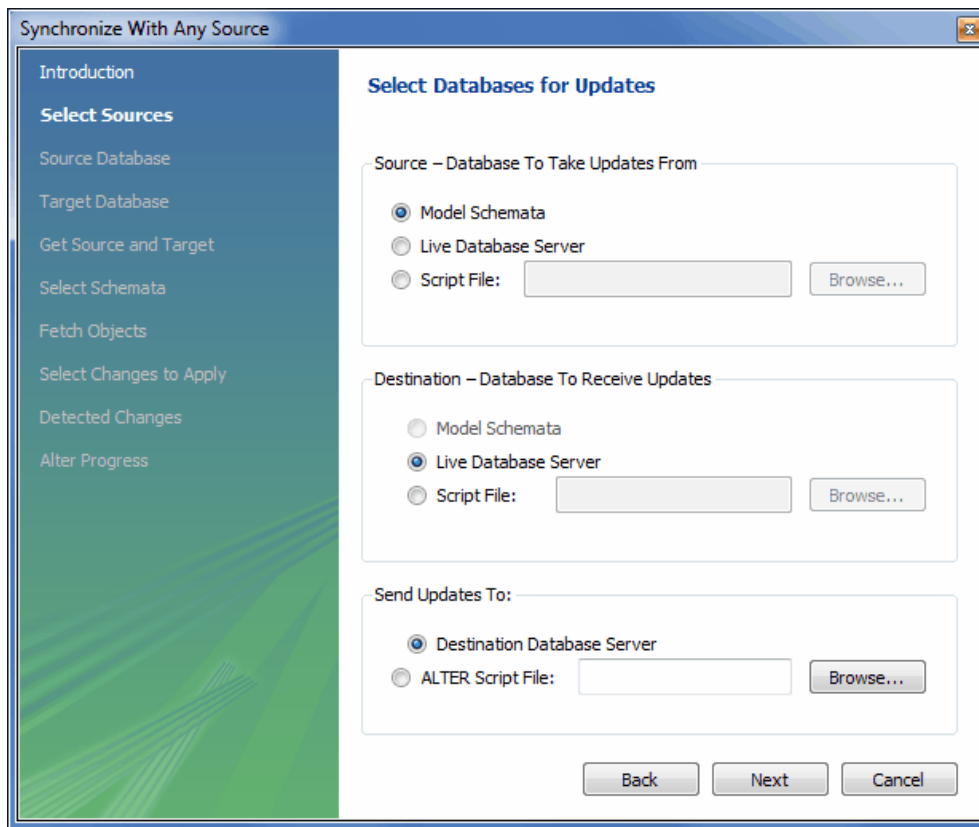
Figure 9.60 Previewing The Synchronization SQL Statement



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** and then **Synchronize With Any Source** from the main menu. The steps are similar to the [Synchronize Model](#) wizard, but with additional options to create SQL script files, use SQL script files, or both. The following figure shows the **Select Sources** settings.

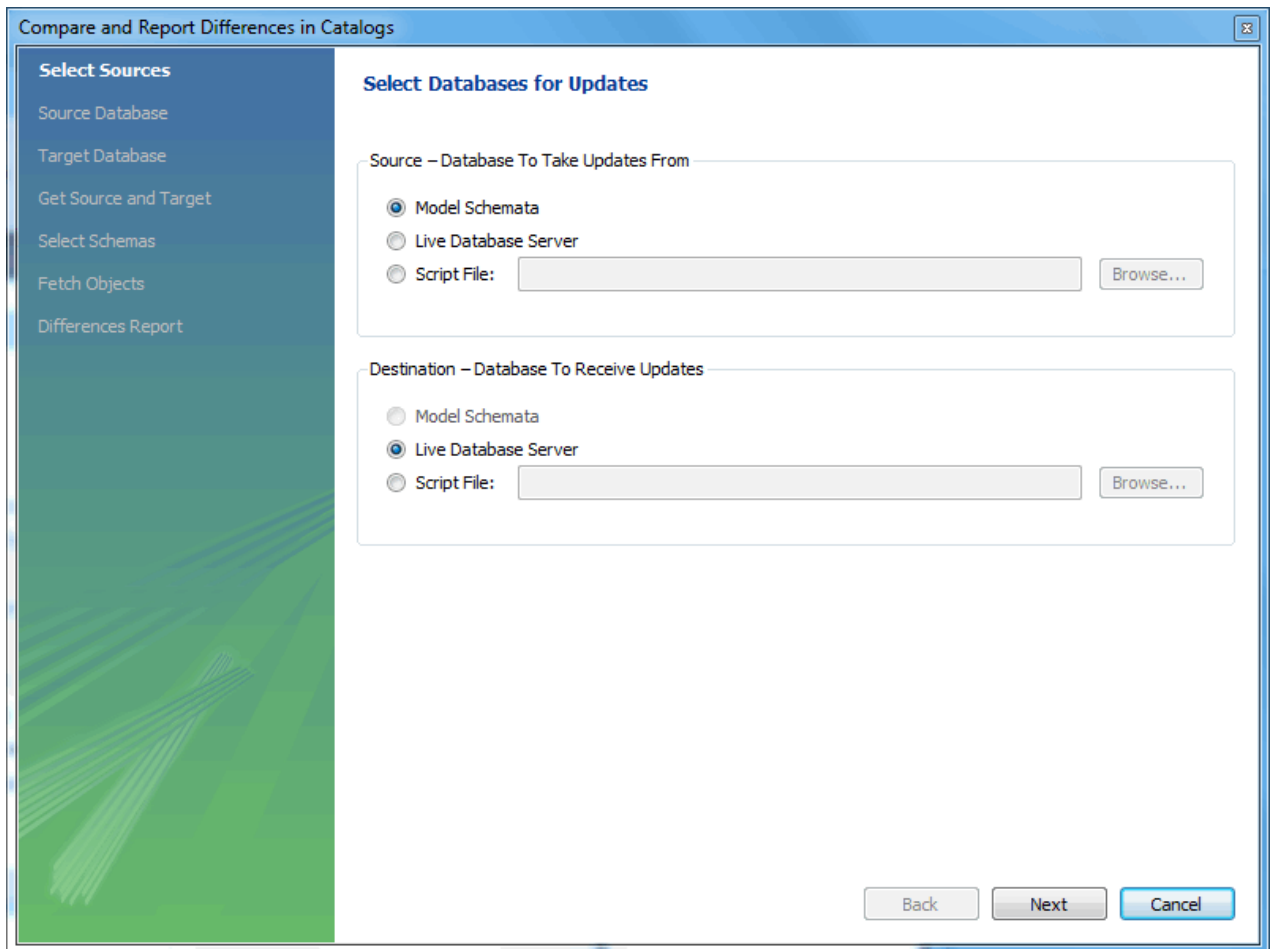
Figure 9.61 Synchronize With Any Source: Select Sources

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

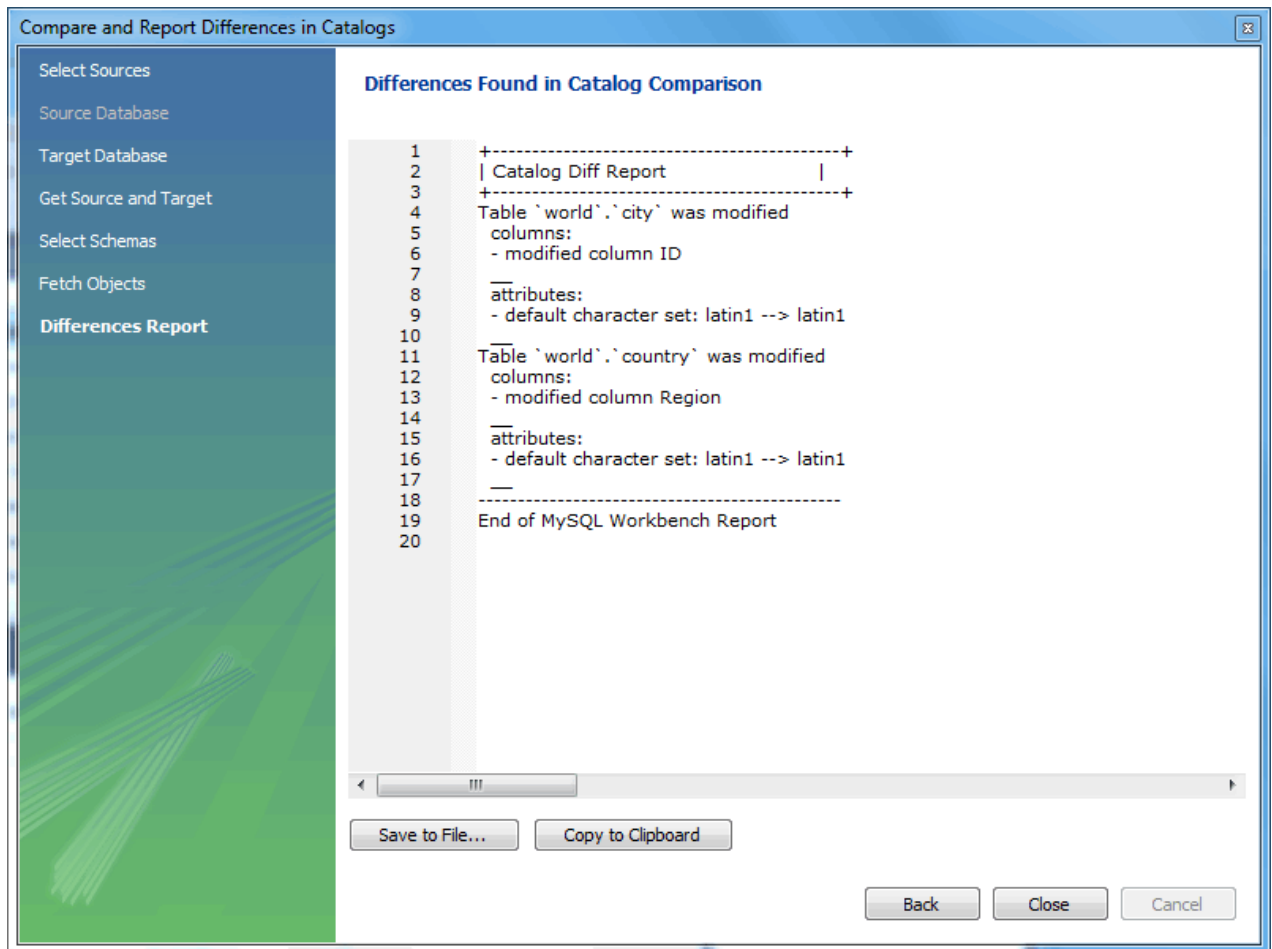
This facility enables you to create a report detailing the differences between your MySQL Workbench model, and a live database or script. Choose **Database, Compare Schemas** from the main menu to run the **Compare and Report Differences in Catalogs** wizard.

As the following figure shows, the first step in the wizard enables you to specify which catalogs to compare. For example, you may choose to compare your live database against your current MySQL Workbench model.

Figure 9.62 Catalog Sources Selection

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, as the next figure shows.

Figure 9.63 Catalog Differences Report

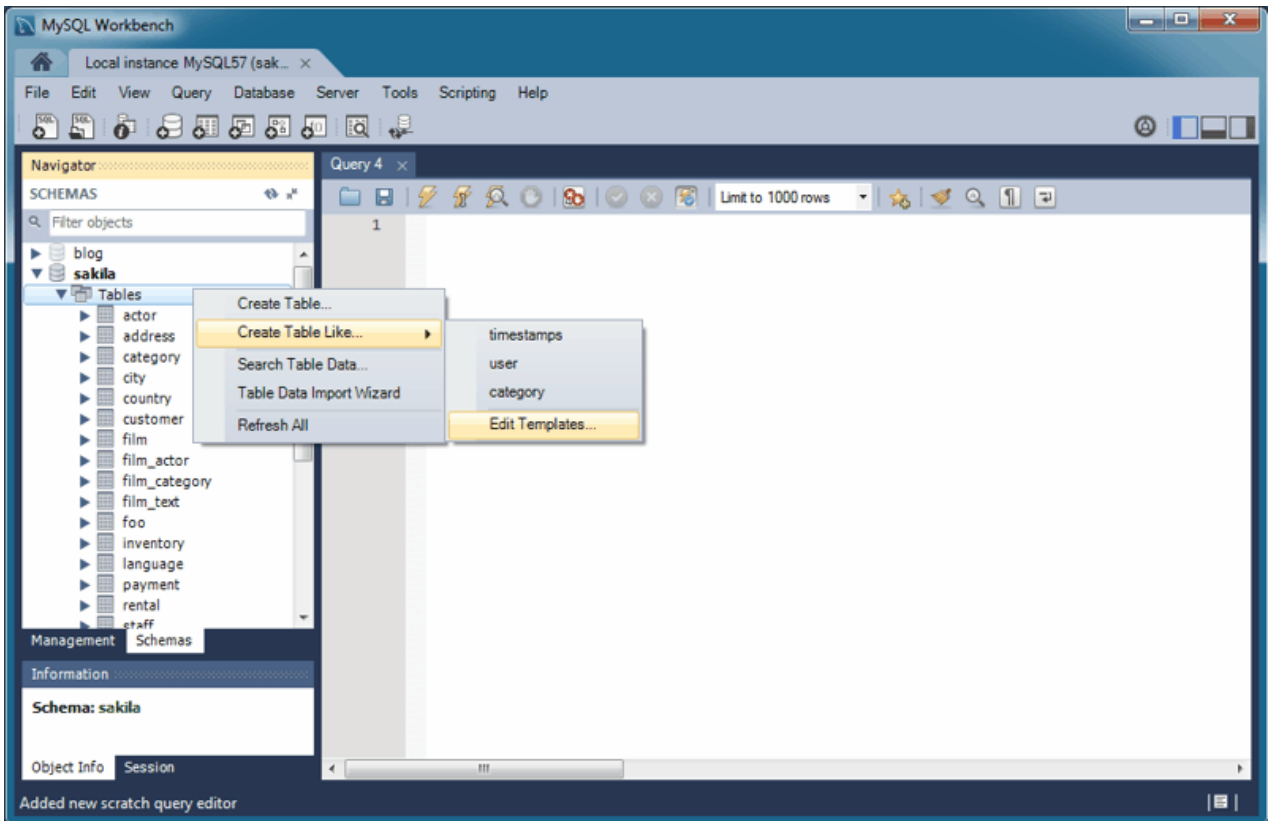


9.6 Table Templates

Define table templates with commonly used columns and settings to create new tables from either a live connection or while creating an EER model.

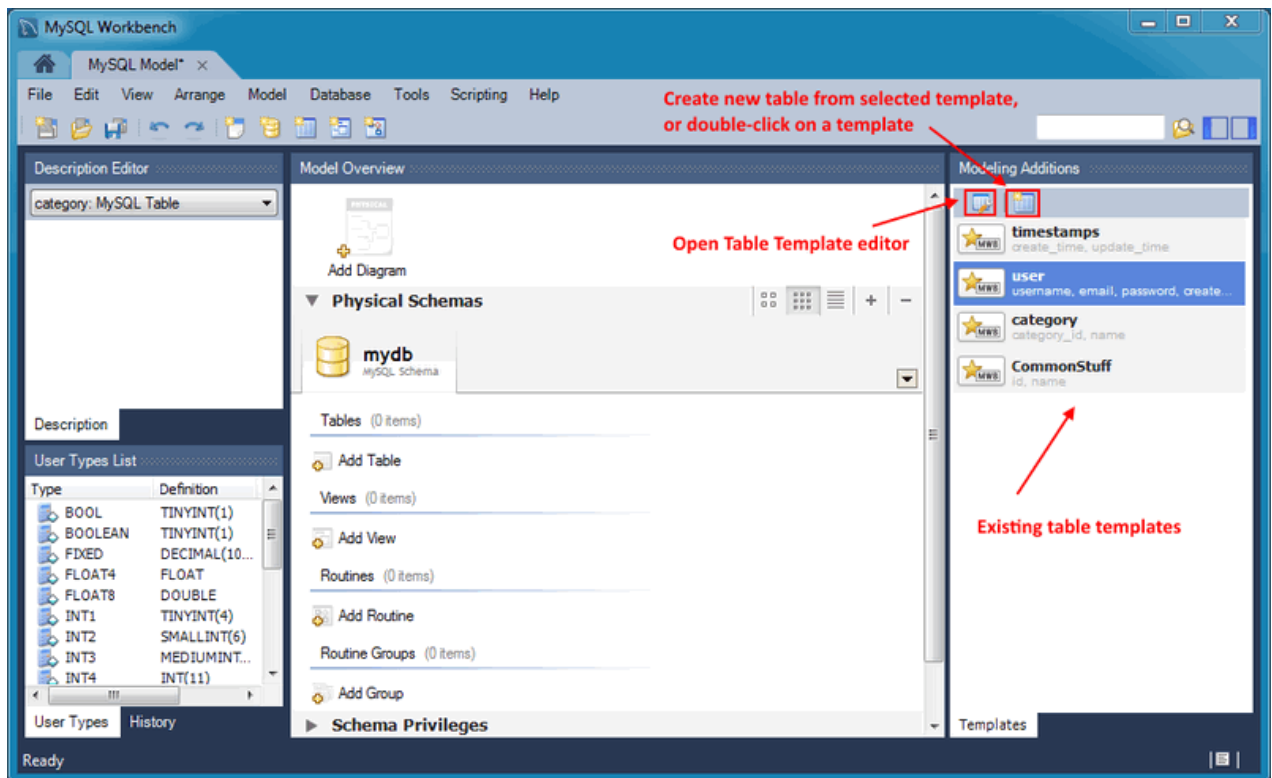
From the SQL editor, select **Create Table Like** from the **Tables** context menu, as shown in the next figure.

Figure 9.64 New Table Template: SQL Editor



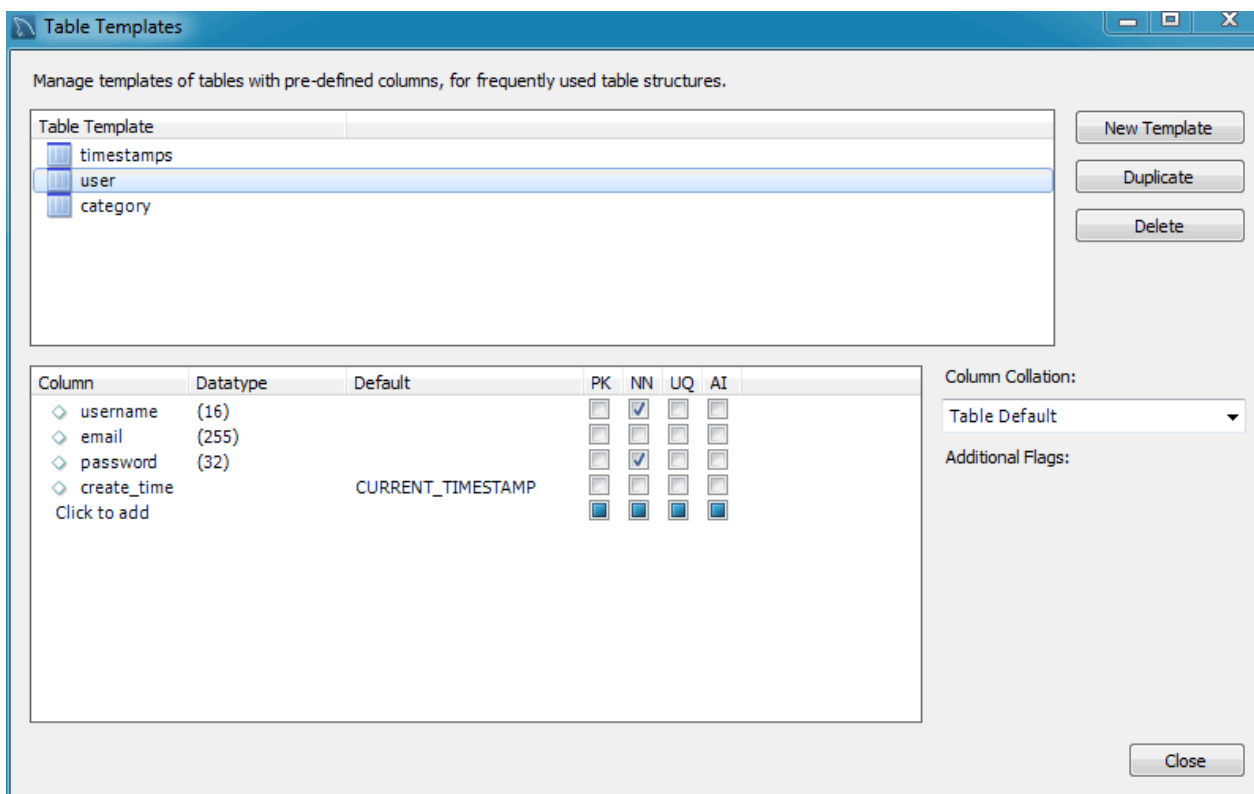
Or while modeling, click the "Open the table template editor" icon under **Modeling Additions**, as the following figure shows.

Figure 9.65 New Table Template: Modeling



After opening the **Table Templates** manager, make the adjustments and then click **Apply** to commit the changes. The following figure shows an example of column, data type, and default values that you can adjust for the user template.

Figure 9.66 Table Templates Manager



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 any of the following 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**

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 Schemas: {{SCHEMA_COUNT}}
```

In the generated output file, the variable `{{SCHEMA_COUNT}}` is replaced by the number of schemata in the model:

```
Total number of Schemas: 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 schemas, 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 9.2 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 9.2, "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 9.3 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
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

Marker text	Type	Data Dictionary or Parent Dictionary	Corresponding data
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
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

Marker text	Type	Data Dictionary or Parent Dictionary	Corresponding data
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
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

Marker text	Type	Data Dictionary or Parent Dictionary	Corresponding 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 is `C:\Program Files\MySQL\MySQL Workbench 8.0 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 two 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 Schemas: {{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

10.1	General Installation Requirements	346
10.1.1	ODBC Libraries	346
10.1.2	ODBC Drivers	347
10.2	Migration Overview	348
10.2.1	A Visual Guide to Performing a Database Migration	348
10.2.2	Migrating from Supported Databases	366
10.2.3	Migrating from Unsupported (Generic) Databases	367
10.3	Conceptual DBMS Equivalents	367
10.4	Microsoft Access Migration	369
10.5	Microsoft SQL Server Migration	386
10.5.1	Preparations	386
10.5.2	Drivers	386
10.5.3	Connection Setup	390
10.5.4	Microsoft SQL Server Type Mapping	393
10.6	PostgreSQL migration	394
10.6.1	Preparations	394
10.6.2	Drivers	395
10.6.3	Connection Setup	396
10.6.4	PostgreSQL Type Mapping	397
10.7	MySQL Migration	399
10.8	Using the MySQL Workbench Migration Wizard	403
10.8.1	Connecting to the Databases	403
10.8.2	Schema Retrieval and Selection	405
10.8.3	Reverse Engineering	407
10.8.4	Object Selection	408
10.8.5	Migration	409
10.8.6	Manual Editing	410
10.8.7	Target Creation Options	413
10.8.8	Schema Creation	414
10.8.9	Create Target Results	415
10.8.10	Data Transfer and Migration Setup	416
10.8.11	Bulk Data Transfer	417
10.8.12	Migration Report	418
10.9	MySQL Workbench Migration Wizard FAQ	419

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, macOS, and Microsoft Windows.

This is not an exhaustive list. The following sections discuss these and additional migration capabilities.

Setup 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, macOS, and Microsoft

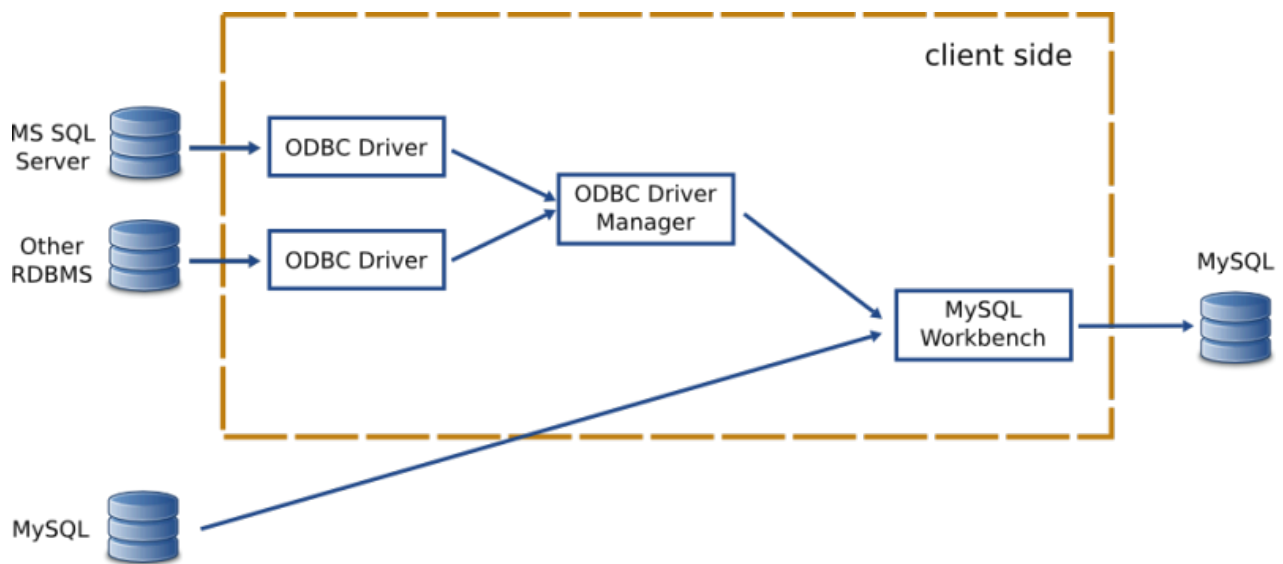
Windows, and the [Database Product Specific Notes](#) section that references setup conditions for each RDBMS.

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 macOS.

The following figure shows the general components involved in an ODBC connection: An ODBC driver for MySQL server and the other relational database management system, the ODBC Driver Manager, and MySQL Workbench.

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 macOS 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 macOS, 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 macOS, 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`.
- **macOS:** 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 macOS, 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 an 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.

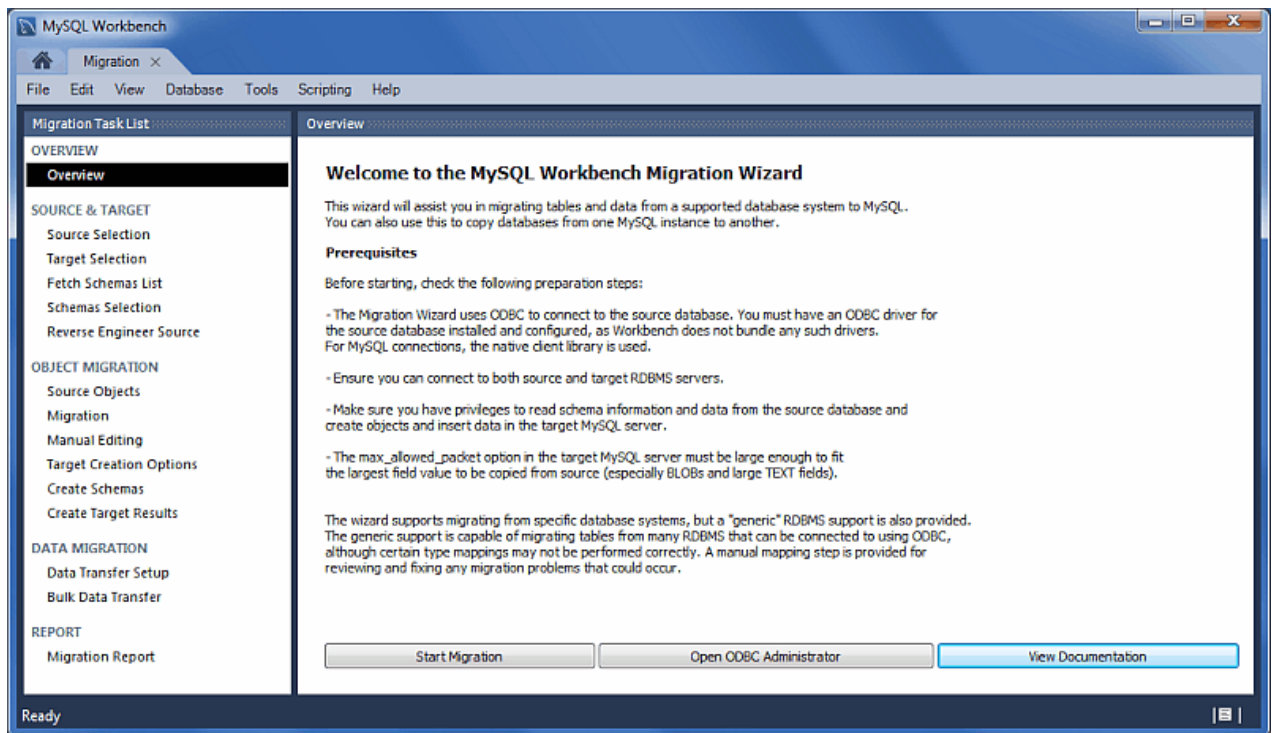
10.2.1 A Visual Guide to Performing a Database Migration

This example migrates a Microsoft SQL Server database to MySQL and includes an image for each step.

From MySQL Workbench, choose **Database** and then **Migrate** to open the migration wizard and display the migration wizard overview (see the figure that follows).

Overview

Figure 10.2 MySQL Workbench migration: Overview



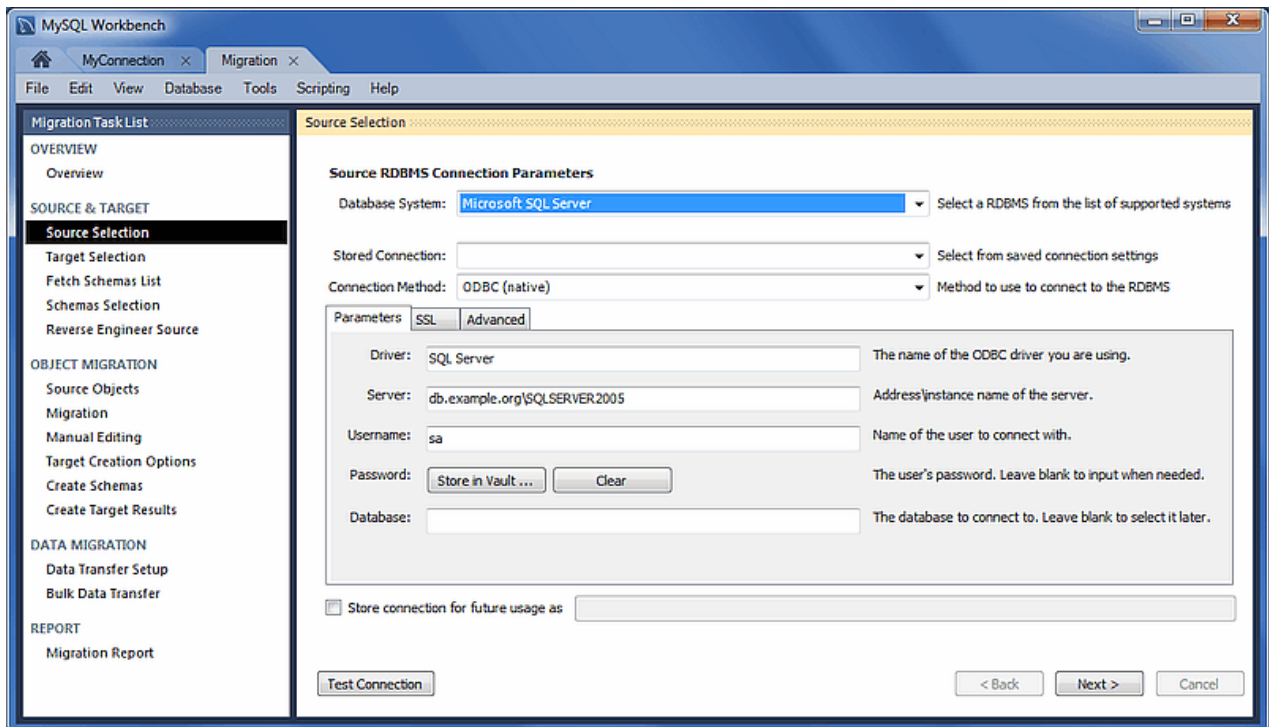
It describes the prerequisites and requirements that should be understood before proceeding further. The **Open ODBC Administrator** option will load `odbcad32.exe`, and is used to confirm that the ODBC Driver for SQL Server is installed, and to make configuration changes if needed.

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. The following figure shows an example of the connection session.

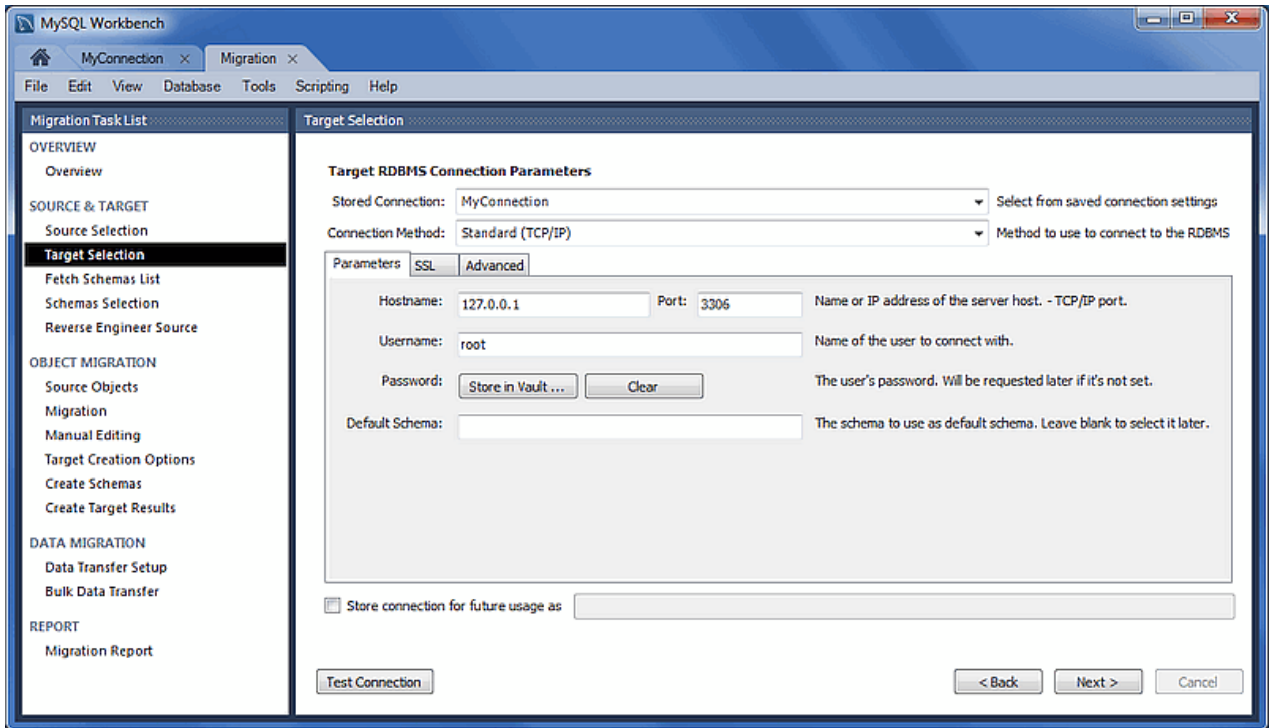
Figure 10.3 MySQL Workbench migration: Source Selection (Parameters)



Target Selection

The target is the MySQL database that will contain the newly migrated database (see the figure that follows). The current Workbench MySQL connections will be available here, or you can choose **Manage DB Connections** to create a new connection.

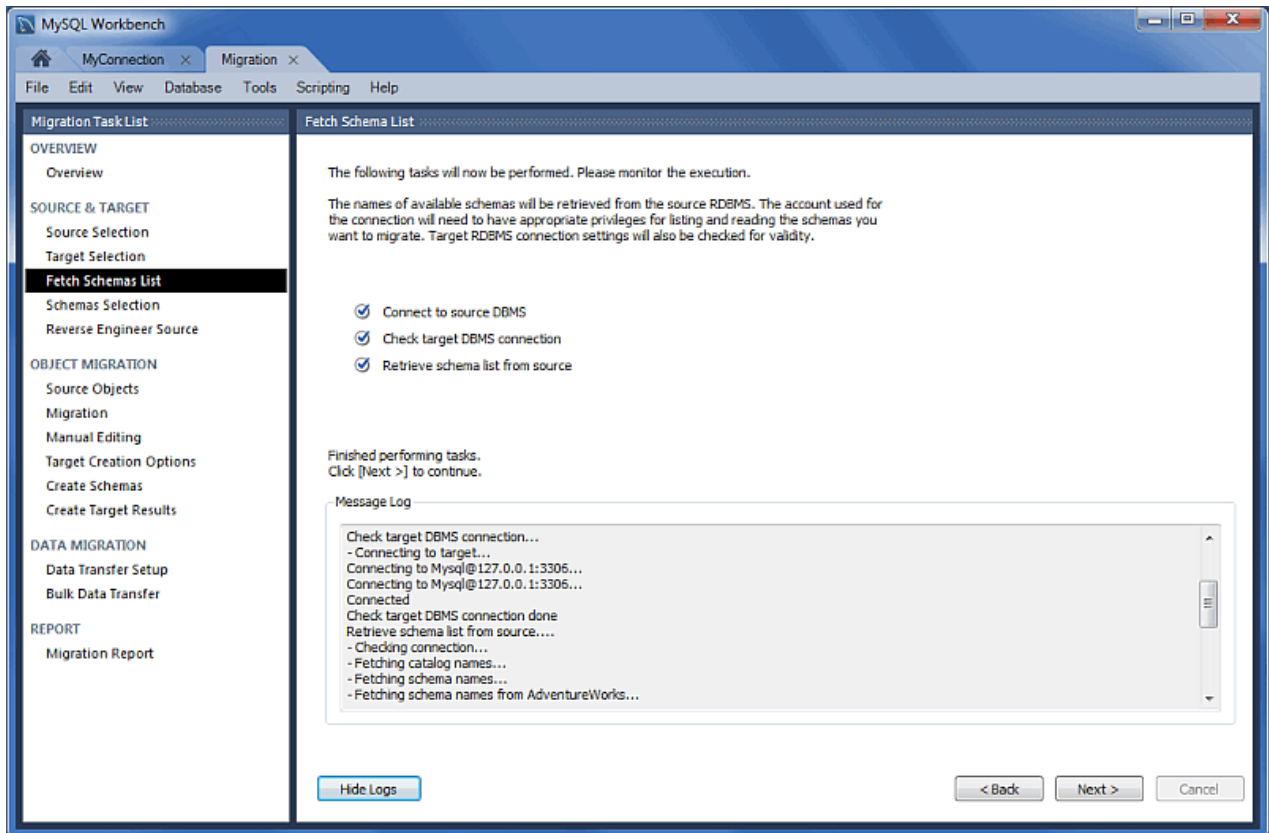
Figure 10.4 MySQL Workbench migration: Target selection



Fetch Schemas List

The Schemas list is retrieved from both the source and target RDBMS (see the figure that follows). This is an automated and informational step that reports connection related errors and/or general log information. Click **Next** to continue.

Figure 10.5 MySQL Workbench migration: Fetch Schemas List



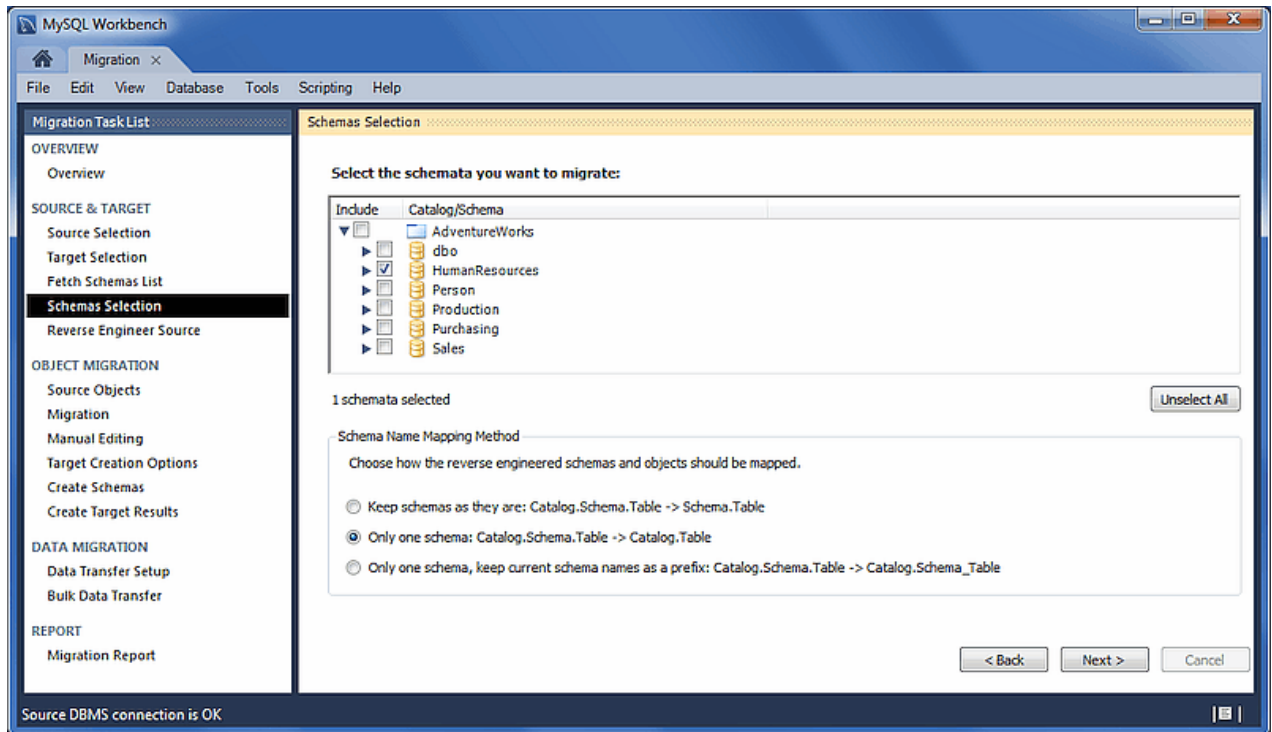
Schemas Selection

Choose the schemas you want to migrate.

"Schema Name Mapping Method" options while migrating Microsoft SQL Server:

- Keep schemas 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. The following figure shows an example of this mapping method.
- Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema_table: Preserves the schema name as a prefix.

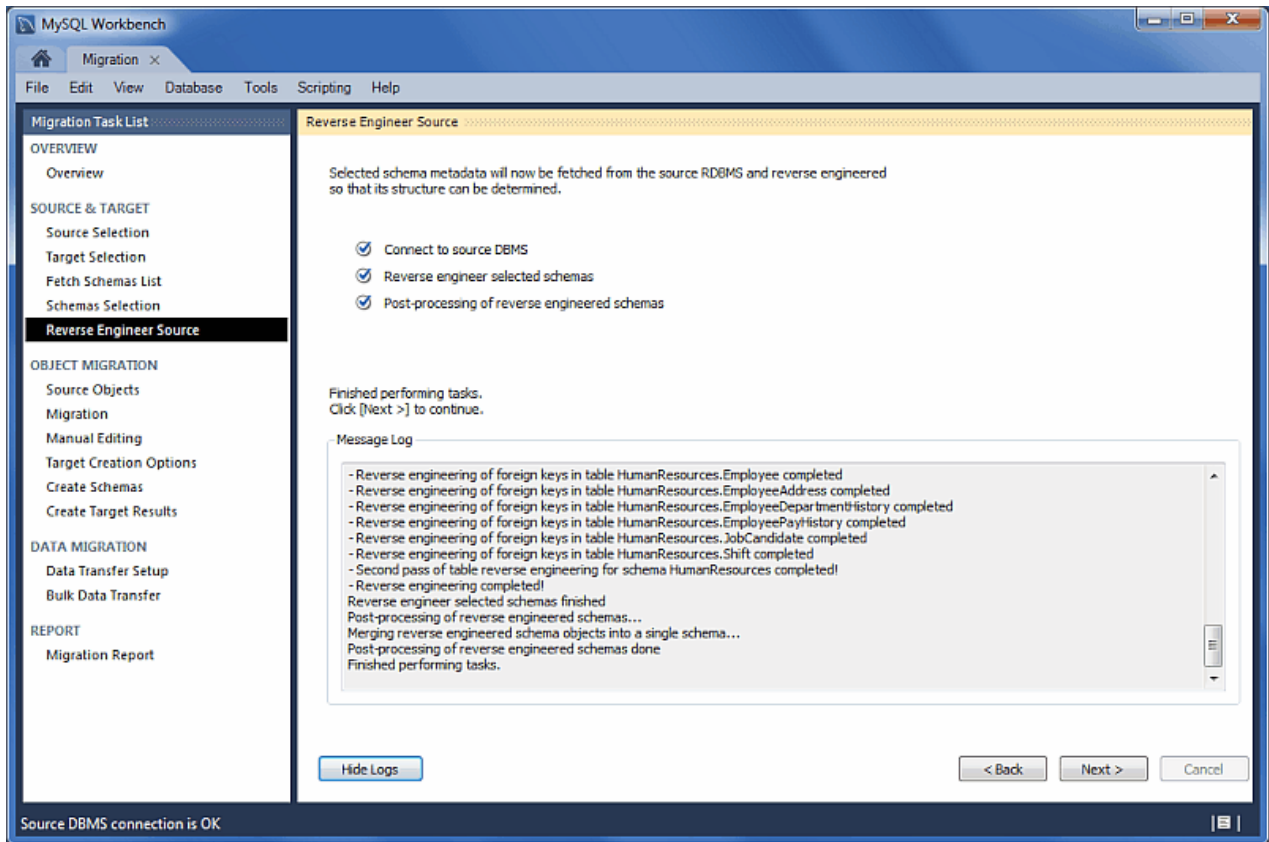
Figure 10.6 MySQL Workbench migration: Schemas Selection



Reverse Engineer Source

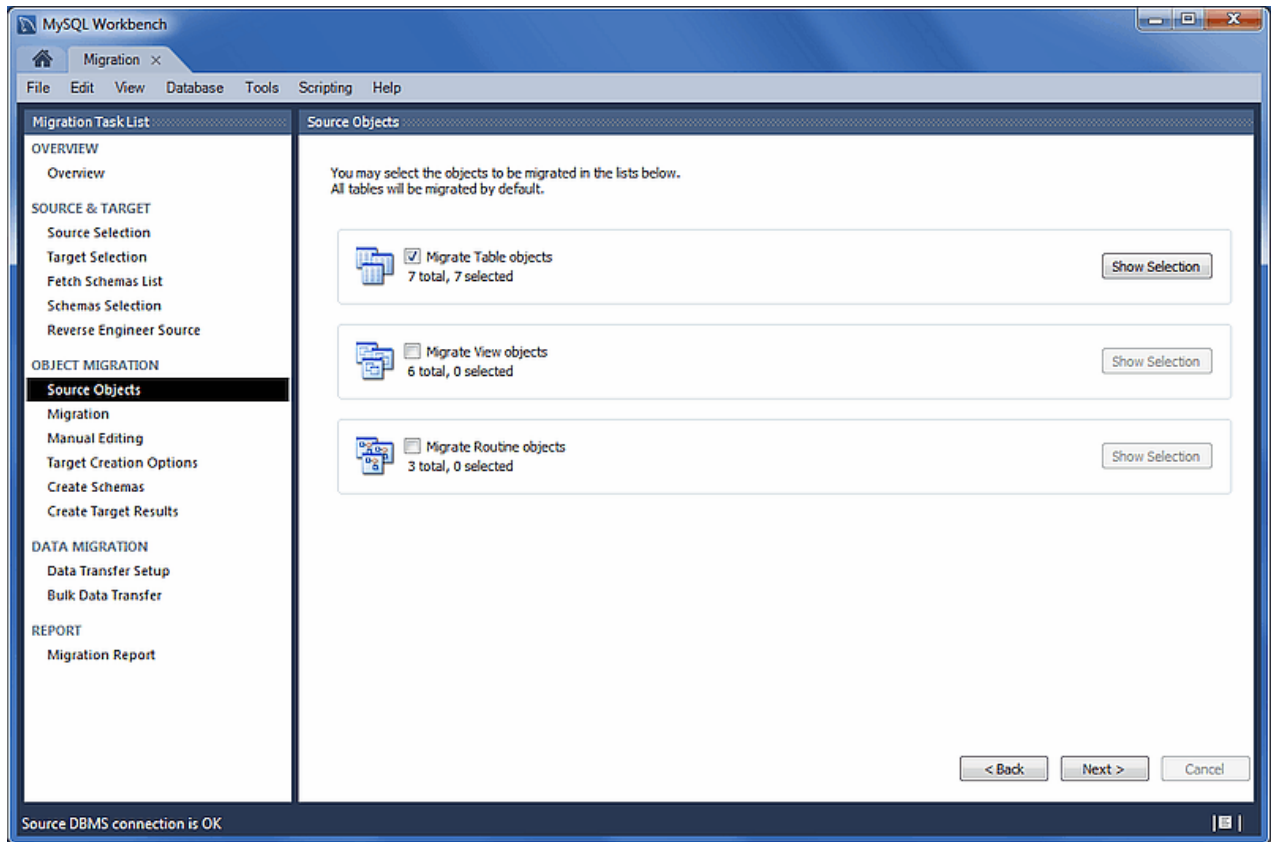
The source metadata is fetched from the source RDBMS and then is reverse-engineered. This is an automated and informational step that reports related errors, general log information, or both (see the figure that follows). View the logs and then click **Next** to continue.

Figure 10.7 MySQL Workbench migration: Reverse Engineer Source



Source Objects

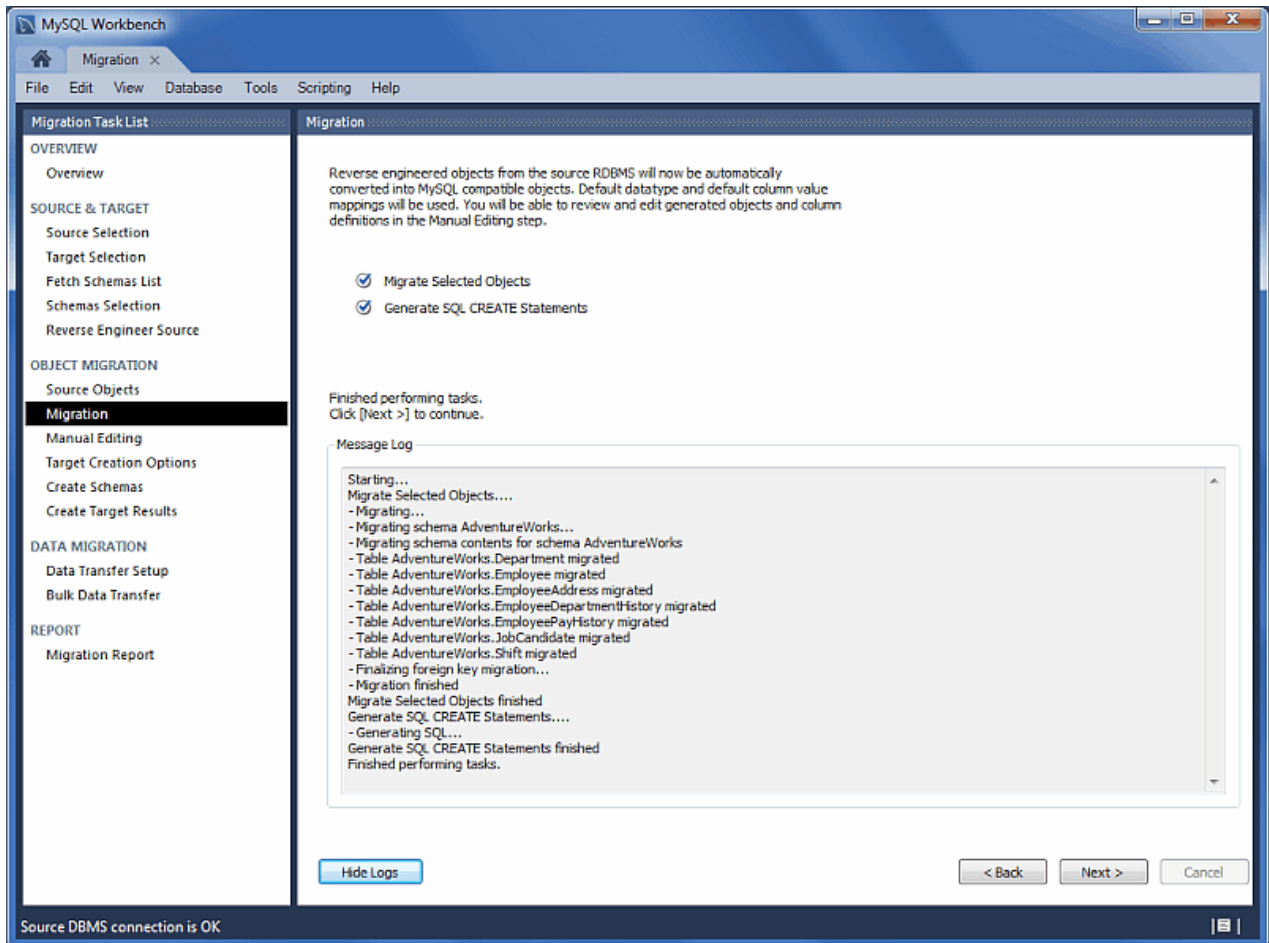
The discovered objects from the **Reverse Engineer Source** stage are revealed and made available. As the next figure shows, the results include Table, View, and Routine objects, with only the Table objects being selected by default.

Figure 10.8 MySQL Workbench migration: Source Objects

Migration

The migration process now converts the selected objects into MySQL compatible objects (see the figure that follows). View the logs and then proceed.

Figure 10.9 MySQL Workbench migration: Migration

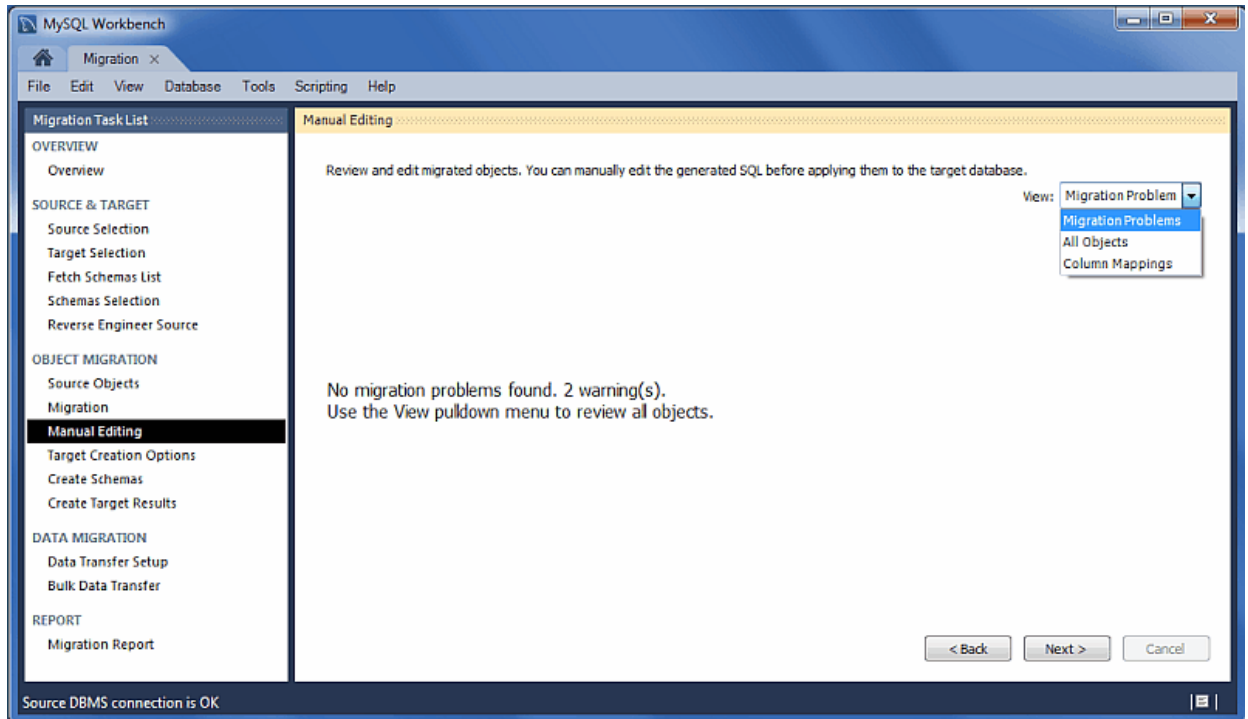


Manual Editing

There are three sections to edit, which are selected by using 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.

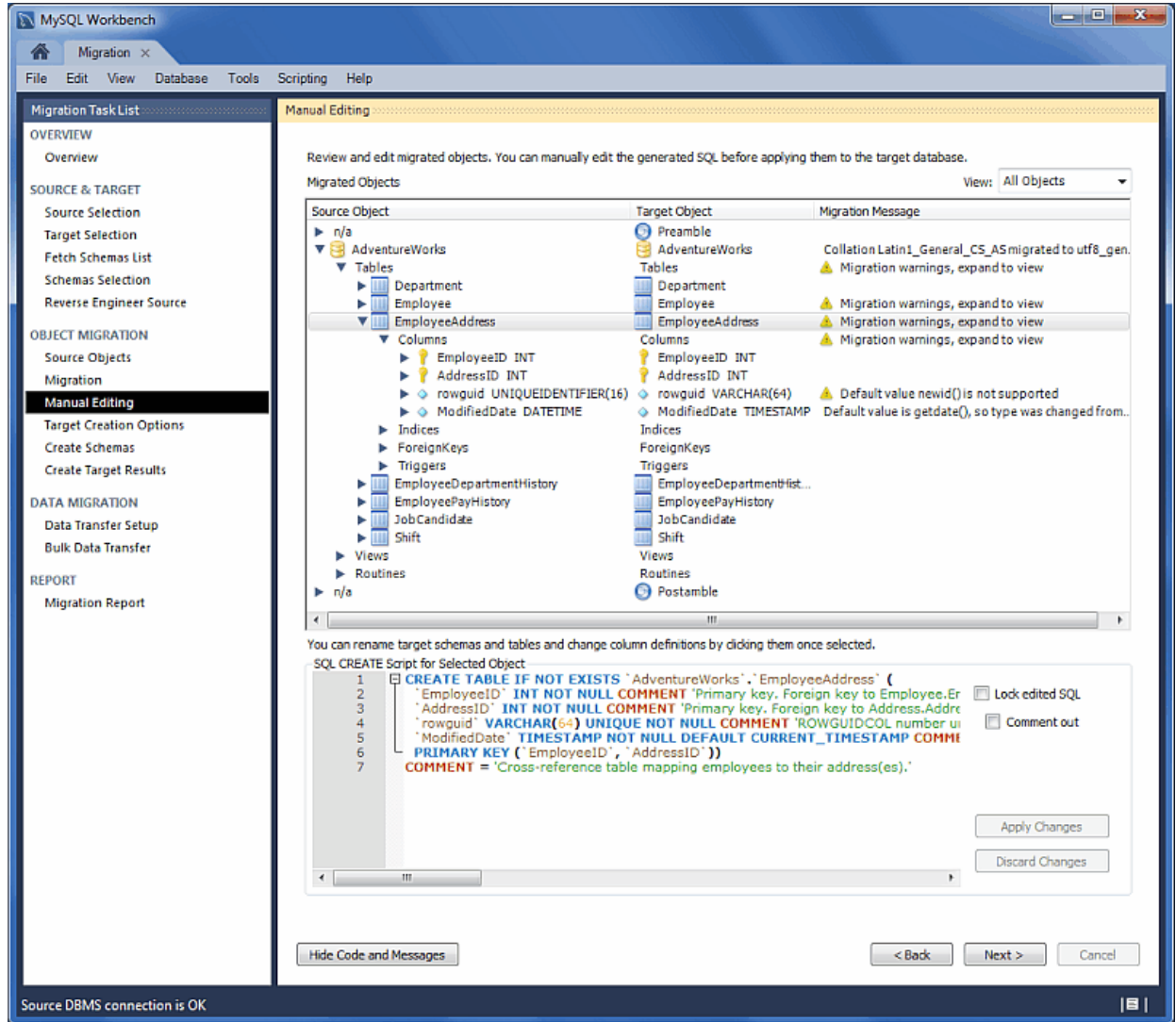
- **Migration Problems:** This will either report problems or display "No mapping problems found." It is an informational screen. The following figure shows an example of this type of message.

Figure 10.10 MySQL Workbench migration: Manual Editing (Migration Problems)



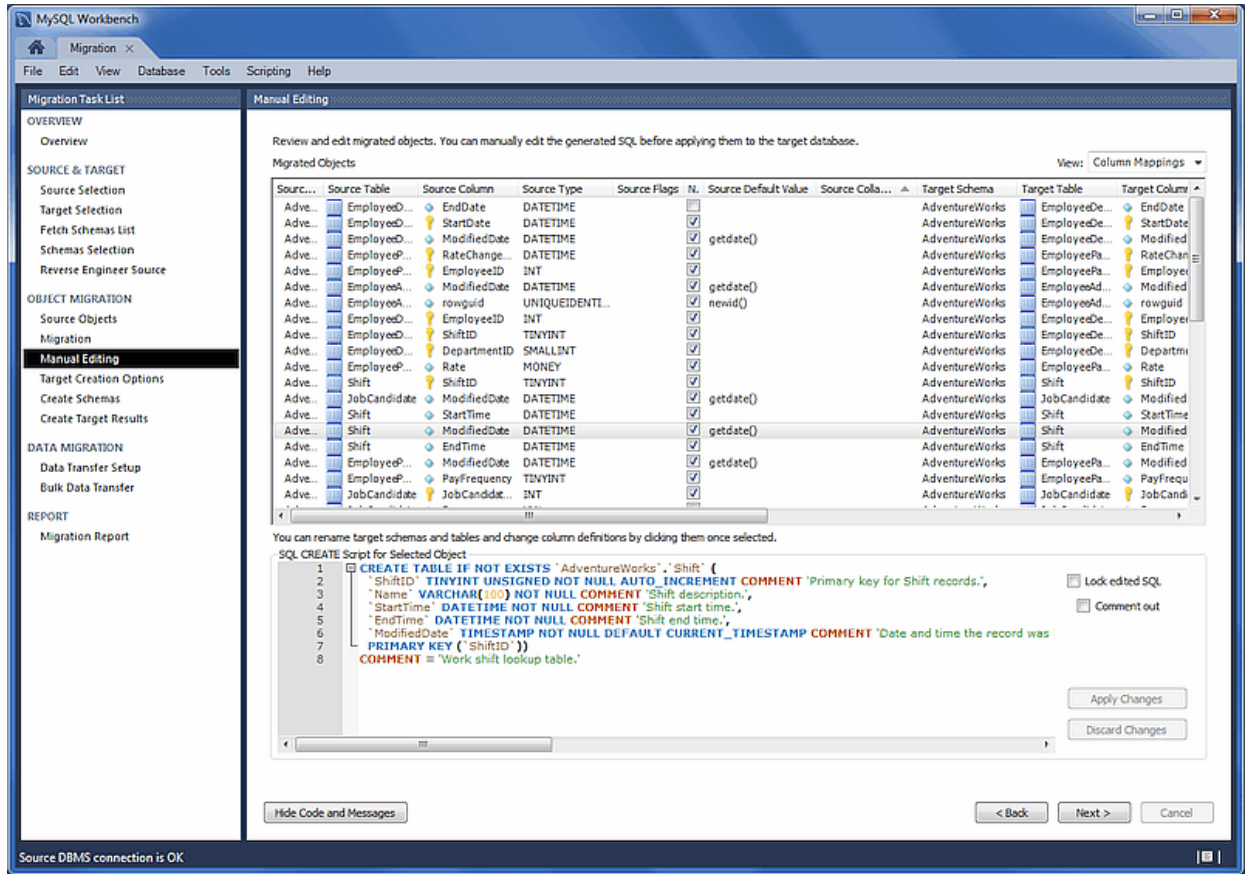
- **All Objects:** An object view that allows you to view and edit the object definitions. Double-click on a row to modify a target objects name. The following figure shows an example of this type of message.

Figure 10.11 MySQL Workbench migration: Manual Editing (All Objects)



- **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. The following figure shows an example of this type of message.

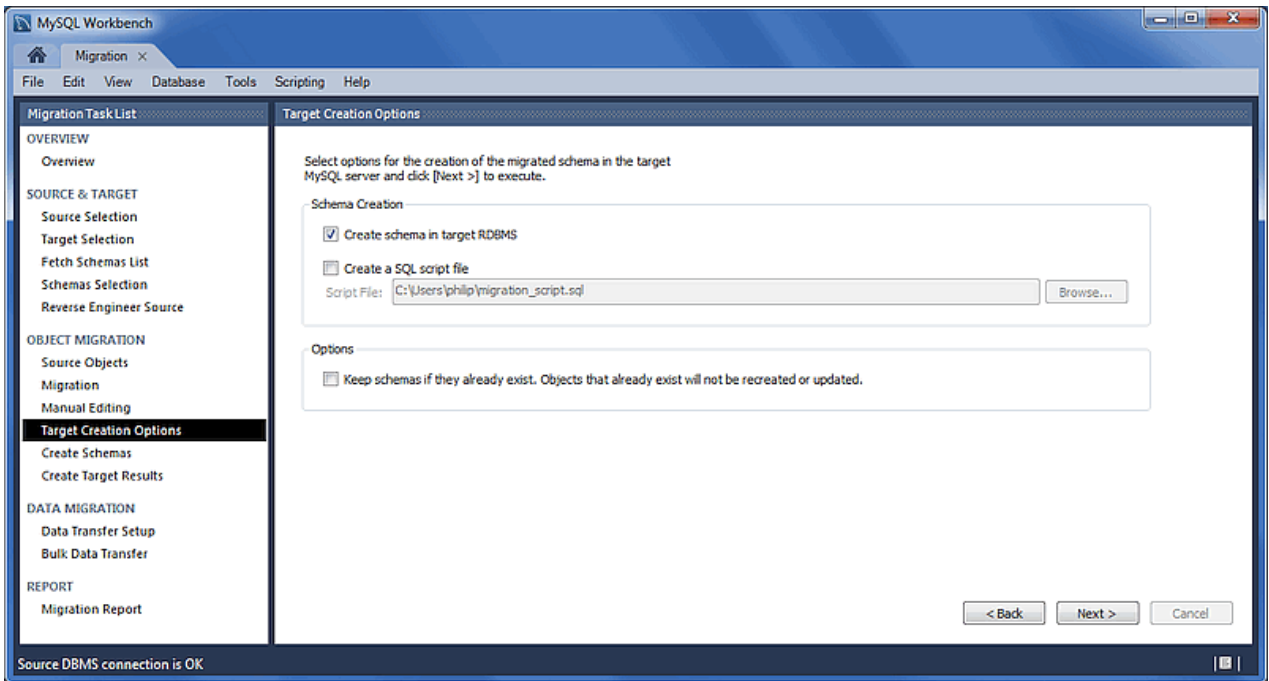
Figure 10.12 MySQL Workbench migration: Manual Editing (Column Mappings)



Target Creation Options

The schema may be created by either adding it to the target RDBMS, creating an SQL script file, or both. The following figure shows the target creation options.

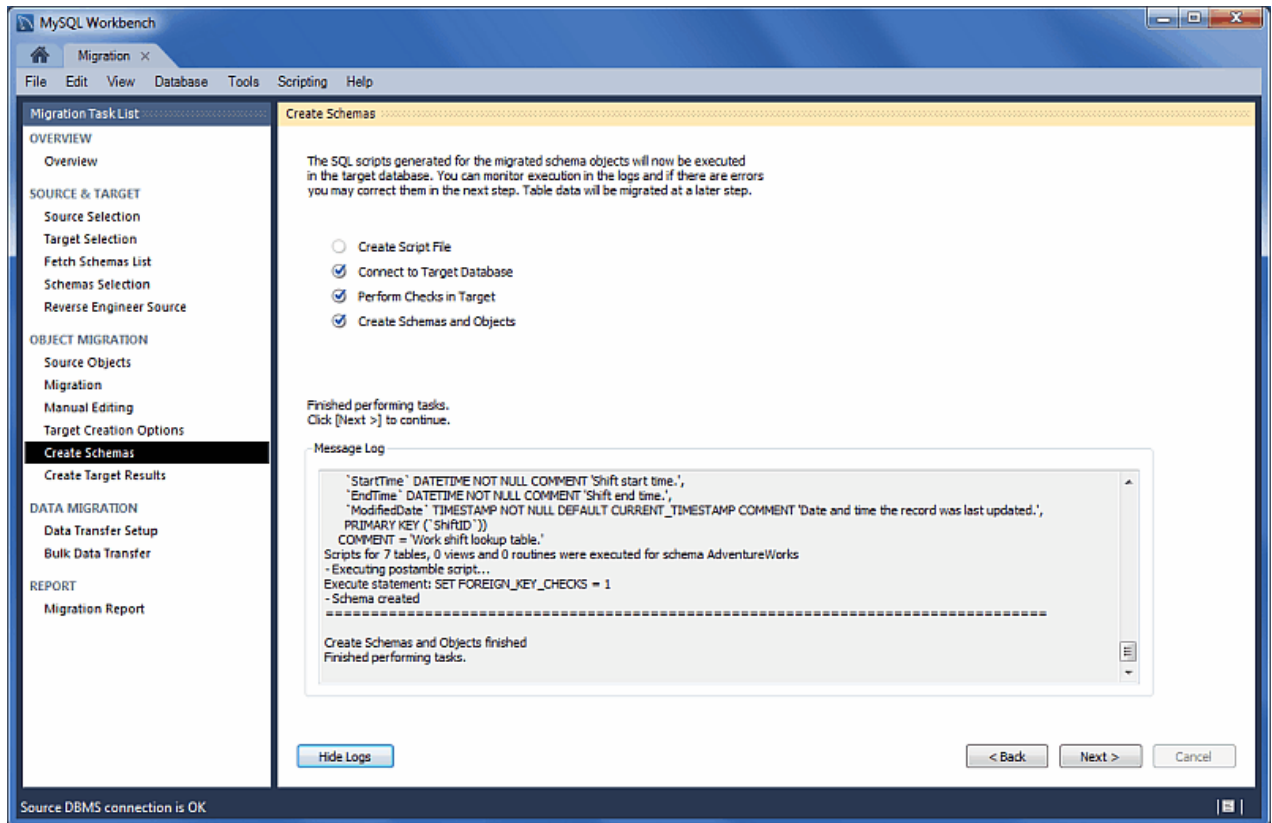
Figure 10.13 MySQL Workbench migration: Target Creation Options



Create Schema

Now the schema is created (see the figure that follows). The complete log is also available here.

Figure 10.14 MySQL Workbench migration: Create Schema



Create Target Results

The generated objects in this example are listed in the figure that follows, along with the error messages if any exist.

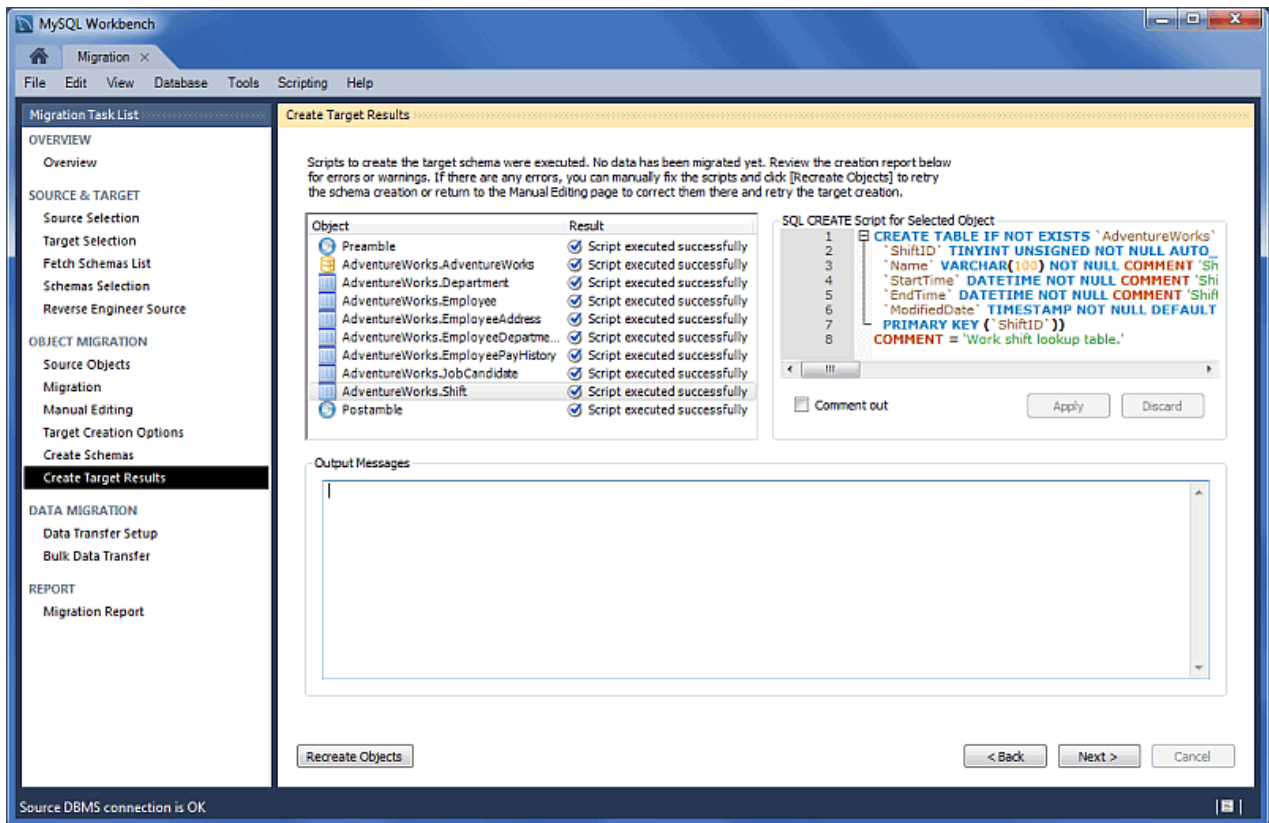
The migration code may also be viewed and edited here. To make changes, select an object, edit the query code and then click **Apply**. Repeat this process for each object that will be edited. Finally, click **Recreate Objects** to save the results.



Note

The **Recreate Objects** operation is required to save any changes here. It will then execute the previous migration step (**Create Schema**) with the modified code, and then continue the migration process. This also means that the previously saved schema will be dropped.

Figure 10.15 MySQL Workbench Migration: Create Target Results



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 method (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.



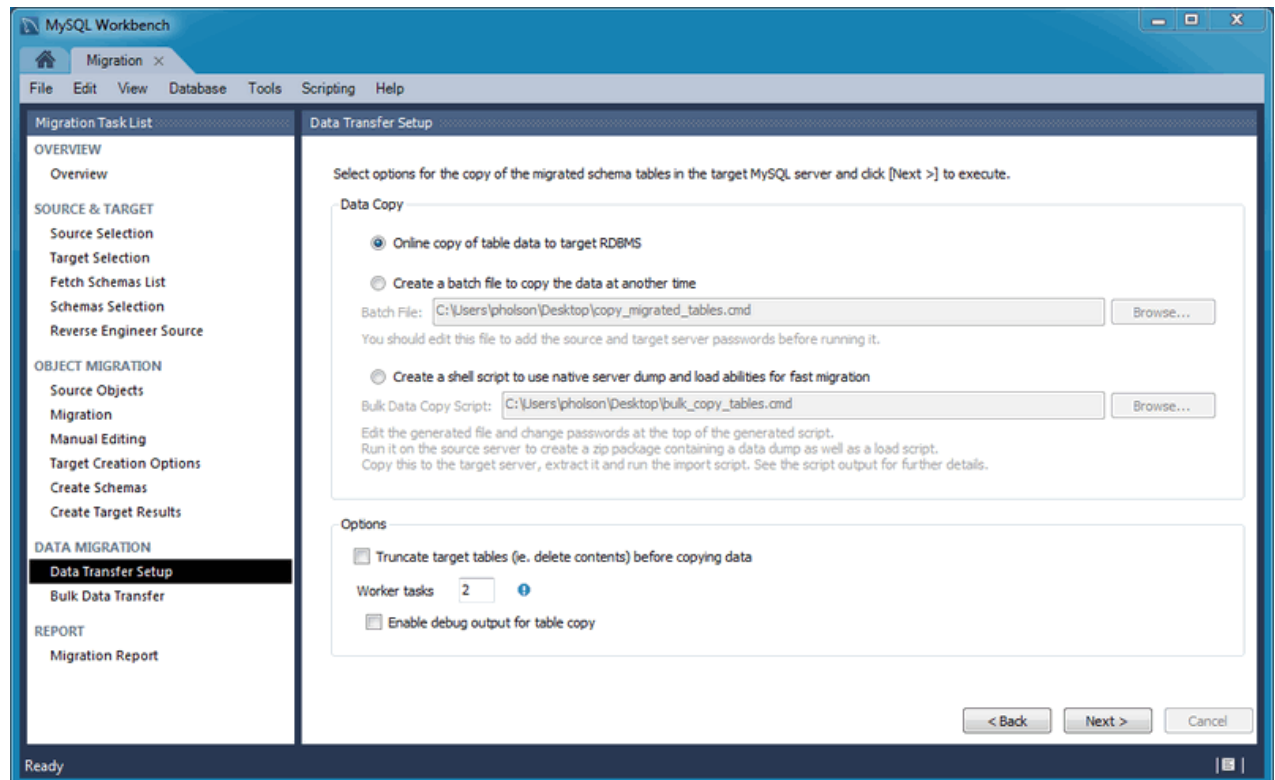
Note

This option was added in MySQL Workbench 6.3.0.

Options (see the figure that follows for an example):

- 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.

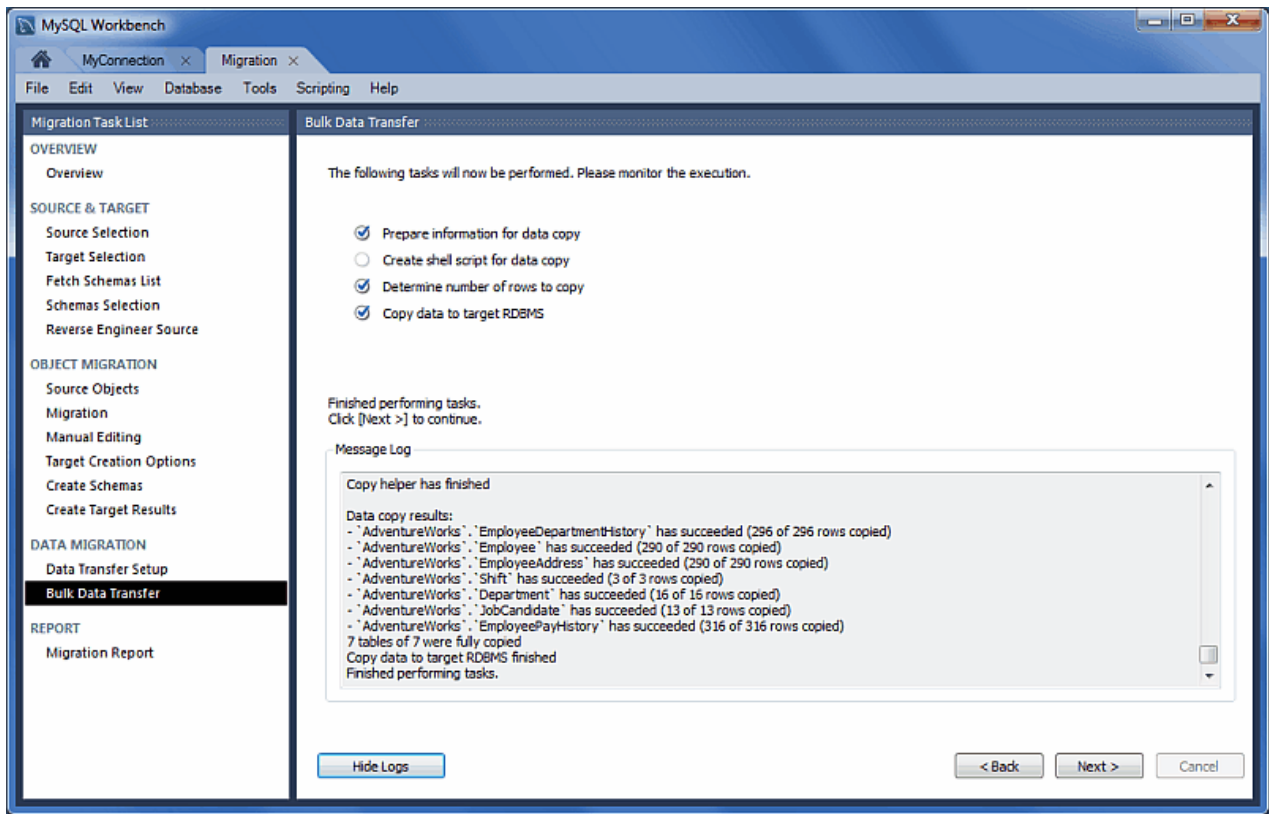
Figure 10.16 MySQL Workbench Migration: Data Transfer Setup



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 (see the figure that follows).

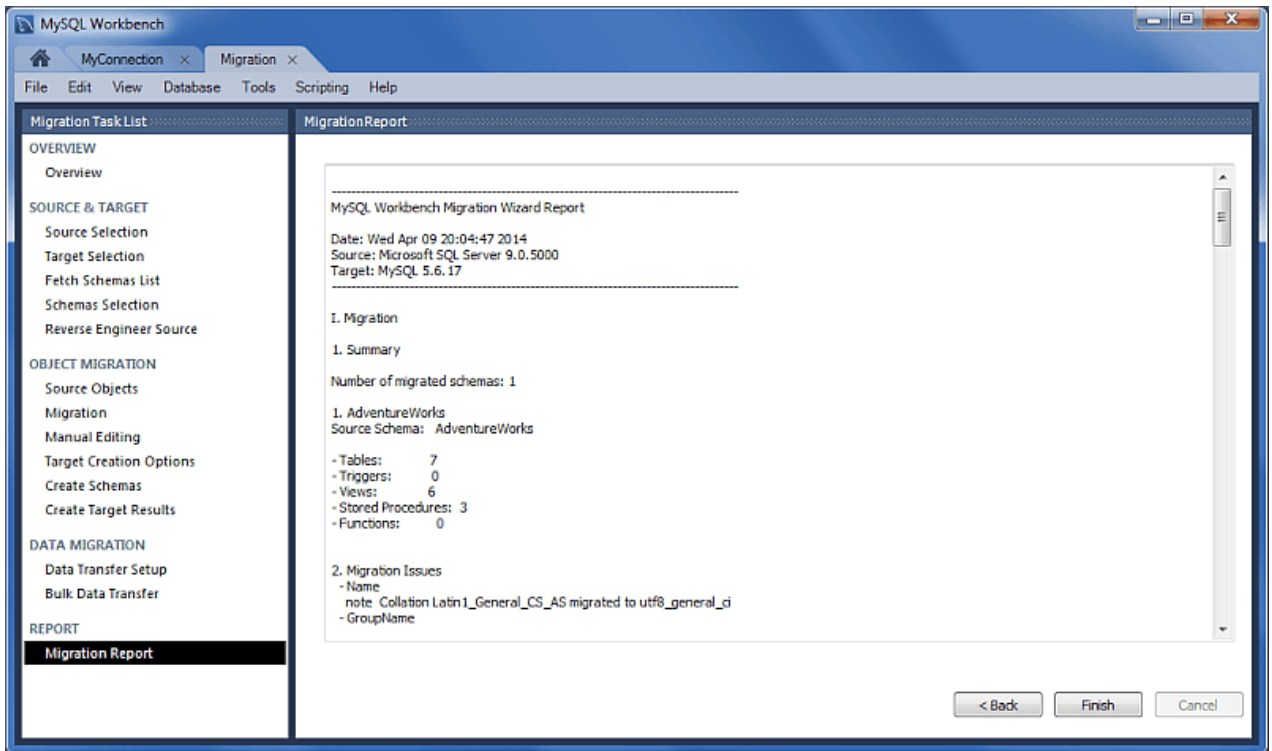
Figure 10.17 MySQL Workbench Migration: Bulk Data Transfer



Migration Report

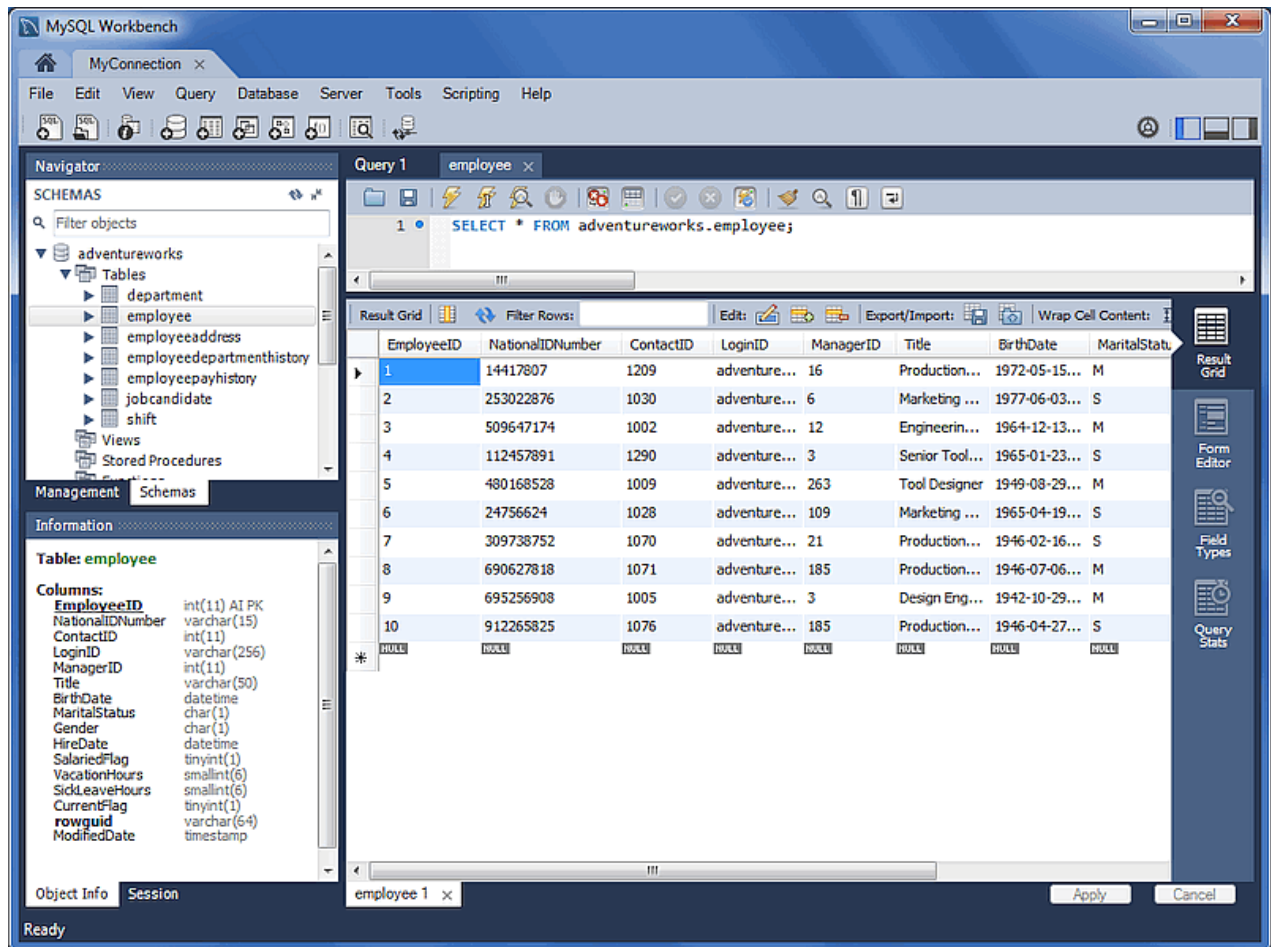
And finally, the migration report is available and summarizes the entire migration process as depicted in the following figure.

Figure 10.18 MySQL Workbench Migration: Migration Report



Clicking **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 as the following figure shows.

Figure 10.19 MySQL Workbench Migration: Viewing the Migrated Database



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 and later

- Microsoft Access 2007 and later
- MySQL Server 5.6 and higher as the source, and MySQL Server 5.6 and higher as the target
- PostgreSQL 8.0 and later
- SQL Anywhere
- SQLite
- Sybase Adaptive Server Enterprise 15.x and later

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

The following table shows a comparison between each DBMS product supported by the Migration Wizard and MySQL.

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 ownerobject, use the owner as the schema name, or merge the owner and object name together.
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	

Concept	MS SQL Server	Sybase ASE	PostgreSQL	MySQL	Note
Foreign Keys	Yes	Yes	Yes	Yes	
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 not case-sensitive. PostgreSQL: as specified in the SQL-99 standard, unquoted object names are treated as not case-sensitive 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: Schemas 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	

Concept	MS SQL Server	Sybase ASE	PostgreSQL	MySQL	Note
Synonyms	N/A	N/A	N/A	N/A	
Table	Yes	Yes	Yes	Yes	
Tablespace	Yes	Yes*	Yes	N/A	MSSQL groups tables in schemas (unless referring to CREATE TABLESPACE). Sybase ASE: tables are grouped in schemas that 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 schemas. 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 schemas for the source (Microsoft SQL Server) database. It can either keep all of the schemas 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

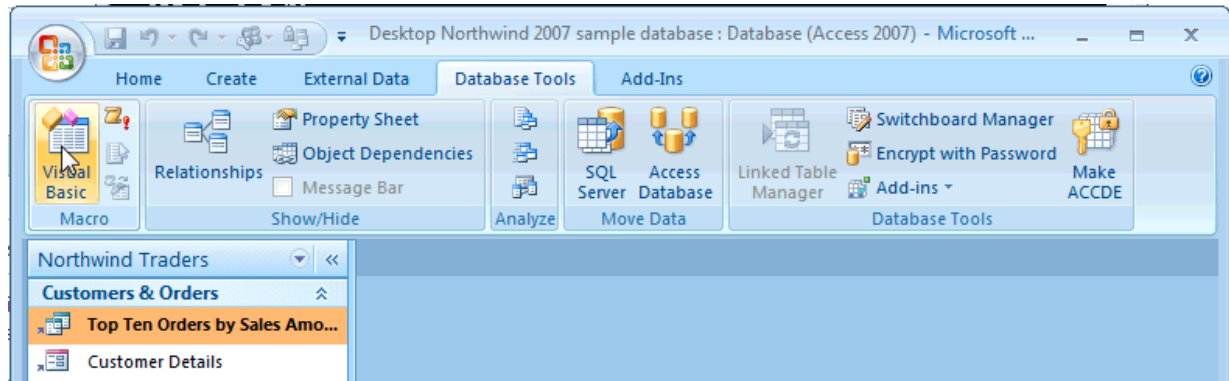
Microsoft Access stores relationship/foreign key information in an internal table called **MSysRelationships**. That table is protected against read access even to the Admin user, so if you try to migrate without opening up access to it, then you will get an error like this:

```
[42000] [Microsoft][ODBC Microsoft Access Driver] Record(s) cannot be read; no read permission on 'msysobj
```

The steps to grant read access to the Admin role (using Microsoft Access 2007) are summarized as follows:

- Open up database in Microsoft Access
- From the **Database Tools** ribbon, click the **Visual Basic** to open the Visual Basic (VB) console. The following figure shows the location of this button in the Macro area.

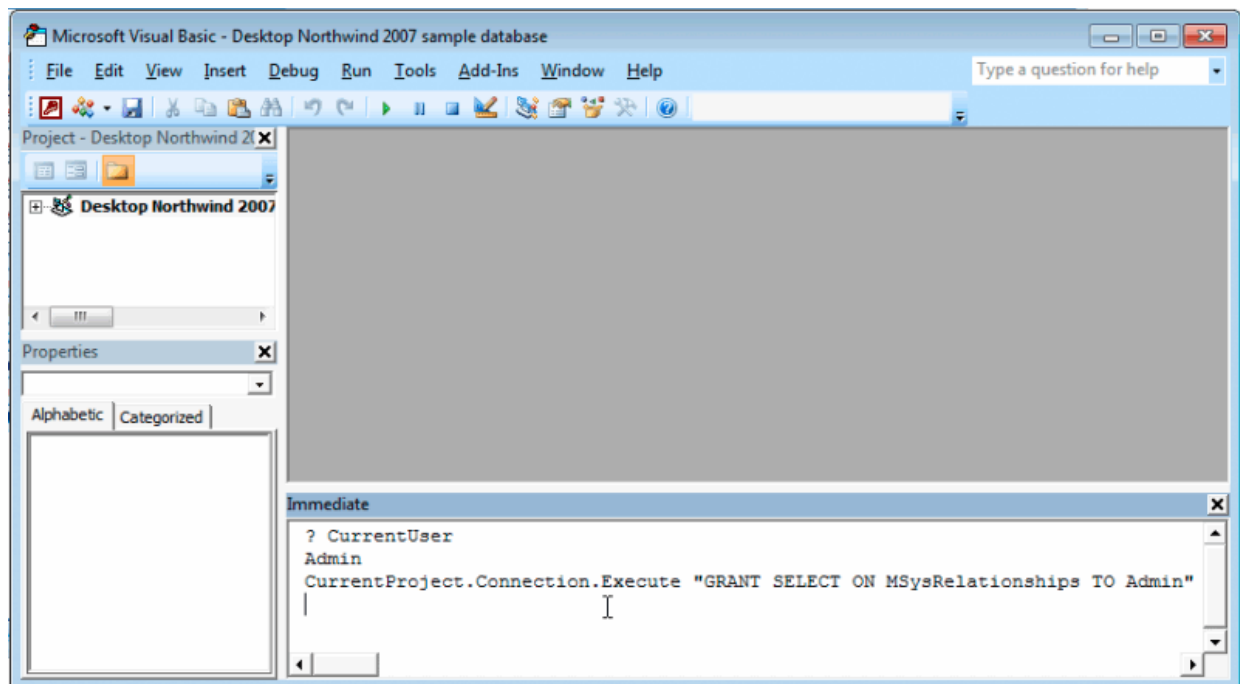
Figure 10.20 Locating the Visual Basic Macro Database Tool



- 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 (see the figure that follows).
- Also in the **Immediate** panel, type the following command to grant access:

```
CurrentProject.Connection.Execute "GRANT SELECT ON MSysRelationships TO Admin"
```

Figure 10.21 GRANT SELECT ON MSysRelationships TO Admin



- Quit the Microsoft Access application

Start the MySQL Workbench Migration Wizard


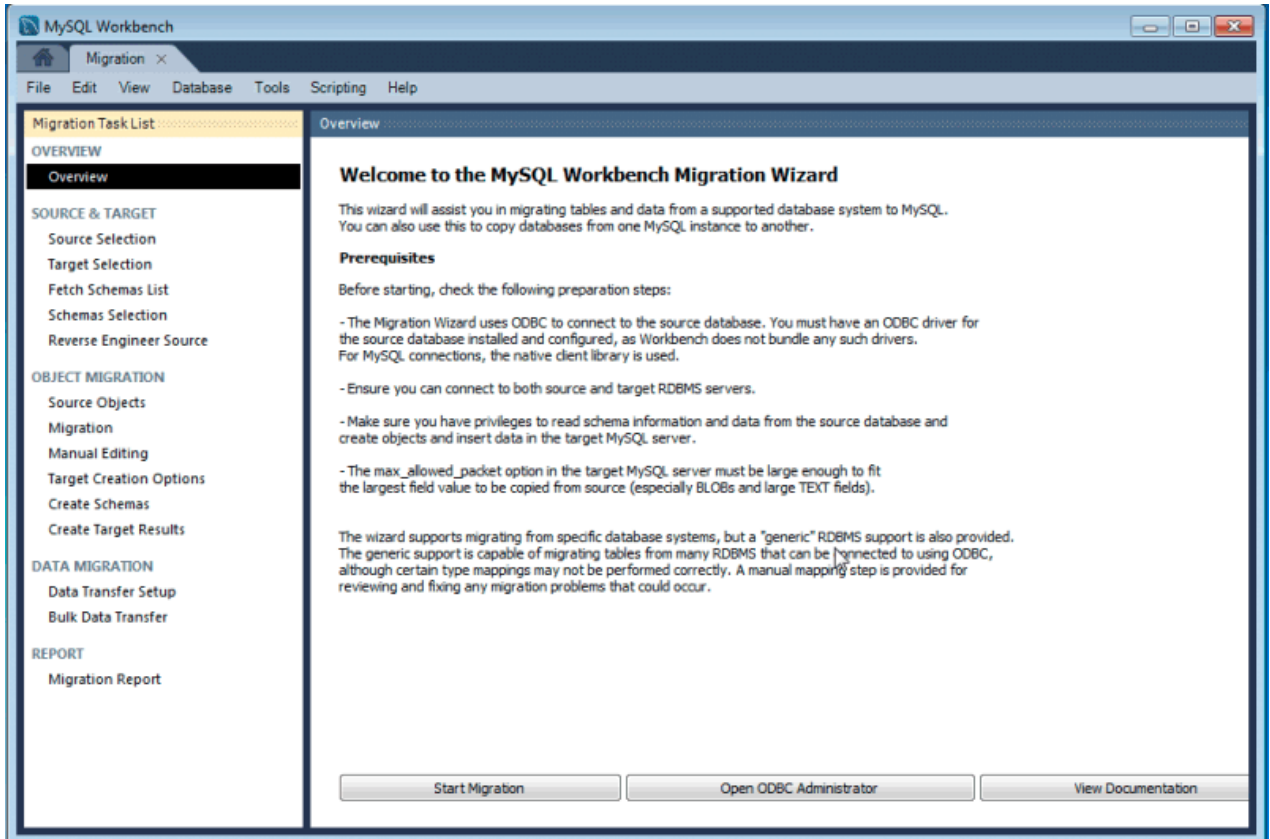
From the main MySQL Workbench screen you can start the Migration Wizard by clicking the database-migration launcher () in the Workbench side panel or by clicking **Database** and then **Migration Wizard** from the main menu. As the following figure shows, a new tab showing the Overview page of the Migration Wizard is displayed.

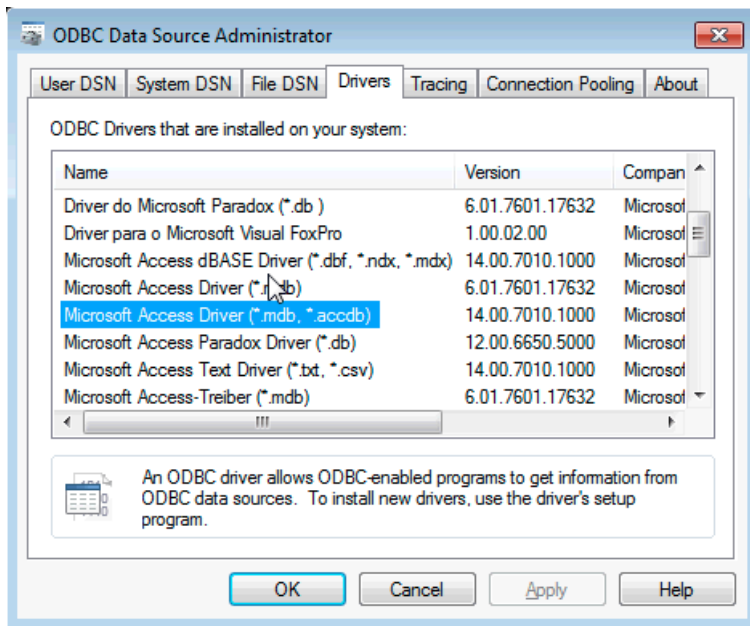
Figure 10.22 Migration Overview Page



Setting Up ODBC Drivers

To verify that 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 (see the figure that follows).

Figure 10.23 Checking the ODBC Drivers for Access Support

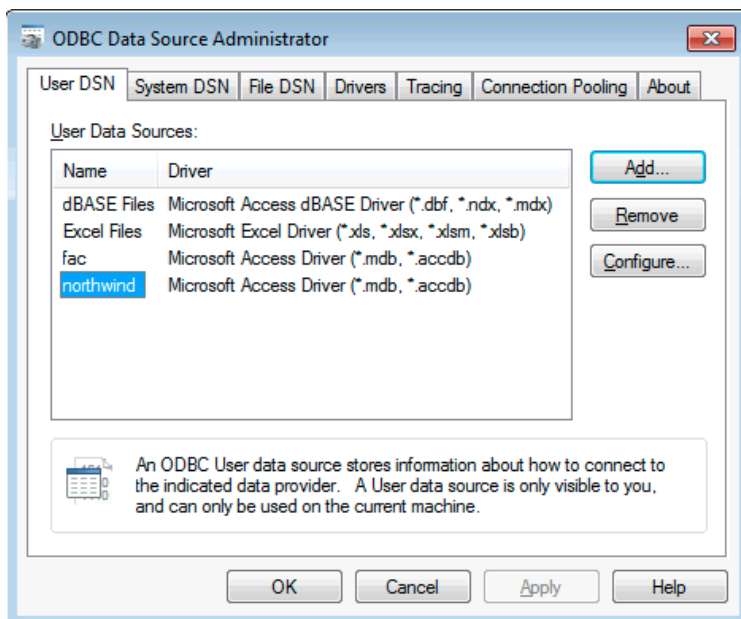


Important

MySQL Workbench has a 64-bit executable. The ODBC drivers you use must be of the same architecture as the MySQL Workbench binaries you are using. 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 **Add** to create a DSN for your database file. As the next figure shows, a new data source was created for the northwind database sample.

Figure 10.24 Adding a New DSN



Setting Up Source Parameters

Click **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.

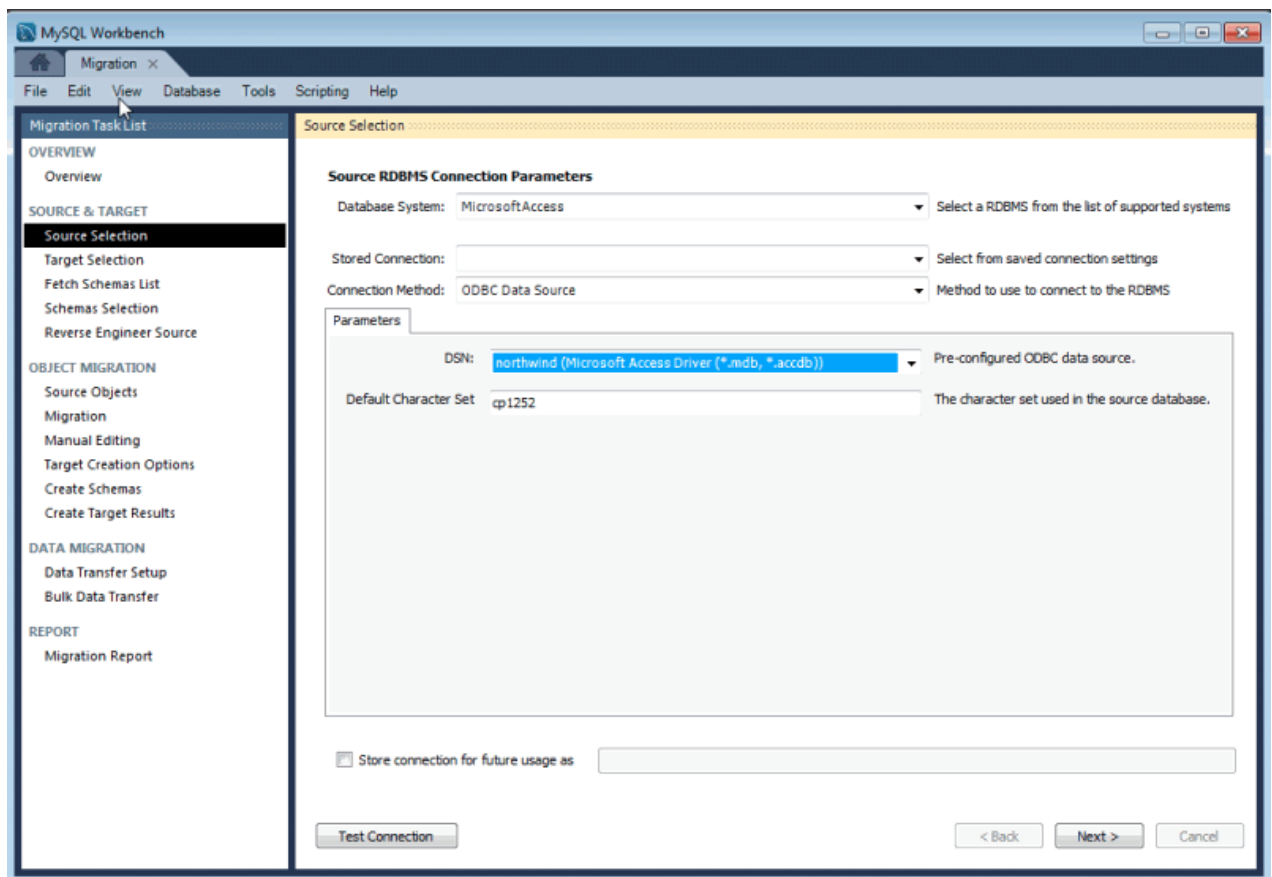
Open the **Database System** combo box for a list of supported RDBMSes, and select **Microsoft Access** from the list. There is another combo box below it named **Stored Connection**. It lists saved connection settings for that RDBMS. You can save connections by marking the check box at the bottom of 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** drop-down list 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 (see the figure that follows). 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 10.25 Access Source Selection

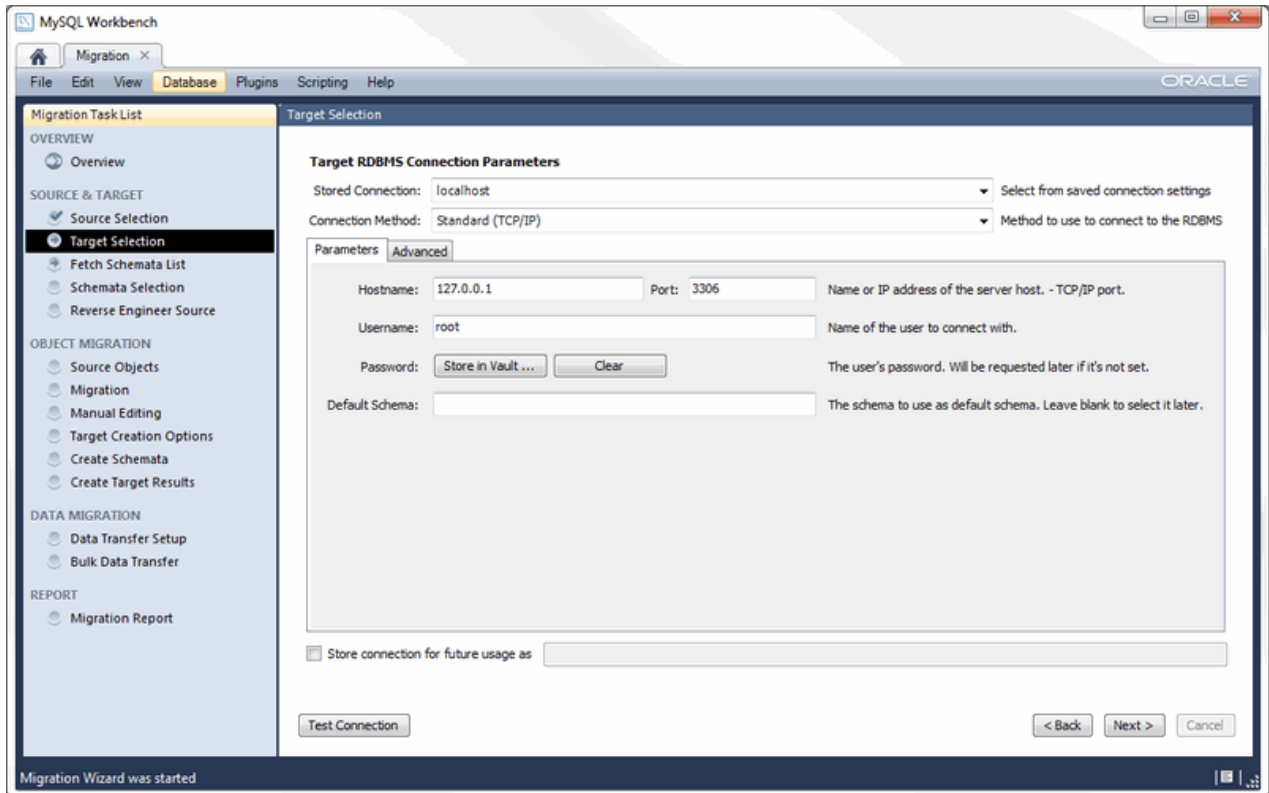


Lastly, click **Test Connection** to check whether an ODBC connection can be established. If you entered the correct parameters then you should see a message reporting a successful connection attempt.

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. The following figure shows the **Parameters** tab.

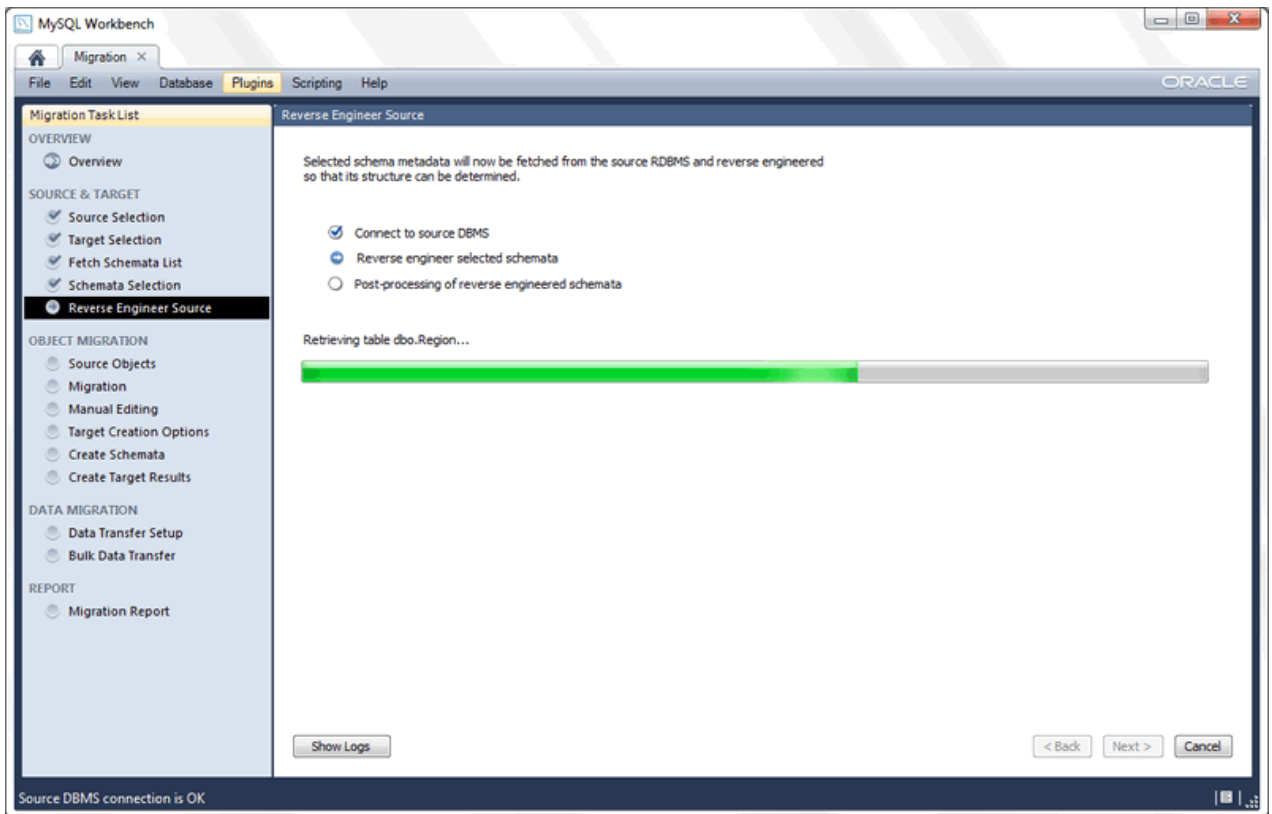
Figure 10.26 Target Database Selection



Select the Objects to Migrate

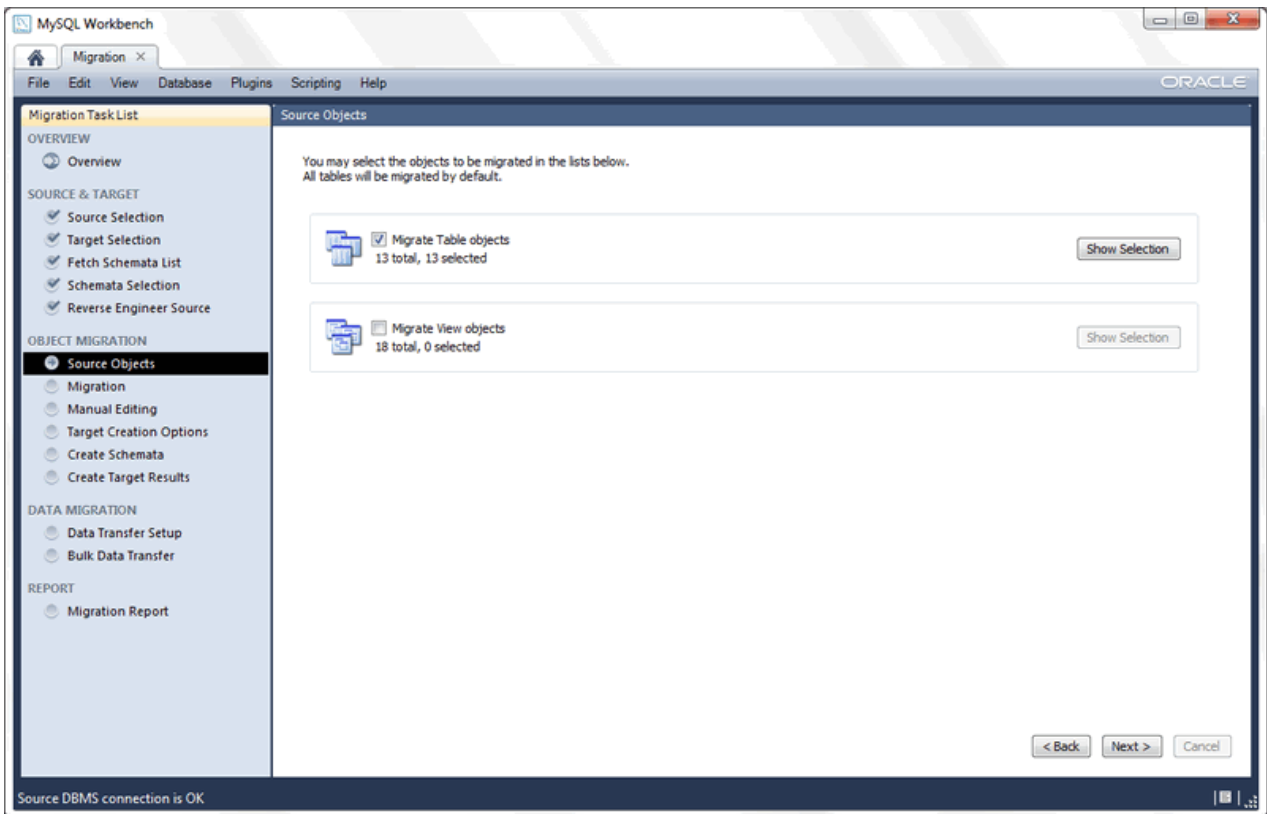
Next, you should see the reverse engineering of the selected database objects progress. At this point, the migration wizard is retrieving relevant information about the involved database objects (such as table names, table columns, primary and foreign keys, indexes, triggers, views, and more). You will be presented a page showing the progress as shown in the next figure.

Figure 10.27 Reverse Engineer Source



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 the figure that follows.

Figure 10.28 Source Objects



In the previous example, the migration wizard discovered table and view objects for our source database. Only the table objects are selected by default for migration.

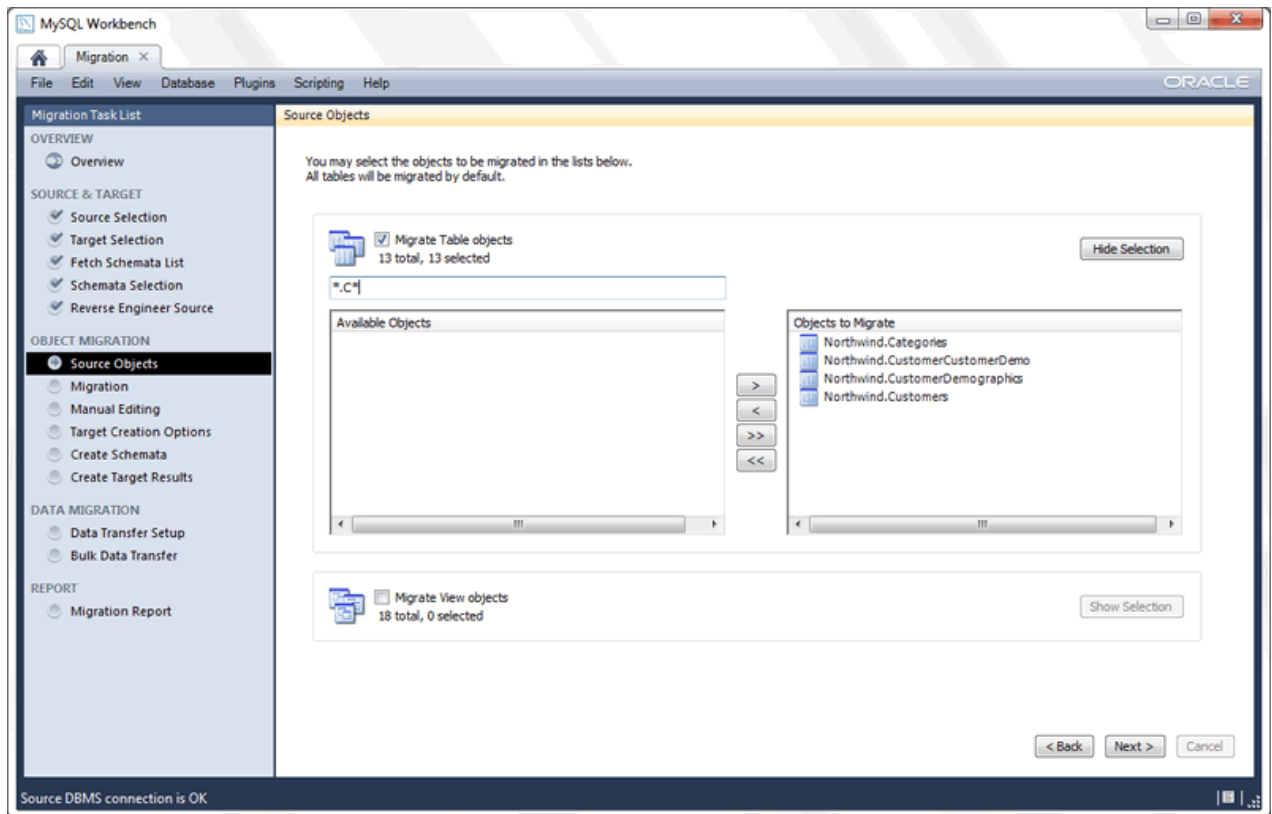


Note

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 the next figure shows.

Figure 10.29 Source Objects Selection

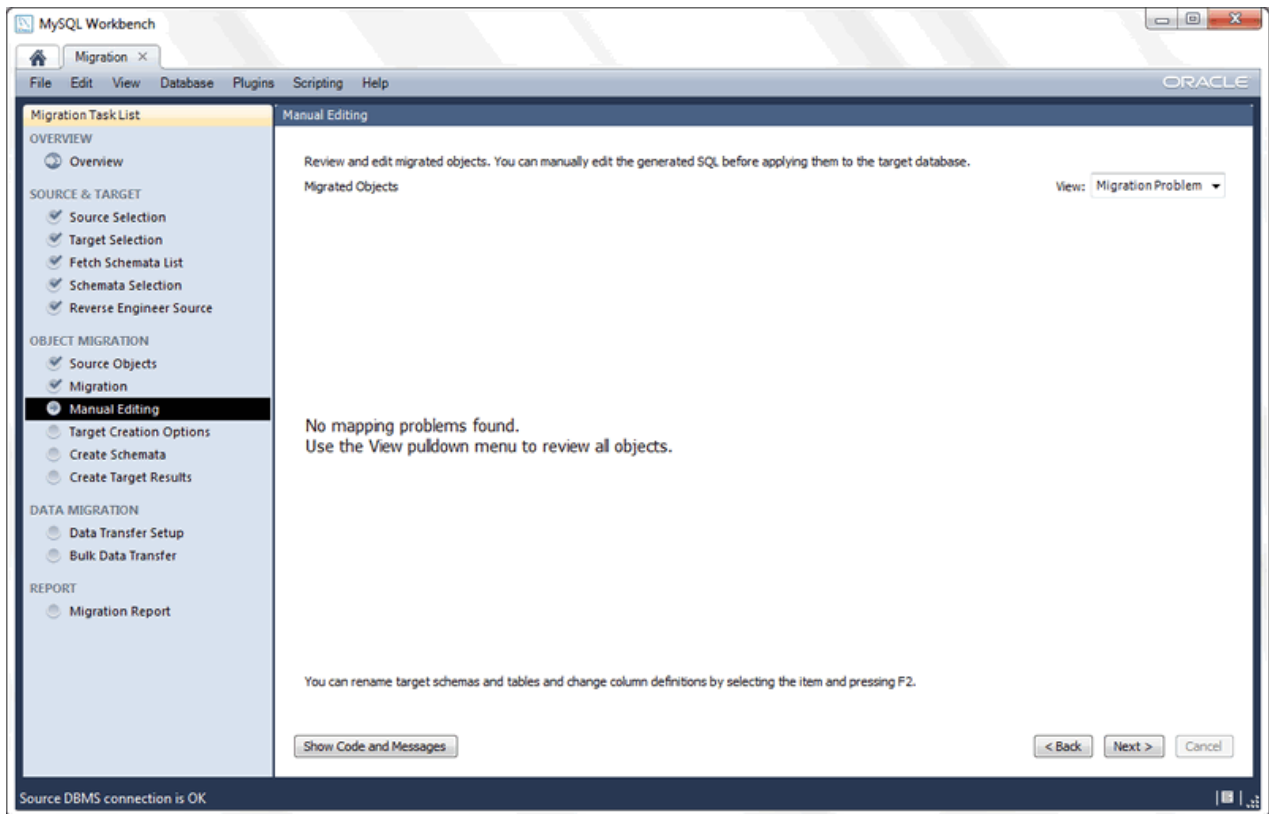


The objects on the right will be migrated. The filter box can filter the list (wildcards are allowed, as demonstrated above). By using the arrow buttons you can filter out the objects that you do not want to migrate. Before continuing, clear the filter text box to check the full list of the selected 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** check box is checked.

Review the Proposed Migration

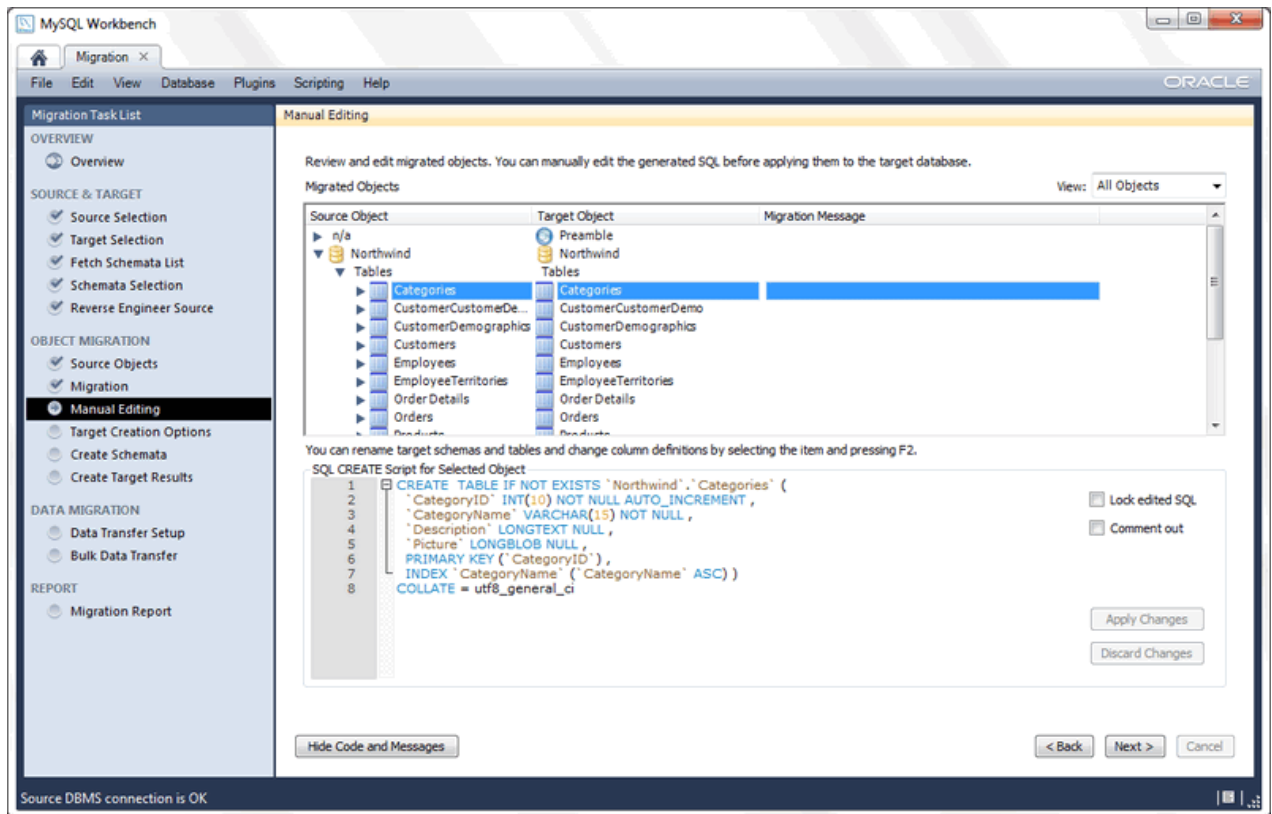
At this point, the migration wizard converts the selected objects into their equivalent objects into the target MySQL server, and it also generates the MySQL code needed to create them. You might have to wait before the **Manual Editing** step displays the initial page shown in the next figure.

Figure 10.30 Manual Editing: Initial Page



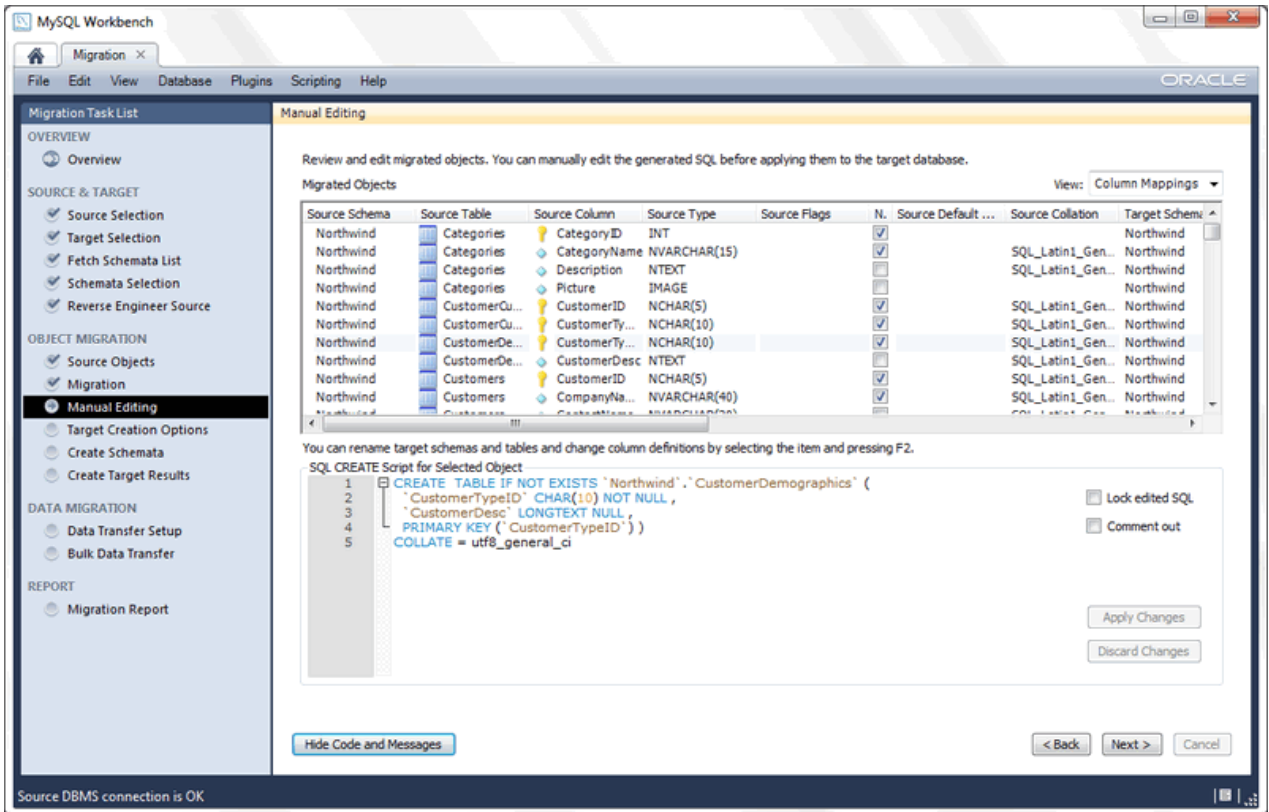
The **View** combo box changes the way the migrated database objects are shown (see the figure that follows). Click **Show Code** to view and edit the generated MySQL code that corresponds to the selected object. Additionally, you can double-click on a row in the object tree to edit the object name, or double-click the database row to change its name.

Figure 10.31 Manual Editing: All Objects



The **View** combo box also has a **Column Mappings** option. As the following figure indicates, it shows the table columns and allows you to review and fix the mapping of column types, default values, and other attributes.

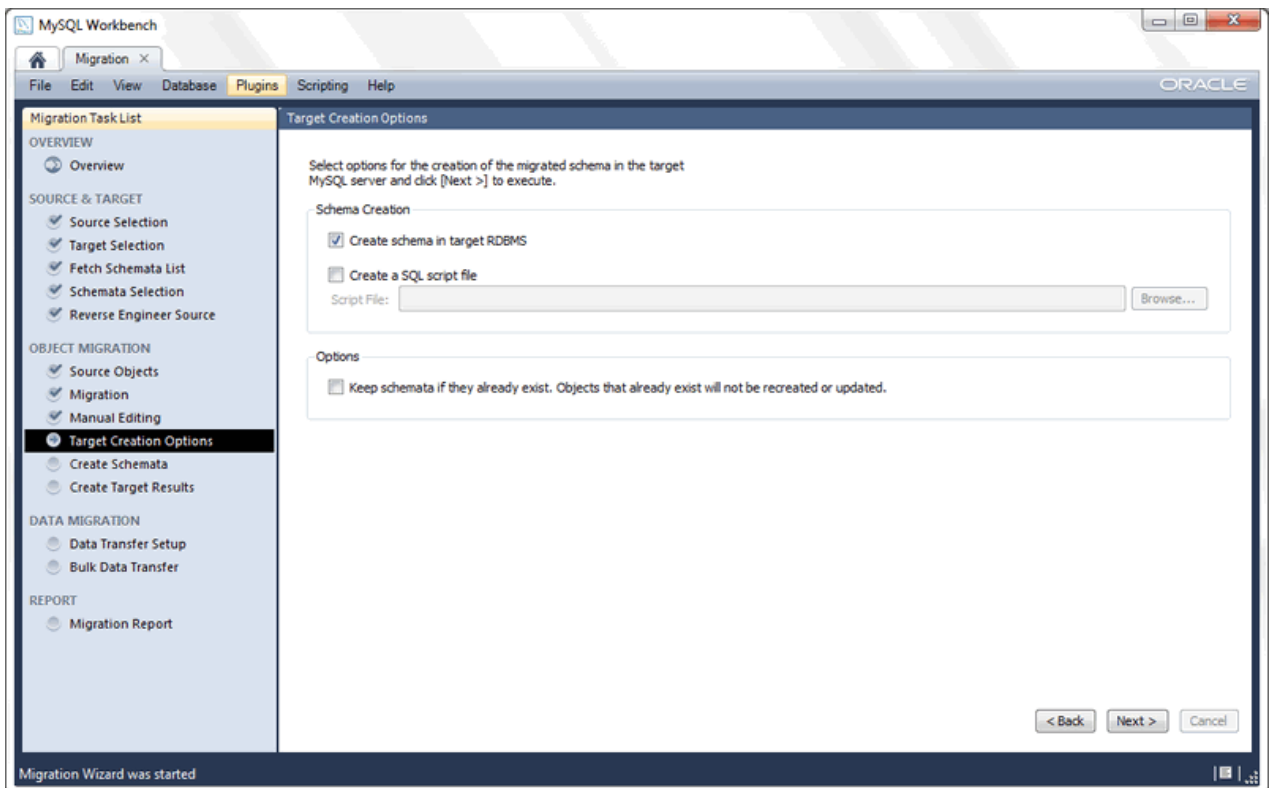
Figure 10.32 Manual Editing: Column Mappings



Create the Database Objects

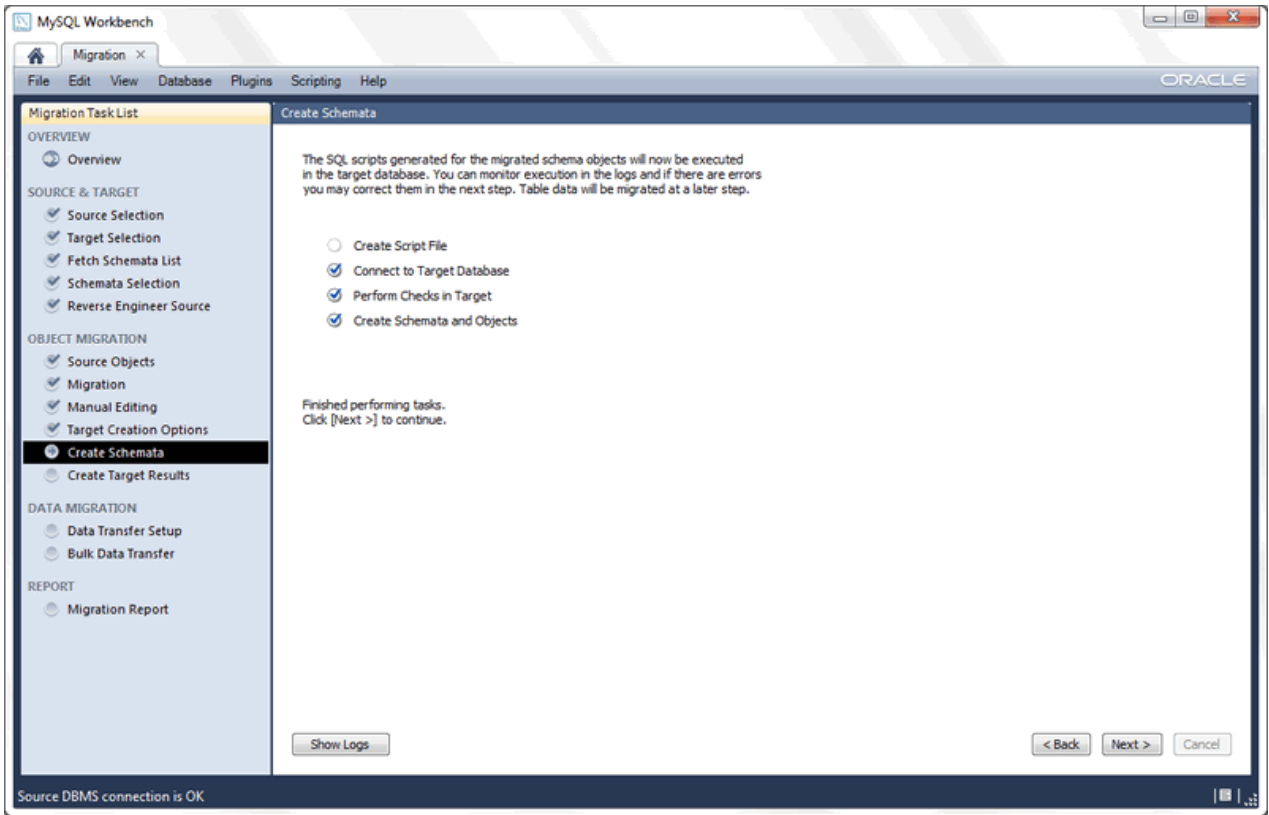
Next is the **Target Creation Options** page, as shown in the following figure.

Figure 10.33 Target Creation Options



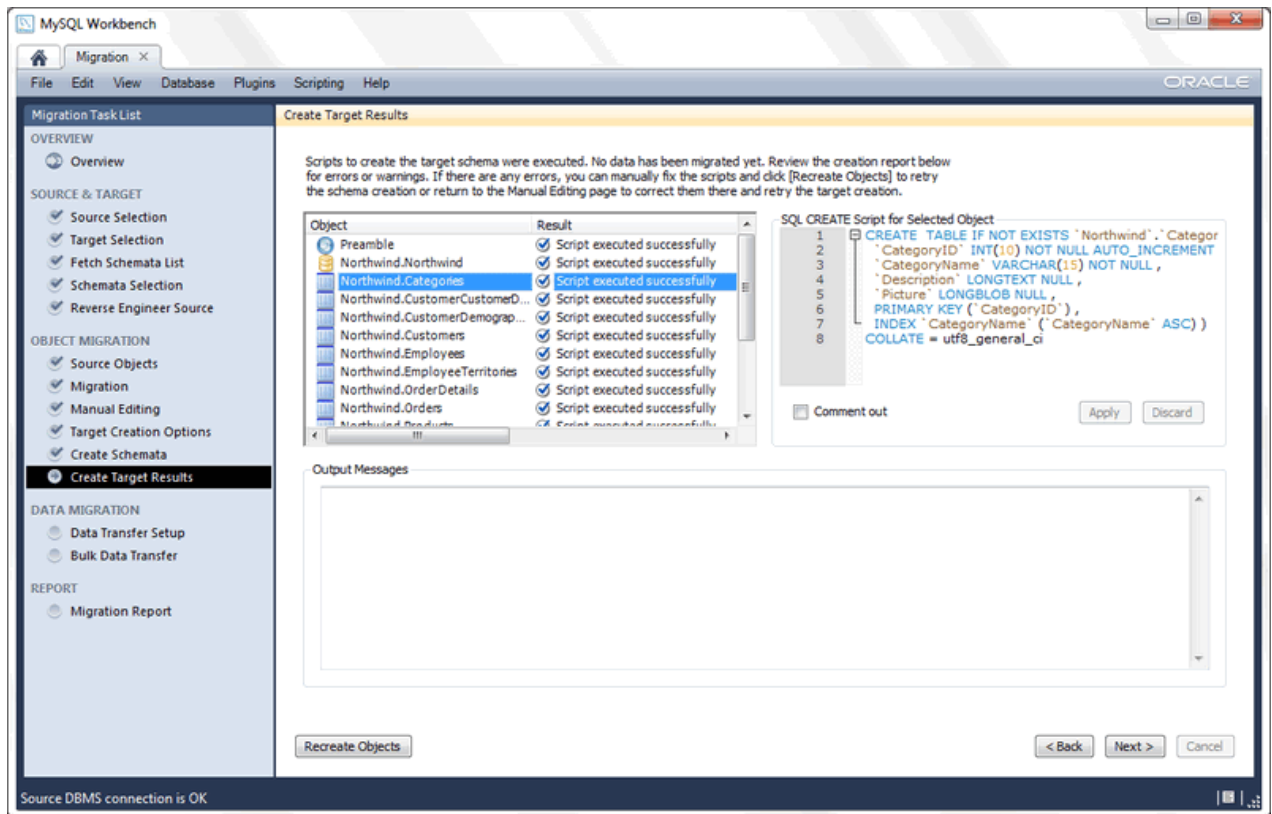
Use target-creation options to execute the generated code in the target RDBMS (your MySQL instance from the second step) or you can dump it to an SQL script file. Leave it as shown in the previous figure and move to the next page. The migrated SQL code will be executed on the target MySQL server. You can view its progress in the **Create Schemas** page shown in the next figure.

Figure 10.34 Create Schemas



When the creation of the schemas 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 the following figure.

Figure 10.35 Create Target Results

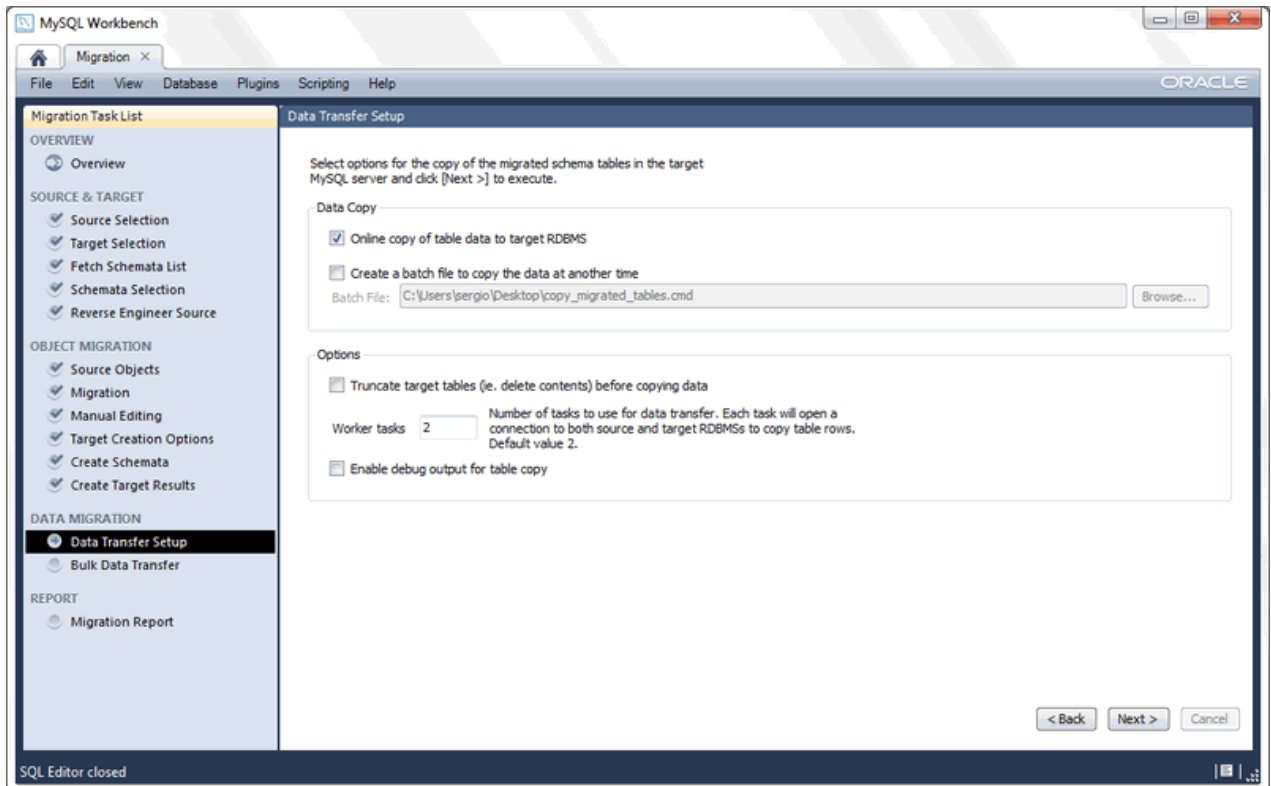


You can edit the migration code using the code box to the right, and save your changes by clicking **Apply**. If edits were made, you still need to recreate the objects with the modified code in order to perform the changes. This is done by clicking **Recreate Objects**. In this tutorial we are not changing 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 (see the figure that follows).

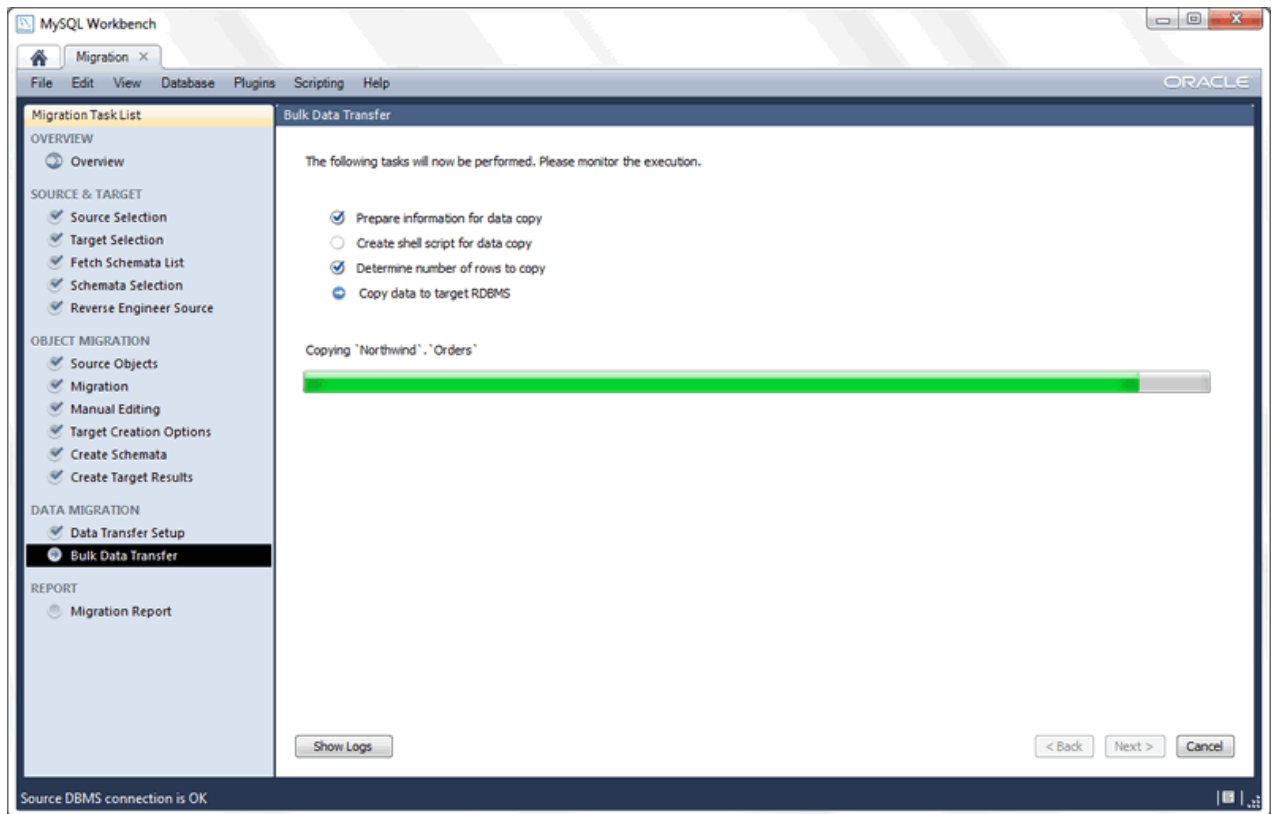
Figure 10.36 Data Transfer Setup



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 previous figure. Next, the data is transferred. At this point the corresponding progress page confirms the tasks being performed (see the figure that follows).

Figure 10.37 Bulk Data Transfer

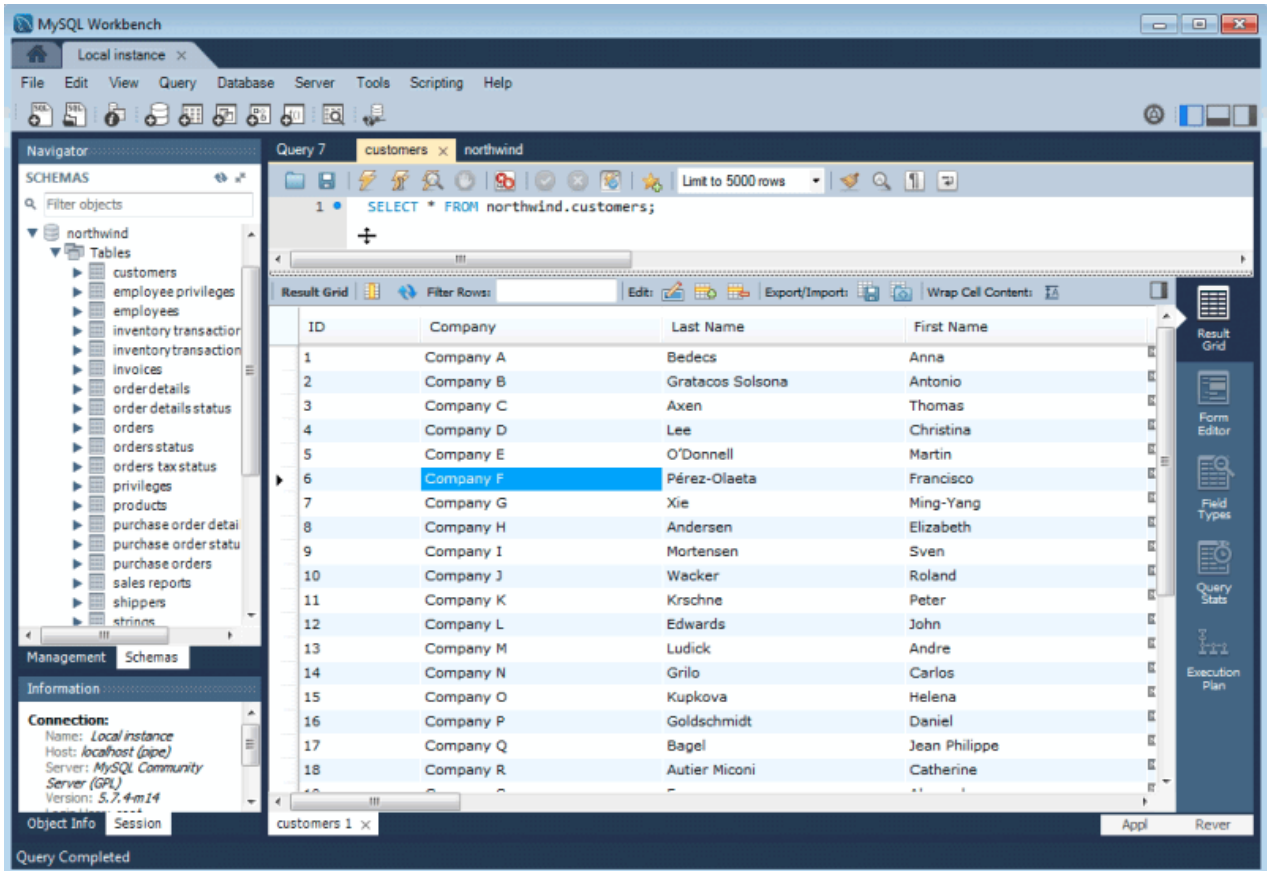


Once it finishes, move to the next page. You will be presented a report page summarizing the whole process. Now, review and click **Finish** to close the wizard.

Verification

Now that the northwind database was successfully migrated, you can view the results. Open an SQL Editor that is associated with your MySQL server instance and then query the northwind database. For example, execute a query like `SELECT * FROM northwind.customers`, which populates the result grid as shown in the next figure.

Figure 10.38 Verify Your Results



10.5 Microsoft SQL Server Migration

The MySQL Workbench Migration Wizard is tested with Microsoft SQL Server 2016.

10.5.1 Preparations

To migrate schemas and data from Microsoft SQL Server for use with MySQL, 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.

Known limitation with Ubuntu 20.04: An error related to FreeTDS/iODBC in Ubuntu prevents the migration of Microsoft SQL Server databases to MySQL using the MySQL Workbench Migration Wizard.

10.5.2 Drivers

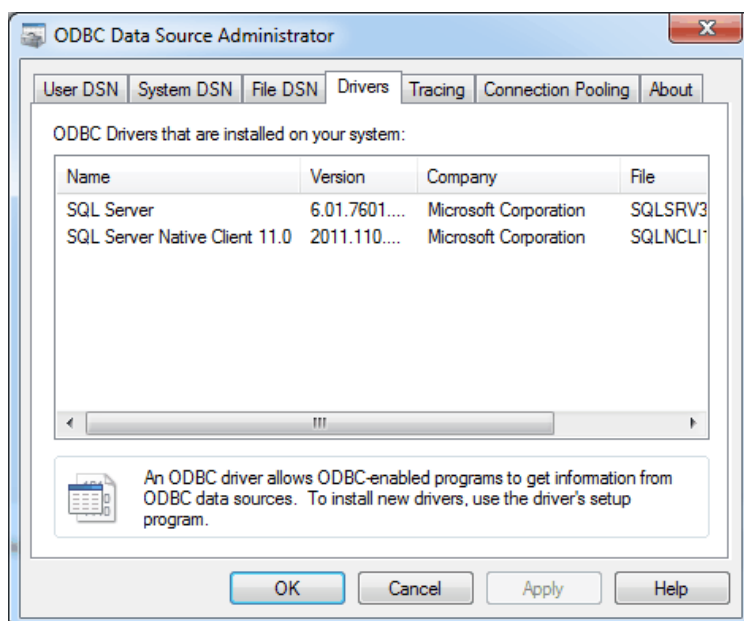
Microsoft Windows does not require additional drivers to be installed and configured, but Linux (and macOS) do. The following sections include specific instructions for each type of system.

10.5.2.1 Microsoft 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 in this section.

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 the following figure.

Figure 10.39 Windows ODBC Data Source Administrator: SQL Server Drivers



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 macOS. 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.

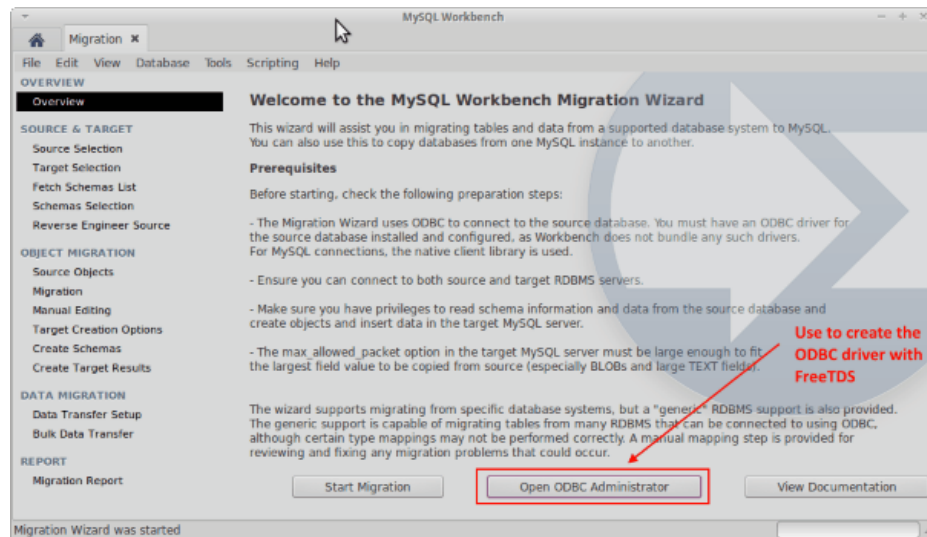
1. For compiling, make sure you have the iODBC headers installed. For Linux, the name depends on your system's package manager but common names are `libiodbc-devel` (RPM based systems) or `libiodbc2-dev` (Debian based systems). For macOS, the headers come with the system and no additional action is required 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](#).

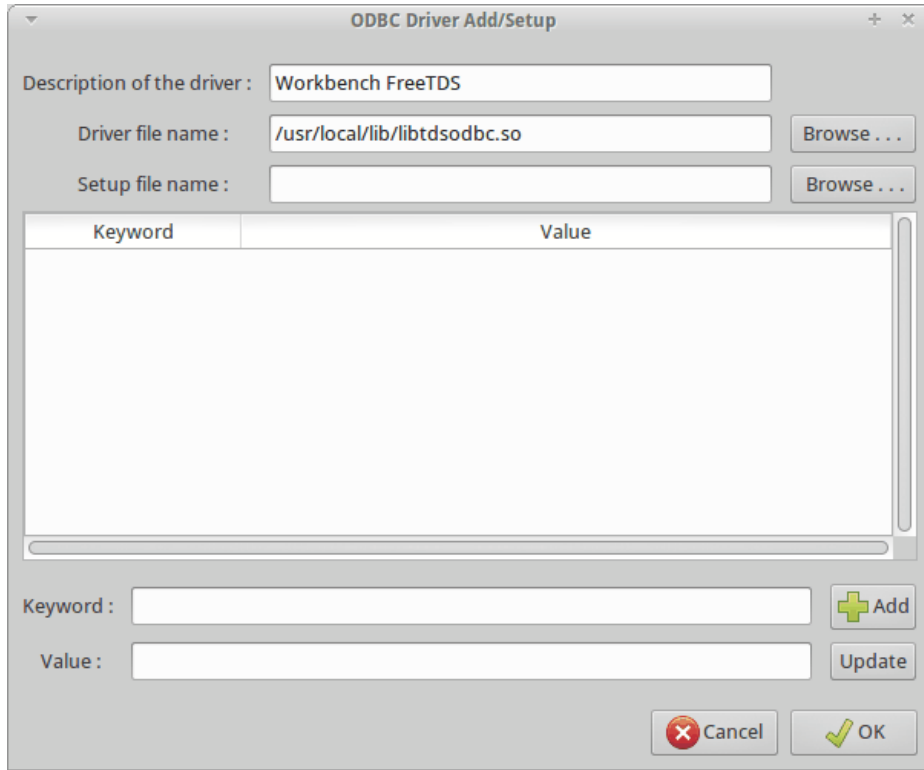
2. `mkdir ~/freetds` to create a directory - within the users home directory.
3. Copy the `build_freetds.sh` script to `~/freetds`
4. Get the latest FreeTDS sources from <ftp://ftp.freetds.org/pub/freetds/> and place the `.tar.gz` source file into the `~/freetds` directory. Make sure to get FreeTDS version 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 (see the figure that follows).

Figure 10.40 Open the ODBC Administrator

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 (see the figure that follows). Remember the name you define here as it will be needed later on. Save the driver.

Figure 10.41 ODBC Driver Add/Setup



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 macOS

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.



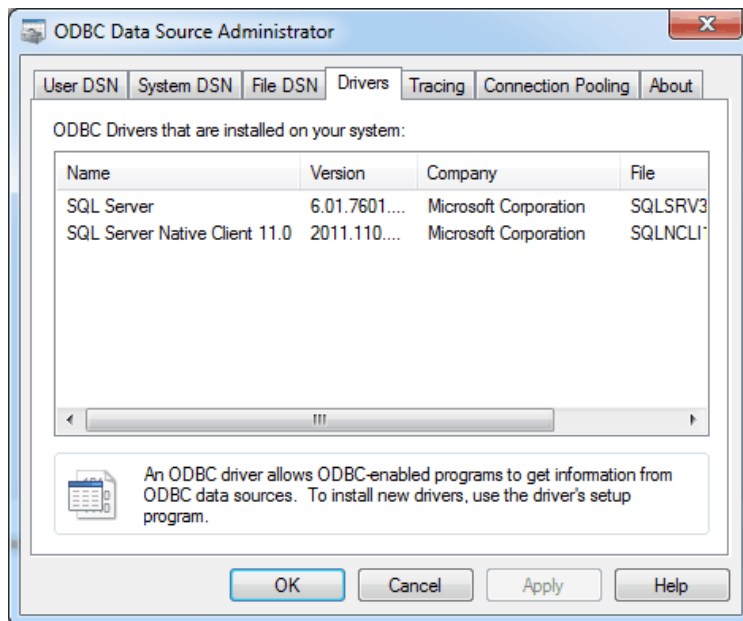
Note

Prerequisite: that you already installed and configured the required [Microsoft SQL Server driver](#) on the system running MySQL Workbench.

10.5.3.1 Microsoft Windows

Select **Microsoft SQL Server** as the database system and fill out the remaining options as described in this section. The following figure shows an example of the **Drivers** tab.

Figure 10.42 SQL Server Connection Parameters Example on Windows



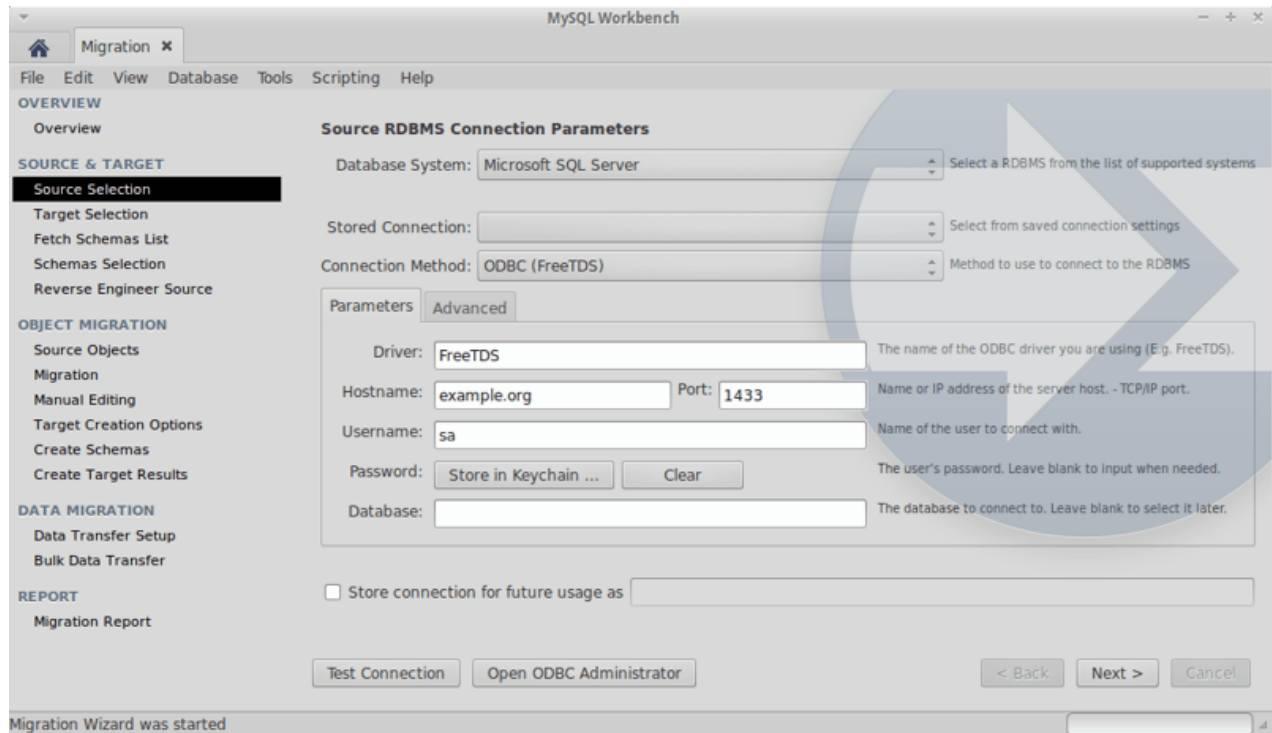
- **Database System:** Microsoft SQL Server
- **Connection Method:** choose **ODBC (native)** to use the native ODBC driver that is provided by Microsoft. Alternatives include "ODBC data sources" and "ODBC FreeTDS". FreeTDS is a popular open source driver for SQL Server and Sybase.
- **Driver:** use the SQL Server driver name, as described in the documentation titled [Section 10.5.2.1, "Microsoft 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 in this section. The following figure shows an example of the connection parameters.

Figure 10.43 SQL Server Connection Parameters Example on Linux



Source RDBMS connection parameters include:

- **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 macOS

Connection parameters are similar to Linux, see [Section 10.5.3.2, “Linux”](#).

10.5.4 Microsoft SQL Server Type Mapping

The following table shows the mapping between Microsoft SQL Server (source) data types and MySQL data types.

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	
SMALLMONEY	DECIMAL	
CHAR	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.6 and higher 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 5.6 and higher can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, a character set of strings depends on the column character set instead of the data type.
VARCHAR	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length. MySQL Server 5.6 and higher 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 5.6 and higher can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated

Source Type	MySQL Type	Comment
		to one of the TEXT blob types. In MySQL, a character set of strings depends on the column character set instead of the data type.
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 and higher.
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.
IMAGE	TINYBLOB/ MEDIUMBLOB/ LONGBLOB	Depending on its length.
SQL_VARIANT	not migrated	There is not specific support for this data type.
TABLE	not migrated	There is not specific support for this data type.
HIERARCHYID	not migrated	There is not specific support for this data type.
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 later, although older versions may work.
- Access to a running MySQL Server instance with privileges to the database you want to migrate.

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:

```
$> cd the/src/directory
$> ./configure --with-iodbc --enable-pthreads
$> make
$> sudo make install
```

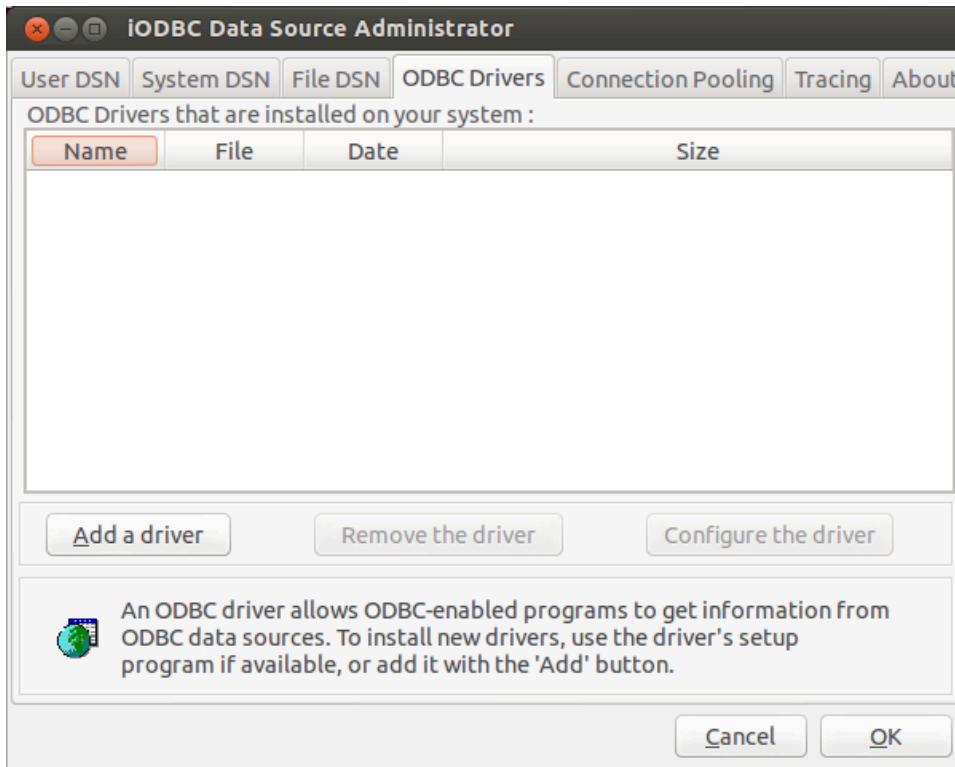
Verify the installation by confirming that the file `psqlodbcw.so` is in the `/usr/local/lib` directory.

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 the example in the figure that follows.

Figure 10.44 The iODBC Data Source Administrator



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 macOS

To compile `psqlODBC` on macOS, 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:

```
$> cd the/src/directory
$> ./configure --with-iodbc --enable-pthreads
$> CFLAGS="-arch i386 -arch x86_64" make
$> 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 list reveals each RDBMS that is supported on your system. Select PostgreSQL from the list. Below that is the **Stored Connection** dropdown list, 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** check box 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

The following table shows the mapping between PostgreSQL (source) data types and MySQL data types.

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)	
CHAR	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.6 and higher can have CHAR columns with a length up to 255 characters. Anything larger is migrated as LONGTEXT.

Source Type	MySQL Type	Comment
NATIONAL CHARACTER	CHAR/LONGTEXT	Depending on its length. MySQL Server 5.6 and higher can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, a character set of strings depends on the column character set instead of the data type.
VARCHAR	VARCHAR/ MEDIUMTEXT/ LONGTEXT	Depending on its length. MySQL Server 5.6 and higher 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 5.6 and higher can have VARCHAR columns with a length up to 65535 characters. Anything larger is migrated to one of the TEXT blob types. In MySQL, a character set of strings depends on the column character set instead of the data type.
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	
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.

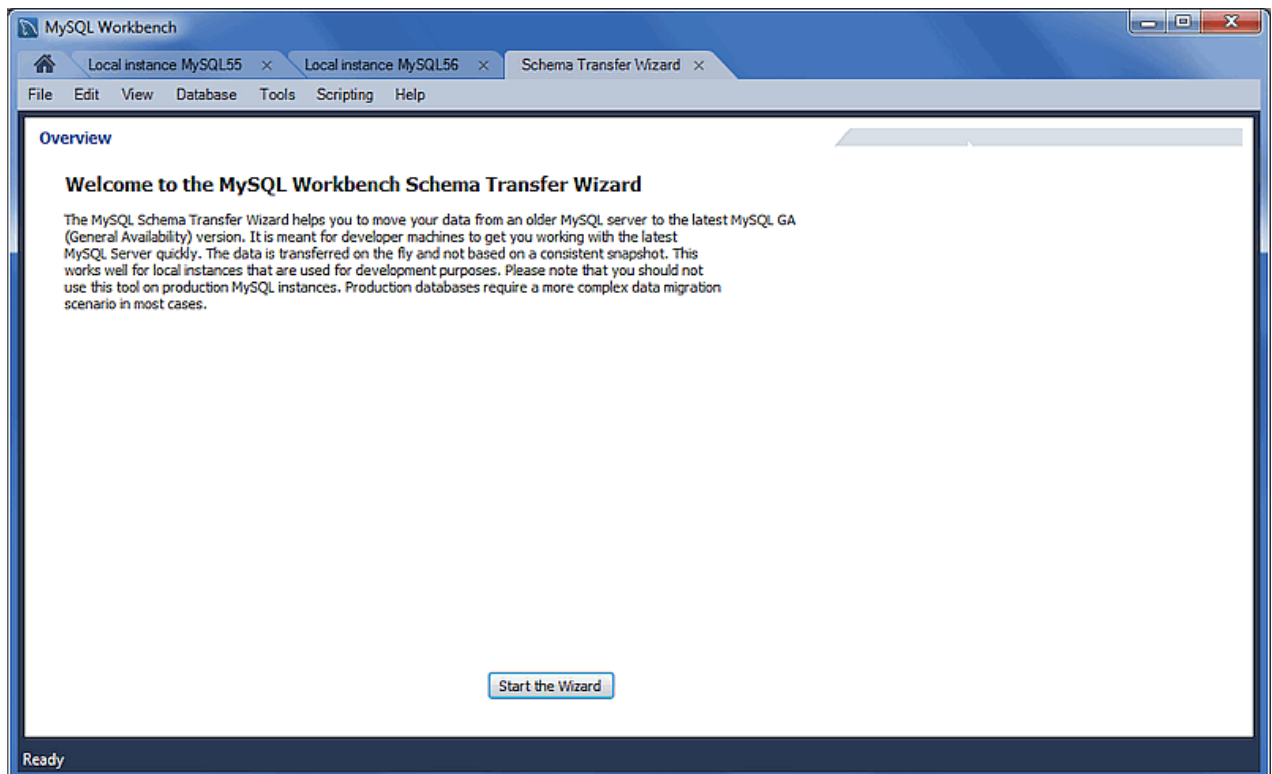


Note

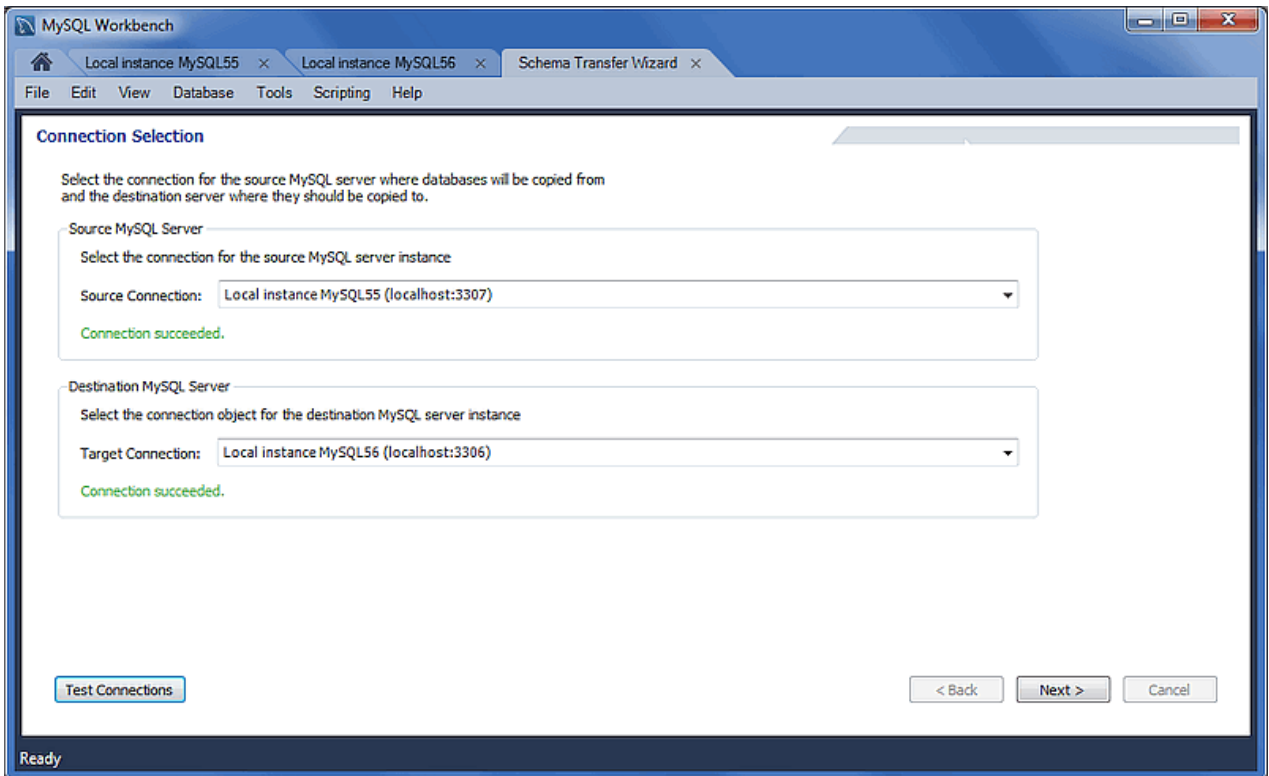
You should not use this wizard on production MySQL instances.

To open the wizard, select **Database** and then **Schema Transfer Wizard** from the main menu. The next figure shows the initial screen.

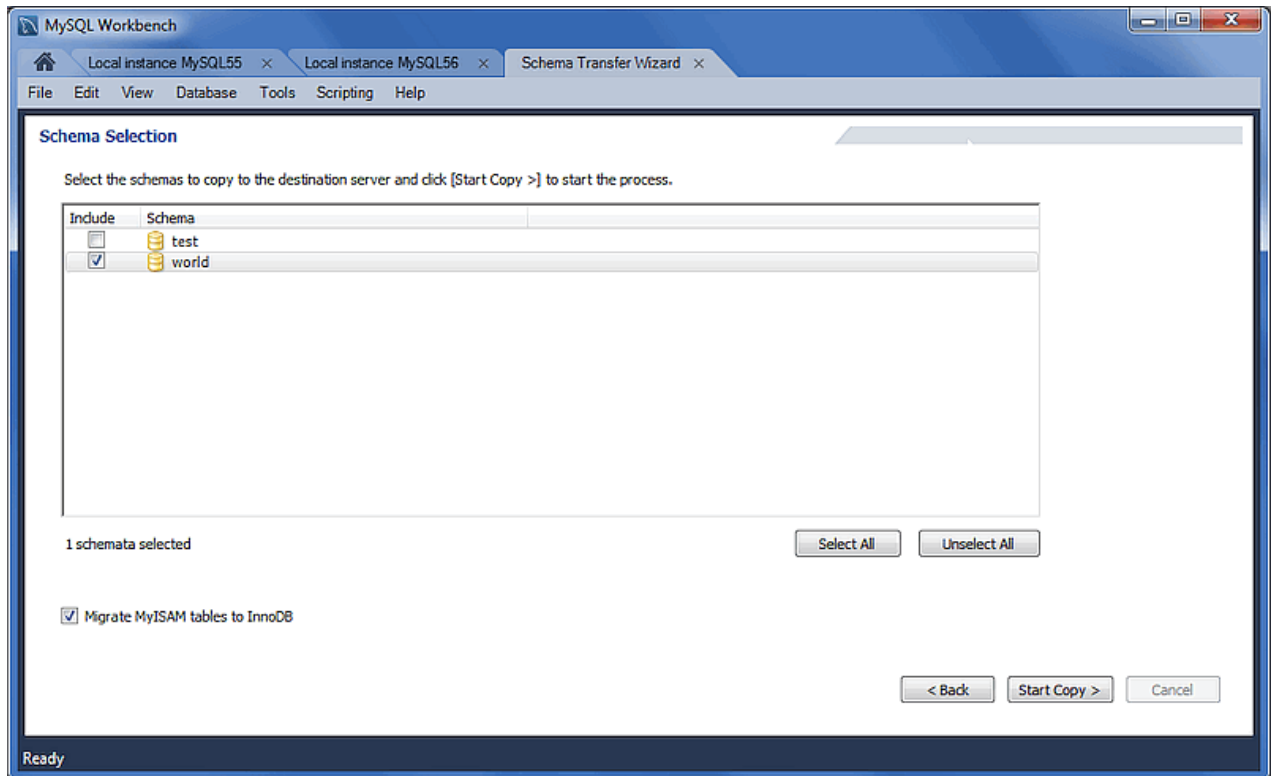
Figure 10.45 MySQL Schema Transfer Wizard: Overview



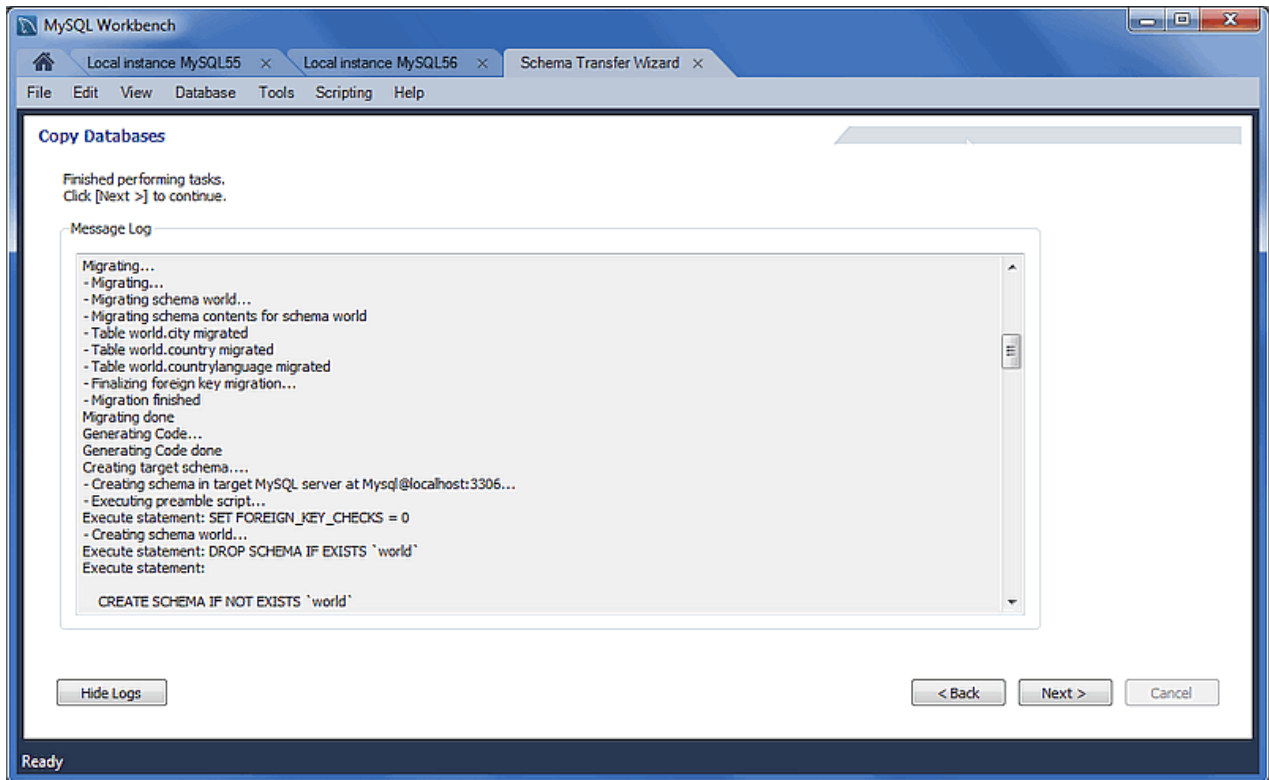
Read the overview text and click **Start the Wizard** to begin. An example transfer appears in the figure that follows.

Figure 10.46 MySQL Schema Transfer Wizard: Connection Selection

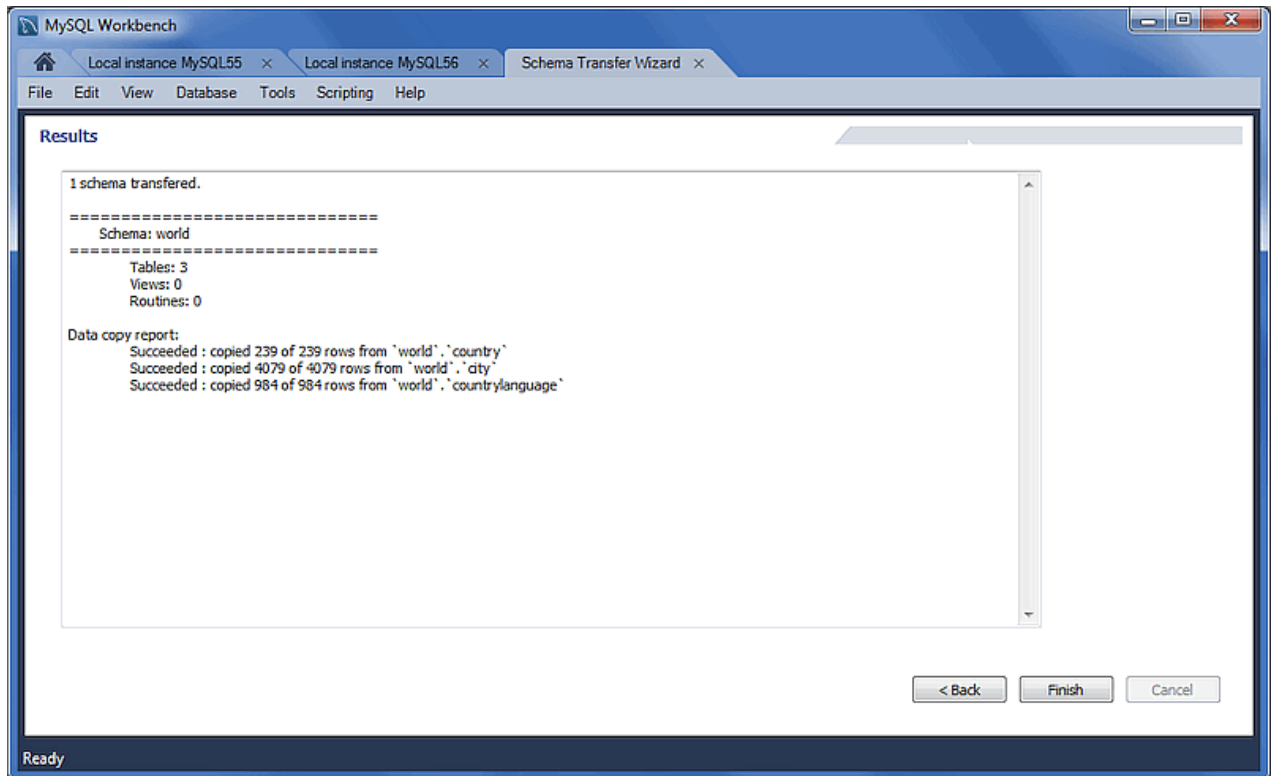
Choose your target and source MySQL connections (see the figure that follows). After choosing and testing your MySQL connections, click **Next** to continue.

Figure 10.47 MySQL Schema Transfer Wizard: Schema Selection

Choose the schemas to migrate, and click **Start Copy** to begin copying the selected schemas from the source to target MySQL server. The next figure shows the copy status.

Figure 10.48 MySQL Schema Transfer Wizard: Copy Databases

Review the **Message Log** to confirm that the migration finished with success. Click **Next** to view a summary of the results. The following figure shows an example of the copy results.

Figure 10.49 MySQL Schema Transfer Wizard: Results

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

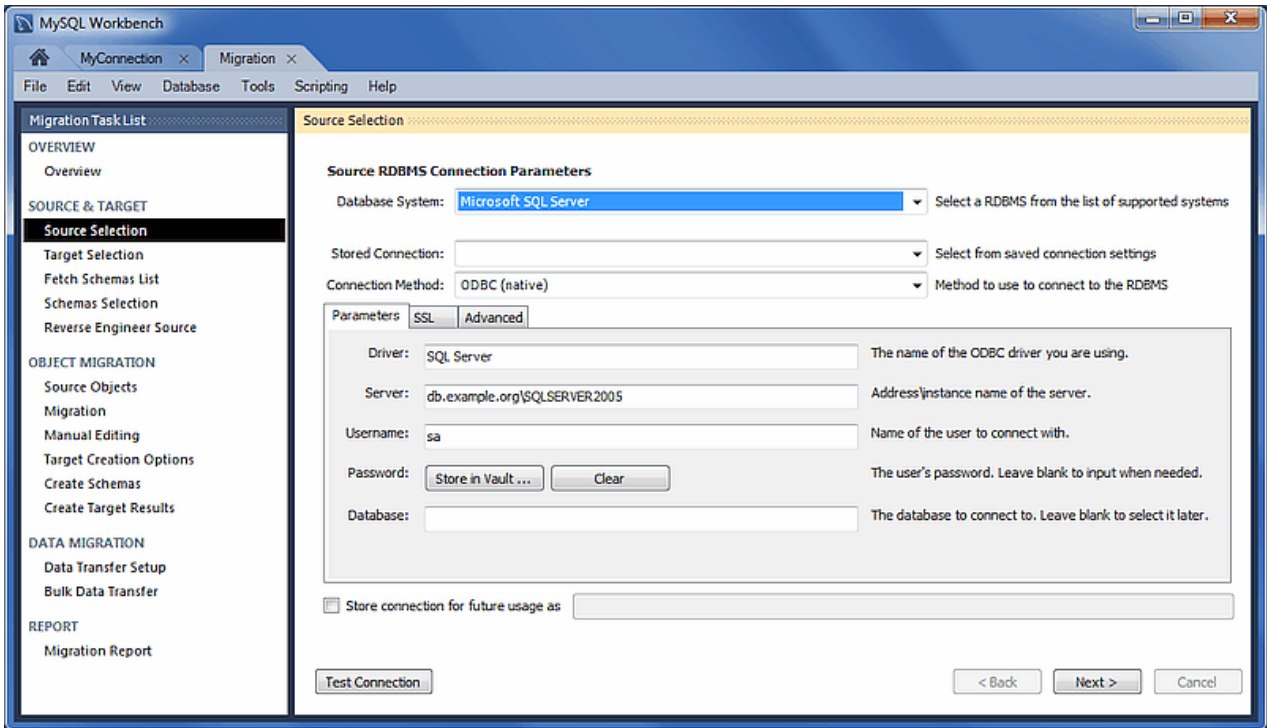
Select the source RDBMS that is migrating to MySQL. Choose the **Database System** that is being migrated and the other connection parameters will change accordingly (see the figure that follows).



Note

This connection definition may be saved using the [Store connection for future use as](#) option, and there is also the **Test Connection** option.

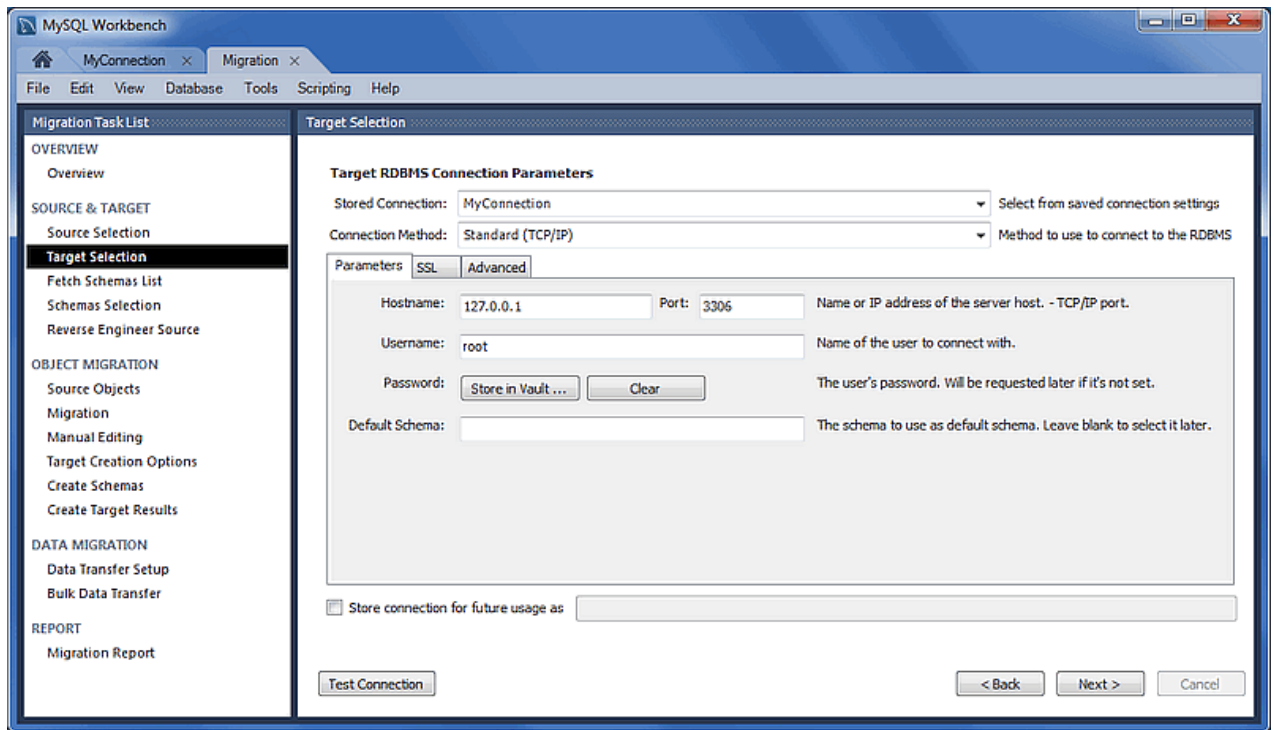
Figure 10.50 MySQL Workbench migration: Source Selection (Parameters)



Target Selection

The target is the MySQL database that will contain the migrated data. Choose an existing MySQL Workbench connection or select **Manage Stored Connections** from drop-down list to create a new MySQL connection. An example stored connection appears in the following figure.

Figure 10.51 MySQL Workbench migration: Target selection



10.8.2 Schema Retrieval and Selection

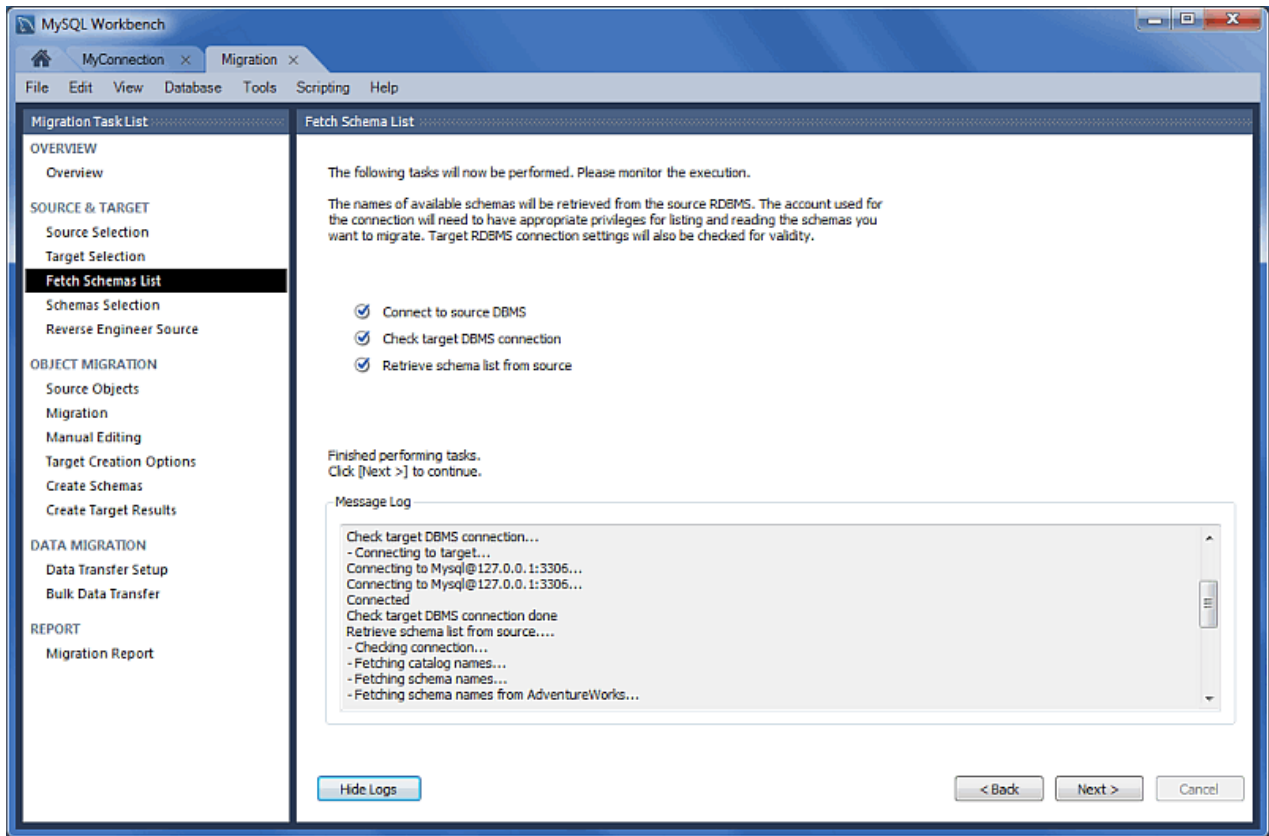
Retrieve a list of available databases and choose the specific databases (and tables) that you want to migrate to MySQL.

Fetch Schemas List

The Schemas 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, general log information, or both (see the figure that follows).

The steps that are performed include: connects to the source DBMS, checks the connection, and retrieves the schema list from the source.

Figure 10.52 MySQL Workbench migration: Fetch Schemas List



Schemas Selection

Choose the databases you want to migrate over to MySQL.

The **Schema Name Mapping Method** step provides the following options for migrating from Microsoft SQL Server:

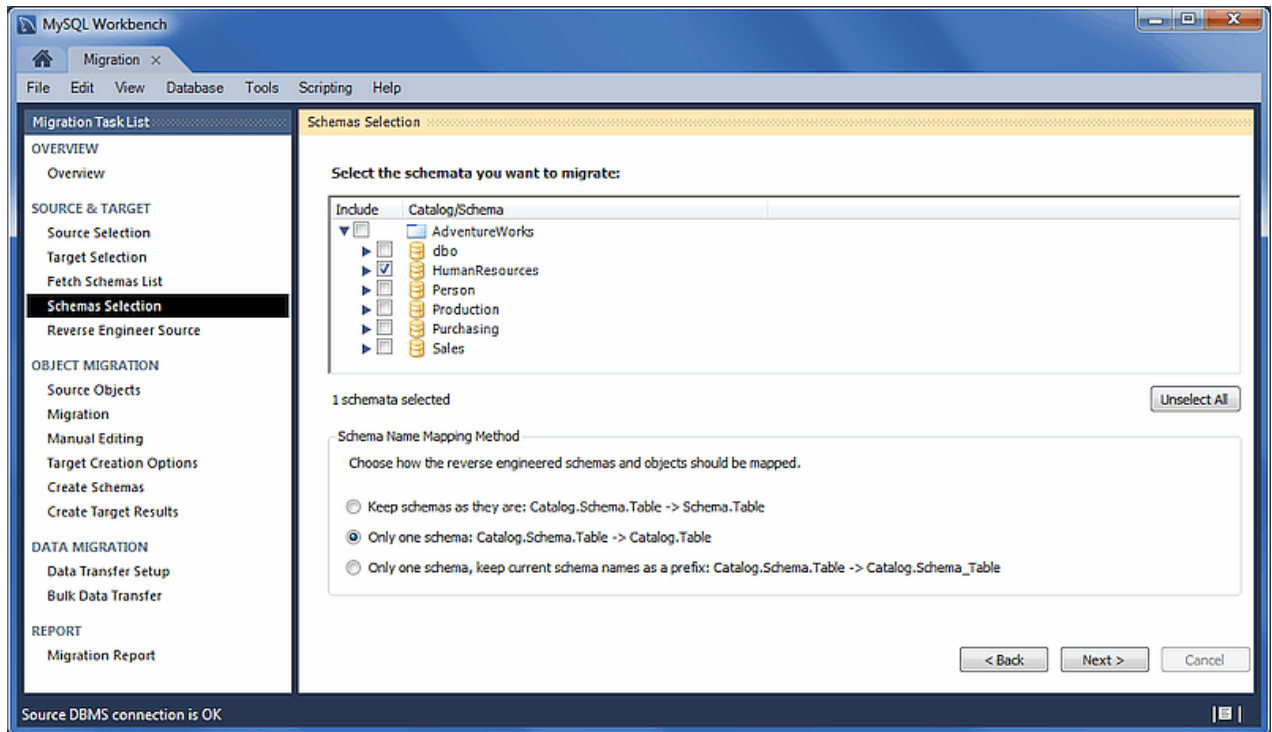


Note

This example uses Microsoft SQL Server as the source RDBMS. Although the options will be different for other database systems, the concept remains the same.

- **Keep schemas 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 (see the figure that follows).
- **Only one schema, keep current schema names as a prefix: Catalog.Schema.Table -> Catalog.Schema_table:** Preserves the schema name as a prefix.

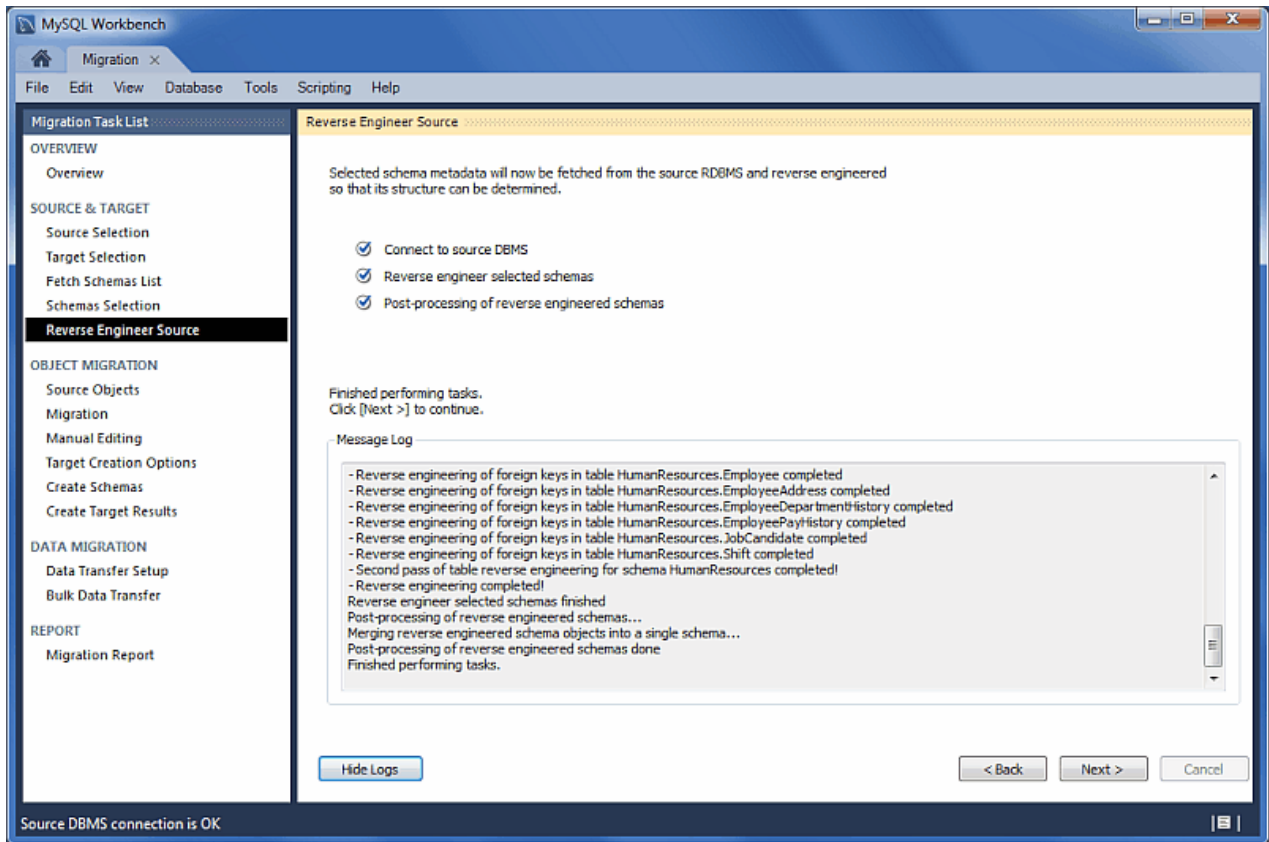
Figure 10.53 MySQL Workbench Migration: Schemas Selection



10.8.3 Reverse Engineering

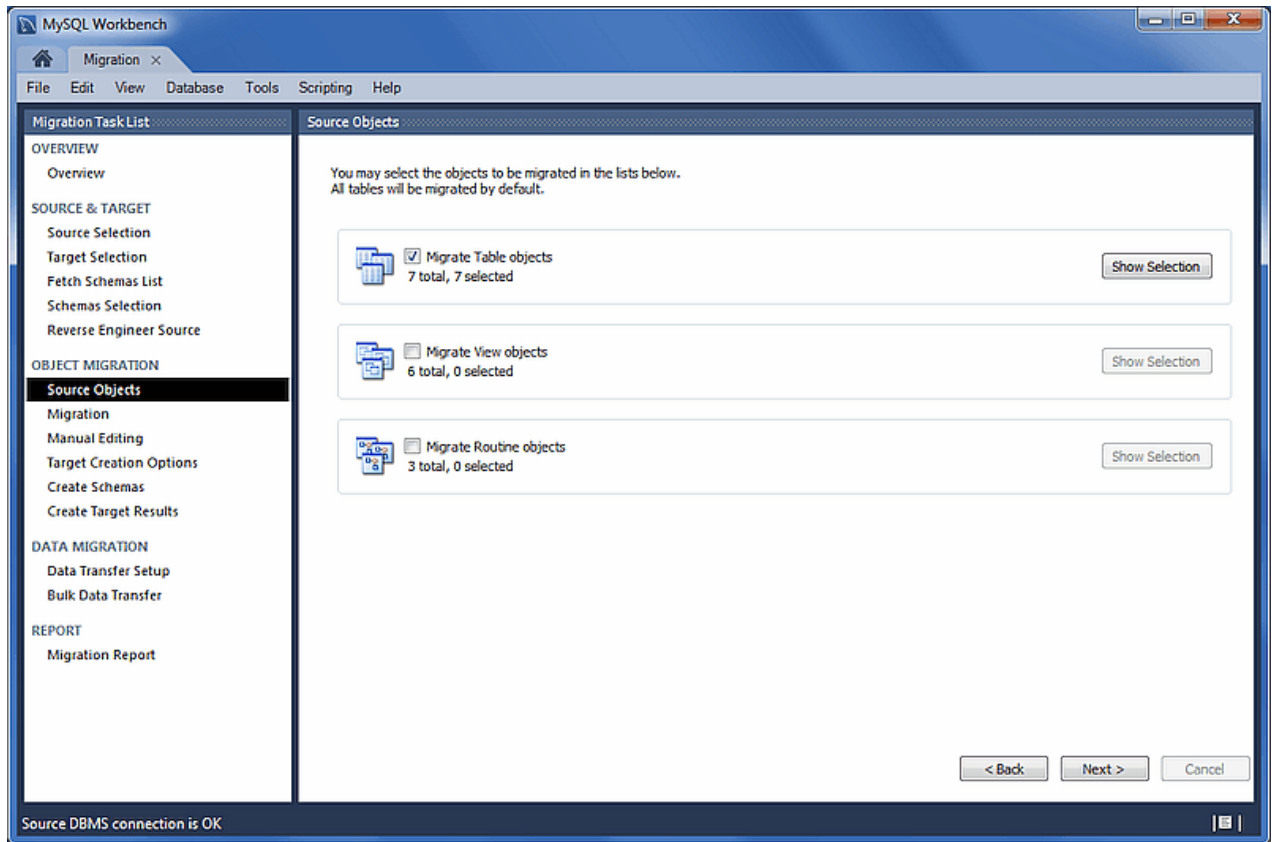
The source metadata is fetched from the source RDBMS, and reverse engineered. This is an automated and informational step that reports related errors, general log information, or both (see the figure that follows). View the logs and then click **Next** to continue.

Figure 10.54 MySQL Workbench migration: Reverse Engineer Source



10.8.4 Object Selection

Objects discovered by the **Reverse Engineer Source** stage are made available here. Valid objects include Table, View, and Routine objects, with only the Table objects being selected by default (see the figure that follows). Use the **Show Selection** button in order to disable individual table objects from being migrated.

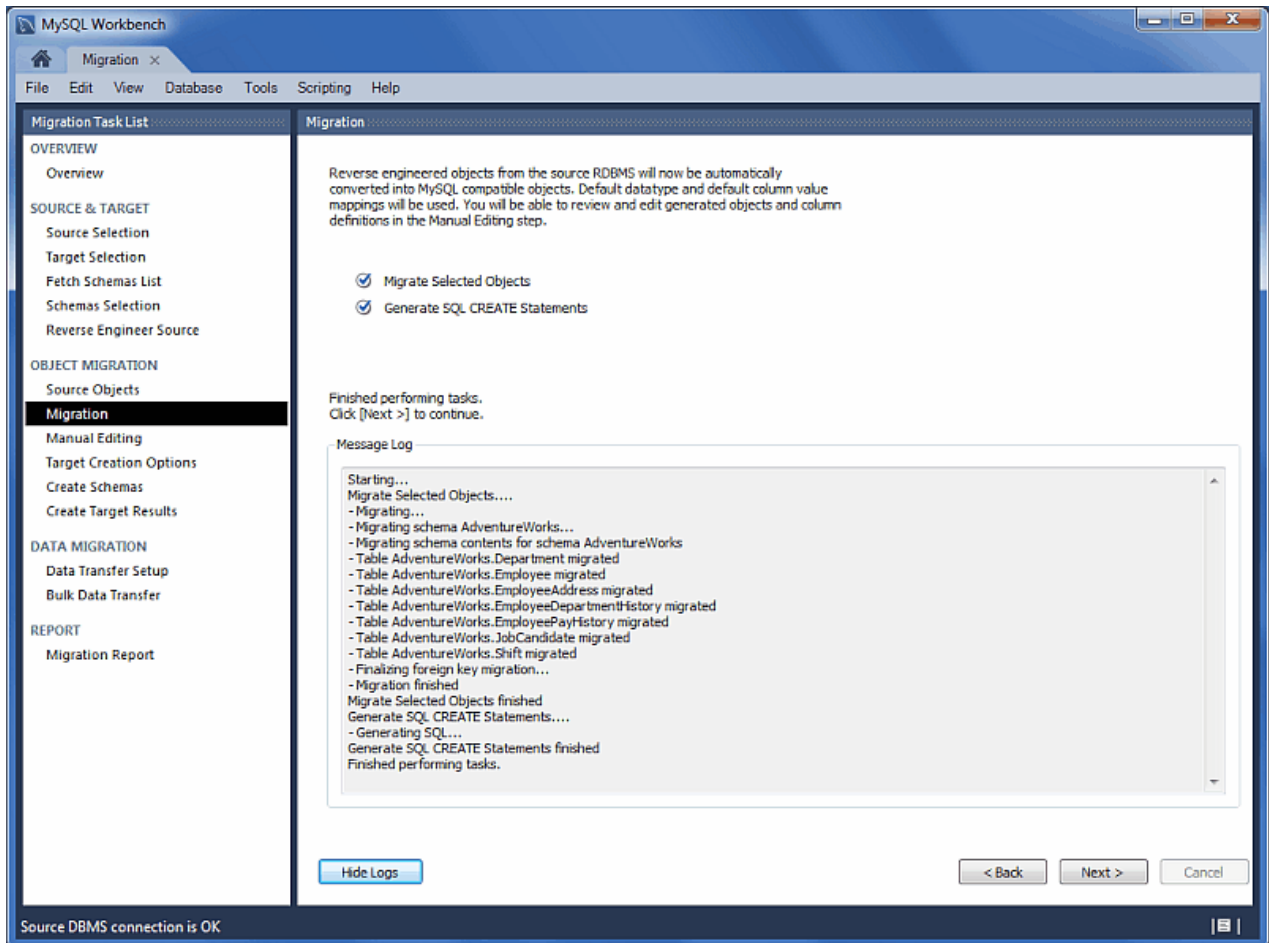
Figure 10.55 MySQL Workbench migration: Source Objects

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, as indicated in the next figure.

Figure 10.56 MySQL Workbench Migration: Migration

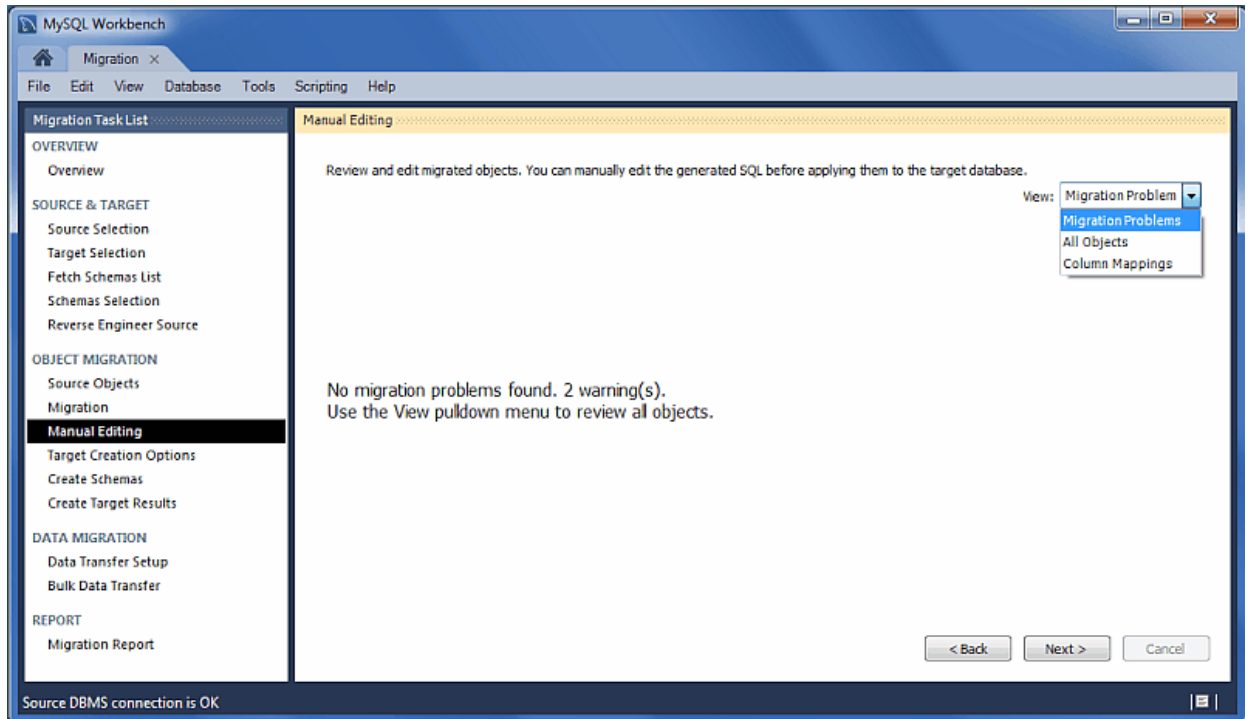


10.8.6 Manual Editing

Use the **View** select box to choose the section to edit (see the figure that follows). The **Show Code and Messages** button is available on every page and it shows the generated MySQL code that corresponds to the selected object.

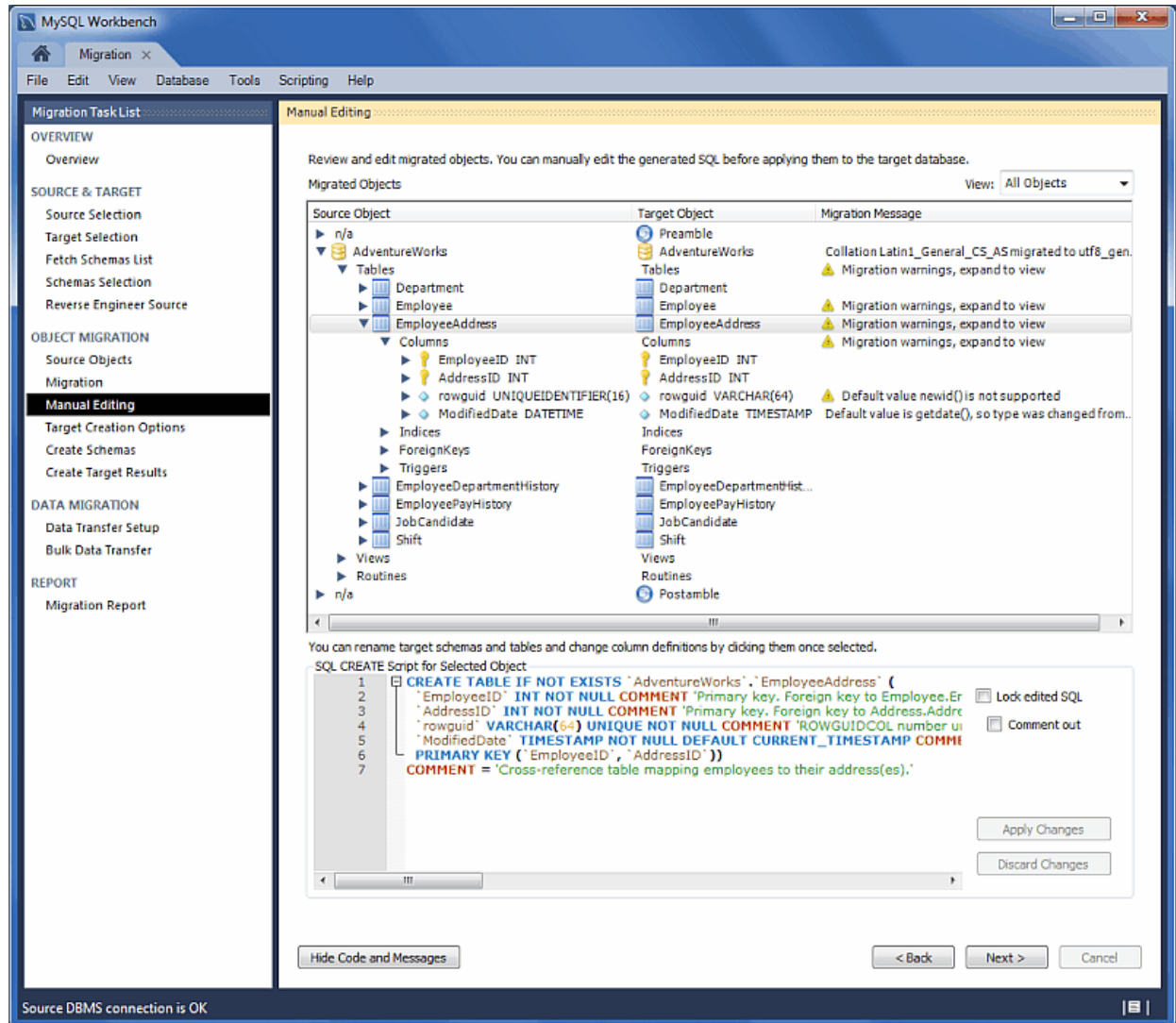
- **Migration Problems:** This either reports problems or displays "No mapping problems found." As the following figure shows, this is an informational screen.

Figure 10.57 MySQL Workbench migration: Manual Editing (Migration Problems)



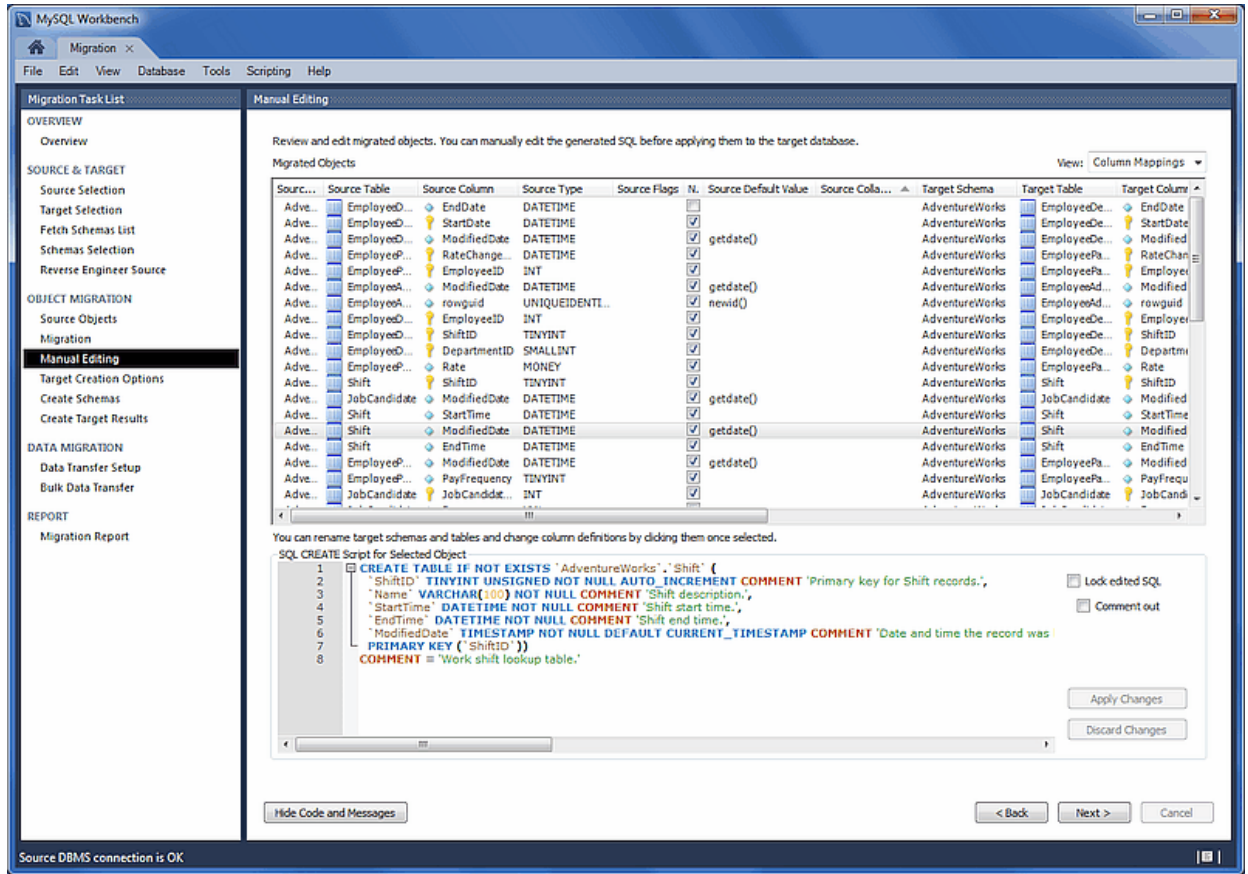
- **All Objects:** An object view that allows you to view and edit the object definitions. Double-click on a row to modify a target objects name (see the figure that follows).

Figure 10.58 MySQL Workbench Migration: Manual Editing (All Objects)



- **Column Mappings:** Shows all of the table column mappings and enables you to individually review and fix the mapping for all column types, default values, and other attributes. The following figure shows an example of the manual editing session.

Figure 10.59 MySQL Workbench Migration: Manual Editing (Column Mappings)

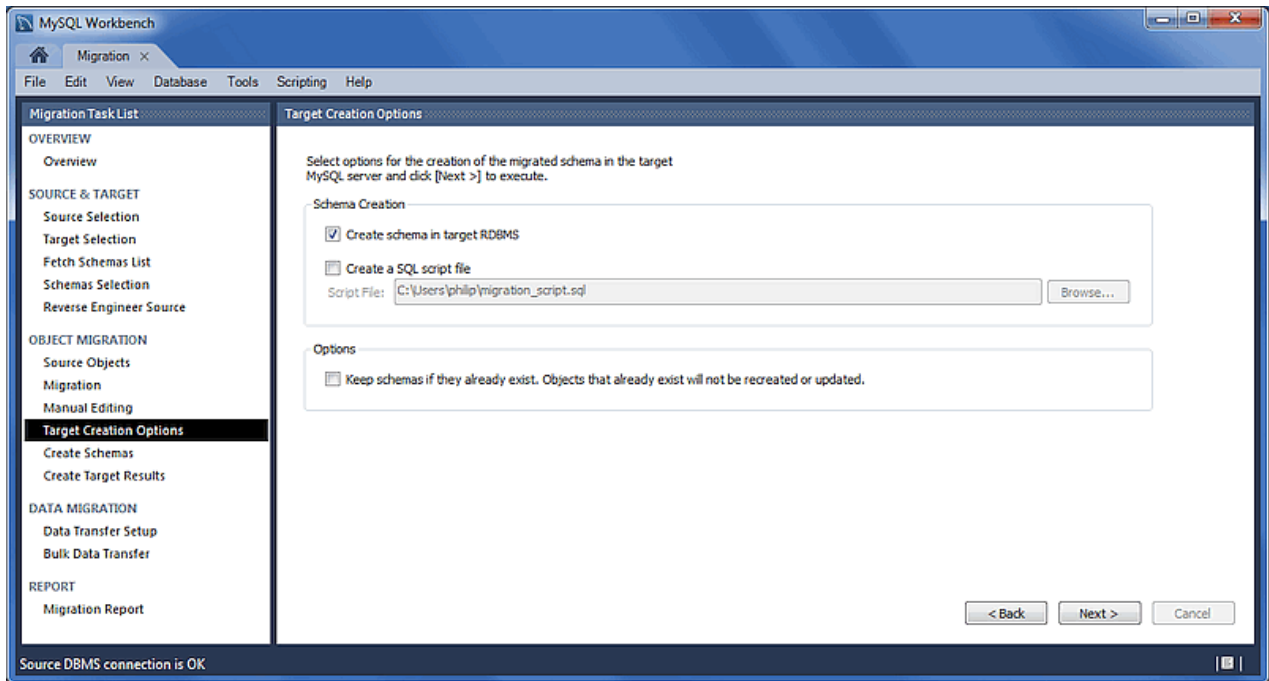


10.8.7 Target Creation Options

Defines addition settings for the target schema. Configuration options include:

- **Create schema in target RDBMS**
- **Create an SQL script file**
- **Keep the schemas if they already exist. Objects that already exist will not be recreated or update.**

Figure 10.60 MySQL Workbench Migration: Target Creation Options

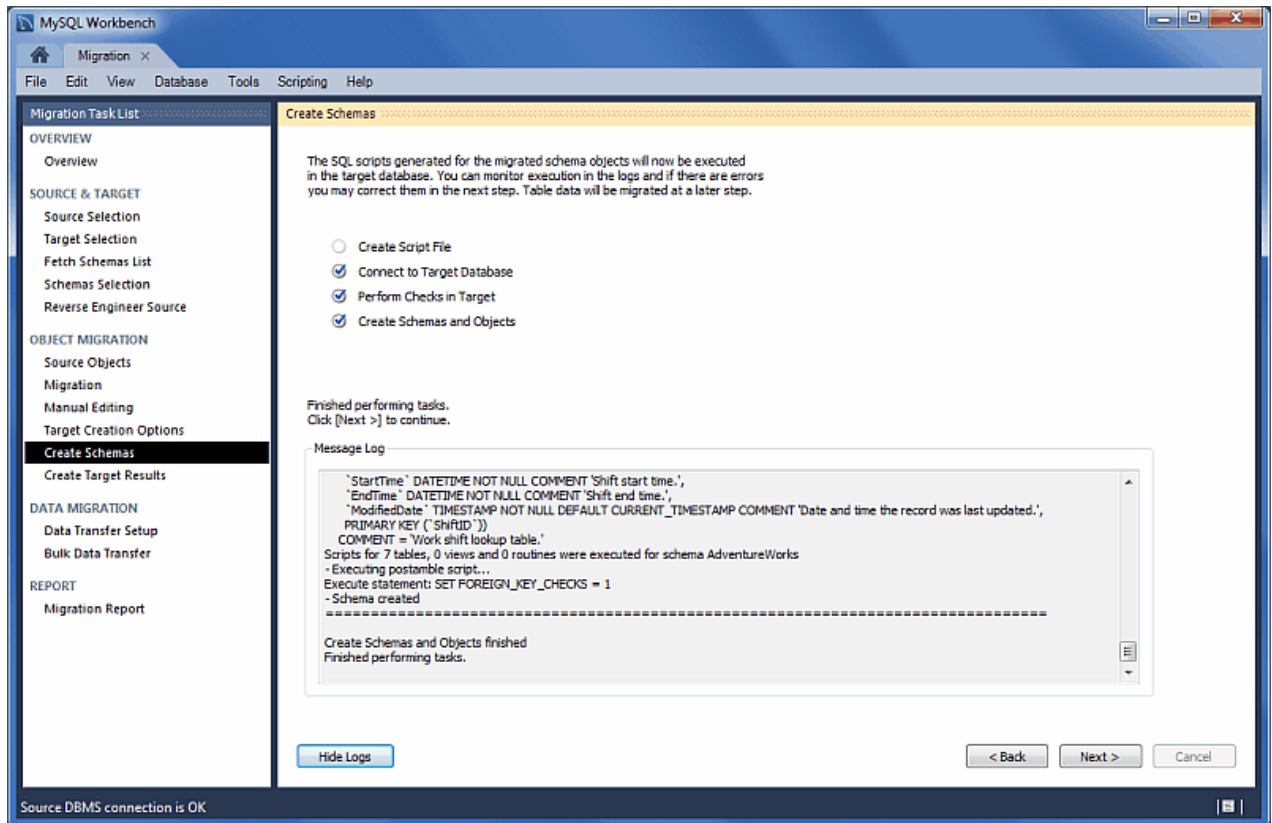


10.8.8 Schema Creation

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.

As the following figure shows, this is an automated step and the actions include: Create Script File, Connect to Target Database, and Create Schemas and Objects.

Figure 10.61 MySQL Workbench Migration: Create Schemas



10.8.9 Create Target Results

The generated objects are listed here, along with the error messages if any exist.

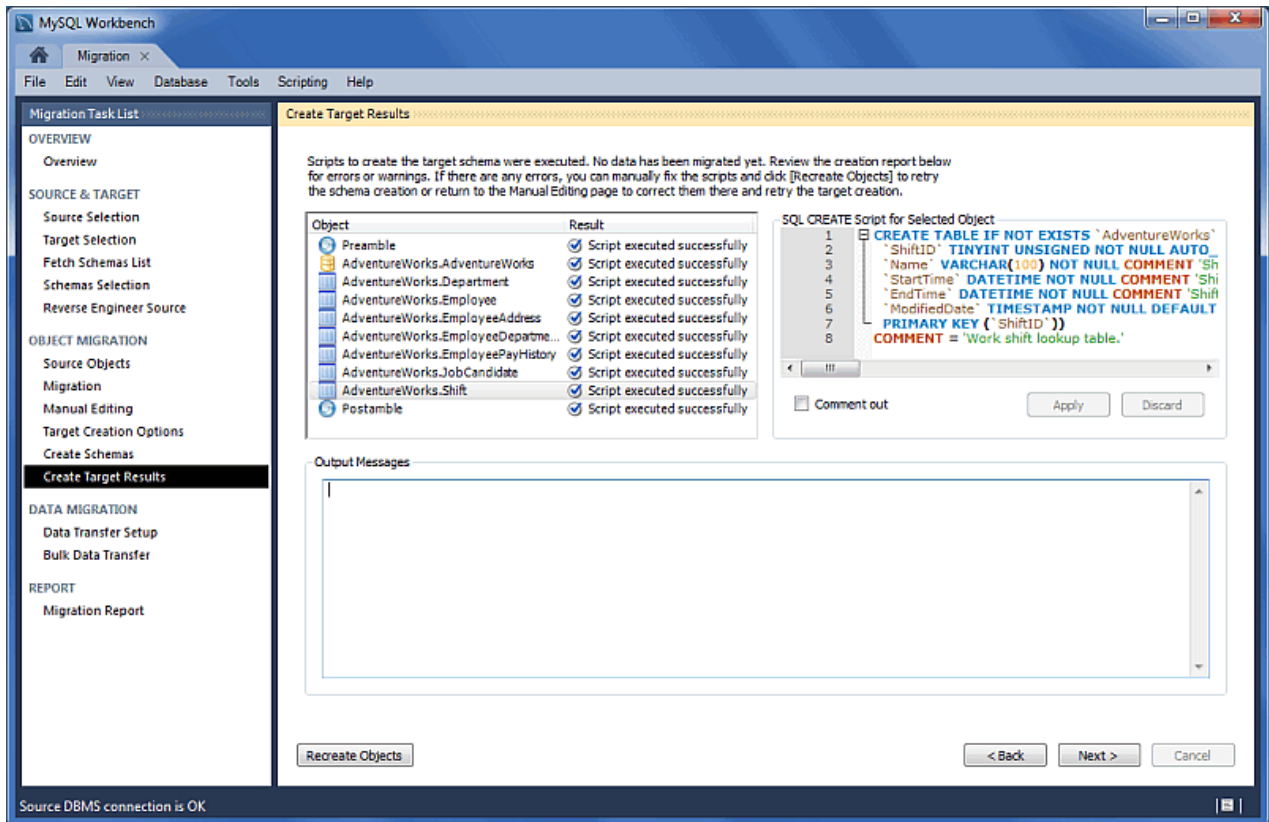
The migration code may also be viewed and edited here (see the figure that follows). To make changes, select an object, edit the query code, and click **Apply**. Repeat this process for each object that will be edited. Finally, click **Recreate Objects** to save the results.



Note

The **Recreate Objects** operation is required to save any changes here. It will then execute the previous migration step (**Create Schemas**) with the modified code, and then continue the migration process. This also means that the previously saved schema will be dropped.

Figure 10.62 MySQL Workbench Migration: Create Target Results



10.8.10 Data Transfer and Migration Setup

Transfers data from the source RDBMS to the target MySQL database (see the figure that follows). 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.



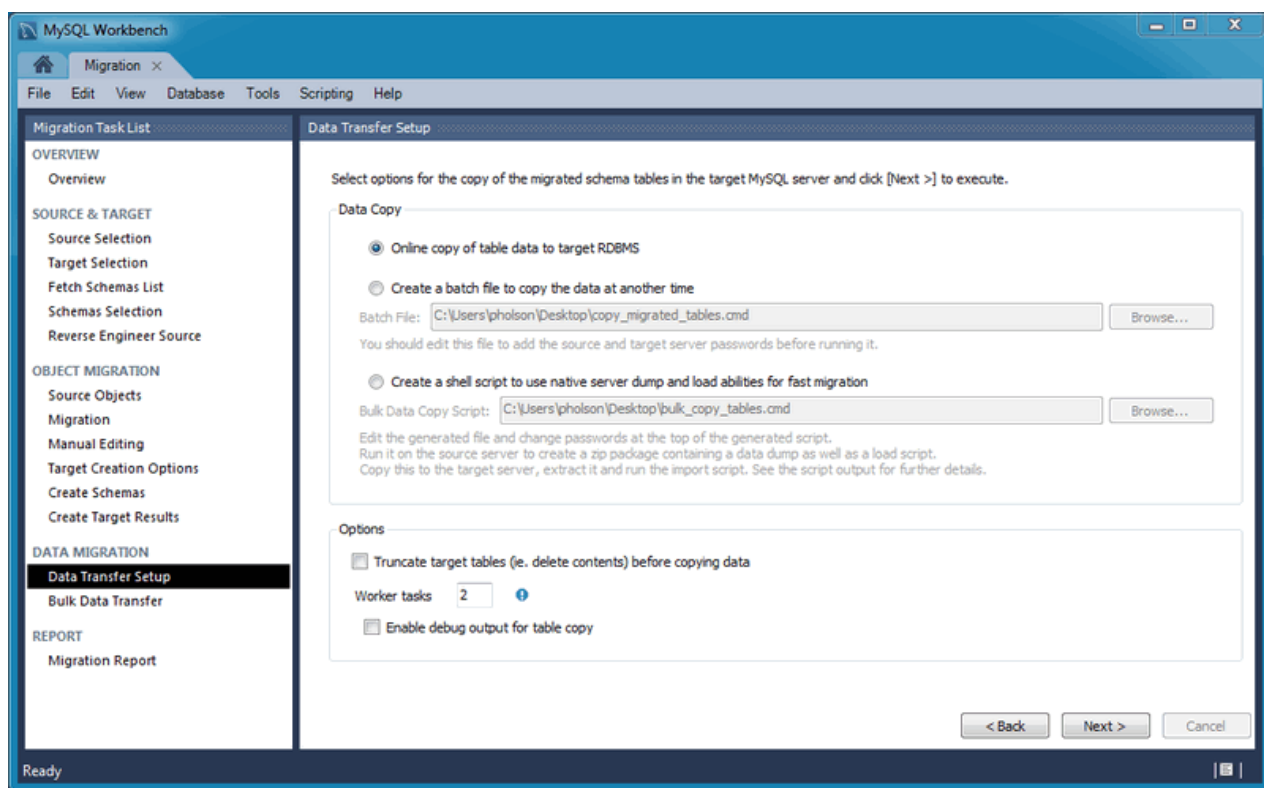
Note

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.

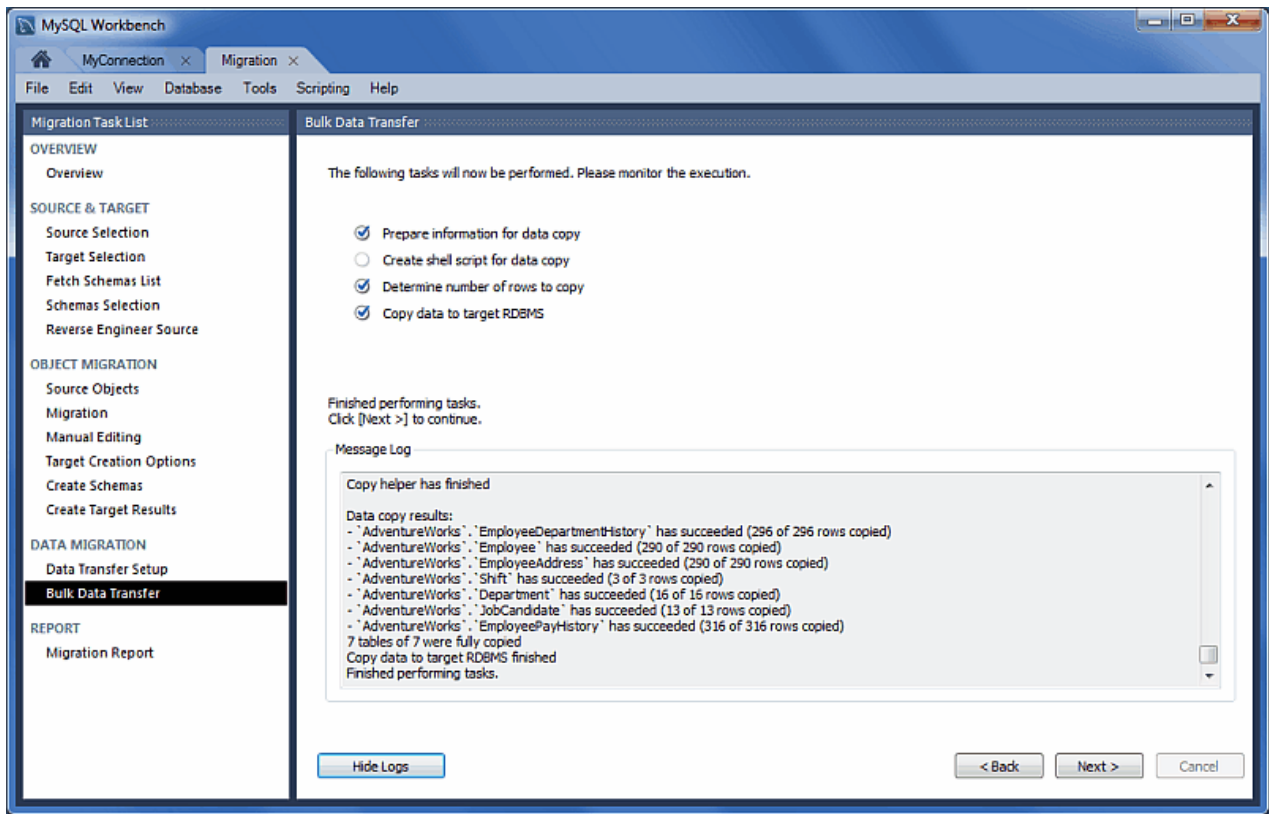
Figure 10.63 MySQL Workbench Migration: Data Transfer Setup



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 (see the figure that follows).

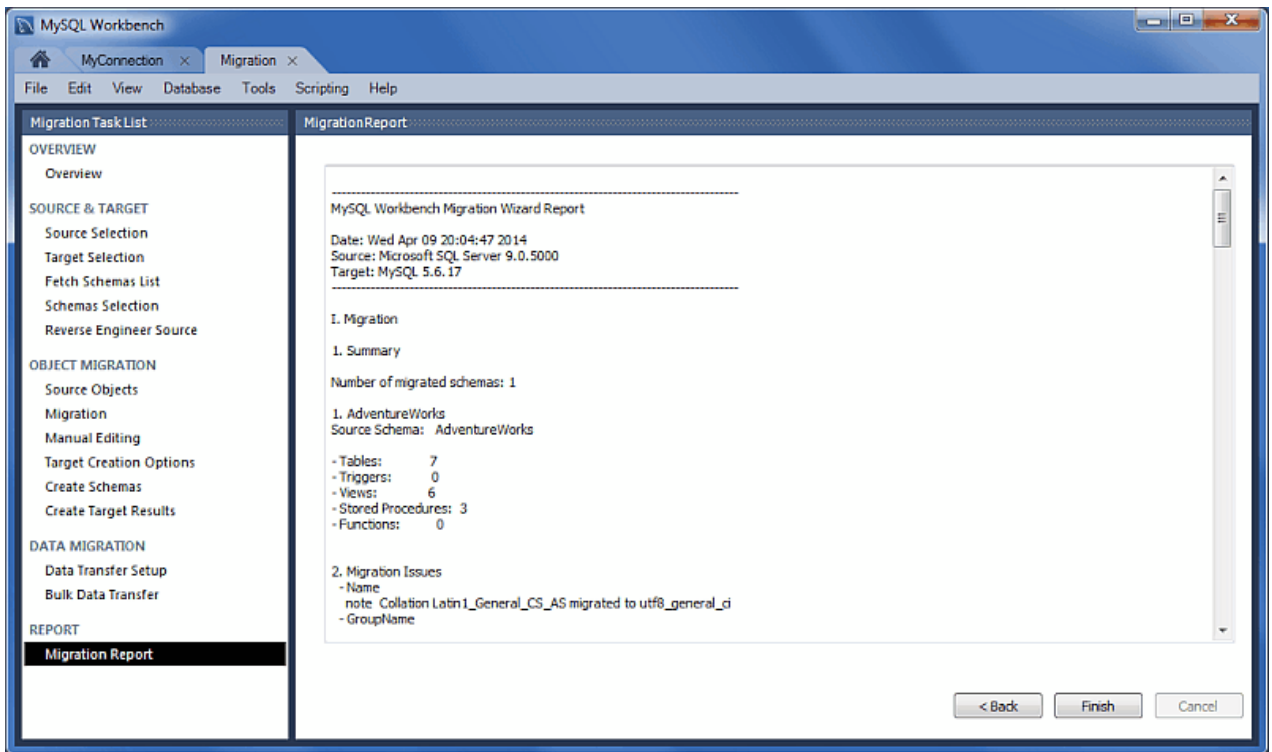
Figure 10.64 MySQL Workbench Migration: Bulk Data Transfer



10.8.12 Migration Report

Displays the final report that summarizes the migration process. The following figure shows an example of the migration report.

Figure 10.65 MySQL Workbench Migration: Migration Report



10.9 MySQL Workbench Migration Wizard FAQ

Frequently Asked Questions with answers.

10.9.1 While using the Postgresql psqldb driver, I see the following error: ('08001', '[08001] Already connected. (202) (SQLDriverConnect)') 419

10.9.1.While using the Postgresql psqldb 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?	421
A.2 Does MySQL Workbench support the TLSv1.2 protocol?	421
A.3 How do I create a MySQL database (schema) in MySQL Workbench?	421
A.4 Is there an easy way to select all data from a table, and then see the results?	422

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 should be identical, although this is not a common use case. For additional information about MySQL connections in MySQL Workbench, see [Chapter 5, Connections in MySQL Workbench](#).

A.2. Does MySQL Workbench support the TLSv1.2 protocol?


Because TLSv1.2 requires OpenSSL, support for this protocol is available for MySQL Workbench Commercial Editions, and not for the Community Edition (which is compiled using yaSSL and supports TLSv1.1 only).

A.3. 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.4. 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** from 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?	422
A.2 How do I copy my saved MySQL connections in Workbench to a different computer?	422
A.3 How can I view my MySQL Workbench query history?	422
A.4 Can I preserve a results tab rather than have it refresh every time I execute a statement?	422
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.	423
A.6 How does MySQL Workbench increase import performance?	423

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 5.3.5, "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 macOS, 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.

- 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?	423
A.2 I'm attempting to execute a DELETE query but the query fails with an "Error Code: 1175" error. How do I proceed?	424
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?	424
A.4 What do the column flag acronyms (PK, NN, UQ, BIN, UN, ZF, AI, G) in the MySQL Workbench Table Editor mean?	424

- 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` is read-only if `type` is not a primary key, but `SELECT id,`

`type FROM food` is 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, move the mouse pointer 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 effect.

- 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 alters the table column by assigning the checked constraints to the designated columns.

Move the pointer over an acronym to view a description, and see [Section 8.1.10.2, "Columns Tab"](#), and the documentation for the `CREATE TABLE` for additional details.

Data Management

- A.1 How do I import comma-separated values (CSV) data into MySQL using Workbench? 424
A.2 How do I export MySQL data to a plain text file with a format such as CSV, JSON, or XML? 424
A.3 How to export (save) a MySQL database to a text file? 425

- 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"](#).

- 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”](#).

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? 425

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 macOS, **Control** on other platforms. On macOS, the **Alt** key is **Option**.

There are keyboard shortcuts for the different menus in MySQL Workbench:

- [File Menu](#)
- [Edit Menu](#)
- [View Menu](#)
- [Arrange Menu](#)
- [Model Menu](#)
- [Query Menu](#)
- [Database Menu](#)
- [Scripting Menu](#)
- [Help Menu](#)
- [EER Diagram Mode](#)

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 (macOS)	Model, EER Diagram
Cut	Modifier+X	All
Copy	Modifier+C	All
Paste	Modifier+V	All
Delete	Modifier+Delete, Command+BackSpace (macOS)	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
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
Beautify Query	Modifier+B	SQL Editor
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 (macOS)	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 macOS)	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 macOS)	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 macOS)	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

Appendix C Extending Workbench

Table of Contents

C.1 GRT and Workbench Data Organization	431
C.2 Modules	432
C.3 Plugins and Tools	434
C.4 Adding a GUI to a Plugin Using MForms	435
C.5 The Workbench Scripting Shell	435
C.5.1 Exploring the Workbench Scripting Shell	436
C.5.2 The Shell Window	436
C.5.3 Files, Globals, Classes, Modules, and Notifications Tabs	438
C.6 Tutorial: Writing Plugins	443
C.6.1 Tutorial: Generate PHP Code to Create a Connection with PDO_MySQL	443
C.6.2 Tutorial: Generating Foreign Keys with MyISAM	445

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 schemas
- Create custom Workbench features

C.1 GRT and Workbench Data Organization

The Generic RunTime (GRT) is the internal system used by MySQL 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, schemas, 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.

Application Objects Tree (GRT Tree)

As mentioned previously, MySQL 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.

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 schemas.
wb.doc.physicalModels[0].catalog.schemata	List of schemas 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, perform 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**.

Table C.2 Default User Module File Location

Operating System	File Path
Windows	%AppData%\MySQL\Workbench\modules
macOS	~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 and 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
macOS	~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 / macOS. 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 Generic RunTime (GRT) 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

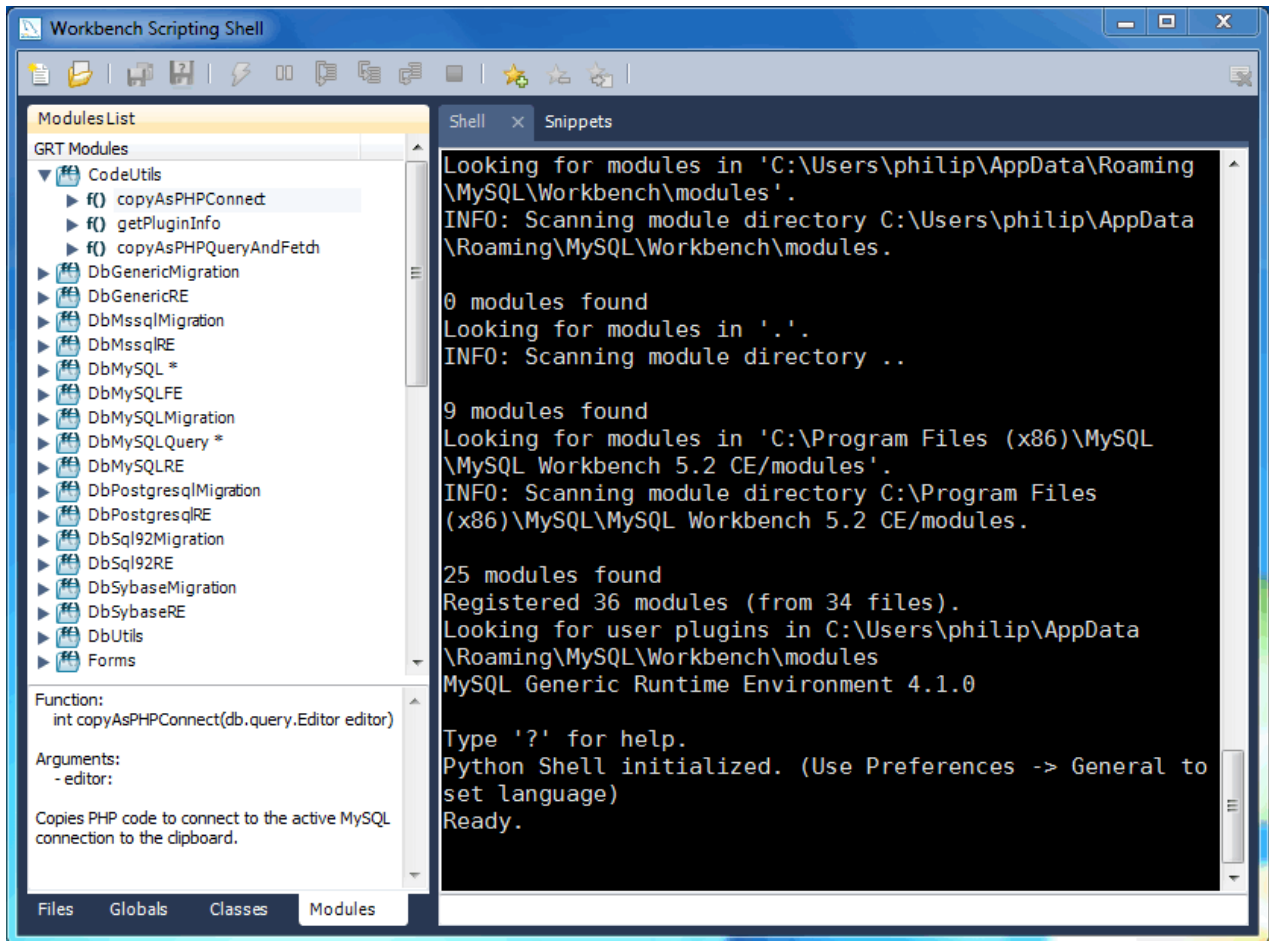
MySQL also has a product named MySQL Utilities, which is different than Workbench Scripting Shell.

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 macOS, or by clicking the shell button above the EER diagram navigator. The Workbench Scripting Shell will then open in a new dialog.

The following figure shows the Workbench Scripting Shell dialog.

Figure C.1 The Workbench Scripting Shell

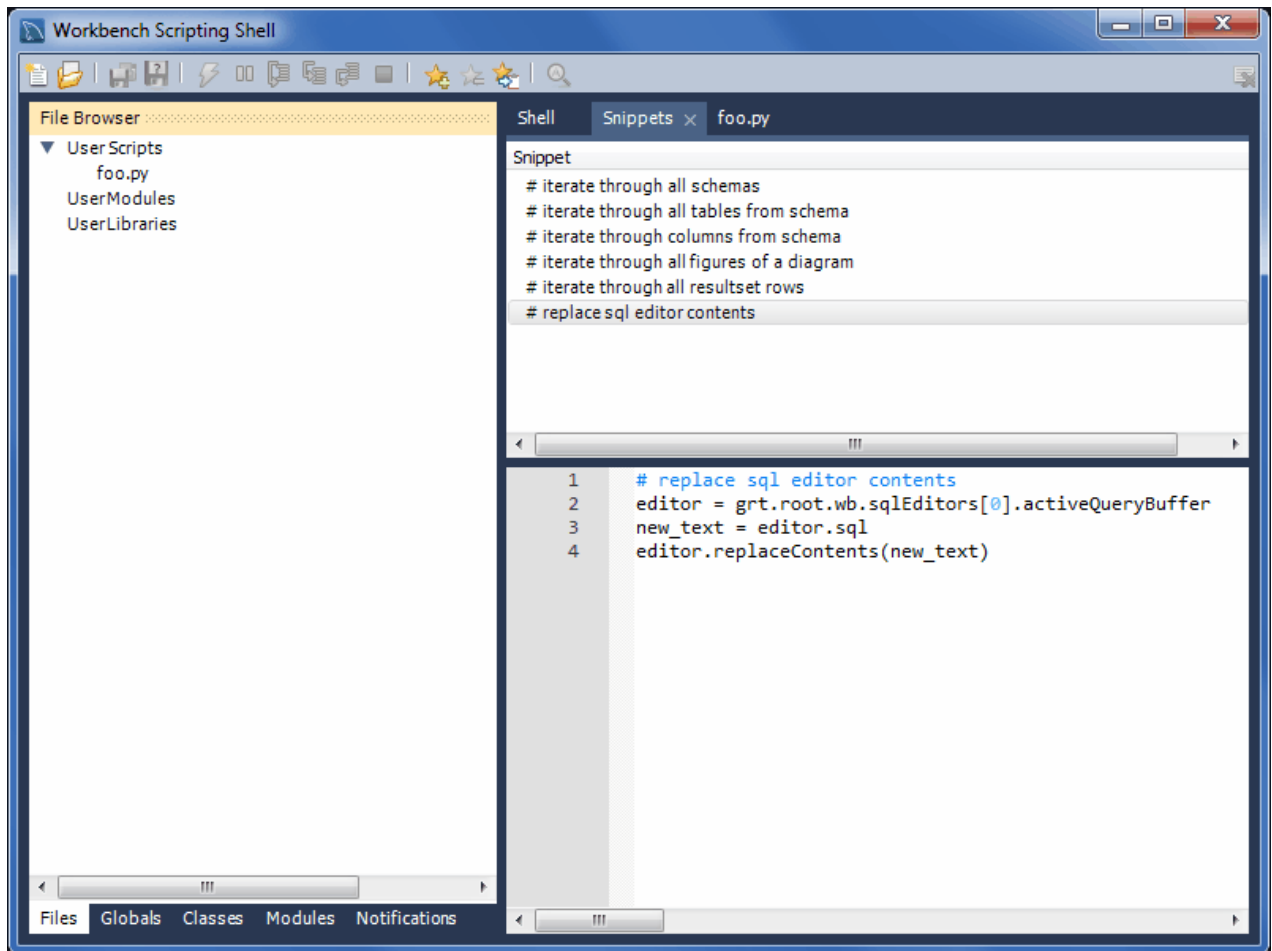


C.5.2 The Shell Window

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, which makes it easy to reuse and execute code in MySQL Workbench. The following figure shows the Snippets tab selected.

Figure C.2 The Workbench Scripting Shell: Snippets



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 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.

Files Tab

The **Files** tab lists folders and files for user-defined (custom) script files. The file-browser categories are **User Scripts**, **User Modules**, and **User Libraries**, as the following figure shows.

Figure C.3 The Workbench Scripting Shell tab: Files

By default, scripts are stored in the `scripts/` folder of your MySQL Workbench configuration folder. The following table lists the default location for each platform.

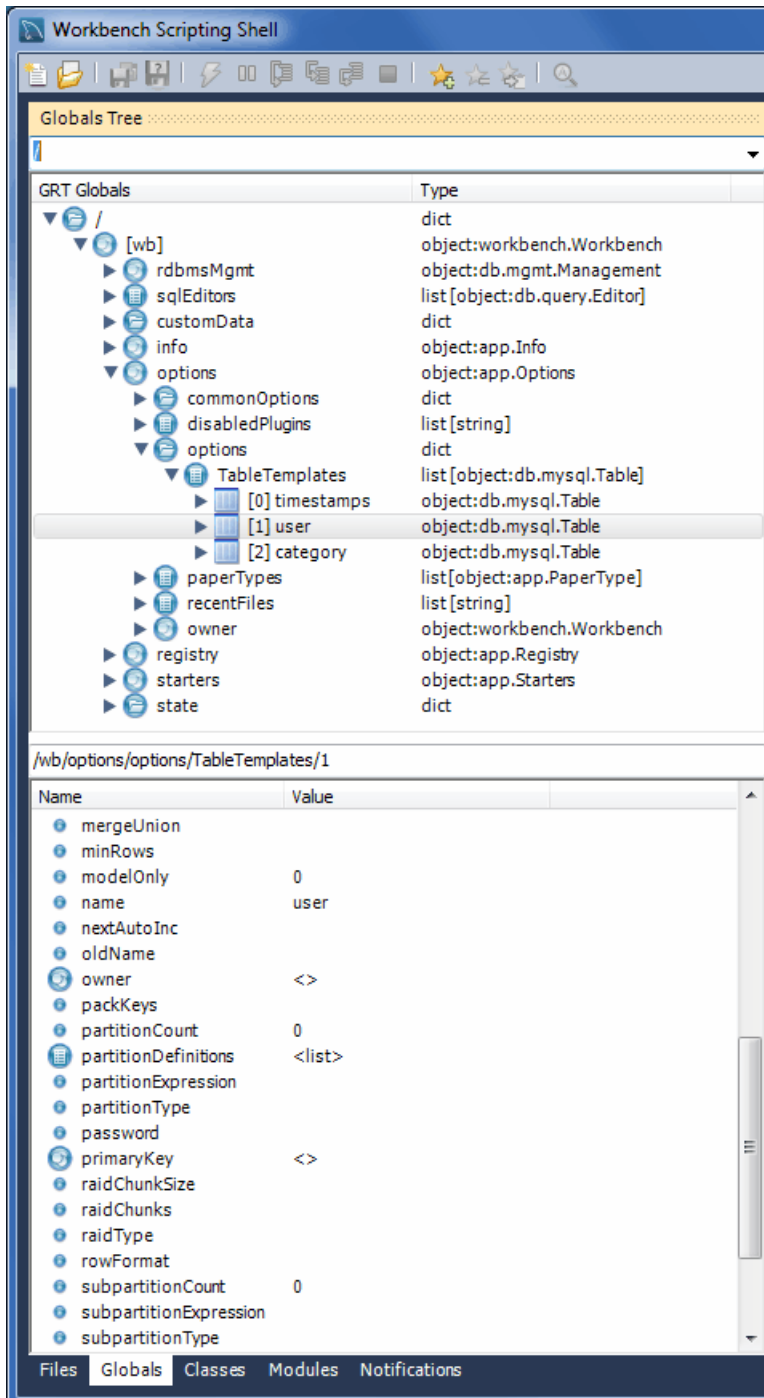
Table C.4 Default Scripts Location

Operating System	Default <code>scripts/</code> path
Linux	<code>~/ .mysql/workbench/scripts</code>
macOS	<code>~/Library/Application\ Support/MySQL/Workbench/scripts/</code>
Windows 7	<code>C:\Users\[user]\AppData\Roaming\MySQL\Workbench\scripts\</code>

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 (see the following figure). 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.

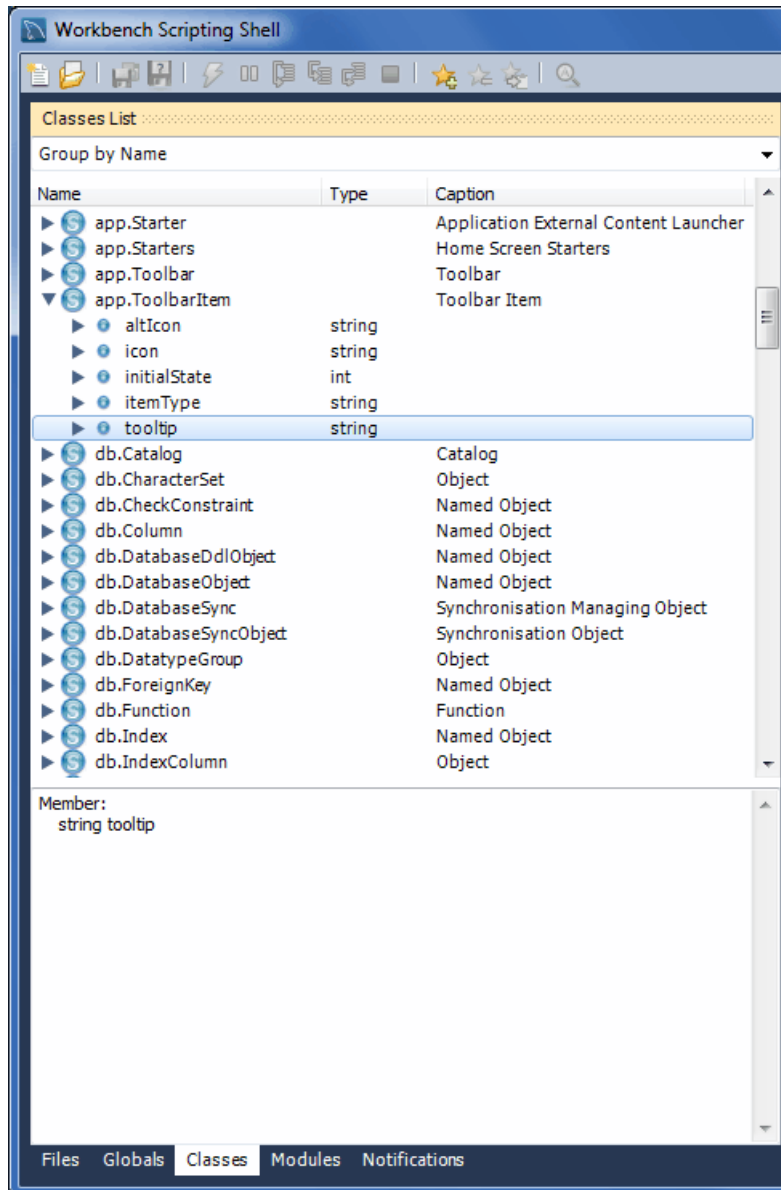
Figure C.4 The Workbench Scripting Shell Tab: Globals



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, as shown in the next figure.

Figure C.5 The Workbench Scripting Shell Tab: Classes



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

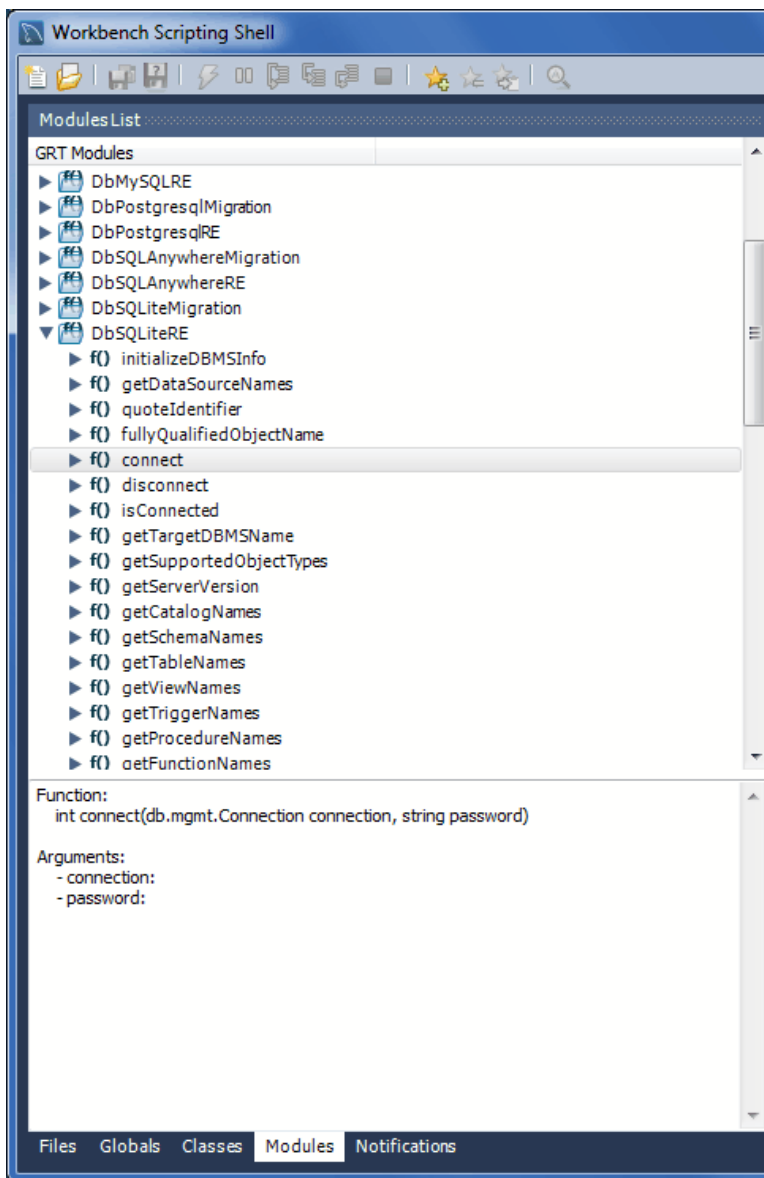
The default view for this tab is **Group By Name**. This view shows all the different objects arranged alphabetically. Click the **+** icon or double-click a package to show the properties of the struct.

If you switch to the hierarchical view, you will see `GrtObject`: the parent object from which all other objects are derived.

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, as the following figure shows. 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.

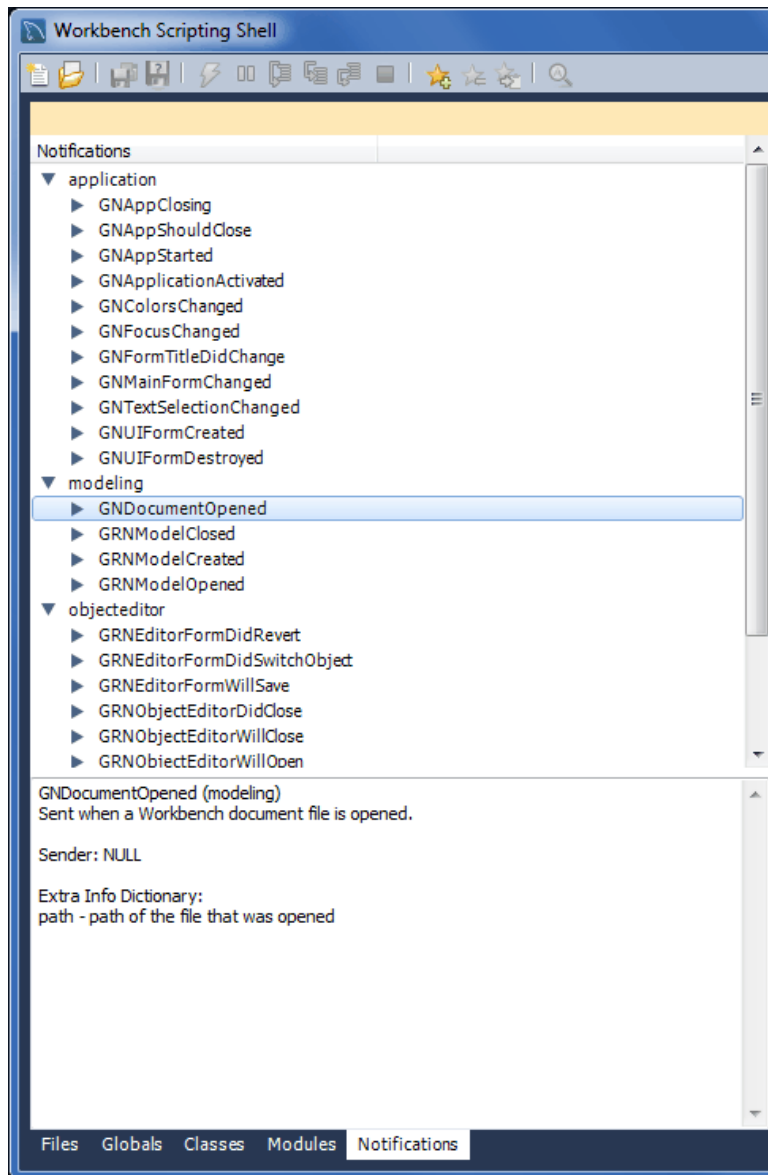
Figure C.6 The Workbench Scripting Shell Tab: Modules



Notifications Tab

The **Notification** tab includes the set of `notification` classes used by MySQL Workbench modules. Click a notification class for a description of its use, as demonstrated in the next figure.

Figure C.7 The Workbench Scripting Shell Tab: Notifications



C.6 Tutorial: Writing Plugins

The tutorials in this section demonstrate 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 tutorial shows how to generate code with the `PDO_MySQL` extension for PHP. You might choose a different extension or a different language altogether, so adjust the generated code accordingly.

To begin, review the plugin code shown in the example that follows.

```
# 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 generate PHP code for a connection, first install the plugin as follows:

1. Copy the plugin code into a new file. The file name used in this example is [php-pdo-connect_grt.py](#), but you can use a different name as long as [_grt.py](#) is the suffix.

2. Start MySQL Workbench. Click **Scripting** and then **Install Plugin/Module** from the menu to open a file browser. Select the plugin file created by the code in the previous step, `php-pdo-connect_grt.py` in this case.

**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.

3. When prompted, restart MySQL Workbench. This step generates a compiled bytecode file (`.pyc`) from your source file. In this example, it generates `php-pdo-connect_grt.pyc`.
4. After restarting MySQL Workbench, load the MySQL connection to use to generate the PHP code. From the menu, click **Tools, Utilities**, and then **MySQL PDO (Connect to Server)**, which is the `Caption` defined within the plugin code.

This action copies the generated PHP code into the clipboard on your system. The following connection example defines "sakila" as the default database in the generated code.

```
$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.createGuesseForeignKeys",
    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.
- A copy of 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, by using a command-line parameter, or by specifying the log level as a preference option (see [Other Preferences](#)).

The environment variable, command-line variant, and preference option accept a single error level, but enabling a more verbose option implicitly enables the levels below it. For example, passing in [info](#) also enables the [error](#) and [warning](#) levels.

- Environment variable: [WB_LOG_LEVEL](#)

Command line option: `--log-level`



Note

If both the command line and environment variable are set, the command line takes precedence.

For example:

```
# Microsoft Windows
$> cd "C:\Program Files (x86)\MySQL\MySQL Workbench CE 8.0.38\"
$> MySQLWorkbench.exe --log-level=debug3

# macOS
$> cd /Applications
```

```
$> MySQLWorkbench --log-level=debug3
# Linux (Ubuntu)
$> cd /usr/bin
$> 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 10.
- 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.

macOS

- 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/log`
- For a crash, we might ask for a stack trace that can be 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, and bundles them so they can be used to perform macro operations with a single command. They 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 for Python v3.1.

The utilities are available under the GPLv2 license, and are extendable using the supplied library.

Installing The MySQL Utilities

MySQL Utilities development is managed elsewhere, and requires 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 `PATH` to determine if the MySQL Utilities are installed.

Opening MySQL Utilities From MySQL Workbench

To open the MySQL Utility `mysqluc` (MySQL Utilities Unified Console) from MySQL Workbench, click **Tools** and then **Start Shell for MySQL Utilities** from the menu. The following output shows the MySQL Utilities console window with the `help` command executed.

```
Welcome to the MySQL Utilities Client (mysqluc) version 1.6.5
Copyright (c) 2010, 2017 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 a commercial license and the GNU General Public License does not apply.
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).