MySQL and Windows

Abstract

This is the MySQL Windows extract from the MySQL 5.1 Reference Manual.

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

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

Document generated on: 2016-08-17 (revision: 48587)

Table of Contents

Preface and Legal Notices

This is the MySQL Windows extract from the MySQL 5.1 Reference Manual.

Legal Notices

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at

http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

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 Installing MySQL on Microsoft Windows

Table of Contents

> **Important**
>
> The MySQL server 5.1 branch is old and not recommended for new installations. Consider installing the latest stable branch, which today is MySQL server 5.7.

MySQL is available for Microsoft Windows, for both 32-bit and 64-bit versions. For supported Windows platform information, see http://www.mysql.com/support/supportedplatforms/database.html.

It is possible to run MySQL as a standard application or as a Windows service. By using a service, you can monitor and control the operation of the server through the standard Windows service management tools. For more information, see Section 1.6.7, “Starting MySQL Server as a Microsoft Windows Service”.

Generally, you should install MySQL on Windows using an account that has administrator rights. Otherwise, you may encounter problems with certain operations such as editing the `PATH` environment variable or accessing the `Service Control Manager`. Once installed, MySQL does not need to be executed using a user with Administrator privileges.

For a list of limitations on the use of MySQL on the Windows platform, see Windows Platform Limitations.

In addition to the MySQL Server package, you may need or want additional components to use MySQL with your application or development environment. These include, but are not limited to:

- To connect to the MySQL server using ODBC, you must have a Connector/ODBC driver. For more information, including installation and configuration instructions, see MySQL Connector/ODBC Developer Guide.
- To use MySQL server with .NET applications, you must have the Connector/Net driver. For more information, including installation and configuration instructions, see MySQL Connector/Net Developer Guide.

MySQL distributions for Windows can be downloaded from http://dev.mysql.com/downloads/. See How to Get MySQL.

MySQL for Windows is available in several distribution formats, detailed here. Generally speaking, you should use a binary distribution that includes an installer. It is simpler to use than the others, and you need no additional tools to get MySQL up and running. The installer for the Windows version of MySQL, combined with a GUI Config Wizard, automatically installs MySQL, creates an option file, starts the server, and secures the default user accounts.

- Binary installer distribution. The installable distribution comes packaged as a Microsoft Windows Installer (MSI) package that you can install manually or automatically on your systems. Two formats are available, an essentials package that contains all the files you need to install and configure MySQL, but no additional components, and a complete package that includes MySQL, configuration tools, benchmarks and other components. For more information on the specific differences, see Section 1.2, “Choosing the Installation Package for Microsoft Windows”

  For instructions on installing MySQL using one of the MSI installation packages, see Section 1.4, “Installing MySQL on Microsoft Windows Using an MSI Package”.

- The standard binary distribution (packaged as a Zip file) contains all of the necessary files that you unpack into your chosen location. This package contains all of the files in the full Windows MSI Installer package, but does not include an installation program.

  For instructions on installing MySQL using the Zip file, see Section 1.6, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”.

- The source distribution format contains all the code and support files for building the executables using the Visual Studio compiler system.

  For instructions on building MySQL from source on Windows, see Chapter 2, *Installing MySQL from Source on Windows*.

MySQL on Windows considerations:

- **Large Table Support**

  If you need tables with a size larger than 4GB, install MySQL on an NTFS or newer file system. Do not forget to use `MAX_ROWS` and `AVG_ROW_LENGTH` when you create tables. See CREATE TABLE Syntax.

- **MySQL and Virus Checking Software**

Virus-scanning software such as Norton/Symantec Anti-Virus on directories containing MySQL data and temporary tables can cause issues, both in terms of the performance of MySQL and the virus-scanning software misidentifying the contents of the files as containing spam. This is due to the fingerprinting mechanism used by the virus-scanning software, and the way in which MySQL rapidly updates different files, which may be identified as a potential security risk.

After installing MySQL Server, it is recommended that you disable virus scanning on the main directory (`datadir`) used to store your MySQL table data. There is usually a system built into the virus-scanning software to enable specific directories to be ignored.

In addition, by default, MySQL creates temporary files in the standard Windows temporary directory. To prevent the temporary files also being scanned, configure a separate temporary directory for MySQL temporary files and add this directory to the virus scanning exclusion list. To do this, add a configuration option for the `tmpdir` parameter to your `my.ini` configuration file. For more information, see Section 1.6.2, “Creating an Option File”.

1.1 MySQL Installation Layout on Microsoft Windows

For MySQL 5.1 on Windows, the default installation directory is `C:\Program Files\MySQL\MySQL Server 5.1`. Some Windows users prefer to install in `C:\mysql`, the directory that formerly was used as the default. However, the layout of the subdirectories remains the same.

For MySQL 5.1.23 and earlier, all of the files are located within the parent directory, using the structure shown in the following table.

Table 1.1 Installation Layout for Windows Using MySQL 5.1.23 and Earlier

Directory	Contents of Directory
`bin`	Client programs and the `mysqld` server
`data`	Log files, databases
`examples`	Example programs and scripts
`include`	Include (header) files
`lib`	Libraries
`scripts`	Utility scripts
`share`	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation

For MySQL 5.1.24 and later, the default location of data directory was changed. The remainder of the directory structure remains the same.

Table 1.2 Installation Layout for Microsoft Windows using MySQL 5.1.24 and later

Directory	Contents of Directory	Notes
`bin`, `scripts`	`mysqld` server, client and utility programs	
`%ALLUSERSPROFILE%\MySQL\MySQL Server 5.1\`	Log files, databases (Windows XP, Windows Server 2003)	The Windows system variable `%ALLUSERSPROFILE%` defaults to `C:\Documents and Settings\All Users\Application Data`

Directory	Contents of Directory	Notes
`%PROGRAMDATA%\MySQL\MySQL Server 5.1\`	Log files, databases (Vista, Windows 7, Windows Server 2008, and newer)	The Windows system variable `%PROGRAMDATA%` defaults to `C:\ProgramData`
`examples`	Example programs and scripts	
`include`	Include (header) files	
`lib`	Libraries	
`share`	Miscellaneous support files, including error messages, character set files, sample configuration files, SQL for database installation	

1.2 Choosing the Installation Package for Microsoft Windows

For MySQL 5.1, there are multiple installation package formats to choose from when installing MySQL on Windows.

> **Note**
>
> Using MySQL Installer is the recommended installation method for Microsoft Windows users. The MySQL Server 5.1 release does not include its own MySQL Installer release, but a MySQL Installer version 5.5 and above can optionally install MySQL Server 5.1. Follow the standard Installing MySQL on Microsoft Windows Using MySQL Installer documentation but choose **Custom Install** after executing it. A MySQL Server 5.1 option will be available, and choosing it will cause MySQL Installer to download it for you.

Table 1.3 Microsoft Windows MySQL Installation package comparison

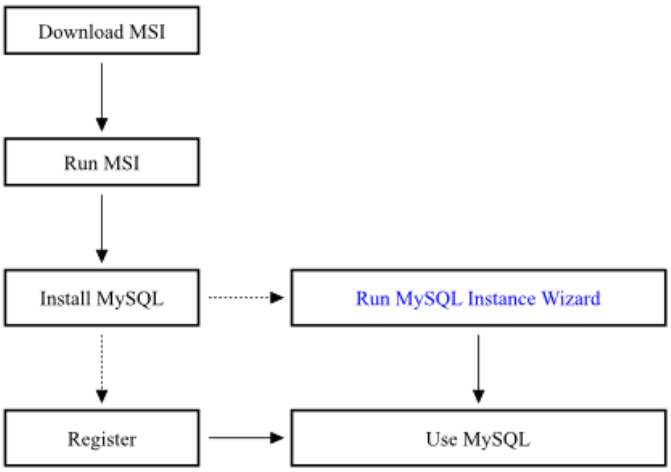
	Packaging		
Feature	**Essentials**	**Complete**	**Zip (No-install)**
Installer	Yes	Yes	No
Directory-only			
MySQL Server Instance Config Wizard	Yes	Yes	No
Test Suite	No	Yes	Yes
MySQL Server	Yes	Yes	Yes
MySQL Client Programs	Yes	Yes	Yes
C Headers/Libraries	Yes	Yes	Yes
Embedded Server	No	Optional	Yes
Scripts and Examples	No	Optional	Yes

In the above table:

- *Yes* indicates that the component is installed by default.
- *No* indicates that the component is not installed or included.
- *Optional* indicates that the component is included with the package, but not installed unless explicitly requested using the Custom installation mode.

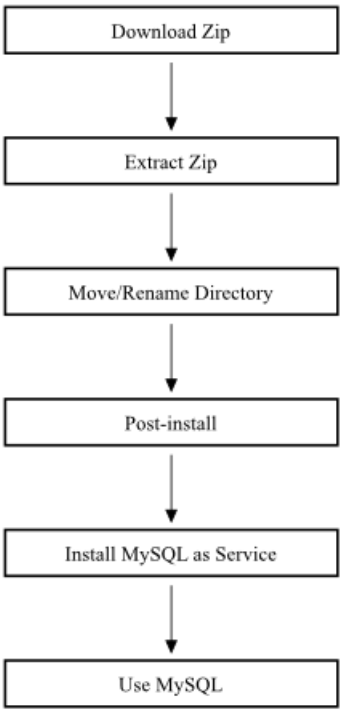
The workflow for installing using the MSI installer is shown here:

Figure 1.1 Installation Workflow for Windows Using MSI



The workflow for installing using the Zip package is shown here:

Figure 1.2 Installation Workflow for Windows Using Zip



> **Note**
>
> For the Essentials and Complete packages in the MSI installer, you can select individual components to be installed by using the Custom mode, including disable the components confiurated for installation by default.

Full details on the components are suggested uses are provided here for reference:

- **Windows Essentials**: This package has a file name similar to `mysql-essential-5.1.73-win32.msi` and is supplied as a Microsoft Installer (MSI) package. The package includes the minimum set of files needed to install MySQL on Windows, including the MySQL Server Instance Config Wizard. This package does not include optional components such as the embedded server, developer headers and libraries or benchmark suite.

  To install using this package, see Section 1.4, “Installing MySQL on Microsoft Windows Using an MSI Package”.

- **Windows MSI Installer (Complete)**: This package has a file name similar to `mysql-5.1.73-win32.msi` and contains all files needed for a complete Windows installation, including the MySQL Server Instance Config Wizard. This package includes optional components such as the embedded server and benchmark suite.

  To install using this package, see Section 1.4, “Installing MySQL on Microsoft Windows Using an MSI Package”.

- **Without installer**: This package has a file name similar to `mysql-noinstall-5.1.73-win32.zip` and contains all the files found in the Complete install package, with the exception of the MySQL Server Instance Config Wizard. This package does not include an automated installer, and must be manually installed and configured.

The Essentials package is recommended for most users. Both the Essentials and Complete distributions are available as an `.msi` file for use with the Windows Installer. The Noinstall distribution is packaged as a Zip archive. To use a Zip archive, you must have a tool that can unpack `.zip` files.

When using the MSI installers you can automate the installation process. For more information, see Section 1.4.2, “Automating MySQL Installation on Microsoft Windows Using the MSI Package”. To automate the creation of a MySQL instance, see Section 1.5.13, “MySQL Server Instance Config Wizard: Creating an Instance from the Command Line”.

Your choice of install package affects the installation process you must follow. If you choose to install either an Essentials or Complete install package, see Section 1.4, “Installing MySQL on Microsoft Windows Using an MSI Package”. If you choose to install a Noinstall archive, see Section 1.6, “Installing MySQL on Microsoft Windows Using a noinstall Zip Archive”.

1.3 MySQL Notifier

MySQL Notifier is a tool that enables you to monitor and adjust the status of your local and remote MySQL server instances through an indicator that resides in the system tray. MySQL Notifier also gives quick access to MySQL Workbench through its context menu.

The MySQL Notifier is installed by MySQL Installer, and (by default) will start-up when Microsoft Windows is started.

To install, download and execute the MySQL Installer, be sure the MySQL Notifier product is selected, then proceed with the installation. See the MySQL Installer manual for additional details.

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

Visit the MySQL Notifier forum for additional MySQL Notifier help and support.

Features include:

- Start, Stop, and Restart instances of the MySQL Server.

- Automatically detects (and adds) new MySQL Server services. These are listed under **Manage Monitored Items**, and may also be configured.
- The Tray icon changes, depending on the status. It's green if all monitored MySQL Server instances are running, or red if at least one service is stopped. The **Update MySQL Notifier tray icon based on service status** option, which dictates this behavior, is enabled by default for each service.
- Links to other applications like MySQL Workbench, MySQL Installer, and the MySQL Utilities. For example, choosing **Manage Instance** will load the MySQL Workbench Server Administration window for that particular instance.
- If MySQL Workbench is also installed, then the **Manage Instance** and **SQL Editor** options are available for local (but not remote) MySQL instances.
- Monitors both local and remote MySQL instances.

1.3.1 MySQL Notifier Usage

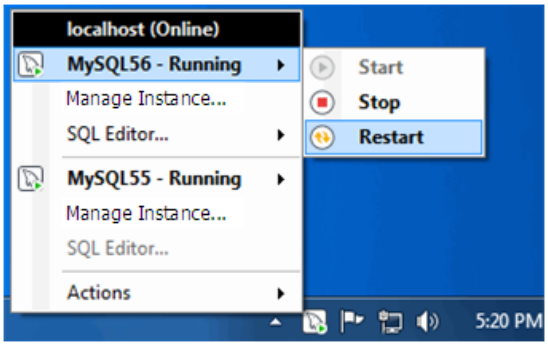
MySQL Notifier resides in the system tray and provides visual status information for your MySQL server instances. A green icon is displayed at the top left corner of the tray icon if the current MySQL server is running, or a red icon if the service is stopped.

MySQL Notifier automatically adds discovered MySQL services on the local machine, and each service is saved and configurable. By default, the **Automatically add new services whose name contains** option is enabled and set to `mysql`. Related **Notifications Options** include being notified when new services are either discovered or experience status changes, and are also enabled by default. And uninstalling a service will also remove the service from MySQL Notifier.

Clicking the system tray icon will reveal several options, as the follow figures show:

The Service Instance menu is the main MySQL Notifier window, and enables you to Stop, Start, and Restart the MySQL server.

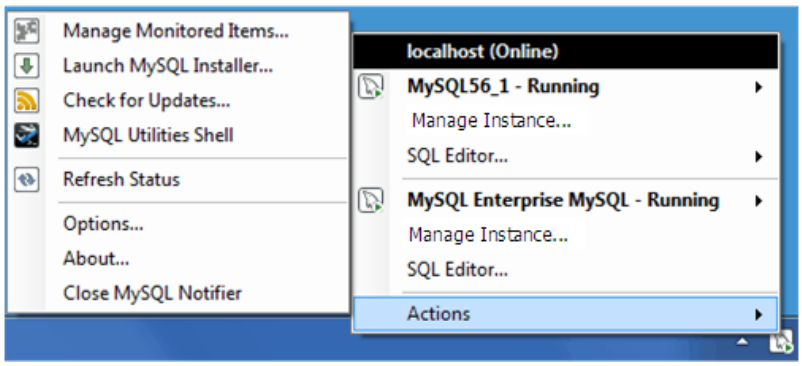
Figure 1.3 MySQL Notifier Service Instance menu



The **Actions** menu includes several links to external applications (if they are installed), and a **Refresh Status** option to manually refresh the status of all monitored services (in both local and remote computers) and MySQL instances.

> **Note**
>
> The main menu will not show the **Actions** menu when there are no services being monitored by MySQL Notifier.

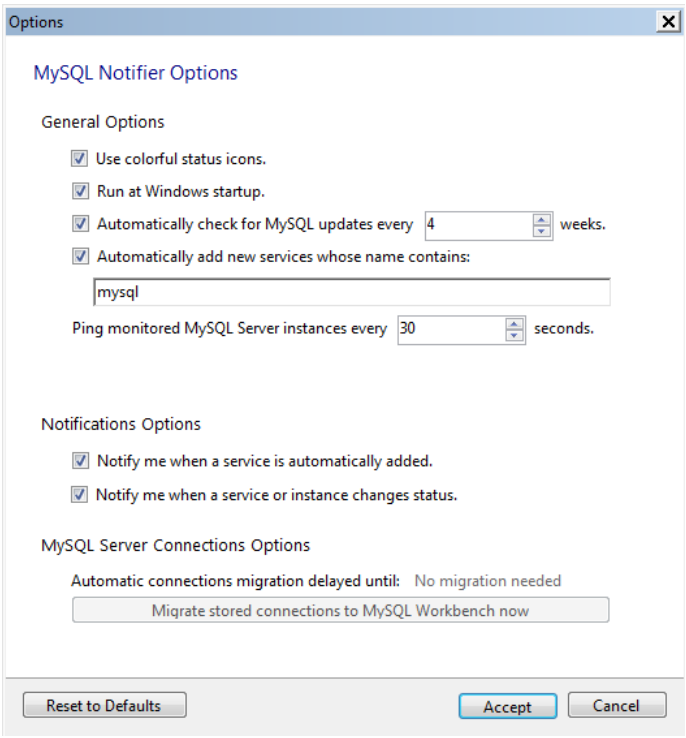
Figure 1.4 MySQL Notifier Actions menu



The **Actions**, **Options** menu configures MySQL Notifier and includes options to:

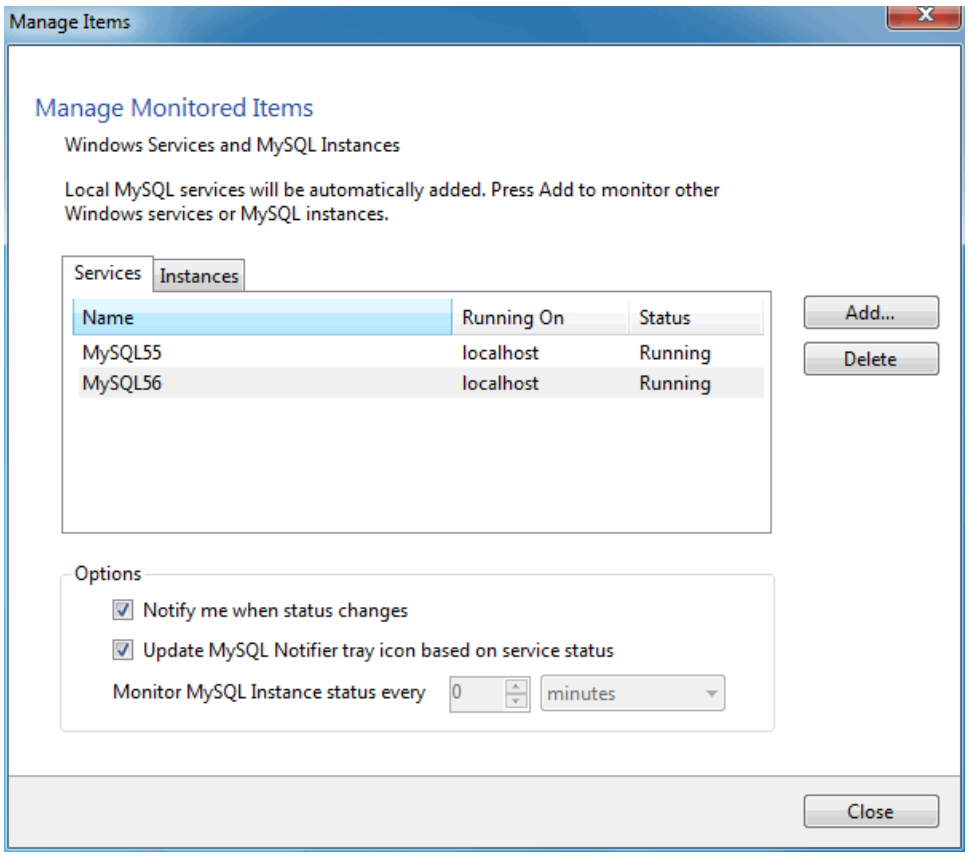
- **Use colorful status icons**: Enables a colorful style of icons for the tray of MySQL Notifier.
- **Run at Windows Startup**: Allows the application to be loaded when Microsoft Windows starts.
- **Automatically Check For Updates Every** # **Weeks**: Checks for a new version of MySQL Notifier, and runs this check every # weeks.
- **Automatically add new services whose name contains:** The text used to filter services and add them automatically to the monitored list of the local computer running MySQL Notifier, and on remote computers already monitoring Windows services.
- **Ping monitored MySQL Server instances every** # **seconds**: The interval (in seconds) to ping monitored MySQL Server instances for status changes. Longer intervals might be necessary if the list of monitored remote instances is large.
- **Notify me when a service is automatically added**: Will display a balloon notification from the taskbar when a newly discovered service is added to the monitored services list.
- **Notify me when a service changes status**: Will display a balloon notification from the taskbar when a monitored service changes its status.
- **Automatic connections migration delayed until:** When there are connections to migrate, postpone the migration by one hour, one day, one week, one month, or indefinitely.

Figure 1.5 MySQL Notifier Options menu



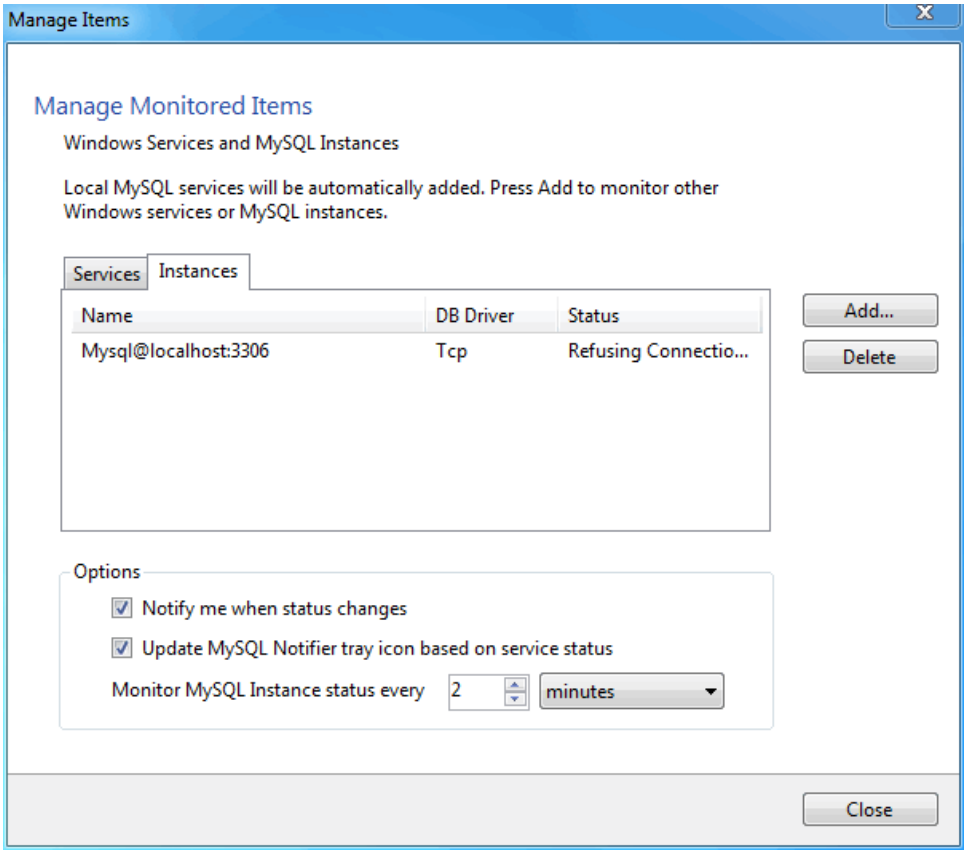
The **Actions**, **Manage Monitored Items** menu enables you to configure the monitored services and MySQL instances. First, with the **Services** tab open:

Figure 1.6 MySQL Notifier Manage Services menu



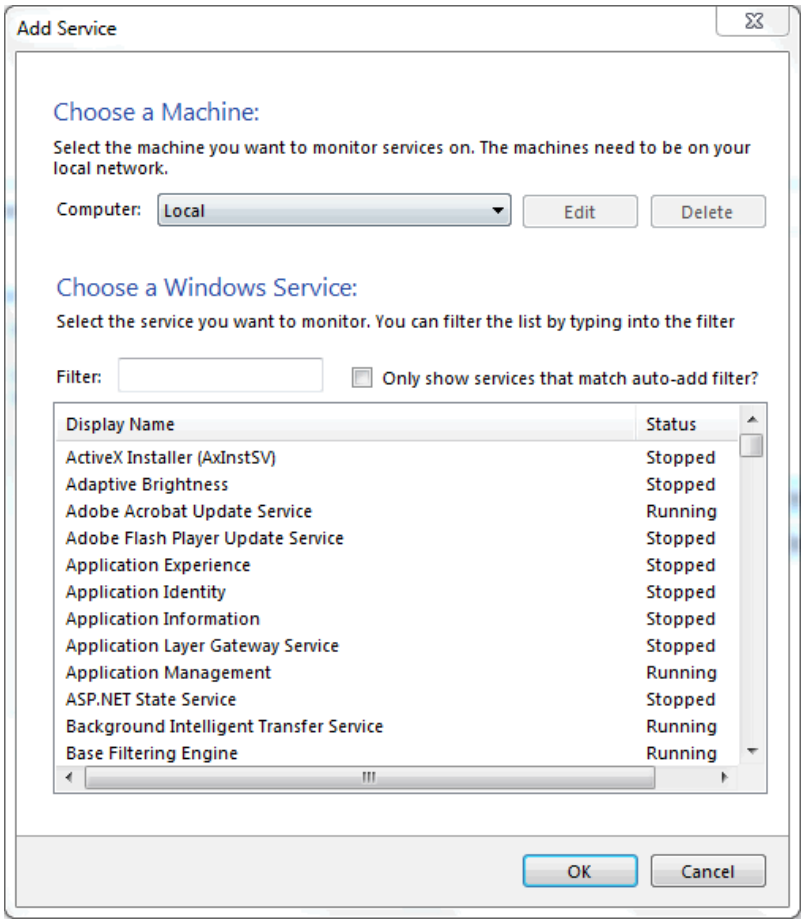
The **Instances** tab is similar:

Figure 1.7 MySQL Notifier Manage Instances menu



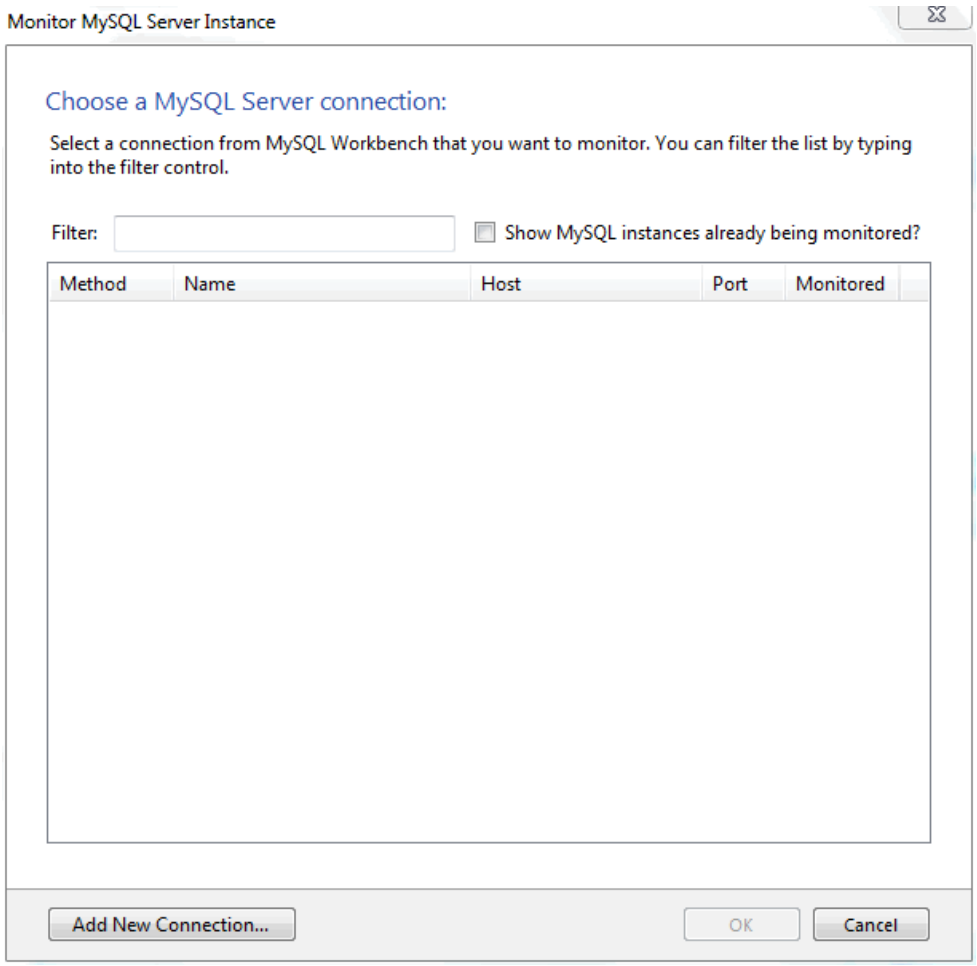
Adding a service or instance (after clicking **Add** in the **Manage Monitored Items** window) enables you to select a running Microsoft Windows service or instance connection, and configure MySQL Notifier to monitor it. Add a new service or instance by clicking service name from the list, then **OK** to accept. Multiple services and instances may be selected.

Figure 1.8 MySQL Notifier Adding new services



Add instances:

Figure 1.9 MySQL Notifier Adding new instances



Troubleshooting

For issues that are not documented here, visit the MySQL Notifier Support Forum for MySQL Notifier help and support.

- *Problem*: attempting to start/stop/restart a MySQL service might generate an error similar to "The Service **MySQL*VERSION*** failed the most recent status change request with the message "The service **mysql*VERSION*** was not found in the Windows Services".

  Explanation: this is a case-sensitivity issue, in that the service name is **MySQL*VERSION*** compared to having **mysql*VERSION*** in the configuration file.

  Solution: either update your MySQL Notifier configuration file with the correct information, or stop MySQL Notifier and delete this configuration file. The MySQL Notifier configuration file is located at `%APPDATA%\Oracle\MySQL Notifier\settings.config` where `%APPDATA%` is a variable and depends on your system. A typical location is "C:\Users*YourUsername*\AppData\Running\Oracle\MySQL Notifier\settings.config" where *YourUsername* is your system's user name. In this file, and within the ServerList section, change the ServerName values from lowercase to the actual service names. For example, change **mysql*VERSION*** to **MySQL*VERSION***, save, and then restart MySQL Notifier. Alternatively, stop MySQL Notifier, delete this file, then restart MySQL Notifier.

- *Problem*: when connecting to a remote computer for the purpose of monitoring a remote Windows service, the **Add Service** dialog does not always show all the services shown in the Windows Services console.

  Explanation: this behavior is governed by the operating system and the outcome is expected when working with nondomain user accounts. For a complete description of the behavior, see the User Account Control and WMI article from Microsoft.

  Solution: when the remote computer is in a compatible domain, it is recommended that domain user accounts are used to connect through WMI to a remote computer. For detailed setup instructions using WMI, see Section 1.3.2, “Setting Up Remote Monitoring in MySQL Notifier”.

  Alternatively, when domain user accounts are not available, Microsoft provides a less secure workaround that should only be implemented with caution. For more information, see the Description of User Account Control and remote restrictions in Windows Vista KB article from Microsoft.

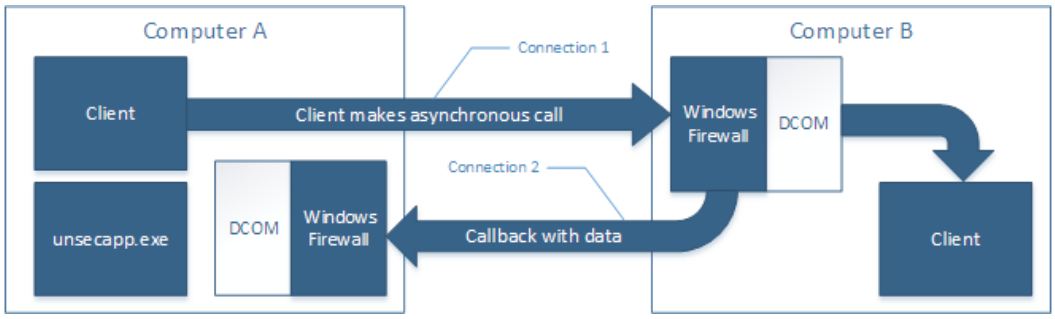
1.3.2 Setting Up Remote Monitoring in MySQL Notifier

MySQL Notifier uses Windows Management Instrumentation (WMI) to manage and monitor services on remote computers. This section explains how it works and how to set up your system to monitor remote MySQL instances.

In order to configure WMI, it is important to understand that the underlying Distributed Component Object Model (DCOM) architecture is doing the WMI work. Specifically, MySQL Notifier is using asynchronous notification queries on remote Microsoft Windows hosts as .NET events. These events send an asynchronous callback to the computer running MySQL Notifier so it knows when a service status has changed on the remote computer. Asynchronous notifications offer the best performance compared to semisynchronous notifications or synchronous notifications that use timers.

Asynchronous notification requires the remote computer to send a callback to the client computer (thus opening a reverse connection), so the Windows Firewall and DCOM settings must be properly configured for the communication to function properly.

Figure 1.10 MySQL Notifier Distributed Component Object Model (DCOM)



Most of the common errors thrown by asynchronous WMI notifications are related to Windows Firewall blocking the communication, or to DCOM / WMI settings not being set up properly. For a list of common errors with solutions, see Common Errors.

The following steps are required to make WMI function. These steps are divided between two machines. A single host computer that runs MySQL Notifier (Computer A), and multiple remote machines that are being monitored (Computer B).

Computer running MySQL Notifier (Computer A)

1. Enable remote administration by either editing the **Group Policy Editor**, or using `NETSH`:

Using the **Group Policy Editor**:

a. Click **Start**, click **Run**, type `GPEDIT.MSC`, and then click **OK**.

b. Under the **Local Computer Policy** heading, expand **Computer Configuration**.

c. Expand **Administrative Templates**, then **Network**, **Network Connections**, and then **Windows Firewall**.

d. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.

e. Double-click **Windows Firewall: Allow inbound remote administration exception** to open a configuration window.

f. Check the **Enabled** option button and then click **OK**.

Using the `NETSH` command:

> **Note**
>
> The "netsh firewall" command is deprecated as of Microsoft Server 2008 and Vista, and replaced with "netsh advfirewall firewall".

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and select **Run as Administrator**).

b. Execute the following command:

```
NETSH advfirewall firewall set service RemoteAdmin enable
```

2. Open the DCOM port TCP 135:

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and select **Run as Administrator**).

b. Execute the following command:

```
NETSH advfirewall firewall add rule name=DCOM_TCP135 protocol=TCP localport=135 dir=in action=allow
```

3. Add the client application that contains the sink for the callback (`MySqlNotifier.exe`) to the Windows Firewall Exceptions List (use either the Windows Firewall configuration or `NETSH`):

Using the Windows Firewall configuration:

a. In the Control Panel, double-click **Windows Firewall**.

b. In the Windows Firewall window's left panel, click **Allow a program or feature through Windows Firewall**.

c. In the Allowed Programs window, click **Change Settings** and do one of the following:

- If `MySqlNotifier.exe` is in the Allowed programs and features list, make sure it is checked for the type of networks the computer connects to (Private, Public or both).
- If `MySqlNotifier.exe` is not in the list, click **Allow another program...**.

i. In the **Add a Program** window, select the `MySqlNotifier.exe` if it exists in the Programs list, otherwise click Browse... and go to the directory where `MySqlNotifier.exe` was installed to select it, then click **Add**.

ii. Make sure `MySqlNotifier.exe` is checked for the type of networks the computer connects to (Private, Public or both).

Using the `NETSH` command:

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and click **Run as Administrator**).

b. Execute the following command, where you change "*`[YOUR_INSTALL_DIRECTORY]`*":

```
NETSH advfirewall firewall add rule name=MySqlNotifier program=[YOUR_INSTALL_DIRECTORY]\MySqlNotifier.ex
```

4. If Computer B is either a member of `WORKGROUP` or is in a different domain that is untrusted by Computer A, then the callback connection (Connection 2) is created as an Anonymous connection. To grant Anonymous connections DCOM Remote Access permissions:

a. Click **Start**, click **Run**, type `DCOMCNFG`, and then click **OK**.

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Access Permissions, click **Edit Limits**.

e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

Monitored Remote Computer (Computer B)

If the user account that is logged on to the computer running the MySQL Notifier (Computer A) is a local administrator on the remote computer (Computer B), such that the same account is an administrator on Computer B, you can skip to the "Allow for remote administration" step.

Setting DCOM security to allow a non-administrator user to access a computer remotely:

1. Grant "DCOM remote launch" and activation permissions for a user or group:

a. Click **Start**, click **Run**, type `DCOMCNFG`, and then click **OK**.

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Launch and Activation Permission, click **Edit Limits**.

e. In the **Launch and Activation Permission** dialog box, follow these steps if your name or your group does not appear in the Groups or user names list:

i. In the **Launch and Activation Permission** dialog box, click **Add**.

ii. In the Select Users or Groups dialog box, add your name and the group in the **Enter the object names to select** box, and then click **OK**.

f. In the **Launch and Activation Permission** dialog box, select your user and group in the Group or user names box. In the Allow column under Permissions for User, select **Remote Launch**, select **Remote Activation**, and then click **OK**.

Grant DCOM remote access permissions:

a. Click **Start**, click **Run**, type `DCOMCNFG`, and then click **OK**.

b. In the Component Services dialog box, expand Component Services, expand Computers, and then right-click **My Computer** and click **Properties**.

c. In the My Computer Properties dialog box, click the **COM Security** tab.

d. Under Access Permissions, click **Edit Limits**.

e. In the Access Permission dialog box, select **ANONYMOUS LOGON name** in the Group or user names box. In the Allow column under Permissions for User, select **Remote Access**, and then click **OK**.

2. Allowing non-administrator users access to a specific WMI namespace:

a. In the Control Panel, double-click **Administrative Tools**.

b. In the Administrative Tools window, double-click **Computer Management**.

c. In the Computer Management window, expand the **Services and Applications** tree.

d. Right-click the WMI Control icon and select **Properties**.

e. In the WMI Control Properties window, click the **Security** tab.

f. In the Security tab, select the namespace and click **Security**. Root/CIMV2 is a commonly used namespace.

g. Locate the appropriate account and check **Remote Enable** in the Permissions list.

3. Allow for remote administration by either editing the **Group Policy Editor** or using `NETSH`:

Using the **Group Policy Editor**:

a. Click **Start**, click **Run**, type `GPEDIT.MSC`, and then click **OK**.

b. Under the Local Computer Policy heading, double-click **Computer Configuration**.

c. Double-click **Administrative Templates**, then **Network**, **Network Connections**, and then **Windows Firewall**.

d. If the computer is in the domain, then double-click **Domain Profile**; otherwise, double-click **Standard Profile**.

e. Click **Windows Firewall: Allow inbound remote administration exception**.

f. On the Action menu either select **Edit**, or double-click the selection from the previous step.

g. Check the **Enabled** radio button, and then click **OK**.

Using the `NETSH` command:

a. Open a command prompt window with Administrative rights (you can right-click the Command Prompt icon and click Run as Administrator).

b. Execute the following command:

```
NETSH advfirewall firewall set service RemoteAdmin enable
```

4. Confirm that the user account you are logging in with uses the `Name` value and not the `Full Name` value:

   a. In the **Control Panel**, double-click **Administrative Tools**.

   b. In the **Administrative Tools** window, double-click **Computer Management**.

   c. In the **Computer Management** window, expand the **System Tools** then **Local Users and Groups**.

   d. Click the **Users** node, and on the right side panel locate your user and make sure it uses the **Name** value to connect, and not the **Full Name** value.

Common Errors

- `0x80070005`
  - DCOM Security was not configured properly (see Computer B, the `Setting DCOM security...` step).
  - The remote computer (Computer B) is a member of WORKGROUP or is in a domain that is untrusted by the client computer (Computer A) (see Computer A, the `Grant Anonymous connections DCOM Remote Access permissions` step).
- `0x8007000E`
  - The remote computer (Computer B) is a member of WORKGROUP or is in a domain that is untrusted by the client computer (Computer A) (see Computer A, the `Grant Anonymous connections DCOM Remote Access permissions` step).
- `0x80041003`
  - Access to the remote WMI namespace was not configured properly (see Computer B, the `Allowing non-administrator users access to a specific WMI namespace` step).
- `0x800706BA`
  - The DCOM port is not open on the client computers (Computer A) firewall. See the `Open the DCOM port TCP 135` step for Computer A.
  - The remote computer (Computer B) is inaccessible because its network location is set to Public. Make sure you can access it through the Windows Explorer.

1.4 Installing MySQL on Microsoft Windows Using an MSI Package

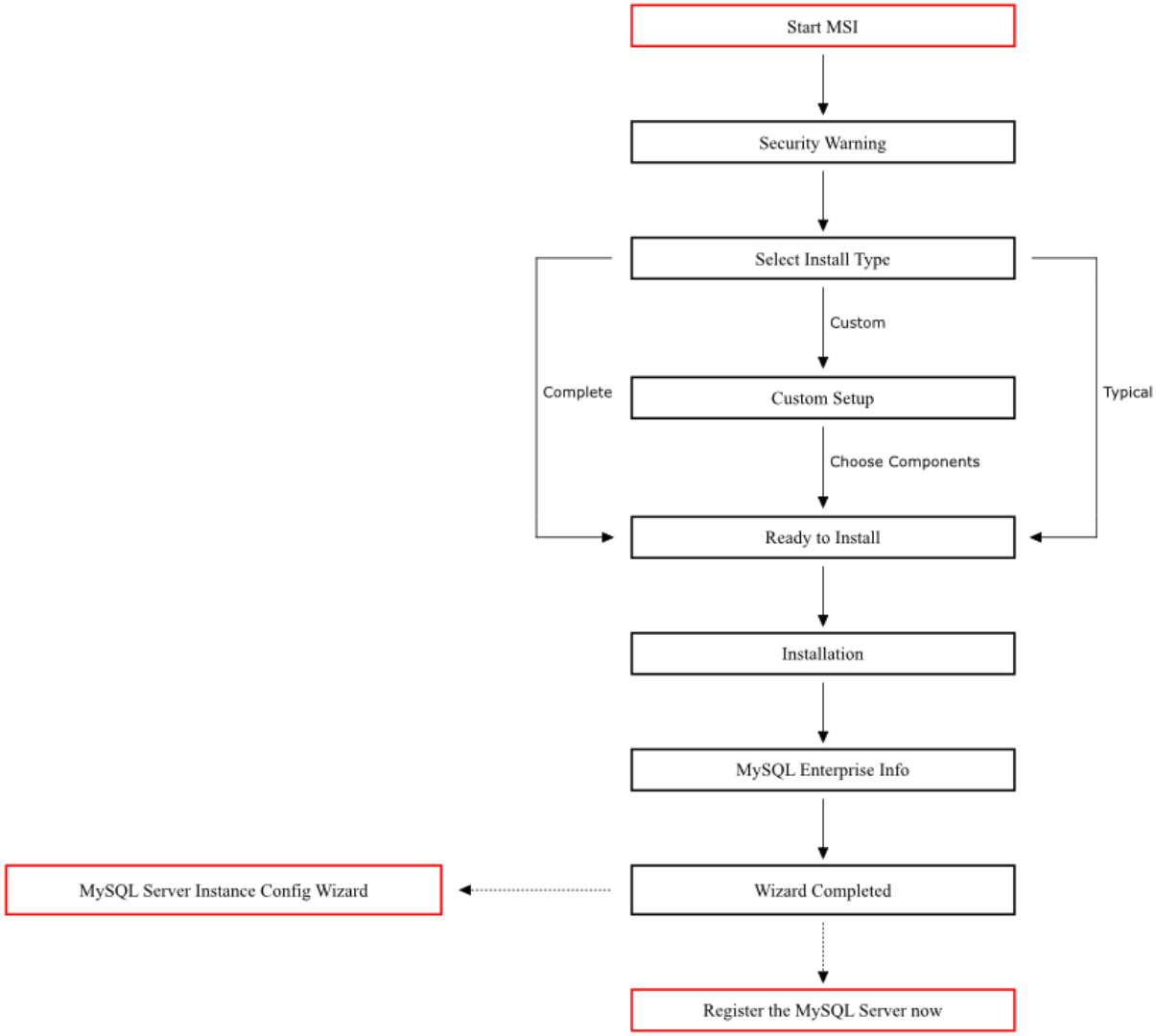
The MSI package is designed to install and configure MySQL in such a way that you can immediately get started using MySQL.

The MySQL Installation Wizard and MySQL Configuration Wizard are available in the Essentials and Complete install packages. They are recommended for most standard MySQL installations. Exceptions include users who need to install multiple instances of MySQL on a single server host and advanced users who want complete control of server configuration.

- For information on installing using the GUI MSI installer process, see Section 1.4.1, “Using the MySQL Installation Wizard for Microsoft Windows”.
- For information on installing using the command line using the MSI package, see Section 1.4.2, “Automating MySQL Installation on Microsoft Windows Using the MSI Package”.
- If you have previously installed MySQL using the MSI package and want to remove MySQL, see Section 1.4.3, “Removing MySQL When Installed from the MSI Package”.

The workflow sequence for using the installer is shown in the figure below:

Figure 1.11 Installation Workflow for Windows Using MSI Installer



Note

Microsoft Windows XP and later include a firewall which specifically blocks ports. If you plan on using MySQL through a network port then you should open and create an exception for this port before performing the installation. To check and if necessary add an exception to the firewall settings:

1. First ensure that you are logged in as an Administrator or a user with Administrator privileges.
2. Go to the **Control Panel**, and double click the **Windows Firewall** icon.
3. Choose the **Allow a program through Windows Firewall** option and click the **Add port** button.
4. Enter `MySQL` into the **Name** text box and `3306` (or the port of your choice) into the **Port number** text box.

5. Also ensure that the **TCP** protocol radio button is selected.
6. If you wish, you can also limit access to the MySQL server by choosing the **Change scope** button.
7. Confirm your choices by clicking the **OK** button.

Additionally, when running the MySQL Installation Wizard on Windows Vista or newer, ensure that you are logged in as a user with administrative rights.

Note

When using Windows Vista or newer, you may want to disable User Account Control (UAC) before performing the installation. If you do not do so, then MySQL may be identified as a security risk, which will mean that you need to enable MySQL. You can disable the security checking by following these instructions:

1. Open **Control Panel**.
2. Under the **User Accounts and Family Safety**, select **Add or remove user accounts.**
3. Click the **Got to the main User Accounts page** link.
4. Click on **Turn User Account Control on or off**. You may be prompted to provide permission to change this setting. Click **Continue**.
5. Deselect or uncheck the check box next to **Use User Account Control (UAC) to help protect your computer**. Click **OK** to save the setting.

You will need to restart to complete the process. Click **Restart Now** to reboot the machine and apply the changes. You can then follow the instructions below for installing Windows.

1.4.1 Using the MySQL Installation Wizard for Microsoft Windows

MySQL Installation Wizard is an installer for the MySQL server that uses the latest installer technologies for Microsoft Windows. The MySQL Installation Wizard, in combination with the MySQL Config Wizard, enables a user to install and configure a MySQL server that is ready for use immediately after installation.

The MySQL Installation Wizard uses the standard Microsoft Installer Engine (MSI) system is the standard installer for all MySQL server distributions. See Section 1.4.1.6, “MySQL Installation Wizard: Upgrading MySQL”, for more information on upgrading from a previous version.

If you are upgrading an installation from MySQL 5.1.31 or earlier to MySQL 5.1.32 or later, read the notes provided in Section 1.4.1.6, “MySQL Installation Wizard: Upgrading MySQL”.

The Microsoft Windows Installer Engine was updated with the release of Windows XP; those using a previous version of Windows can reference this Microsoft Knowledge Base article for information on upgrading to the latest version of the Windows Installer Engine.

In addition, Microsoft has introduced the WiX (Windows Installer XML) toolkit. This is the first highly acknowledged Open Source project from Microsoft. We have switched to WiX because it is an Open Source project and it enables us to handle the complete Windows installation process in a flexible manner using scripts.

Improving the MySQL Installation Wizard depends on the support and feedback of users like you. If you find that the MySQL Installation Wizard is lacking some feature important to you, or if you discover a bug, please report it in our bugs database using the instructions given in How to Report Bugs or Problems.

1.4.1.1 MySQL Installation Wizard: Downloading and Starting

The MySQL installation packages can be downloaded from http://dev.mysql.com/downloads/. If the package you download is contained within a Zip archive, you need to extract the archive first.

The process for starting the wizard depends on the contents of the installation package you download. If there is a `setup.exe` file present, double-click it to start the installation process. If there is an `.msi` file present, double-click it to start the installation process.

1.4.1.2 MySQL Installation Wizard: Choosing an Install Type

There are three installation types available: **Typical**, **Complete**, and **Custom**.

The **Typical** installation type installs the MySQL server, the `mysql` command-line client, and the command-line utilities. The command-line clients and utilities include `mysqldump`, `myisamchk`, and several other tools to help you manage the MySQL server.

The **Complete** installation type installs all components included in the installation package. The full installation package includes components such as the embedded server library, the benchmark suite, support scripts, and documentation.

The **Custom** installation type gives you complete control over which packages you wish to install and the installation path that is used. See Section 1.4.1.3, “MySQL Installation Wizard: The Custom Install Dialog”, for more information on performing a custom install.

If you choose the **Typical** or **Complete** installation types and click the **Next** button, you advance to the confirmation screen to verify your choices and begin the installation. If you choose the **Custom** installation type and click the **Next** button, you advance to the custom installation dialog, described in Section 1.4.1.3, “MySQL Installation Wizard: The Custom Install Dialog”.

1.4.1.3 MySQL Installation Wizard: The Custom Install Dialog

If you wish to change the installation path or the specific components that are installed by the MySQL Installation Wizard, choose the **Custom** installation type.

A tree view on the left side of the custom install dialog lists all available components. Components that are not installed have a red **X** icon; components that are installed have a gray icon. To change whether a component is installed, click that component's icon and choose a new option from the drop-down list that appears.

You can change the default installation path by clicking the **Change...** button to the right of the displayed installation path.

After choosing your installation components and installation path, click the **Next** button to advance to the confirmation dialog.

1.4.1.4 MySQL Installation Wizard: The Confirmation Dialog

Once you choose an installation type and optionally choose your installation components, you advance to the confirmation dialog. Your installation type and installation path are displayed for you to review.

To install MySQL if you are satisfied with your settings, click the **Install** button. To change your settings, click the **Back** button. To exit the MySQL Installation Wizard without installing MySQL, click the **Cancel** button.

In MySQL 5.1.47 and earlier, after installation is complete, you have the option of registering with the MySQL web site. Registration gives you access to post in the MySQL forums at forums.mysql.com, along with the ability to report bugs at bugs.mysql.com and to subscribe to our newsletter.

The final screen of the installer provides a summary of the installation and gives you the option to launch the MySQL Config Wizard, which you can use to create a configuration file, install the MySQL service, and configure security settings.

1.4.1.5 MySQL Installation Wizard: Changes Made

Once you click the **Install** button, the MySQL Installation Wizard begins the installation process and makes certain changes to your system which are described in the sections that follow.

Changes to the Registry

The MySQL Installation Wizard creates one Windows registry key in a typical install situation, located in `HKEY_LOCAL_MACHINE\SOFTWARE\MySQL AB`. For 64-bit Windows, the registry location is `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\MYSQL AB`. A server version specific entry will be created for each release series of MySQL that you install.

The MySQL Installation Wizard creates a key named after the release series of the server that is being installed, such as `MySQL Server 5.1`. It contains two string values, `Location` and `Version`. The `Location` string contains the path to the installation directory. In a default installation it contains `C:\Program Files\MySQL\MySQL Server 5.1\`. The `Version` string contains the release number. For example, for an installation of MySQL Server 5.1.73, the key contains a value of `5.1.73`.

These registry keys are used to help external tools identify the installed location of the MySQL server, preventing a complete scan of the hard-disk to determine the installation path of the MySQL server. The registry keys are not required to run the server, and if you install MySQL using the `noinstall` Zip archive, the registry keys are not created.

Changes to the Start Menu

The MySQL Installation Wizard creates a new entry in the Windows **Start** menu under a common MySQL menu heading named after the release series of MySQL that you have installed. For example, if you install MySQL 5.1, the MySQL Installation Wizard creates a **MySQL Server 5.1** section in the **Start** menu.

The following entries are created within the new **Start** menu section:

- **MySQL Command-Line Client**: This is a shortcut to the `mysql` command-line client and is configured to connect as the `root` user. The shortcut prompts for a `root` user password when you connect.
- **MySQL Server Instance Config Wizard**: This is a shortcut to the MySQL Config Wizard. Use this shortcut to configure a newly installed server, or to reconfigure an existing server.
- **MySQL Documentation**: This is a link to the MySQL server documentation that is stored locally in the MySQL server installation directory. This option is not available when the MySQL server is installed using the Essentials installation package.

Changes to the File System

The MySQL Installation Wizard by default installs the MySQL 5.1 server to `C:\Program Files\MySQL\MySQL Server 5.1`, where `Program Files` is the default location for applications in your system, and `5.1` is the release series of your MySQL server. This is the recommended location for the MySQL server, replacing the former default location `C:\mysql`.

By default, all MySQL applications are stored in a common directory at `C:\Program Files\MySQL`, where `Program Files` is the default location for applications in your Windows installation. A typical MySQL installation on a developer machine might look like this:

```
C:\Program Files\MySQL\MySQL Server 5.1
C:\Program Files\MySQL\MySQL Workbench 5.1 OSS
```

This approach makes it easier to manage and maintain all MySQL applications installed on a particular system.

In MySQL 5.1.23 and earlier, the default location for the data files used by MySQL is located within the corresponding MySQL Server installation directory. For MySQL 5.1.24 and later, the default location of the data directory is the `AppData` directory configured for the user that installed the MySQL application.

1.4.1.6 MySQL Installation Wizard: Upgrading MySQL

The MySQL Installation Wizard can perform server upgrades automatically using the upgrade capabilities of MSI. That means you do not need to remove a previous installation manually before installing a new release. The installer automatically shuts down and removes the previous MySQL service before installing the new version.

Automatic upgrades are available only when upgrading between installations that have the same major and minor version numbers. For example, you can upgrade automatically from MySQL 5.1.5 to MySQL 5.1.6, but not from MySQL 5.0 to MySQL 5.1.

In MySQL 5.1.32 and later, the `EXE` version of the MSI installer packages were removed. When upgrading an existing MySQL installation from the old EXE based installer to the MSI based installer, please keep the following notes in mind:

- The MSI installer will not identify an existing installation that was installed using the old EXE installer. This means that the installer will not stop the existing server, or detect that the existing password is required before installing the new version. To work around this:

  1. Stop the current server manually using `net stop` or `mysqladmin shutdown`.
  2. Remove the existing installation manually by using the **Add/Remove Programs** control panel. This will keep the existing configuration and data files, as these are not removed automatically.
  3. Install the new version of MySQL using the MSI installer. When running the installation, skip updating the security by deselecting the check box on the security screen.
  4. Complete the installation, and then start the server again. You should be able to login with your existing user and password credentials.

- You can only upgrade the version and release using the MSI installer. For example, you can upgrade an open source installation with an open source installer. You cannot upgrade an open source installation using the enterprise installer.

See Section 1.9, “Upgrading MySQL Server on Microsoft Windows”.

1.4.2 Automating MySQL Installation on Microsoft Windows Using the MSI Package

The Microsoft Installer (MSI) supports a both a *quiet* and a *passive* mode that can be used to install MySQL automatically without requiring intervention. You can use this either in scripts to automatically

install MySQL or through a terminal connection such as Telnet where you do not have access to the standard Windows user interface. The MSI packages can also be used in combination with Microsoft's Group Policy system (part of Windows Server 2003 and Windows Server 2008) to install MySQL across multiple machines.

To install MySQL from one of the MSI packages automatically from the command line (or within a script), you need to use the `msiexec.exe` tool. For example, to perform a quiet installation (which shows no dialog boxes or progress):

```
shell> msiexec /i mysql-5.1.73.msi /quiet
```

The `/i` indicates that you want to perform an installation. The `/quiet` option indicates that you want no interactive elements.

To provide a dialog box showing the progress during installation, and the dialog boxes providing information on the installation and registration of MySQL, use `/passive` mode instead of `/quiet`:

```
shell> msiexec /i mysql-5.1.73.msi /passive
```

Regardless of the mode of the installation, installing the package in this manner performs a 'Typical' installation, and installs the default components into the standard location.

You can also use this method to uninstall MySQL by using the `/uninstall` or `/x` options:

```
shell> msiexec /x mysql-5.1.73.msi /uninstall
```

To install MySQL and configure a MySQL instance from the command line, see Section 1.5.13, “MySQL Server Instance Config Wizard: Creating an Instance from the Command Line”.

For information on using MSI packages to install software automatically using Group Policy, see How to use Group Policy to remotely install software in Windows Server 2003.

1.4.3 Removing MySQL When Installed from the MSI Package

To uninstall a MySQL where you have used the MSI packages, you must use the **Add/Remove Programs** tool within **Control Panel**. To do this:

1. Right-click the **start** menu and choose **Control Panel**.
2. If the Control Panel is set to category mode (you will see **Pick a category** at the top of the **Control Panel** window), double-click **Add or Remove Programs**. If the Control is set to classic mode, double-click the **Add or Remove Programs** icon.
3. Find MySQL in the list of installed software. MySQL Server is installed against release series numbers (MySQL 5.0, MySQL 5.1, etc.). Select the version that you want to remove and click **Remove**.
4. You will be prompted to confirm the removal. Click **Yes** to remove MySQL.

When MySQL is removed using this method, only the installed components are removed. Any database information (including the tables and data), import or export files, log files, and binary logs produced during execution are kept in their configured location.

If you try to install MySQL again the information will be retained and you will be prompted to enter the password configured with the original installation.

If you want to delete MySQL completely:

- Delete the associated data directory. On Windows XP and Windows Server 2003, before MySQL 5.1.24, the default data directory would be located within the MySQL installation directory. On MySQL 5.1.24 and later, the default data directory is the configured AppData directory, which is `C:\Documents and Settings\All Users\Application Data\MySQL` by default.
- On Windows 7 and Windows Server 2008, the default data directory location is `C:\ProgramData\Mysql`.

> **Note**
>
> The `C:\ProgramData` directory is hidden by default. You must change your folder options to view the hidden file. Choose **Organize**, **Folder and search options**, **Show hidden folders**.

1.5 Using the MySQL Server Instance Config Wizard

The MySQL Server Instance Config Wizard helps automate the process of configuring your server. It creates a custom MySQL configuration file (`my.ini` or `my.cnf`) by asking you a series of questions and then applying your responses to a template to generate the configuration file that is tuned to your installation.

The complete and essential MSI installation packages include the MySQL Server Instance Config Wizard in the MySQL 5.1 server. The MySQL Server Instance Config Wizard is only available for Windows.

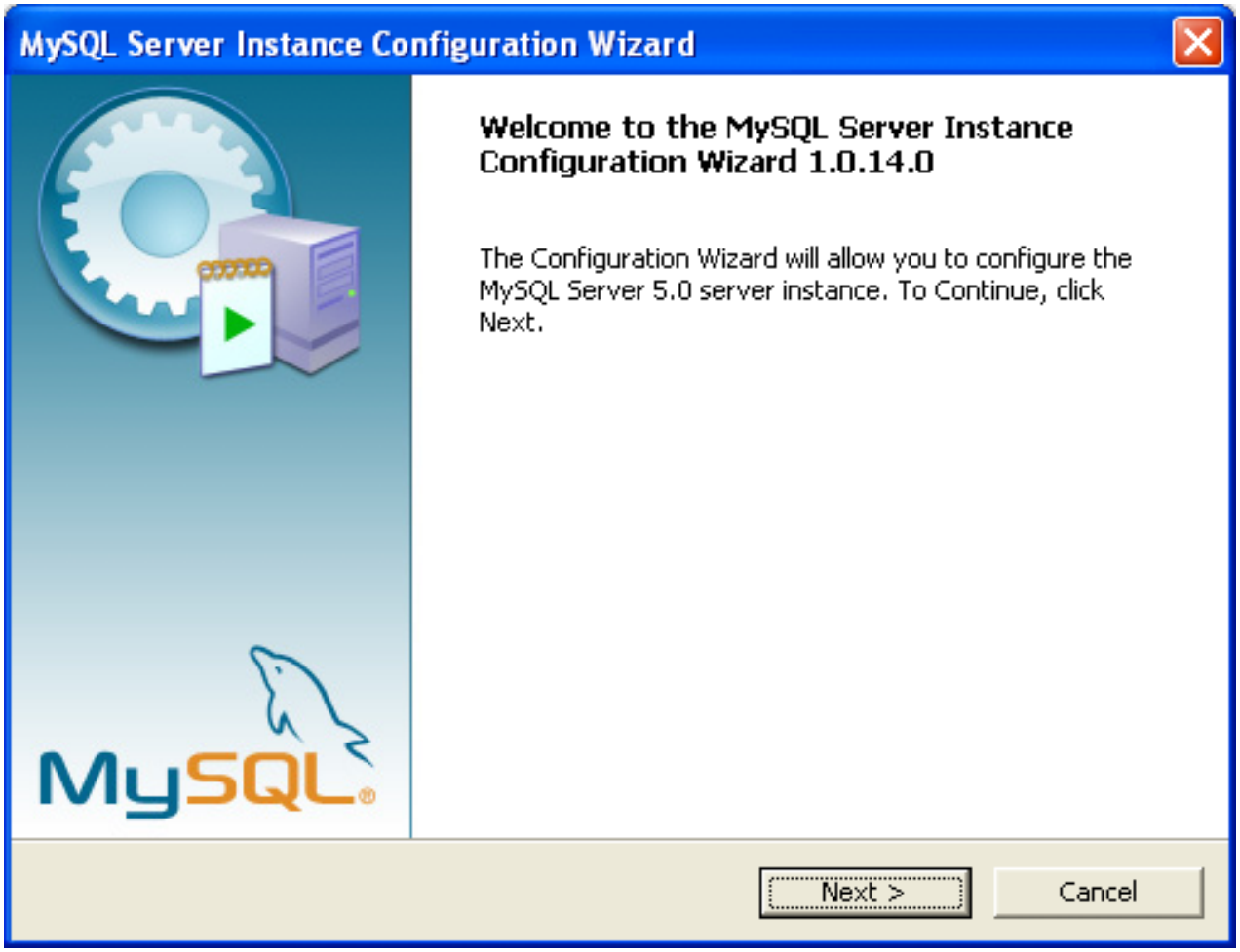
The workflow sequence for using the MySQL Server Instance Config Wizard is shown in the figure below:


Start Wizard
Remove Instance
Confirm Removal
New Installation
Reconfigure Instance
Configuration Type
Detailed Installation
Server Type
Database Usage
Multifunction/Transactional Database
Non-transactional Database
Standard Installation
InnoDB Settings
Connection Settings
Networking Options
Character Set Options
Windows Options
Security Options
Confirm Installation
Execute
Execute
Completed

1.5.1 Starting the MySQL Server Instance Config Wizard

The MySQL Server Instance Config Wizard is normally started as part of the installation process. You should only need to run the MySQL Server Instance Config Wizard again when you need to change the configuration parameters of your server.

If you chose not to open a port prior to installing MySQL on Windows Vista or newer, you can choose to use the MySQL Server Instance Config Wizard after installation. However, you must open a port in the Windows Firewall. To do this see the instructions given in Section 1.4.1.1, “MySQL Installation Wizard: Downloading and Starting”. Rather than opening a port, you also have the option of adding MySQL as a program that bypasses the Windows Firewall. One or the other option is sufficient—you need not do both. Additionally, when running the MySQL Server Config Wizard on Windows Vista or newer, ensure that you are logged in as a user with administrative rights.



You can launch the MySQL Config Wizard by clicking the **MySQL Server Instance Config Wizard** entry in the **MySQL** section of the Windows **Start** menu.

Alternatively, you can navigate to the `bin` directory of your MySQL installation and launch the `MySQLInstanceConfig.exe` file directly.

The MySQL Server Instance Config Wizard places the `my.ini` file in the installation directory for the MySQL server. This helps associate configuration files with particular server instances.

To ensure that the MySQL server knows where to look for the `my.ini` file, an argument similar to this is passed to the MySQL server as part of the service installation:

```
--defaults-file="C:\Program Files\MySQL\MySQL Server 5.1\my.ini"
```

Here, *`C:\Program Files\MySQL\MySQL Server 5.1`* is replaced with the installation path to the MySQL Server. The `--defaults-file` option instructs the MySQL server to read the specified file for configuration options when it starts.

Apart from making changes to the `my.ini` file by running the MySQL Server Instance Config Wizard again, you can modify it by opening it with a text editor and making any necessary changes. You can also modify the server configuration with the http://www.mysql.com/products/administrator/ utility. For more information about server configuration, see Server Command Options.

MySQL clients and utilities such as the `mysql` and `mysqldump` command-line clients are not able to locate the `my.ini` file located in the server installation directory. To configure the client and utility applications, create a new `my.ini` file in the Windows installation directory (for example, `C:\WINDOWS`).

Under Windows Server 2003, Windows Server 2000, Windows XP, and Windows Vista, MySQL Server Instance Config Wizard will configure MySQL to work as a Windows service. To start and stop MySQL you use the `Services` application that is supplied as part of the Windows Administrator Tools.

1.5.2 MySQL Server Instance Config Wizard: Choosing a Maintenance Option

If the MySQL Server Instance Config Wizard detects an existing configuration file, you have the option of either reconfiguring your existing server, or removing the server instance by deleting the configuration file and stopping and removing the MySQL service.

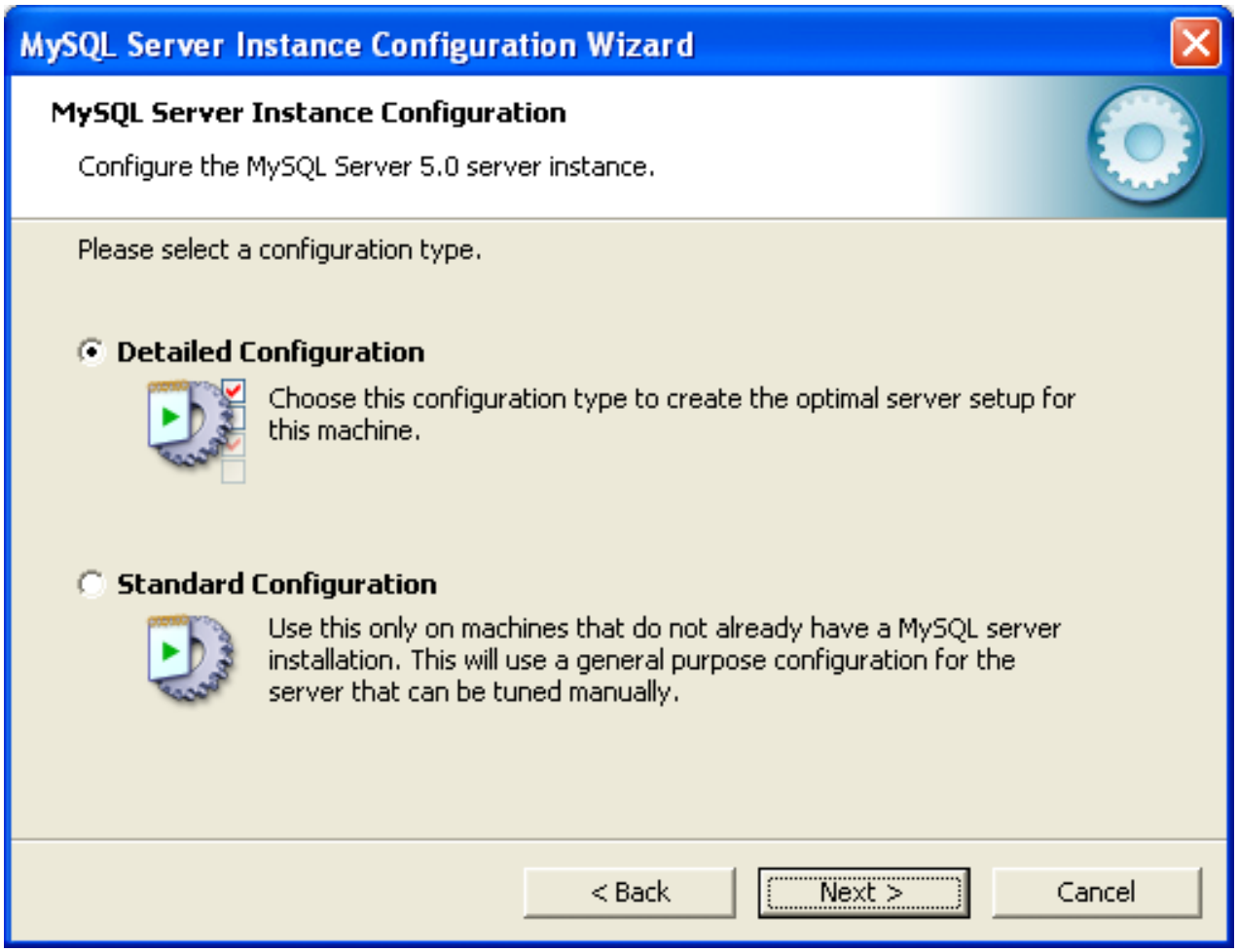
To reconfigure an existing server, choose the **Re-configure Instance** option and click the **Next** button. Any existing configuration file is not overwritten, but renamed (within the same directory) using a timestamp (Windows) or sequential number (Linux). To remove the existing server instance, choose the **Remove Instance** option and click the **Next** button.

If you choose the **Remove Instance** option, you advance to a confirmation window. Click the **Execute** button. The MySQL Server Config Wizard stops and removes the MySQL service, and then deletes the configuration file. The server installation and its `data` folder are not removed.

If you choose the **Re-configure Instance** option, you advance to the **Configuration Type** dialog where you can choose the type of installation that you wish to configure.

1.5.3 MySQL Server Instance Config Wizard: Choosing a Configuration Type

When you start the MySQL Server Instance Config Wizard for a new MySQL installation, or choose the **Re-configure Instance** option for an existing installation, you advance to the **Configuration Type** dialog.



There are two configuration types available: **Detailed Configuration** and **Standard Configuration**. The **Standard Configuration** option is intended for new users who want to get started with MySQL quickly without having to make many decisions about server configuration. The **Detailed Configuration** option is intended for advanced users who want more fine-grained control over server configuration.

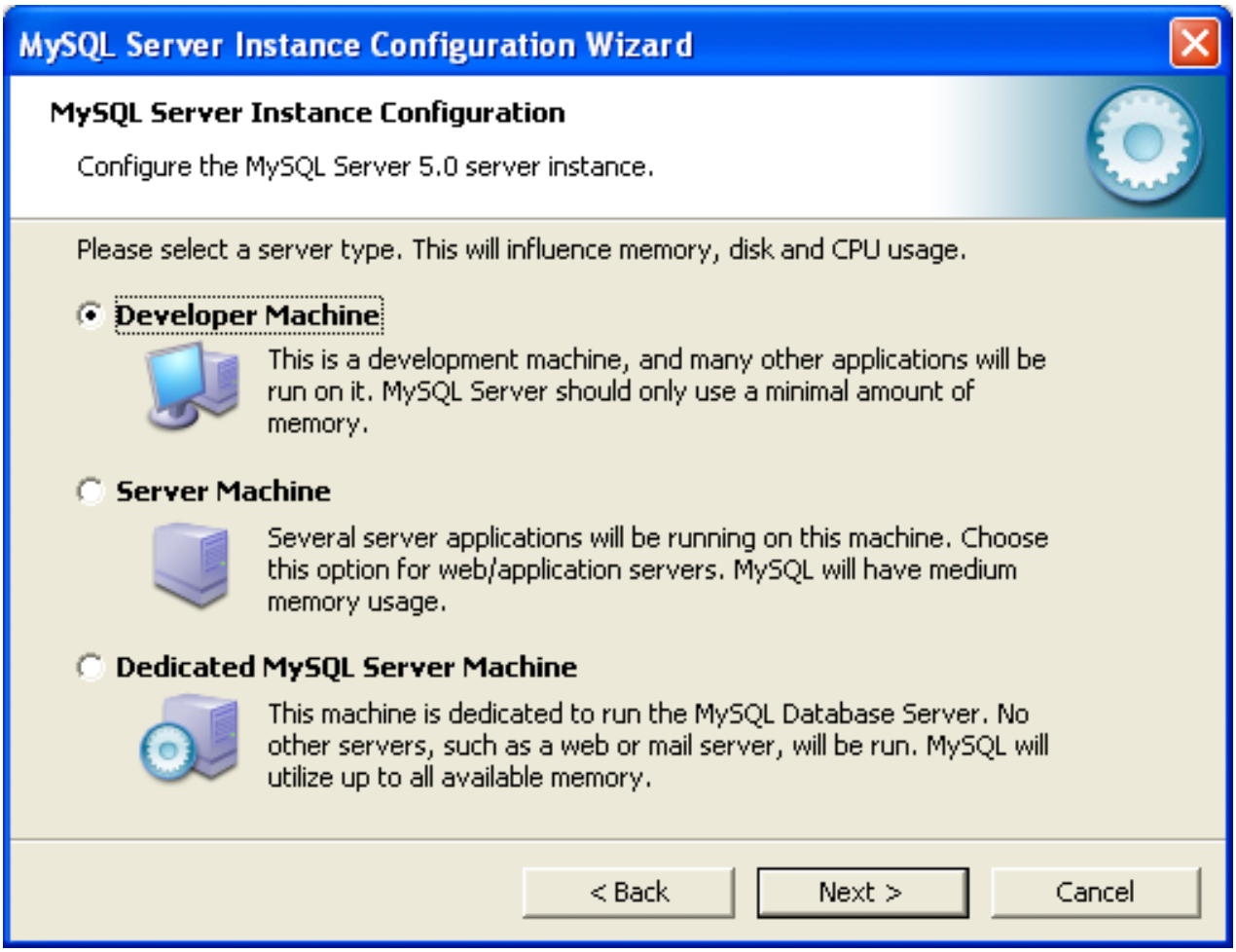
If you are new to MySQL and need a server configured as a single-user developer machine, the **Standard Configuration** should suit your needs. Choosing the **Standard Configuration** option causes the MySQL Config Wizard to set all configuration options automatically with the exception of **Service Options** and **Security Options**.

The **Standard Configuration** sets options that may be incompatible with systems where there are existing MySQL installations. If you have an existing MySQL installation on your system in addition to the installation you wish to configure, the **Detailed Configuration** option is recommended.

To complete the **Standard Configuration**, please refer to the sections on **Service Options** and **Security Options** in Section 1.5.10, “MySQL Server Instance Config Wizard: The Service Options Dialog”, and Section 1.5.11, “MySQL Server Instance Config Wizard: The Security Options Dialog”, respectively.

1.5.4 MySQL Server Instance Config Wizard: The Server Type Dialog

There are three different server types available to choose from. The server type that you choose affects the decisions that the MySQL Server Instance Config Wizard makes with regard to memory, disk, and processor usage.

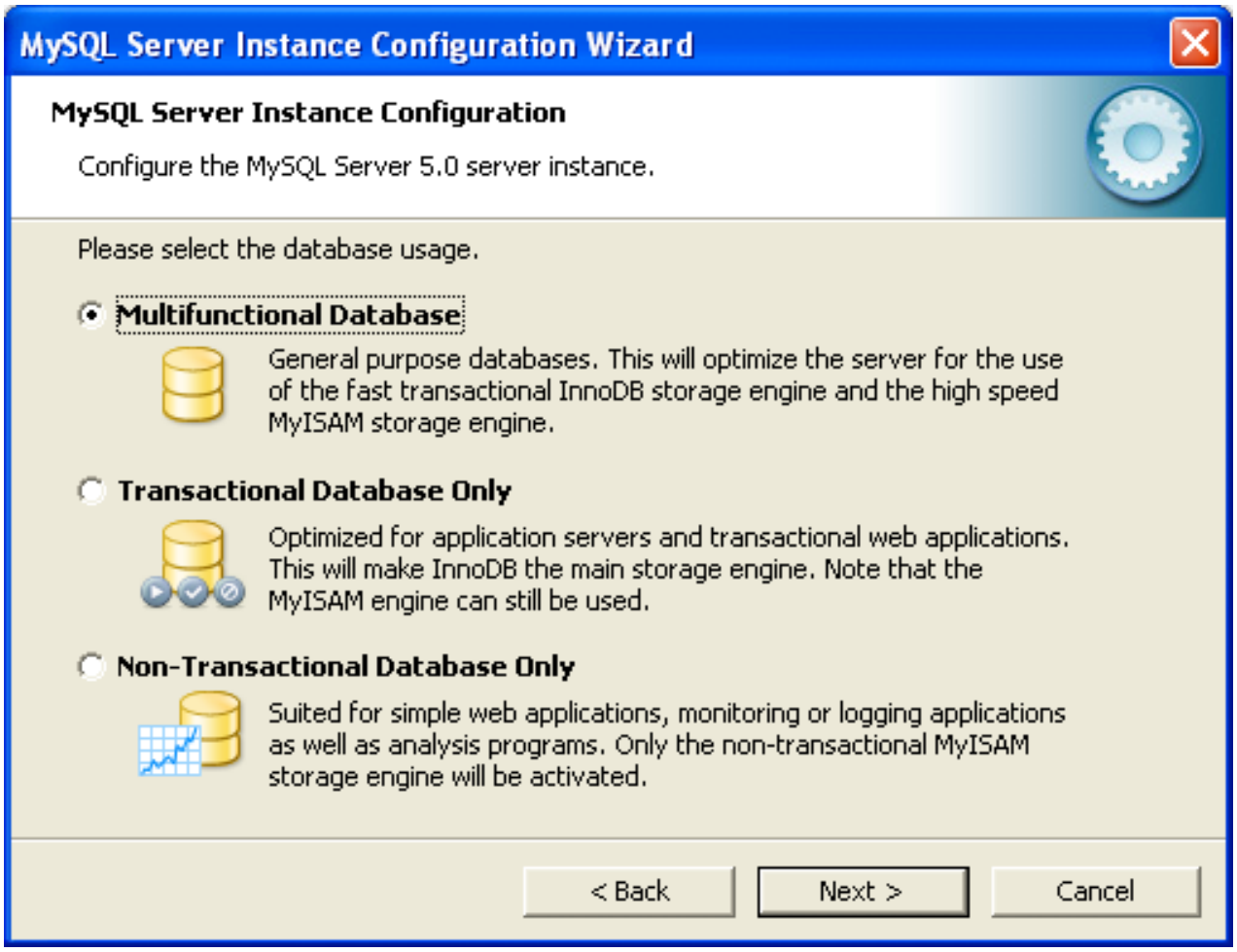


- **Developer Machine**: Choose this option for a typical desktop workstation where MySQL is intended only for personal use. It is assumed that many other desktop applications are running. The MySQL server is configured to use minimal system resources.
- **Server Machine**: Choose this option for a server machine where the MySQL server is running alongside other server applications such as FTP, email, and Web servers. The MySQL server is configured to use a moderate portion of the system resources.
- **Dedicated MySQL Server Machine**: Choose this option for a server machine that is intended to run only the MySQL server. It is assumed that no other applications are running. The MySQL server is configured to use all available system resources.

> **Note**
>
> By selecting one of the preconfigured configurations, the values and settings of various options in your `my.cnf` or `my.ini` will be altered accordingly. The default values and options as described in the reference manual may therefore be different to the options and values that were created during the execution of the Config Wizard.

1.5.5 MySQL Server Instance Config Wizard: The Database Usage Dialog

The **Database Usage** dialog enables you to indicate the storage engines that you expect to use when creating MySQL tables. The option you choose determines whether the `InnoDB` storage engine is available and what percentage of the server resources are available to `InnoDB`.



- **Multifunctional Database**: This option enables both the `InnoDB` and `MyISAM` storage engines and divides resources evenly between the two. This option is recommended for users who use both storage engines on a regular basis.
- **Transactional Database Only**: This option enables both the `InnoDB` and `MyISAM` storage engines, but dedicates most server resources to the `InnoDB` storage engine. This option is recommended for users who use `InnoDB` almost exclusively and make only minimal use of `MyISAM`.
- **Non-Transactional Database Only**: This option disables the `InnoDB` storage engine completely and dedicates all server resources to the `MyISAM` storage engine. This option is recommended for users who do not use `InnoDB`.

The Config Wizard uses a template to generate the server configuration file. The **Database Usage** dialog sets one of the following option strings:

```
Multifunctional Database:        MIXED
Transactional Database Only:     INNODB
Non-Transactional Database Only: MYISAM
```

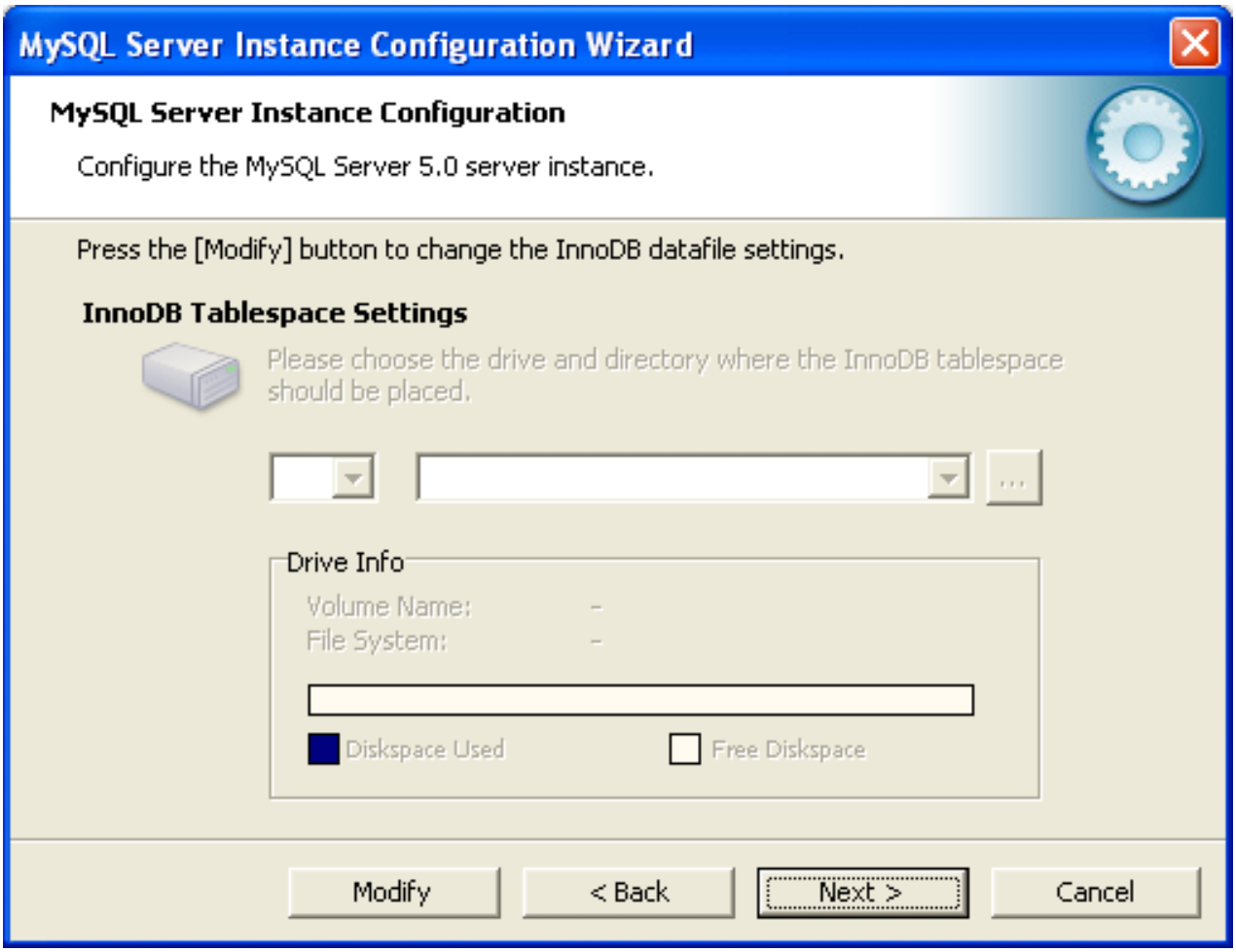
When these options are processed through the default template (my-template.ini) the result is:

```
Multifunctional Database:
default-storage-engine=InnoDB
_myisam_pct=50

Transactional Database Only:
default-storage-engine=InnoDB
_myisam_pct=5

Non-Transactional Database Only:
default-storage-engine=MyISAM
_myisam_pct=100
skip-innodb
```

The `_myisam_pct` value is used to calculate the percentage of resources dedicated to `MyISAM`. The remaining resources are allocated to `InnoDB`.

1.5.6 MySQL Server Instance Config Wizard: The InnoDB Tablespace Dialog

Some users may want to locate the `InnoDB` tablespace files in a different location than the MySQL server data directory. Placing the tablespace files in a separate location can be desirable if your system has a higher capacity or higher performance storage device available, such as a RAID storage system.

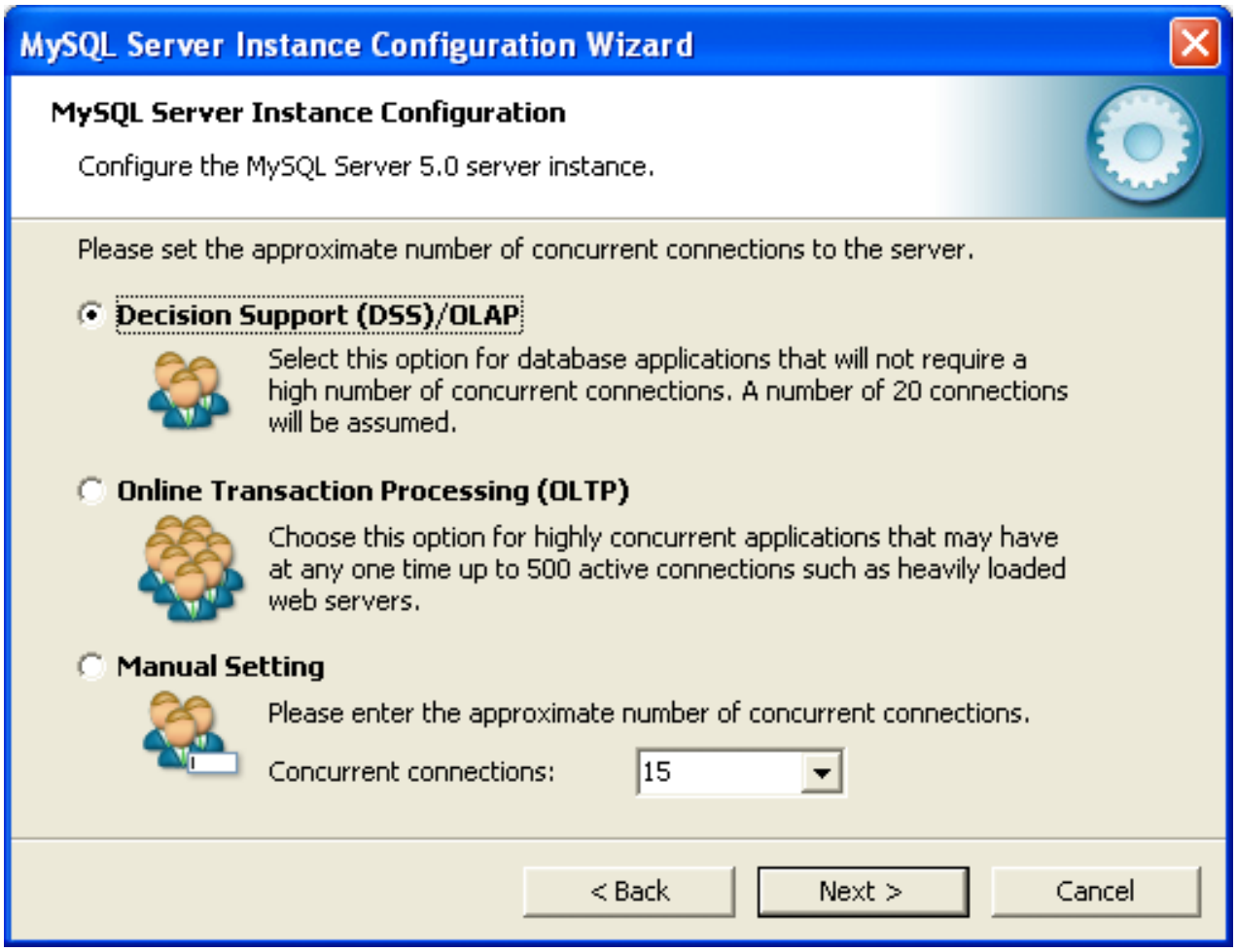


To change the default location for the `InnoDB` tablespace files, choose a new drive from the drop-down list of drive letters and choose a new path from the drop-down list of paths. To create a custom path, click the **...** button.

If you are modifying the configuration of an existing server, you must click the **Modify** button before you change the path. In this situation you must move the existing tablespace files to the new location manually before starting the server.

1.5.7 MySQL Server Instance Config Wizard: The Concurrent Connections Dialog

To prevent the server from running out of resources, it is important to limit the number of concurrent connections to the MySQL server that can be established. The **Concurrent Connections** dialog enables you to choose the expected usage of your server, and sets the limit for concurrent connections accordingly. It is also possible to set the concurrent connection limit manually.

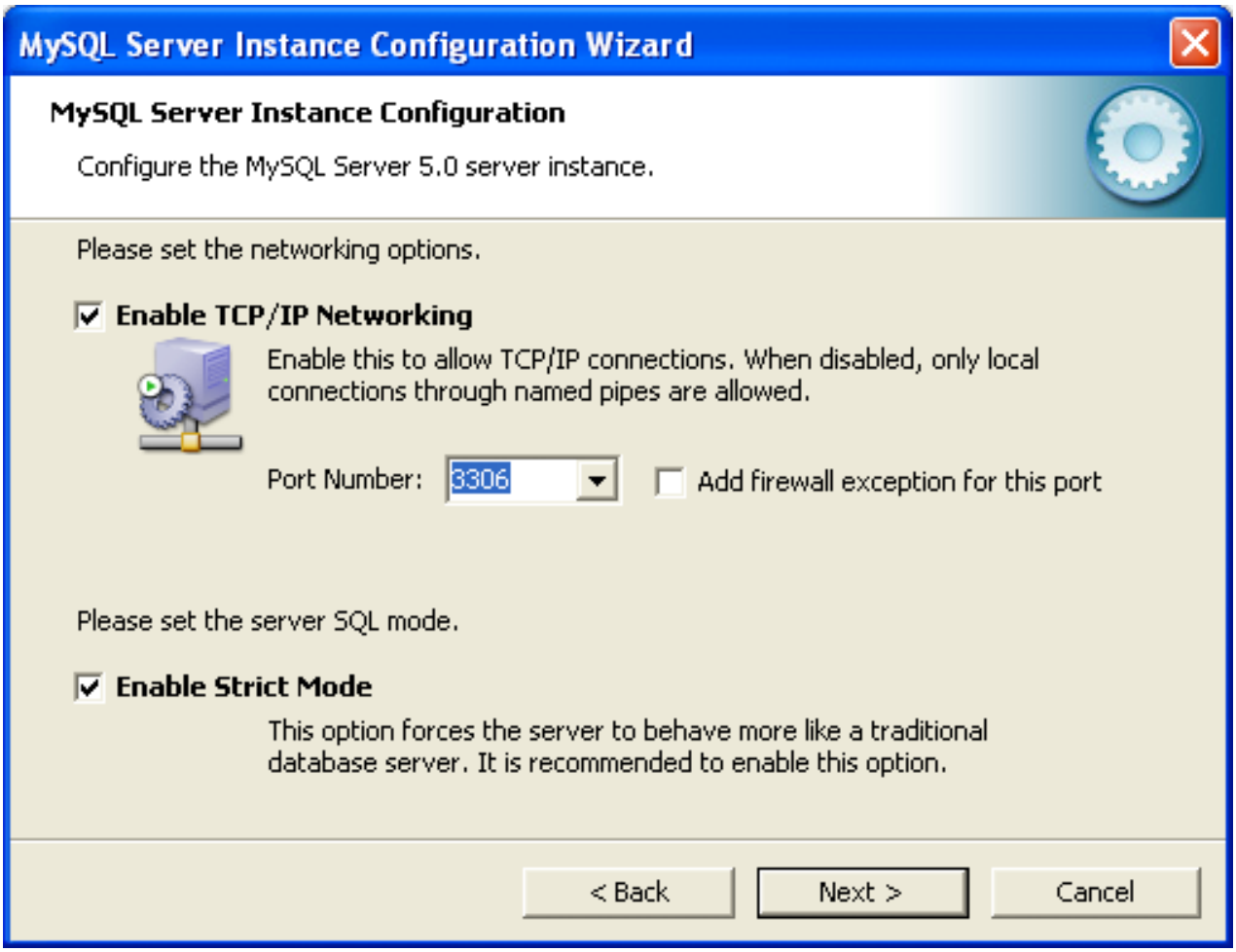


- **Decision Support (DSS)/OLAP**: Choose this option if your server does not require a large number of concurrent connections. The maximum number of connections is set at 100, with an average of 20 concurrent connections assumed.
- **Online Transaction Processing (OLTP)**: Choose this option if your server requires a large number of concurrent connections. The maximum number of connections is set at 500.

- **Manual Setting**: Choose this option to set the maximum number of concurrent connections to the server manually. Choose the number of concurrent connections from the drop-down box provided, or enter the maximum number of connections into the drop-down box if the number you desire is not listed.

1.5.8 MySQL Server Instance Config Wizard: The Networking and Strict Mode Options Dialog

Use the **Networking Options** dialog to enable or disable TCP/IP networking and to configure the port number that is used to connect to the MySQL server.



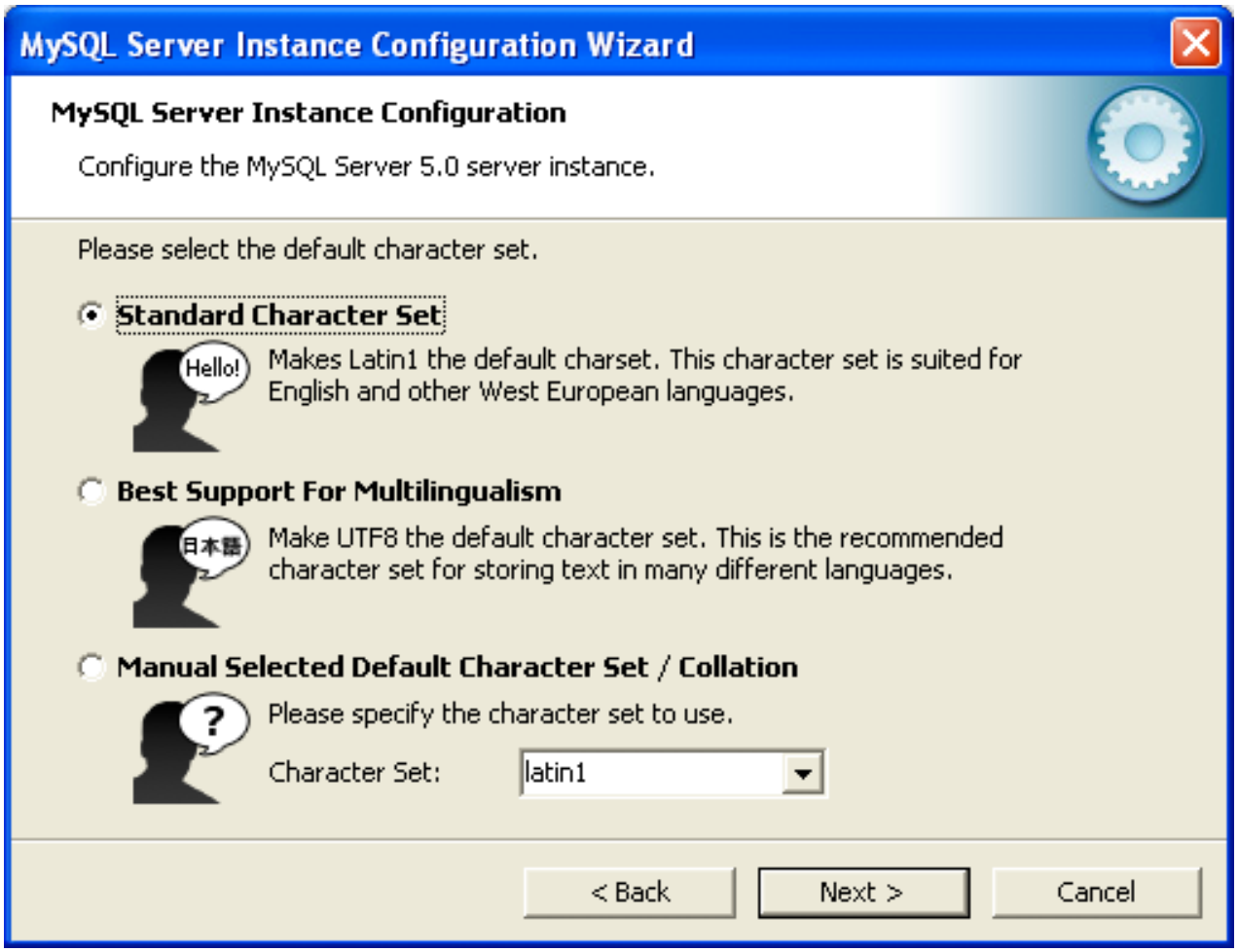
TCP/IP networking is enabled by default. To disable TCP/IP networking, uncheck the box next to the **Enable TCP/IP Networking** option.

Port 3306 is used by default. To change the port used to access MySQL, choose a new port number from the drop-down box or type a new port number directly into the drop-down box. If the port number you choose is in use, you are prompted to confirm your choice of port number.

Set the **Server SQL Mode** to either enable or disable strict mode. Enabling strict mode (default) makes MySQL behave more like other database management systems. *If you run applications that rely on MySQL's old "forgiving" behavior, make sure to either adapt those applications or to disable strict mode.* For more information about strict mode, see Server SQL Modes.

1.5.9 MySQL Server Instance Config Wizard: The Character Set Dialog

The MySQL server supports multiple character sets and it is possible to set a default server character set that is applied to all tables, columns, and databases unless overridden. Use the **Character Set** dialog to change the default character set of the MySQL server.



- **Standard Character Set**: Choose this option if you want to use `latin1` as the default server character set. `latin1` is used for English and many Western European languages.
- **Best Support For Multilingualism**: Choose this option if you want to use `utf8` as the default server character set. This is a Unicode character set that can store characters from many different languages.
- **Manual Selected Default Character Set / Collation**: Choose this option if you want to pick the server's default character set manually. Choose the desired character set from the provided drop-down list.

1.5.10 MySQL Server Instance Config Wizard: The Service Options Dialog

On Windows platforms, the MySQL server can be installed as a Windows service. When installed this way, the MySQL server can be started automatically during system startup, and even restarted automatically by Windows in the event of a service failure.

The MySQL Server Instance Config Wizard installs the MySQL server as a service by default, using the service name `MySQL`. If you do not wish to install the service, uncheck the box next to the **Install As**

Windows Service option. You can change the service name by picking a new service name from the drop-down box provided or by entering a new service name into the drop-down box.

> **Note**
>
> Service names can include any legal character except forward (`/`) or backward (`\`) slashes, and must be less than 256 characters long.

> **Warning**
>
> If you are installing multiple versions of MySQL onto the same machine, you *must* choose a different service name for each version that you install. If you do not choose a different service for each installed version then the service manager information will be inconsistent and this will cause problems when you try to uninstall a previous version.
>
> If you have already installed multiple versions using the same service name, you must manually edit the contents of the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services` parameters within the Windows registry to update the association of the service name with the correct server version.
>
> Typically, when installing multiple versions you create a service name based on the version information. For example, you might install MySQL 5.x as `mysql5`, or specific versions such as MySQL 5.1.30 as `mysql50130`.

To install the MySQL server as a service but not have it started automatically at startup, uncheck the box next to the **Launch the MySQL Server Automatically** option.

1.5.11 MySQL Server Instance Config Wizard: The Security Options Dialog

The content of the security options portion of the MySQL Server Instance Configuration Wizard will depend on whether this is a new installation, or modifying an existing installation.

- **Setting the root password for a new installation**

  It is strongly recommended that you set a `root` password for your MySQL server, and the MySQL Server Instance Config Wizard requires by default that you do so. If you do not wish to set a `root` password, uncheck the box next to the **Modify Security Settings** option.

  > **Note**
  >
  > If you have previously installed MySQL, but not deleted the data directory associated with the previous installation, you may be prompted to provide the current root password. The password will be the one configured with your old data directory. If you do not want to use this data, or do not know the root password, you should cancel the installation, delete the previous installation data, and then restart the installation process. For more information on deleting MySQL data on Microsoft Windows, see Section 1.4.3, “Removing MySQL When Installed from the MSI Package”.

- To set the `root` password, enter the desired password into both the **New root password** and **Confirm** boxes.

  Setting the root password for an existing installation

  If you are modifying the configuration of an existing configuration, or you are installing an upgrade and the MySQL Server Instance Configuration Wizard has detected an existing MySQL system, then you must enter the existing password for `root` before changing the configuration information.

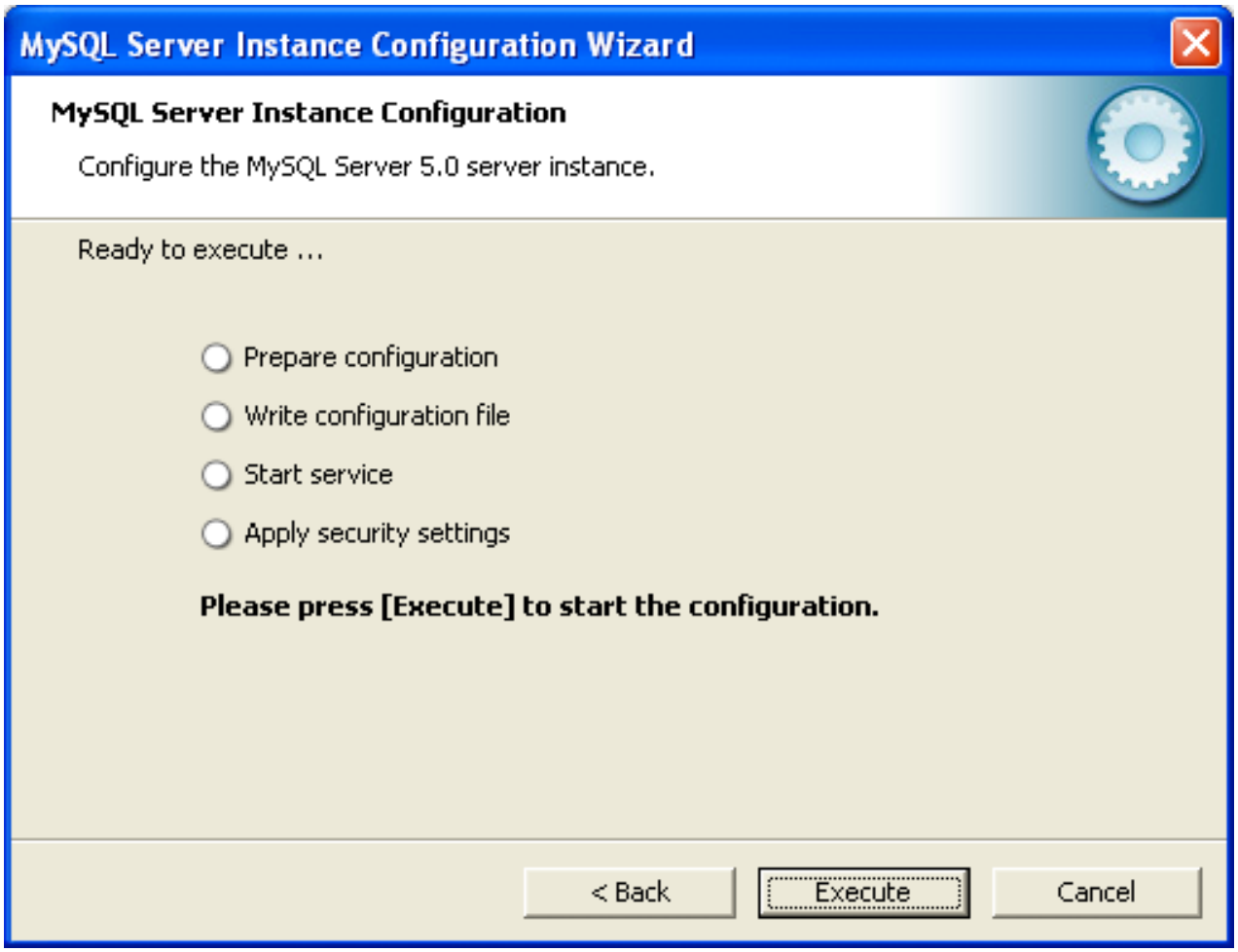
If you want to change the current `root` password, enter the desired new password into both the **New root password** and **Confirm** boxes.

To permit `root` logins from across the network, check the box next to the **Enable root access from remote machines** option. This decreases the security of your `root` account.

To create an anonymous user account, check the box next to the **Create An Anonymous Account** option. Creating an anonymous account can decrease server security and cause login and permission difficulties. For this reason, it is not recommended.

1.5.12 MySQL Server Instance Config Wizard: The Confirmation Dialog

The final dialog in the MySQL Server Instance Config Wizard is the **Confirmation Dialog**. To start the configuration process, click the **Execute** button. To return to a previous dialog, click the **Back** button. To exit the MySQL Server Instance Config Wizard without configuring the server, click the **Cancel** button.



After you click the **Execute** button, the MySQL Server Instance Config Wizard performs a series of tasks and displays the progress onscreen as the tasks are performed.

The MySQL Server Instance Config Wizard first determines configuration file options based on your choices using a template prepared by MySQL developers and engineers. This template is named `my-template.ini` and is located in your server installation directory.

The MySQL Config Wizard then writes these options to the corresponding configuration file.

If you chose to create a service for the MySQL server, the MySQL Server Instance Config Wizard creates and starts the service. If you are reconfiguring an existing service, the MySQL Server Instance Config Wizard restarts the service to apply your configuration changes.

If you chose to set a `root` password, the MySQL Config Wizard connects to the server, sets your new `root` password, and applies any other security settings you may have selected.

After the MySQL Server Instance Config Wizard has completed its tasks, it displays a summary. Click the **Finish** button to exit the MySQL Server Config Wizard.

1.5.13 MySQL Server Instance Config Wizard: Creating an Instance from the Command Line

In addition to using the GUI interface to the MySQL Server Instance Config Wizard, you can also create instances automatically from the command line.

To use the MySQL Server Instance Config Wizard on the command line, you need to use the `MySQLInstanceConfig.exe` command that is installed with MySQL in the `bin` directory within the installation directory. `MySQLInstanceConfig.exe` takes a number of command-line arguments the set the properties that would normally be selected through the GUI interface, and then creates a new configuration file (`my.ini`) by combining these selections with a template configuration file to produce the working configuration file.

The main command line options are provided in the table below. Some of the options are required, while some options are optional.

Table 1.4 MySQL Server Instance Config Wizard Command Line Options

Option	Description
Required Parameters	
`-nPRODUCTNAME`	The name of the instance when installed
`-pPATH`	Path of the base directory for installation. This is equivalent to the directory when using the `basedir` configuration parameter
`-vVERSION`	The version tag to use for this installation
Action to Perform	
`-i`	Install an instance
`-r`	Remove an instance
`-s`	Stop an existing instance
`-q`	Perform the operation quietly
`-lFILENAME`	Sae the installation progress in a logfile
Config File to Use	
`-tFILENAME`	Path to the template config file that will be used to generate the installed configuration file
`-cFILENAME`	Path to a config file to be generated

The `-t` and `-c` options work together to set the configuration parameters for a new instance. The `-t` option specifies the template configuration file to use as the basic configuration, which are then merged with the configuration parameters generated by the MySQL Server Instance Config Wizard into the configuration file specified by the `-c` option.

A sample template file, `my-template.ini` is provided in the toplevel MySQL installation directory. The file contains elements are replaced automatically by the MySQL Server Instance Config Wizard during configuration.

If you specify a configuration file that already exists, the existing configuration file will be saved in the file with the original, with the date and time added. For example, the `mysql.ini` will be copied to `mysql 2009-10-27 1646.ini.bak`.

The parameters that you can specify on the command line are listed in the table below.

Table 1.5 MySQL Server Instance Config Wizard Parameters

Parameter	Description
`ServiceName=$`	Specify the name of the service to be created

Parameter	Description
`AddBinToPath={yes | no}`	Specifies whether to add the binary directory of MySQL to the standard `PATH` environment variable
`ServerType={DEVELOPMENT | SERVER | DEDICATED}`	Specify the server type. For more information, see Section 1.5.4, “MySQL Server Instance Config Wizard: The Server Type Dialog”
`DatabaseType={MIXED | INNODB | MYISAM}`	Specify the default database type. For more information, see Section 1.5.5, “MySQL Server Instance Config Wizard: The Database Usage Dialog”
`ConnectionUsage={DSS | OLTP}`	Specify the type of connection support, this automates the setting for the number of concurrent connections (see the `ConnectionCount` parameter). For more information, see Section 1.5.7, “MySQL Server Instance Config Wizard: The Concurrent Connections Dialog”
`ConnectionCount=#`	Specify the number of concurrent connections to support. For more information, see Section 1.5.4, “MySQL Server Instance Config Wizard: The Server Type Dialog”
`SkipNetworking={yes | no}`	Specify whether network support should be supported. Specifying `yes` disables network access altogether
`Port=#`	Specify the network port number to use for network connections. For more information, see Section 1.5.8, “MySQL Server Instance Config Wizard: The Networking and Strict Mode Options Dialog”
`StrictMode={yes | no}`	Specify whether to use the `strict` SQL mode. For more information, see Section 1.5.8, “MySQL Server Instance Config Wizard: The Networking and Strict Mode Options Dialog”
`Charset=$`	Specify the default character set. For more information, see Section 1.5.9, “MySQL Server Instance Config Wizard: The Character Set Dialog”
`RootPassword=$`	Specify the root password
`RootCurrentPassword=$`	Specify the current root password then stopping or reconfiguring an existing service

> **Note**
>
> When specifying options on the command line, you can enclose the entire command-line option and the value you are specifying using double quotation marks. This enables you to use spaces in the options. For example, `"-cC:\mysql.ini"`.

The following command installs a MySQL Server 5.1 instance from the directory `C:\Program Files\MySQL\MySQL Server 5.1` using the service name `MySQL51` and setting the root password to 1234.

```
shell> MySQLInstanceConfig.exe -i -q "-lC:\mysql_install_log.txt" »
   "-nMySQL Server 5.1" "-pC:\Program Files\MySQL\MySQL Server 5.1" -v5.1.73 »
   "-tmy-template.ini" "-cC:\mytest.ini" ServerType=DEVELOPMENT DatabaseType=MIXED »
   ConnectionUsage=DSS Port=3311 ServiceName=MySQL51 RootPassword=1234
```

In the above example, a log file will be generated in `mysql_install_log.txt` containing the information about the instance creation process. The log file generated by the above example is shown below:

```
Welcome to the MySQL Server Instance Configuration Wizard 1.0.16.0
Date: 2009-10-27 17:07:21
```

```
Installing service ...

Product Name:         MySQL Server 5.1
Version:              5.1.73
Installation Path:    C:\Program Files\MySQL\MySQL Server 5.1\

Creating configuration file C:\mytest.ini using template my-template.ini.
Options:
DEVELOPMENT
MIXED
DSS
STRICTMODE

Variables:
port: 3311
default-character-set: latin1
basedir: "C:/Program Files/MySQL/MySQL Server 5.1/"
datadir: "C:/Program Files/MySQL/MySQL Server 5.1/Data/"


Creating Windows service entry.
Service name: "MySQL51"
Parameters:   "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld" --defaults-file="C:\mytest.ini" MySQL51
Windows service MySQL51 installed.
```

When using the command line, the return values in the following table indicate an error performing the specified option.

Table 1.6 Return Value from MySQL Server Instance Config Wizard

Value	Description
2	Configuration template file cannot be found
3	The Windows service entry cannot be created
4	Could not connect to the Service Control Manager
5	The MySQL service cannot be started
6	The MySQL service cannot be stopped
7	The security settings cannot be applied
8	The configuration file cannot be written
9	The Windows service entry cannot be removed

You can perform an installation of MySQL automatically using the MSI package. For more information, see Section 1.4.2, “Automating MySQL Installation on Microsoft Windows Using the MSI Package”.

1.6 Installing MySQL on Microsoft Windows Using a noinstall Zip Archive

Users who are installing from the `noinstall` package can use the instructions in this section to manually install MySQL. The process for installing MySQL from a Zip archive is as follows:

1. Extract the archive to the desired install directory
2. Create an option file
3. Choose a MySQL server type
4. Start the MySQL server

5. Secure the default user accounts

This process is described in the sections that follow.

1.6.1 Extracting the Install Archive

To install MySQL manually, do the following:

1. If you are upgrading from a previous version please refer to Section 1.9, “Upgrading MySQL Server on Microsoft Windows”, before beginning the upgrade process.

2. Make sure that you are logged in as a user with administrator privileges.

3. Choose an installation location. Traditionally, the MySQL server is installed in `C:\mysql`. The MySQL Installation Wizard installs MySQL under `C:\Program Files\MySQL`. If you do not install MySQL at `C:\mysql`, you must specify the path to the install directory during startup or in an option file. See Section 1.6.2, “Creating an Option File”.

4. Extract the install archive to the chosen installation location using your preferred Zip archive tool. Some tools may extract the archive to a folder within your chosen installation location. If this occurs, you can move the contents of the subfolder into the chosen installation location.

1.6.2 Creating an Option File

If you need to specify startup options when you run the server, you can indicate them on the command line or place them in an option file. For options that are used every time the server starts, you may find it most convenient to use an option file to specify your MySQL configuration. This is particularly true under the following circumstances:

- The installation or data directory locations are different from the default locations (`C:\Program Files\MySQL\MySQL Server 5.1` and `C:\Program Files\MySQL\MySQL Server 5.1\data`).
- You need to tune the server settings, such as memory, cache, or InnoDB configuration information.

When the MySQL server starts on Windows, it looks for option files in several locations, such as the Windows directory, `C:\`, and the MySQL installation directory (for the full list of locations, see Using Option Files). The Windows directory typically is named something like `C:\WINDOWS`. You can determine its exact location from the value of the `WINDIR` environment variable using the following command:

```
C:\> echo %WINDIR%
```

MySQL looks for options in each location first in the `my.ini` file, and then in the `my.cnf` file. However, to avoid confusion, it is best if you use only one file. If your PC uses a boot loader where `C:` is not the boot drive, your only option is to use the `my.ini` file. Whichever option file you use, it must be a plain text file.

You can also make use of the example option files included with your MySQL distribution; see Server Configuration Defaults.

An option file can be created and modified with any text editor, such as Notepad. For example, if MySQL is installed in `E:\mysql` and the data directory is in `E:\mydata\data`, you can create an option file containing a `[mysqld]` section to specify values for the `basedir` and `datadir` options:

```
[mysqld]
# set basedir to your installation path
```

```
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=E:/mydata/data
```

Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=E:\\mysql
# set datadir to the location of your data directory
datadir=E:\\mydata\\data
```

The rules for use of backslash in option file values are given in Using Option Files.

In MySQL 5.1.23 and earlier, the MySQL installer places the data directory directly under the directory where you install MySQL. On MySQL 5.1.24 and later, the data directory is located within the `AppData` directory for the user running MySQL.

If you would like to use a data directory in a different location, you should copy the entire contents of the `data` directory to the new location. For example, if you want to use `E:\mydata` as the data directory instead, you must do two things:

1. Move the entire `data` directory and all of its contents from the default location (for example `C:\Program Files\MySQL\MySQL Server 5.1\data`) to `E:\mydata`.
2. Use a `--datadir` option to specify the new data directory location each time you start the server.

1.6.3 Selecting a MySQL Server Type

The following table shows the available servers for Windows in MySQL 5.1.20 and earlier.

Binary	Description
`mysqld-nt`	Optimized binary with named-pipe support
`mysqld`	Optimized binary without named-pipe support
`mysqld-debug`	Like `mysqld-nt`, but compiled with full debugging and automatic memory allocation checking

The following table shows the available servers for Windows in MySQL 5.1.21 and later.

Table 1.7 mysqld binary types for Microsoft Windows MySQL 5.1.21 and later

Binary	Description
`mysqld`	Optimized binary with named-pipe support
`mysqld-debug`	Like `mysqld`, but compiled with full debugging and automatic memory allocation checking

All of the preceding binaries are optimized for modern Intel processors, but should work on any Intel i386-class or higher processor.

Each of the servers in a distribution support the same set of storage engines. The `SHOW ENGINES` statement displays which engines a given server supports.

All Windows MySQL 5.1 servers have support for symbolic linking of database directories.

MySQL supports TCP/IP on all Windows platforms. MySQL servers on Windows support named pipes as indicated in the following list. However, the default is to use TCP/IP regardless of platform. (Named pipes are slower than TCP/IP in many Windows configurations.)

Use of named pipes is subject to these conditions:

- Named pipes are enabled only if you start the server with the `--enable-named-pipe` option. It is necessary to use this option explicitly because some users have experienced problems with shutting down the MySQL server when named pipes were used.
- For MySQL 5.1.20 and earlier, named-pipe connections are permitted only by the `mysqld-nt` and `mysqld-debug` servers. For MySQL 5.1.21 and later, the `mysqld` and `mysqld-debug` servers both contain support for named-pipe connections.

> **Note**
>
> Most of the examples in this manual use `mysqld` as the server name. If you choose to use a different server, such as `mysqld-nt` or `mysqld-debug`, make the appropriate substitutions in the commands that are shown in the examples.

1.6.4 Starting MySQL Server on Microsoft Windows for the First Time

This section gives a general overview of starting the MySQL server. The following sections provide more specific information for starting the MySQL server from the command line or as a Windows service.

The information here applies primarily if you installed MySQL using the `Noinstall` version, or if you wish to configure and test MySQL manually rather than with the GUI tools.

> **Note**
>
> The MySQL Notifier GUI can be used to start/stop/restart the MySQL server at any time.

The examples in these sections assume that MySQL is installed under the default location of `C:\Program Files\MySQL\MySQL Server 5.1`. Adjust the path names shown in the examples if you have MySQL installed in a different location.

Clients have two options. They can use TCP/IP, or they can use a named pipe if the server supports named-pipe connections.

MySQL for Windows also supports shared-memory connections if the server is started with the `--shared-memory` option. Clients can connect through shared memory by using the `--protocol=MEMORY` option.

For information about which server binary to run, see Section 1.6.3, “Selecting a MySQL Server Type”.

Testing is best done from a command prompt in a console window (or “DOS window”). In this way you can have the server display status messages in the window where they are easy to see. If something is wrong with your configuration, these messages make it easier for you to identify and fix any problems.

To start the server, enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld" --console
```

For a server that includes `InnoDB` support, you should see the messages similar to those following as it starts (the path names and sizes may differ):

```
InnoDB: The first specified datafile c:\ibdata\ibdata1 did not exist:
InnoDB: a new database to be created!
InnoDB: Setting file c:\ibdata\ibdata1 size to 209715200
InnoDB: Database physically writes the file full: wait...
InnoDB: Log file c:\iblogs\ib_logfile0 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile0 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile1 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile1 size to 31457280
InnoDB: Log file c:\iblogs\ib_logfile2 did not exist: new to be created
InnoDB: Setting log file c:\iblogs\ib_logfile2 size to 31457280
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: creating foreign key constraint system tables
InnoDB: foreign key constraint system tables created
011024 10:58:25  InnoDB: Started
```

When the server finishes its startup sequence, you should see something like this, which indicates that the server is ready to service client connections:

```
mysqld: ready for connections
Version: '5.1.73'  socket: ''  port: 3306
```

The server continues to write to the console any further diagnostic output it produces. You can open a new console window in which to run client programs.

If you omit the `--console` option, the server writes diagnostic output to the error log in the data directory (`C:\Program Files\MySQL\MySQL Server 5.1\data` by default). The error log is the file with the `.err` extension, and may be set using the `--log-error` option.

> **Note**
>
> The accounts that are listed in the MySQL grant tables initially have no passwords. After starting the server, you should set up passwords for them using the instructions in Securing the Initial MySQL Accounts.

1.6.5 Starting MySQL Server from the Windows Command Line

The MySQL server can be started manually from the command line. This can be done on any version of Windows.

> **Note**
>
> The MySQL Notifier GUI can also be used to start/stop/restart the MySQL server.

To start the `mysqld` server from the command line, you should start a console window (or “DOS window”) and enter this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld"
```

The path to `mysqld` may vary depending on the install location of MySQL on your system.

You can stop the MySQL server by executing this command:

```
```

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqladmin" -u root shutdown
```

> **Note**
>
> If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

> **Note**
>
> Users in the MySQL grant system are wholly independent from any login users under Microsoft Windows.

If `mysqld` doesn't start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. By default, the error log is located in the `C:\Program Files\MySQL\MySQL Server 5.1\data` directory. It is the file with a suffix of `.err`, or may be specified by passing in the `--log-error` option. Alternatively, you can try to start the server with the `--console` option; in this case, the server may display some useful information on the screen that will help solve the problem.

The last option is to start `mysqld` with the `--standalone` and `--debug` options. In this case, `mysqld` writes a log file `C:\mysqld.trace` that should contain the reason why `mysqld` doesn't start. See The DBUG Package.

Use `mysqld --verbose --help` to display all the options that `mysqld` supports.

1.6.6 Customizing the PATH for MySQL Tools

To make it easier to invoke MySQL programs, you can add the path name of the MySQL `bin` directory to your Windows system `PATH` environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL `bin` directory (for example, `C:\Program Files\MySQL\MySQL Server 5.1\bin`)

  > **Note**
  >
  > There must be a semicolon separating this path from any values present in this field.

  Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the `mysql` client, and all MySQL command-line utilities such as `mysqladmin` and `mysqldump`.

You should not add the MySQL `bin` directory to your Windows `PATH` if you are running multiple MySQL servers on the same machine.

> **Warning**
>
> You must exercise great care when editing your system `PATH` by hand; accidental deletion or modification of any portion of the existing `PATH` value can leave you with a malfunctioning or even unusable system.

1.6.7 Starting MySQL Server as a Microsoft Windows Service

On Windows, the recommended way to run MySQL is to install it as a Windows service, so that MySQL starts and stops automatically when Windows starts and stops, and can be managed using the service manager framework. A MySQL server installed as a service can also be controlled from the command line using `NET` commands, or with the graphical `Services` utility. Generally, to install MySQL as a Windows service you should be logged in using an account that has administrator rights.

> **Note**
>
> The MySQL Notifier GUI can also be used to monitor the status of the MySQL service.

The `Services` utility (the Windows `Service Control Manager`) can be found in the Windows Control Panel (under **Administrative Tools** on Windows 2000, XP, Vista, and Server 2003). To avoid conflicts, it is advisable to close the `Services` utility while performing server installation or removal operations from the command line.

Installing the service

Before installing MySQL as a Windows service, you should first stop the current server if it is running by using the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqladmin"
          -u root shutdown
```

> **Note**
>
> If the MySQL `root` user account has a password, you need to invoke `mysqladmin` with the `-p` option and supply the password when prompted.

This command invokes the MySQL administrative utility `mysqladmin` to connect to the server and tell it to shut down. The command connects as the MySQL `root` user, which is the default administrative account in the MySQL grant system.

> **Note**
>
> Users in the MySQL grant system are wholly independent from any login users under Windows.

Install the server as a service using this command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld" --install
```

The service-installation command does not start the server. Instructions for that are given later in this section.

To make it easier to invoke MySQL programs, you can add the path name of the MySQL `bin` directory to your Windows system `PATH` environment variable:

- On the Windows desktop, right-click the **My Computer** icon, and select **Properties**.
- Next select the **Advanced** tab from the **System Properties** menu that appears, and click the **Environment Variables** button.
- Under **System Variables**, select **Path**, and then click the **Edit** button. The **Edit System Variable** dialogue should appear.
- Place your cursor at the end of the text appearing in the space marked **Variable Value**. (Use the **End** key to ensure that your cursor is positioned at the very end of the text in this space.) Then enter the complete path name of your MySQL `bin` directory (for example, `C:\Program Files\MySQL\MySQL Server 5.1\bin`), and there should be a semicolon separating this path from any values present in this field. Dismiss this dialogue, and each dialogue in turn, by clicking **OK** until all of the dialogues that were opened have been dismissed. You should now be able to invoke any MySQL executable program by typing its name at the DOS prompt from any directory on the system, without having to supply the path. This includes the servers, the `mysql` client, and all MySQL command-line utilities such as `mysqladmin` and `mysqldump`.

  You should not add the MySQL `bin` directory to your Windows `PATH` if you are running multiple MySQL servers on the same machine.

  > **Warning**
  >
  > You must exercise great care when editing your system `PATH` by hand; accidental deletion or modification of any portion of the existing `PATH` value can leave you with a malfunctioning or even unusable system.

The following additional arguments can be used when installing the service:

- You can specify a service name immediately following the `--install` option. The default service name is `MySQL`.
- If a service name is given, it can be followed by a single option. By convention, this should be `--defaults-file=`*`file_name`* to specify the name of an option file from which the server should read options when it starts.

  The use of a single option other than `--defaults-file` is possible but discouraged. `--defaults-file` is more flexible because it enables you to specify multiple startup options for the server by placing them in the named option file.
- You can also specify a `--local-service` option following the service name. This causes the server to run using the `LocalService` Windows account that has limited system privileges. This account is available only for Windows XP or newer. If both `--defaults-file` and `--local-service` are given following the service name, they can be in any order.

For a MySQL server that is installed as a Windows service, the following rules determine the service name and option files that the server uses:

- If the service-installation command specifies no service name or the default service name (`MySQL`) following the `--install` option, the server uses the a service name of `MySQL` and reads options from the `[mysqld]` group in the standard option files.
- If the service-installation command specifies a service name other than `MySQL` following the `--install` option, the server uses that service name. It reads options from the `[mysqld]` group and the group that

has the same name as the service in the standard option files. This enables you to use the `[mysqld]` group for options that should be used by all MySQL services, and an option group with the service name for use by the server installed with that service name.

- If the service-installation command specifies a `--defaults-file` option after the service name, the server reads options the same way as described in the previous item, except that it reads options only from the named file and ignores the standard option files.

As a more complex example, consider the following command:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld"
         --install MySQL --defaults-file=C:\my-opts.cnf
```

Here, the default service name (`MySQL`) is given after the `--install` option. If no `--defaults-file` option had been given, this command would have the effect of causing the server to read the `[mysqld]` group from the standard option files. However, because the `--defaults-file` option is present, the server reads options from the `[mysqld]` option group, and only from the named file.

> **Note**
>
> On Windows, if the server is started with the `--defaults-file` and `--install` options, `--install` must be first. Otherwise, `mysqld.exe` will attempt to start the MySQL server.

You can also specify options as Start parameters in the Windows `Services` utility before you start the MySQL service.

Starting the service

Once a MySQL server has been installed as a service, Windows starts the service automatically whenever Windows starts. The service also can be started immediately from the `Services` utility, or by using a `NET START MySQL` command. The `NET` command is not case sensitive.

When run as a service, `mysqld` has no access to a console window, so no messages can be seen there. If `mysqld` does not start, check the error log to see whether the server wrote any messages there to indicate the cause of the problem. The error log is located in the MySQL data directory (for example, `C:\Program Files\MySQL\MySQL Server 5.1\data`). It is the file with a suffix of `.err`.

When a MySQL server has been installed as a service, and the service is running, Windows stops the service automatically when Windows shuts down. The server also can be stopped manually by using the `Services` utility, the `NET STOP MySQL` command, or the `mysqladmin shutdown` command.

You also have the choice of installing the server as a manual service if you do not wish for the service to be started automatically during the boot process. To do this, use the `--install-manual` option rather than the `--install` option:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld" --install-manual
```

Removing the service

To remove a server that is installed as a service, first stop it if it is running by executing `NET STOP MySQL`. Then use the `--remove` option to remove it:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqld" --remove
```

If `mysqld` is not running as a service, you can start it from the command line. For instructions, see Section 1.6.5, “Starting MySQL Server from the Windows Command Line”.

If you encounter difficulties during installation, see Section 1.7, “Troubleshooting a Microsoft Windows MySQL Server Installation”.

For more information about stopping or removing a MySQL Windows service, see Starting Multiple MySQL Instances as Windows Services.

1.6.8 Testing The MySQL Server Installation on Microsoft Windows

You can test whether the MySQL server is working by executing any of the following commands:

```
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqlshow"
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqlshow" -u root mysql
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysqladmin" version status proc
C:\> "C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql" test
```

> **Note**
>
> By default, `mysqlshow` will try to connect using the `ODBC` user. This user is not created by default. You should specify a valid user, or `root` with the right password to check the operation of the server.

If `mysqld` is slow to respond to TCP/IP connections from client programs, there is probably a problem with your DNS. In this case, start `mysqld` with the `--skip-name-resolve` option and use only `localhost` and IP addresses in the `Host` column of the MySQL grant tables. (Be sure that an account exists that specifies an IP address or you may not be able to connect.)

You can force a MySQL client to use a named-pipe connection rather than TCP/IP by specifying the `--pipe` or `--protocol=PIPE` option, or by specifying `.` (period) as the host name. Use the `--socket` option to specify the name of the pipe if you do not want to use the default pipe name.

If you have set a password for the `root` account, deleted the anonymous account, or created a new user account, then to connect to the MySQL server you must use the appropriate `-u` and `-p` options with the commands shown previously. See Connecting to the MySQL Server.

For more information about `mysqlshow`, see `mysqlshow` — Display Database, Table, and Column Information.

1.7 Troubleshooting a Microsoft Windows MySQL Server Installation

When installing and running MySQL for the first time, you may encounter certain errors that prevent the MySQL server from starting. This section helps you diagnose and correct some of these errors.

Your first resource when troubleshooting server issues is the error log. The MySQL server uses the error log to record information relevant to the error that prevents the server from starting. The error log is located in the data directory specified in your `my.ini` file. The default data directory location is `C:\Program Files\MySQL\MySQL Server 5.1\data`, or `C:\ProgramData\Mysql` on Windows 7 and Windows Server 2008. The `C:\ProgramData` directory is hidden by default. You need to change your folder options to see the directory and contents. For more information on the error log and understanding the content, see The Error Log.

For information regarding possible errors, also consult the console messages displayed when the MySQL service is starting. Use the `NET START MySQL` command from the command line after installing `mysqld`

as a service to see any error messages regarding the starting of the MySQL server as a service. See Section 1.6.7, “Starting MySQL Server as a Microsoft Windows Service”.

The following examples show other common error messages you might encounter when installing MySQL and starting the server for the first time:

- If the MySQL server cannot find the `mysql` privileges database or other critical files, it displays these messages:

```
System error 1067 has occurred.
Fatal error: Can't open and lock privilege tables:
Table 'mysql.user' doesn't exist
```

  These messages often occur when the MySQL base or data directories are installed in different locations than the default locations (`C:\Program Files\MySQL\MySQL Server 5.1` and `C:\Program Files\MySQL\MySQL Server 5.1\data`, respectively).

  This situation can occur when MySQL is upgraded and installed to a new location, but the configuration file is not updated to reflect the new location. In addition, old and new configuration files might conflict. Be sure to delete or rename any old configuration files when upgrading MySQL.

  If you have installed MySQL to a directory other than `C:\Program Files\MySQL\MySQL Server 5.1`, ensure that the MySQL server is aware of this through the use of a configuration (`my.ini`) file. Put the `my.ini` file in your Windows directory, typically `C:\WINDOWS`. To determine its exact location from the value of the `WINDIR` environment variable, issue the following command from the command prompt:

```
C:\> echo %WINDIR%
```

  You can create or modify an option file with any text editor, such as Notepad. For example, if MySQL is installed in `E:\mysql` and the data directory is `D:\MySQLdata`, you can create the option file and set up a `[mysqld]` section to specify values for the `basedir` and `datadir` options:

```
[mysqld]
# set basedir to your installation path
basedir=E:/mysql
# set datadir to the location of your data directory
datadir=D:/MySQLdata
```

  Microsoft Windows path names are specified in option files using (forward) slashes rather than backslashes. If you do use backslashes, double them:

```
[mysqld]
# set basedir to your installation path
basedir=C:\\Program Files\\MySQL\\MySQL Server 5.1
# set datadir to the location of your data directory
datadir=D:\\MySQLdata
```

  The rules for use of backslash in option file values are given in Using Option Files.

  If you change the `datadir` value in your MySQL configuration file, you must move the contents of the existing MySQL data directory before restarting the MySQL server.

  See Section 1.6.2, “Creating an Option File”.

- If you reinstall or upgrade MySQL without first stopping and removing the existing MySQL service and install MySQL using the MySQL Config Wizard, you may see this error:

```
Error: Cannot create Windows service for MySql. Error: 0
```

This occurs when the Configuration Wizard tries to install the service and finds an existing service with the same name.

One solution to this problem is to choose a service name other than `mysql` when using the configuration wizard. This enables the new service to be installed correctly, but leaves the outdated service in place. Although this is harmless, it is best to remove old services that are no longer in use.

To permanently remove the old `mysql` service, execute the following command as a user with administrative privileges, on the command line:

```
C:\> sc delete mysql
[SC] DeleteService SUCCESS
```

If the `sc` utility is not available for your version of Windows, download the `delsrv` utility from http://www.microsoft.com/windows2000/techinfo/reskit/tools/existing/delsrv-o.asp and use the `delsrv mysql` syntax.

1.8 Windows Postinstallation Procedures

GUI tools exist that perform most of the tasks described in this section, including:

- MySQL Installer: Used to install and upgrade MySQL products.
- MySQL Workbench: Manages the MySQL server and edits SQL queries.
- MySQL Notifier: Starts, stops, or restarts the MySQL server, and monitors its status.
- MySQL for Excel: Edits MySQL data with Microsoft Excel.

On Windows, you need not create the data directory and the grant tables. MySQL Windows distributions include the grant tables with a set of preinitialized accounts in the `mysql` database under the data directory.

Regarding passwords, if you installed MySQL using the Windows Installation Wizard, you may have already assigned passwords to the accounts. (See Section 1.4.1, “Using the MySQL Installation Wizard for Microsoft Windows”.) Otherwise, use the password-assignment procedure given in Securing the Initial MySQL Accounts.

Before assigning passwords, you might want to try running some client programs to make sure that you can connect to the server and that it is operating properly. Make sure that the server is running (see Section 1.6.4, “Starting MySQL Server on Microsoft Windows for the First Time”). You can also set up a MySQL service that runs automatically when Windows starts (see Section 1.6.7, “Starting MySQL Server as a Microsoft Windows Service”).

These instructions assume that your current location is the MySQL installation directory and that it has a `bin` subdirectory containing the MySQL programs used here. If that is not true, adjust the command path names accordingly.

If you installed MySQL using the Windows installation Wizard (see Section 1.4.1, “Using the MySQL Installation Wizard for Microsoft Windows”), the default installation directory is `C:\Program Files\MySQL\MySQL Server 5.1`:

```
C:\> cd "C:\Program Files\MySQL\MySQL Server 5.1"
```

A common installation location for installation from a Zip package is `C:\mysql`:

```
C:\> cd C:\mysql
```

Alternatively, add the `bin` directory to your `PATH` environment variable setting. That enables your command interpreter to find MySQL programs properly, so that you can run a program by typing only its name, not its path name. See Section 1.6.6, “Customizing the PATH for MySQL Tools”.

With the server running, issue the following commands to verify that you can retrieve information from the server. The output should be similar to that shown here.

Use `mysqlshow` to see what databases exist:

```
C:\> bin\mysqlshow
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
```

The list of installed databases may vary, but will always include the minimum of `mysql` and `information_schema`.

The preceding command (and commands for other MySQL programs such as `mysql`) may not work if the correct MySQL account does not exist. For example, the program may fail with an error, or you may not be able to view all databases. If you installed using the MSI packages and used the MySQL Server Instance Config Wizard, then the `root` user will have been created automatically with the password you supplied. In this case, you should use the `-u root` and `-p` options. (You will also need to use the `-u root` and `-p` options if you have already secured the initial MySQL accounts.) With `-p`, you will be prompted for the `root` password. For example:

```
C:\> bin\mysqlshow -u root -p
Enter password: (enter root password here)
+--------------------+
|     Databases      |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
```

If you specify a database name, `mysqlshow` displays a list of the tables within the database:

```
C:\> bin\mysqlshow mysql
Database: mysql
+---------------------------+
|          Tables           |
+---------------------------+
| columns_priv              |
| db                        |
| event                     |
| func                      |
| help_category             |
| help_keyword              |
| help_relation             |
```

```
| help_topic                |
| host                      |
| ndb_binlog_index          |
| plugin                    |
| proc                      |
| procs_priv                |
| servers                   |
| tables_priv               |
| time_zone                 |
| time_zone_leap_second     |
| time_zone_name            |
| time_zone_transition      |
| time_zone_transition_type |
| user                      |
+---------------------------+
```

Use the `mysql` program to select information from a table in the `mysql` database:

```
C:\> bin\mysql -e "SELECT User, Host FROM mysql.user" mysql
+------+-----------+
| User | Host      |
+------+-----------+
| root | localhost |
+------+-----------+
```

For more information about `mysql` and `mysqlshow`, see `mysql` — The MySQL Command-Line Tool, and `mysqlshow` — Display Database, Table, and Column Information.

If you are running a version of Windows that supports services, you can set up the MySQL server to run automatically when Windows starts. See Section 1.6.7, “Starting MySQL Server as a Microsoft Windows Service”.

1.9 Upgrading MySQL Server on Microsoft Windows

To upgrade MySQL on Windows, follow these steps:

1. Review Upgrading MySQL, for additional information on upgrading MySQL that is not specific to Windows.
2. Always back up your current MySQL installation before performing an upgrade. See Database Backup Methods.
3. Download the latest Windows distribution of MySQL from http://dev.mysql.com/downloads/.
4. Before upgrading MySQL, stop the server. If the server is installed as a service, stop the service with the following command from the command prompt:

   ```
   C:\> NET STOP MySQL
   ```

   If you are not running the MySQL server as a service, use `mysqladmin` to stop it. For example, before upgrading from MySQL 5.0 to 5.1, use `mysqladmin` from MySQL 5.0 as follows:

   ```
   C:\> "C:\Program Files\MySQL\MySQL Server 5.0\bin\mysqladmin" -u root shutdown
   ```

   > **Note**
   >
   > If the MySQL `root` user account has a password, invoke `mysqladmin` with the `-p` option and enter the password when prompted.

5. Before upgrading to MySQL 5.1 from a version previous to 4.1.5, or from a version of MySQL installed from a Zip archive to a version of MySQL installed with the MySQL Installation Wizard, you must first manually remove the previous installation and MySQL service (if the server is installed as a service).

   To remove the MySQL service, use the following command:

   ```
   C:\> C:\mysql\bin\mysqld --remove
   ```

   > **Important**
   >
   > If you do not remove the existing service, the MySQL Installation Wizard may fail to properly install the new MySQL service.

6. When upgrading from MySQL 5.1.23 to MySQL 5.1.24, the change in the default location of the data directory from a directory within the MySQL installation to the `AppData` folder means that you must manually copy the data files from your old installation to the new location.

7. If you are using the MySQL Installation Wizard, start the wizard as described in Section 1.4.1, “Using the MySQL Installation Wizard for Microsoft Windows”.

8. If you are upgrading MySQL from a Zip archive, extract the archive. You may either overwrite your existing MySQL installation (usually located at `C:\mysql`), or install it into a different directory, such as `C:\mysql5`. Overwriting the existing installation is recommended. However, for upgrades (as opposed to installing for the first time), you must remove the data directory from your existing MySQL installation to avoid replacing your current data files. To do so, follow these steps:

   a. Unzip the Zip archive in some location other than your current MySQL installation

   b. Remove the data directory

   c. Rezip the Zip archive

   d. Unzip the modified Zip archive on top of your existing installation

   Alternatively:

   a. Unzip the Zip archive in some location other than your current MySQL installation

   b. Remove the data directory

   c. Move the data directory from the current MySQL installation to the location of the just-removed data directory

   d. Remove the current MySQL installation

   e. Move the unzipped installation to the location of the just-removed installation

9. If you were running MySQL as a Windows service and you had to remove the service earlier in this procedure, reinstall the service. (See Section 1.6.7, “Starting MySQL Server as a Microsoft Windows Service”.)

10. Restart the server. For example, use `NET START MySQL` if you run MySQL as a service, or invoke `mysqld` directly otherwise.

11. As Administrator, run `mysql_upgrade` to check your tables, attempt to repair them if necessary, and update your grant tables if they have changed so that you can take advantage of any new capabilities. See `mysql_upgrade` — Check and Upgrade MySQL Tables.

12. If you encounter errors, see Section 1.7, “Troubleshooting a Microsoft Windows MySQL Server Installation”.

Chapter 2 Installing MySQL from Source on Windows

These instructions describe how to build binaries from source for MySQL 5.1 on Windows. Instructions are provided for building binaries from a standard source distribution or from the Bazaar tree that contains the latest development source.

> **Note**
>
> The instructions here are strictly for users who want to test MySQL on Microsoft Windows from the latest source distribution or from the Bazaar tree. For production use, we do not advise using a MySQL server built by yourself from source. Normally, it is best to use precompiled binary distributions of MySQL that are built specifically for optimal performance on Windows by Oracle Corporation. Instructions for installing binary distributions are available in Chapter 1, *Installing MySQL on Microsoft Windows*.

To build MySQL on Windows from source, you must satisfy the following system, compiler, and resource requirements:

- Windows 2000, Windows XP, or newer version.

  Windows Vista is supported when using Visual Studio 2005 provided you have installed the following updates:

  - Microsoft Visual Studio 2005 Professional Edition - ENU Service Pack 1 (KB926601)
  - Security Update for Microsoft Visual Studio 2005 Professional Edition - ENU (KB937061)
  - Update for Microsoft Visual Studio 2005 Professional Edition - ENU (KB932232)

- `CMake`, which can be downloaded from http://www.cmake.org. After installing, modify your `PATH` environment variable to include the directory where `cmake` is located.

- Microsoft Visual C++ 2005 Express Edition, Visual Studio .Net 2003 (7.1), or Visual Studio 2005 (8.0) compiler system.

- If you are using Visual C++ 2005 Express Edition, you must also install an appropriate Platform SDK. More information and links to downloads for various Windows platforms is available from http://www.microsoft.com/downloads/details.aspx?familyid=0baf2b35-c656-4969-ace8-e4c0c0716adb.

- If you are compiling from a Bazaar tree or making changes to the parser, you need `bison` for Windows, which can be downloaded from http://gnuwin32.sourceforge.net/packages/bison.htm. Download the package labeled “Complete package, excluding sources”. After installing the package, modify your `PATH` environment variable to include the directory where `bison` is located.

  > **Note**
  >
  > On Windows, the default location for `bison` is the `C:\Program Files\GnuWin32` directory. Some utilities, including `m4`, may fail to find `bison` because of the space in the directory name. You can resolve this by installing into a directory that does not contain a space; for example `C:\GnuWin32`.

- Cygwin might be necessary if you want to run the test script or package the compiled binaries and support files into a Zip archive. (Cygwin is needed only to test or package the distribution, not to build it.) Cygwin is available from http://cygwin.com.

- 3GB to 5GB of disk space.

You also need a MySQL source distribution for Windows, which can be obtained two ways:

- Obtain a source distribution packaged by Oracle Corporation. These are available from http://dev.mysql.com/downloads/.
- Package a source distribution yourself from the latest Bazaar developer source tree. For instructions on pulling the latest source files, see Installing MySQL Using a Development Source Tree.

If you find something not working as expected, or you have suggestions about ways to improve the current build process on Windows, please send a message to the `win32` mailing list. See MySQL Mailing Lists.

> **Note**
>
> To compile from the source code on Windows you must use the standard source distribution (for example, `mysql-5.1.73.zip`) or `mysql-5.1.73.tar.gz`). You build from the same distribution as used to build MySQL on Unix, Linux and other platforms. Do *not* use the Windows Source distributions as they do not contain the necessary configuration script and other files.

Follow this procedure to build MySQL:

1. If you are installing from a packaged source distribution, create a work directory (for example, `C:\workdir`), and unpack the source distribution there using `WinZip` or another Windows tool that can read `.zip` files. This directory is the work directory in the following instructions.

   > **Note**
   >
   > Commands that are located in the `win` directory should be executed from the top-level source directory. Do not change location into the `win` directory, as the commands will not execute correctly.

2. Start a command shell. If you have not configured the `PATH` and other environment variables for all command shells, you may be able to start a command shell from the **Start Menu** within the Windows Visual Studio menu that contains the necessary environment changes.

3. Within the command shell, navigate to the work directory and run the following command:

   ```
   C:\workdir>win\configure.js options
   ```

   If you have associated the `.js` file extension with an application such as a text editor, then you may need to use the following command to force `configure.js` to be executed as a script:

   ```
   C:\workdir>cscript win\configure.js options
   ```

   These options are available for `configure.js`:

   - `WITH_INNOBASE_STORAGE_ENGINE`: Enable the `InnoDB` storage engine.
   - `WITH_PARTITION_STORAGE_ENGINE`: Enable user-defined partitioning.
   - `WITH_ARCHIVE_STORAGE_ENGINE`: Enable the `ARCHIVE` storage engine.
   - `WITH_BLACKHOLE_STORAGE_ENGINE`: Enable the `BLACKHOLE` storage engine.
   - `WITH_EXAMPLE_STORAGE_ENGINE`: Enable the `EXAMPLE` storage engine.
   - `WITH_FEDERATED_STORAGE_ENGINE`: Enable the `FEDERATED` storage engine.

- `WITH_NDBCLUSTER_STORAGE_ENGINE`: Enable the `NDBCLUSTER` storage engine in the MySQL server; cause binaries for the MySQL Cluster management and data node, management client, and other programs to be built.

  This option is supported only in MySQL Cluster NDB 7.0 and later (`NDBCLUSTER` storage engine versions 6.4.0 and later) using the MySQL Cluster sources. It cannot be used to enable clustering support in other MySQL source trees or distributions.

- `MYSQL_SERVER_SUFFIX=`*`suffix`*: Server suffix, default none.

- `COMPILATION_COMMENT=`*`comment`*: Server comment, default "Source distribution".

- `MYSQL_TCP_PORT=`*`port`*: Server port, default 3306.

- `DISABLE_GRANT_OPTIONS`: Disables the `--bootstrap`, `--skip-grant-tables`, and `--init-file` options for `mysqld`. This option is available as of MySQL 5.1.15.

For example (type the command on one line):

```
C:\workdir>win\configure.js WITH_INNOBASE_STORAGE_ENGINE
            WITH_PARTITION_STORAGE_ENGINE MYSQL_SERVER_SUFFIX=-pro
```

4. From the work directory, execute the `win\build-vs9.bat` (Windows Visual Studio 2008), `win\build-vs8.bat` (Windows Visual Studio 2005), or `win\build-vs71.bat` (Windows Visual Studio 2003) script, depending on the version of Visual Studio you have installed. The script invokes `CMake`, which generates the `mysql.sln` solution file.

   You can also use the corresponding 64-bit file (for example `win\build-vs8_x64.bat` or `win\build-vs9_x64.bat`) to build the 64-bit version of MySQL. However, you cannot build the 64-bit version with Visual Studio Express Edition. You must use Visual Studio 2005 (8.0) or higher.

5. From the work directory, open the generated `mysql.sln` file with Visual Studio and select the proper configuration using the **Configuration** menu. The menu provides **Debug**, **Release**, **RelwithDebInfo**, **MinRelInfo** options. Then select **Solution** > **Build** to build the solution.

   Remember the configuration that you use in this step. It is important later when you run the test script because that script needs to know which configuration you used.

6. Test the server. The server built using the preceding instructions expects that the MySQL base directory and data directory are `C:\mysql` and `C:\mysql\data` by default. If you want to test your server using the source tree root directory and its data directory as the base directory and data directory, you need to tell the server their path names. You can either do this on the command line with the `--basedir` and `--datadir` options, or by placing appropriate options in an option file. (See Using Option Files.) If you have an existing data directory elsewhere that you want to use, you can specify its path name instead.

   When the server is running in standalone fashion or as a service based on your configuration, try to connect to it from the `mysql` interactive command-line utility.

   You can also run the standard test script, `mysql-test-run.pl`. This script is written in Perl, so you'll need either Cygwin or ActiveState Perl to run it. You may also need to install the modules required by the script. To run the test script, change location into the `mysql-test` directory under the work directory, set the `MTR_VS_CONFIG` environment variable to the configuration you selected earlier (or use the `--vs-config` option), and invoke `mysql-test-run.pl`. For example (using Cygwin and the `bash` shell):

```
shell> cd mysql-test
shell> export MTR_VS_CONFIG=debug
shell> ./mysql-test-run.pl --force --timer
shell> ./mysql-test-run.pl --force --timer --ps-protocol
```

When you are satisfied that the programs you have built are working correctly, stop the server. Now you can install the distribution. One way to do this is to use the `make_win_bin_dist` script in the `scripts` directory of the MySQL source distribution (see `make_win_bin_dist` — Package MySQL Distribution as Zip Archive). This is a shell script, so you must have Cygwin installed if you want to use it. It creates a Zip archive of the built executables and support files that you can unpack in the location at which you want to install MySQL.

It is also possible to install MySQL by copying directories and files directly:

1. Create the directories where you want to install MySQL. For example, to install into `C:\mysql`, use these commands:

   ```
   C:\> mkdir C:\mysql
   C:\> mkdir C:\mysql\bin
   C:\> mkdir C:\mysql\data
   C:\> mkdir C:\mysql\share
   C:\> mkdir C:\mysql\scripts
   ```

   If you want to compile other clients and link them to MySQL, you should also create several additional directories:

   ```
   C:\> mkdir C:\mysql\include
   C:\> mkdir C:\mysql\lib
   C:\> mkdir C:\mysql\lib\debug
   C:\> mkdir C:\mysql\lib\opt
   ```

   If you want to benchmark MySQL, create this directory:

   ```
   C:\> mkdir C:\mysql\sql-bench
   ```

   Benchmarking requires Perl support for MySQL. See Perl Installation Notes.

2. From the work directory, copy into the `C:\mysql` directory the following files and directories:

   ```
   C:\> cd \workdir
   C:\workdir> mkdir C:\mysql
   C:\workdir> mkdir C:\mysql\bin
   C:\workdir> copy client\Release\*.exe C:\mysql\bin
   C:\workdir> copy sql\Release\mysqld.exe C:\mysql\bin\mysqld.exe
   C:\workdir> xcopy scripts\*.* C:\mysql\scripts /E
   C:\workdir> xcopy share\*.* C:\mysql\share /E
   ```

   If you want to compile other clients and link them to MySQL, you should also copy several libraries and header files:

   ```
   C:\workdir> copy lib\Release\mysqlclient.lib C:\mysql\lib\debug
   C:\workdir> copy lib\Release\libmysql.* C:\mysql\lib\debug
   C:\workdir> copy lib\Release\zlib.* C:\mysql\lib\debug
   C:\workdir> copy lib\Release\mysqlclient.lib C:\mysql\lib\opt
   C:\workdir> copy lib\Release\libmysql.* C:\mysql\lib\opt
   C:\workdir> copy lib\Release\zlib.* C:\mysql\lib\opt
   ```

```
C:\workdir> copy include\*.h C:\mysql\include
C:\workdir> copy libmysql\libmysql.def C:\mysql\include
```

> **Note**
>
> If you have compiled a Debug solution, rather than a Release solution, install it by replacing `Release` with `Debug` in the source file names just shown.

If you want to benchmark MySQL, you should also do this:

```
C:\workdir> xcopy sql-bench\*.* C:\mysql\bench /E
```

After installation, set up and start the server in the same way as for binary Windows distributions. This includes creating the system tables by running `mysql_install_db`. For more information, see Chapter 1, *Installing MySQL on Microsoft Windows*.

Chapter 3 Connection to MySQL Server Failing on Windows

When you're running a MySQL server on Windows with many TCP/IP connections to it, and you're experiencing that quite often your clients get a `Can't connect to MySQL server` error, the reason might be that Windows does not allow for enough ephemeral (short-lived) ports to serve those connections.

The purpose of `TIME_WAIT` is to keep a connection accepting packets even after the connection has been closed. This is because Internet routing can cause a packet to take a slow route to its destination and it may arrive after both sides have agreed to close. If the port is in use for a new connection, that packet from the old connection could break the protocol or compromise personal information from the original connection. The `TIME_WAIT` delay prevents this by ensuring that the port cannot be reused until after some time has been permitted for those delayed packets to arrive.

It is safe to reduce `TIME_WAIT` greatly on LAN connections because there is little chance of packets arriving at very long delays, as they could through the Internet with its comparatively large distances and latencies.

Windows permits ephemeral (short-lived) TCP ports to the user. After any port is closed it will remain in a `TIME_WAIT` status for 120 seconds. The port will not be available again until this time expires. The default range of port numbers depends on the version of Windows, with a more limited number of ports in older versions:

- Windows through Server 2003: Ports in range 1025–5000
- Windows Vista, Server 2008, and newer: Ports in range 49152–65535

With a small stack of available TCP ports (5000) and a high number of TCP ports being open and closed over a short period of time along with the `TIME_WAIT` status you have a good chance for running out of ports. There are two ways to address this problem:

- Reduce the number of TCP ports consumed quickly by investigating connection pooling or persistent connections where possible
- Tune some settings in the Windows registry (see below)

> **Important**
>
> The following procedure involves modifying the Windows registry. Before you modify the registry, make sure to back it up and make sure that you understand how to restore it if a problem occurs. For information about how to back up, restore, and edit the registry, view the following article in the Microsoft Knowledge Base: http://support.microsoft.com/kb/256986/EN-US/.

1. Start Registry Editor (`Regedt32.exe`).
2. Locate the following key in the registry:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
   ```

3. On the `Edit` menu, click `Add Value`, and then add the following registry value:

   ```
   Value Name: MaxUserPort
   Data Type: REG_DWORD
   Value: 65534
   ```

   This sets the number of ephemeral ports available to any user. The valid range is between 5000 and 65534 (decimal). The default value is 0x1388 (5000 decimal).

4. On the Edit menu, click Add Value, and then add the following registry value:

```
Value Name: TcpTimedWaitDelay
Data Type: REG_DWORD
Value: 30
```

   This sets the number of seconds to hold a TCP port connection in TIME_WAIT state before closing. The valid range is between 30 and 300 decimal, although you may wish to check with Microsoft for the latest permitted values. The default value is 0x78 (120 decimal).

5. Quit Registry Editor.

6. Reboot the machine.

Note: Undoing the above should be as simple as deleting the registry entries you've created.

Chapter 4 Resetting the Root Password: Windows Systems

On Windows, use the following procedure to reset the password for the MySQL `'root'@'localhost'` account. To change the password for a `root` account with a different host name part, modify the instructions to use that host name.

1. Log on to your system as Administrator.
2. Stop the MySQL server if it is running. For a server that is running as a Windows service, go to the Services manager: From the **Start** menu, select **Control Panel**, then **Administrative Tools**, then **Services**. Find the MySQL service in the list and stop it.

   If your server is not running as a service, you may need to use the Task Manager to force it to stop.
3. Create a text file containing the following statement on a single line. Replace the password with the password that you want to use.

   ```
   SET PASSWORD FOR 'root'@'localhost' = PASSWORD('MyNewPass');
   ```
4. Save the file. This example assumes that you name the file `C:\mysql-init.txt`.
5. Open a console window to get to the command prompt: From the **Start** menu, select **Run**, then enter `cmd` as the command to be run.
6. Start the MySQL server with the special `--init-file` option (notice that the backslash in the option value is doubled):

   ```
   C:\> cd "C:\Program Files\MySQL\MySQL Server 5.1\bin"
   C:\> mysqld --init-file=C:\\mysql-init.txt
   ```

   If you installed MySQL to a different location, adjust the `cd` command accordingly.

   The server executes the contents of the file named by the `--init-file` option at startup, changing the `'root'@'localhost'` account password.

   To have server output to appear in the console window rather than in a log file, add the `--console` option to the `mysqld` command.

   If you installed MySQL using the MySQL Installation Wizard, you may need to specify a `--defaults-file` option. For example:

   ```
   C:\> mysqld
           --defaults-file="C:\\Program Files\\MySQL\\MySQL Server 5.1\\my.ini"
           --init-file=C:\\mysql-init.txt
   ```

   The appropriate `--defaults-file` setting can be found using the Services Manager: From the **Start** menu, select **Control Panel**, then **Administrative Tools**, then **Services**. Find the MySQL service in the list, right-click it, and choose the `Properties` option. The `Path to executable` field contains the `--defaults-file` setting.
7. After the server has started successfully, delete `C:\mysql-init.txt`.

You should now be able to connect to the MySQL server as `root` using the new password. Stop the MySQL server and restart it normally. If you run the server as a service, start it from the Windows Services window. If you start the server manually, use whatever command you normally use.

Chapter 5 MySQL for Excel

Chapter 6 Installation

MySQL for Excel is a product for Windows, and it is installed with MySQL Installer. And typically you will not be required to install or configure additional tools to use MySQL for Excel.

MySQL for Excel Requirements

The MySQL Installer installation process will check if these requirements are met, or notify you if further action is required before proceeding with the installation.

- *.NET Framework 4.0* (Client or Full Profile).
- *Microsoft Office Excel* 2007 or greater, for Microsoft Windows.
- *Visual Studio 2010 Tools for Office Runtime*, and MySQL Installer may install this for you.

  > **Note**
  >
  > This requirement is different than *Office Developer Tools for Visual Studio*, which is not a substitute.

- An available MySQL Server connection.

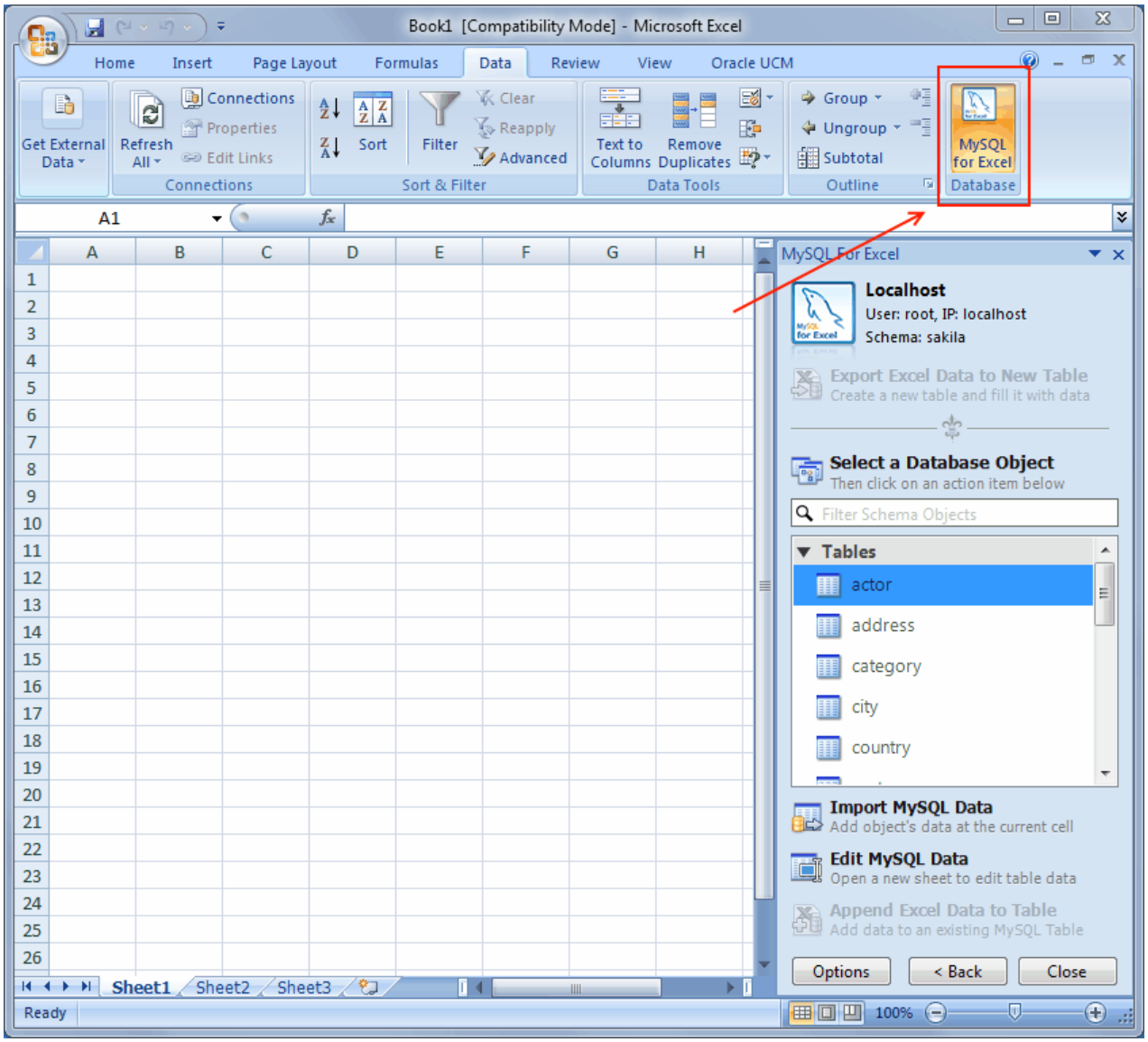
MySQL for Excel Download

Either install MySQL for Excel using the MySQL Installer for Windows (a system that manages installations and updates for all MySQL products on Windows), or download and execute the standalone file. The download links are as follows:

- **MySQL Installer**: Download and execute the MySQL Installer MSI file. Select the MySQL for Excel product and then proceed with the installation. See the MySQL Installer manual for additional details. This is the recommended approach.
- **Standalone**: Download and execute the MySQL for Excel standalone MSI file.

MySQL for Excel is loaded and executed by selecting the **Data** menu tab in Excel, and then choosing the "MySQL for Excel" Database icon. This opens a new Excel sidebar with the available MySQL for Excel options. The navigation bar with the MySQL for Excel icon is shown in the following screenshot:

Figure 6.1 The MySQL for Excel navigation bar



Chapter 7 Configuration

Table of Contents

This section is divided into two overlapping topics; configuring the global options, and managing the MySQL connections.

7.1 Global Options and Preferences

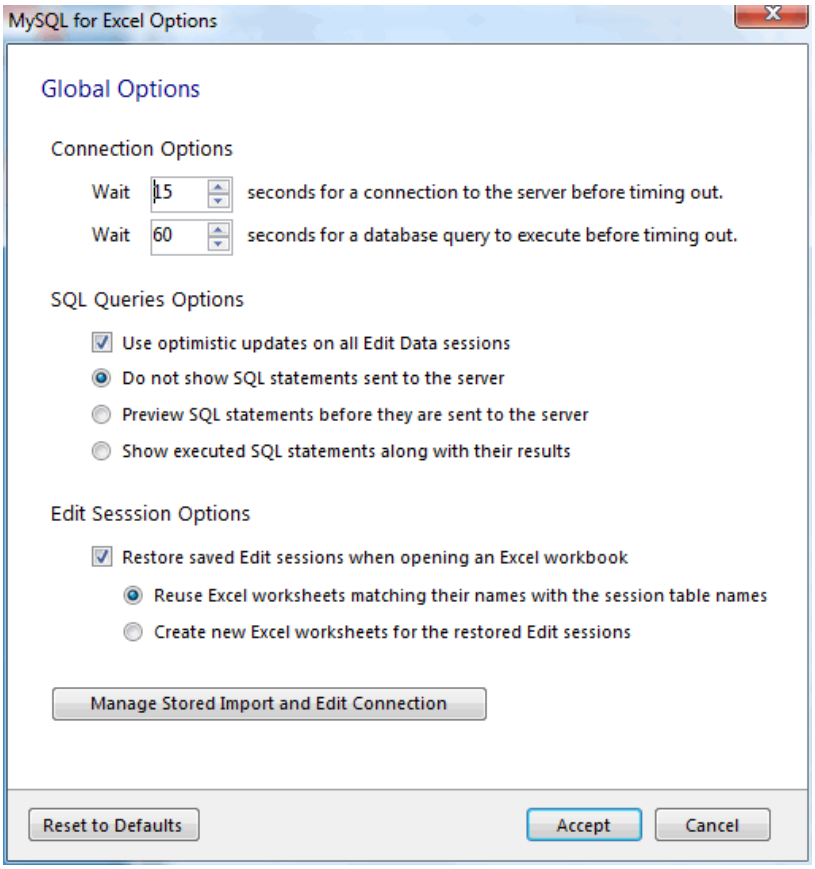
Each action, such as **Import MySQL Data**, has its own set of options. The buttons on these pages include:

Actions include:

- Clicking **Accept**: Saves option changes to your host, and preserves these changes across all sessions and future Excel instances.
- Clicking **Reset to Defaults**: Resets all option values on the current options window to their default settings. Click **Accept** to save the changes.

A set of "global" options affect the entire plugin, as described here:

Figure 7.1 The MySQL for Excel configuration: Global Options



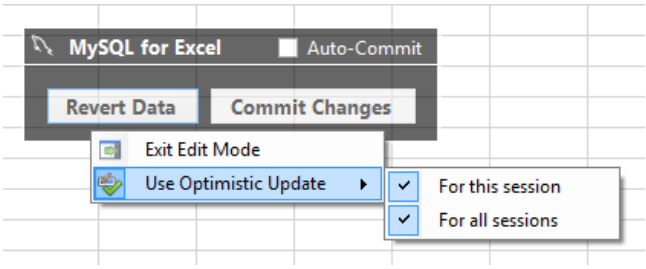
- Connection Options:

- *Wait [ ] seconds for a connection to the server before timing out*. Defaults to 15.
- *Wait [ ] seconds for a database query to execute before timing out*. Defaults to 60.

- SQL Queries Options:
  - *[ ] Use optimistic updates on all Edit Data sessions*. This option helps prevent unintentional data overwrite, in that it checks for external edits before committing your changes. For example, between the time you loaded the data into Excel, made changes in Excel, and committed, a different user could have edited the same cells elsewhere in MySQL using MySQL Workbench or some other means. The optimistic updates feature checks for these changes, and notifies you accordingly.

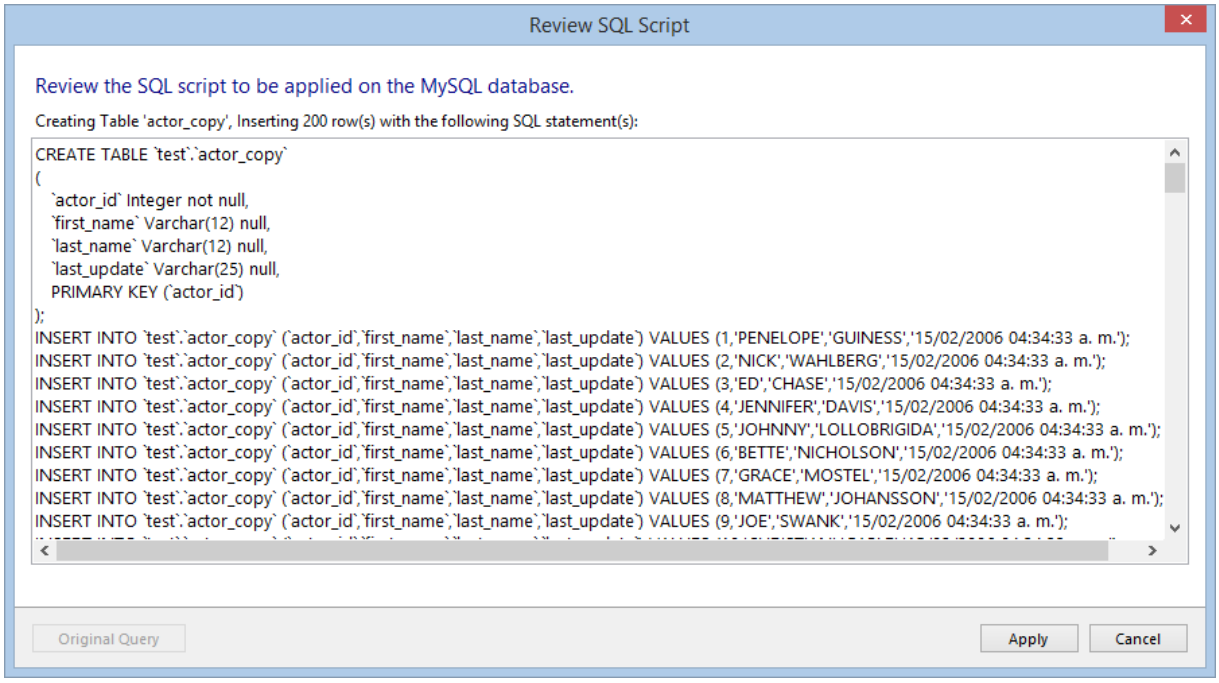
    Optimistic updates can also be configured at runtime for all Edit Sessions, or for a specific edit session by right-clicking the **Edit Session** floating dialog and choosing the desired **Use Optimistic Update** option, as demonstrated below:

    Figure 7.2 Optimistic Updates: Configuring at Runtime

    This option is enabled by default.
  - *( ) Do not show SQL statements sent to the server*: When enabled, SQL statements are now displayed, and only their results are displayed in the information dialog. Enabled by default.
  - *( ) Preview SQL statements before they are sent to the server*: When enabled, it adds an extra step to the Create Schema, Export Data, Append Data and Edit Data operations before a statement is committed to the server. It enables the "Review SQL Script" dialog, as shown below for an "Export Data" operation:

Figure 7.3 The MySQL for Excel configuration: Preview Option



From here you can modify the SQL statements before they are executed, which also enables the **Original Query** button. If clicked, it will revert all modifications to the script to restore the SQL to its original form (when the dialog was first opened).

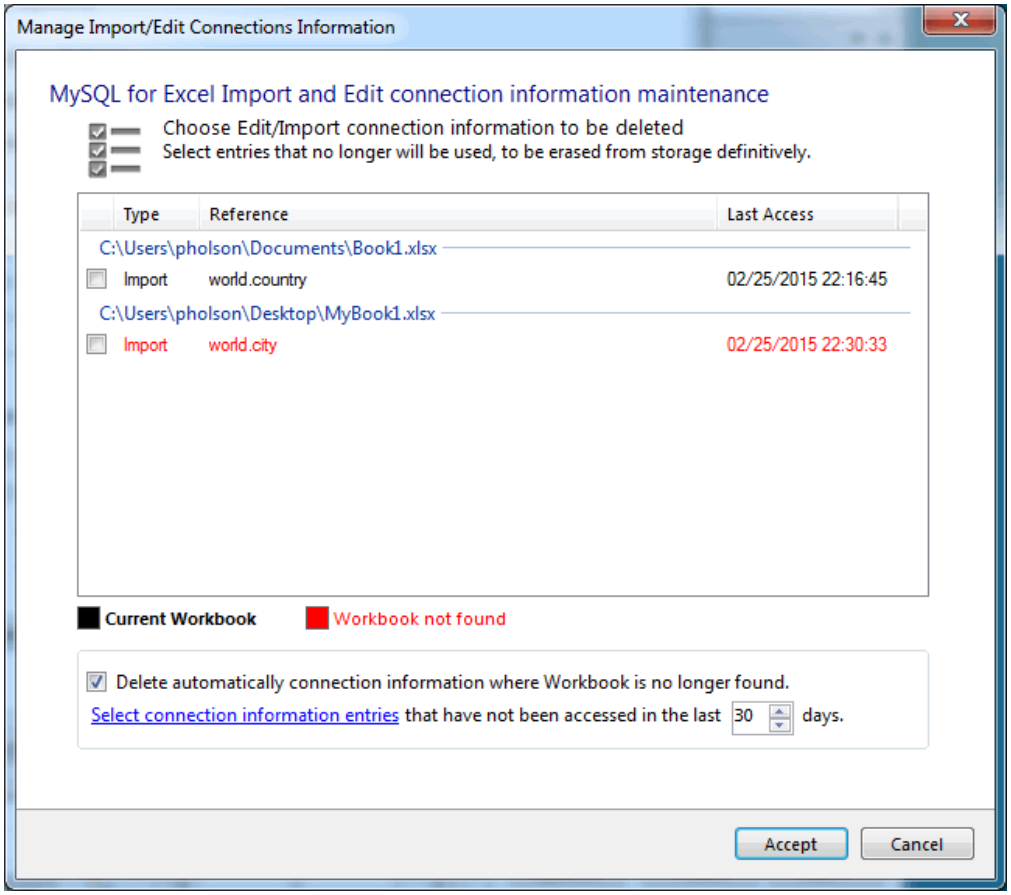
This option is disabled by default.

- *( ) Show executed SQL statements along with their results*: When enabled, SQL statements are first executed and then the information dialog includes both the results and the executed statements. This is helpful when reviewing the recently executed queries when comparing the results.

  This option is disabled by default.

- Edit Session Options:
  - [ ] Restore saved Edit sessions when opening an Excel workbook. Enabled by default.
  - ( ) Reuse Excel worksheets matching their names with the session table names. Enabled by default.
  - ( ) Create new Excel worksheets for the restored Edit sessions. Disabled by default.
- **Manage Stored Import and Edit Connections**: See a list of saved Excel files with linked MySQL connections.

Figure 7.4 MySQL for Excel: Manage Stored MySQL Connections



This lists the connected Excel worksheets that are known to MySQL for Excel. From here you can view these connections, and optionally delete them. By default, clicking **Apply** will delete connections to missing worksheets but this behavior is configurable. Additionally, clicking the *Select connection information entries* link checks (selects for deletion) books that you have not accessed for n days, where n defaults to 30.

> **Note**
>
> This option was added in MySQL for Excel 1.3.0

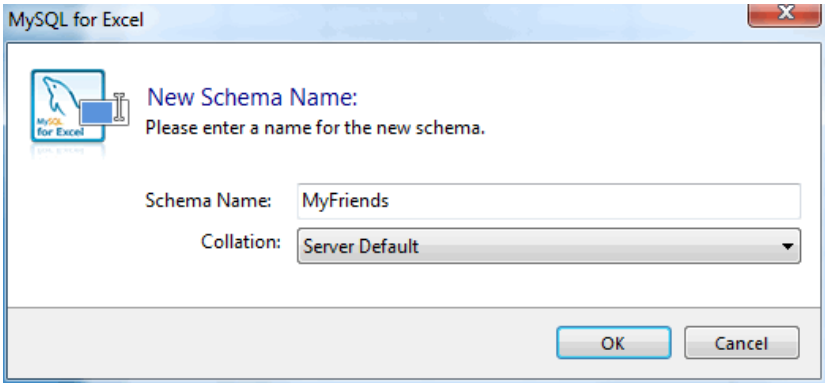
> **Note**
>
> The options to automatically delete missing connections, or delete connections not accessed for n days, were added in MySQL for Excel 1.3.4.

For additional information about managing MySQL connections using MySQL for Excel, see Section 7.2, “Managing MySQL Connections”.

Adding a New Schema

Select a MySQL connection and then click **Create New Schema** from the MySQL for Excel toolbar to add a new and empty MySQL schema.

Figure 7.5 MySQL for Excel: Adding a New Schema



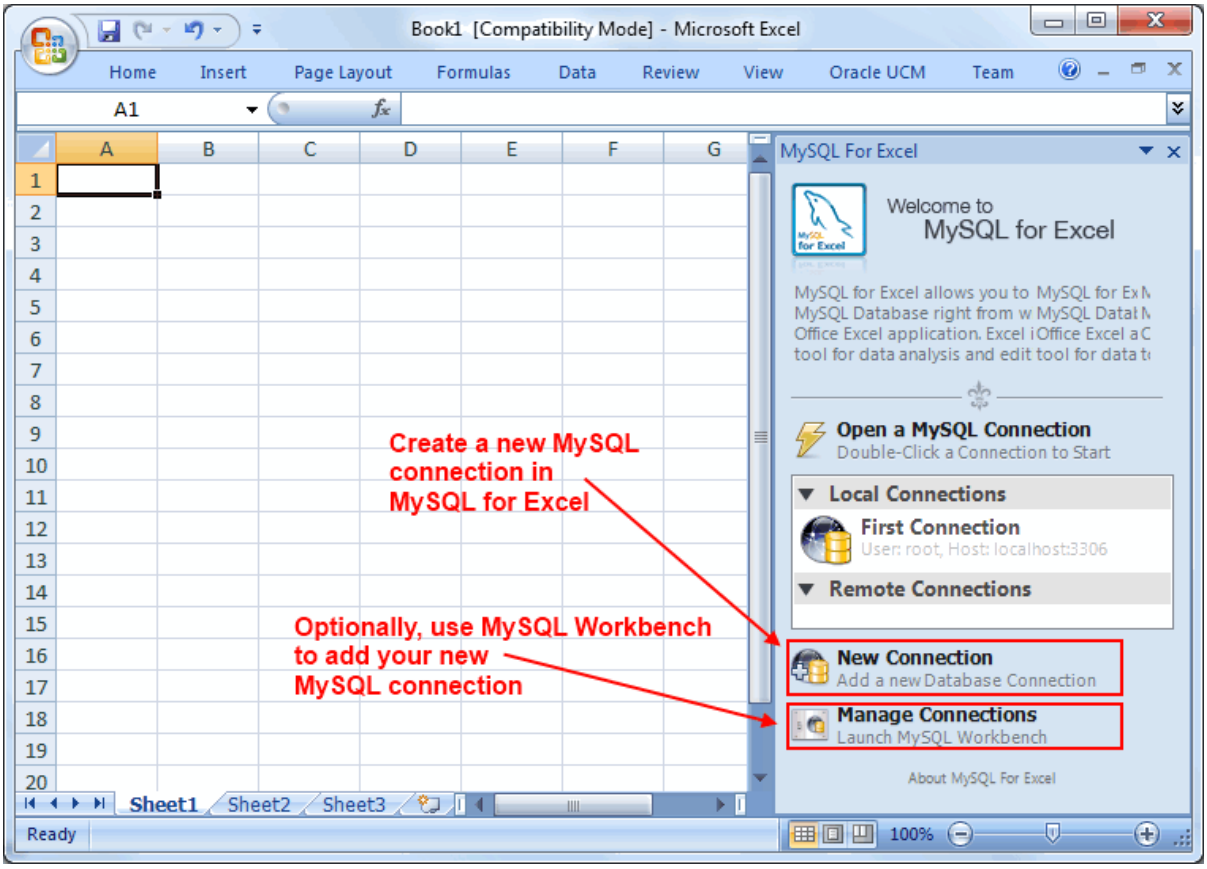
7.2 Managing MySQL Connections

MySQL for Excel shares its MySQL connections with MySQL Workbench, although it is optional to have MySQL Workbench installed. Creating and editing MySQL connections in either application will edit the MySQL connection information for both applications.

Adding MySQL Connections

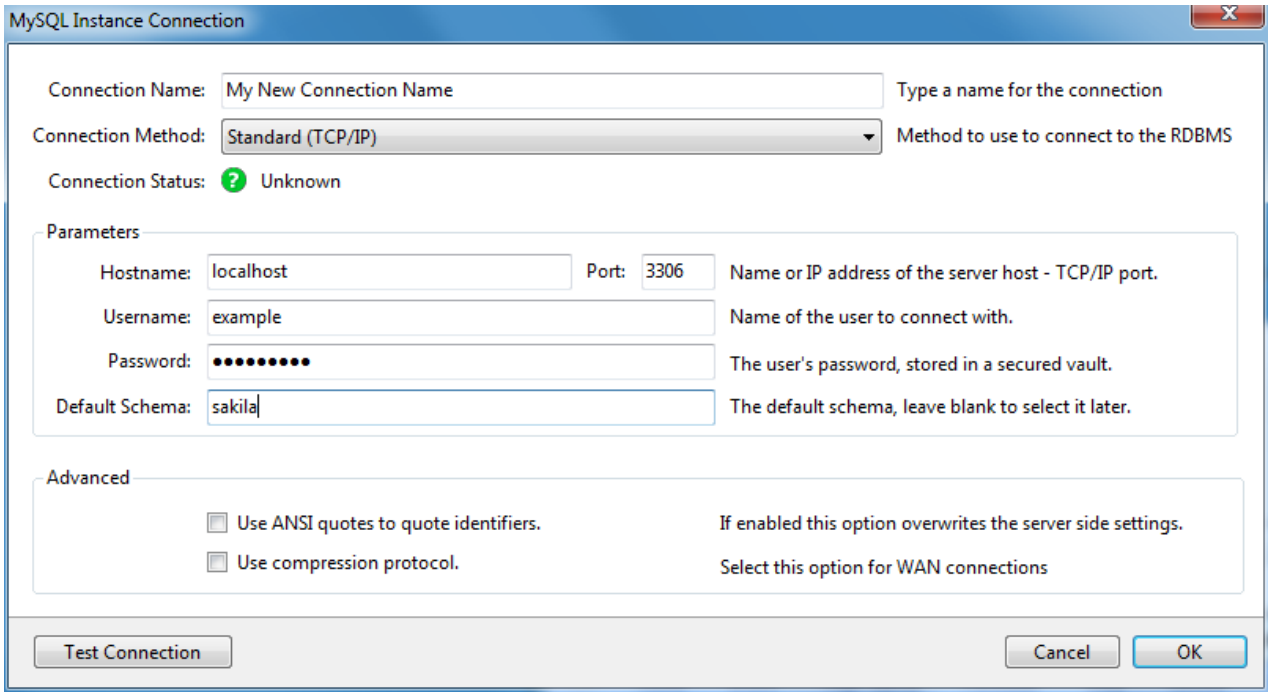
You can use MySQL for Excel or MySQL Workbench to add new MySQL connections.

Figure 7.6 MySQL for Excel: Add a New MySQL Connection



From Excel, click **New Connection** to open the new connection dialog as demonstrated in the following partially filled screenshot:

Figure 7.7 MySQL for Excel: Add a New MySQL Connection Dialog



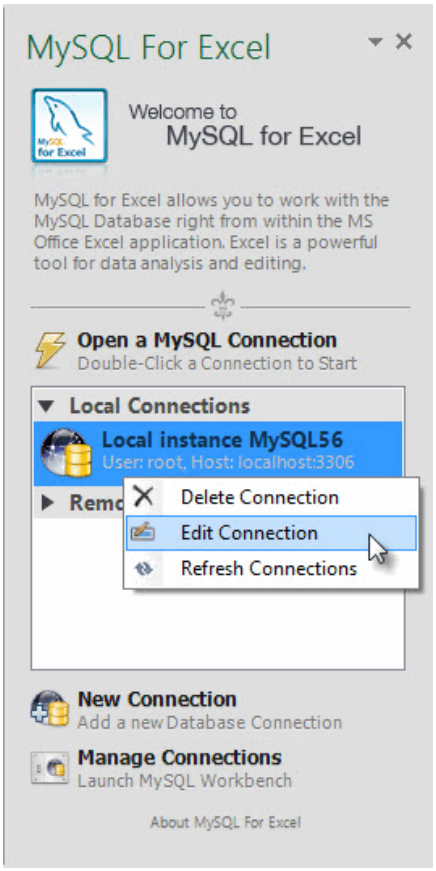
Fill out the connection details, click **Test Connection** to confirm the MySQL connection is valid, and click **OK** to save the new connection.

Editing MySQL Connections

Editing MySQL Connections in MySQL for Excel

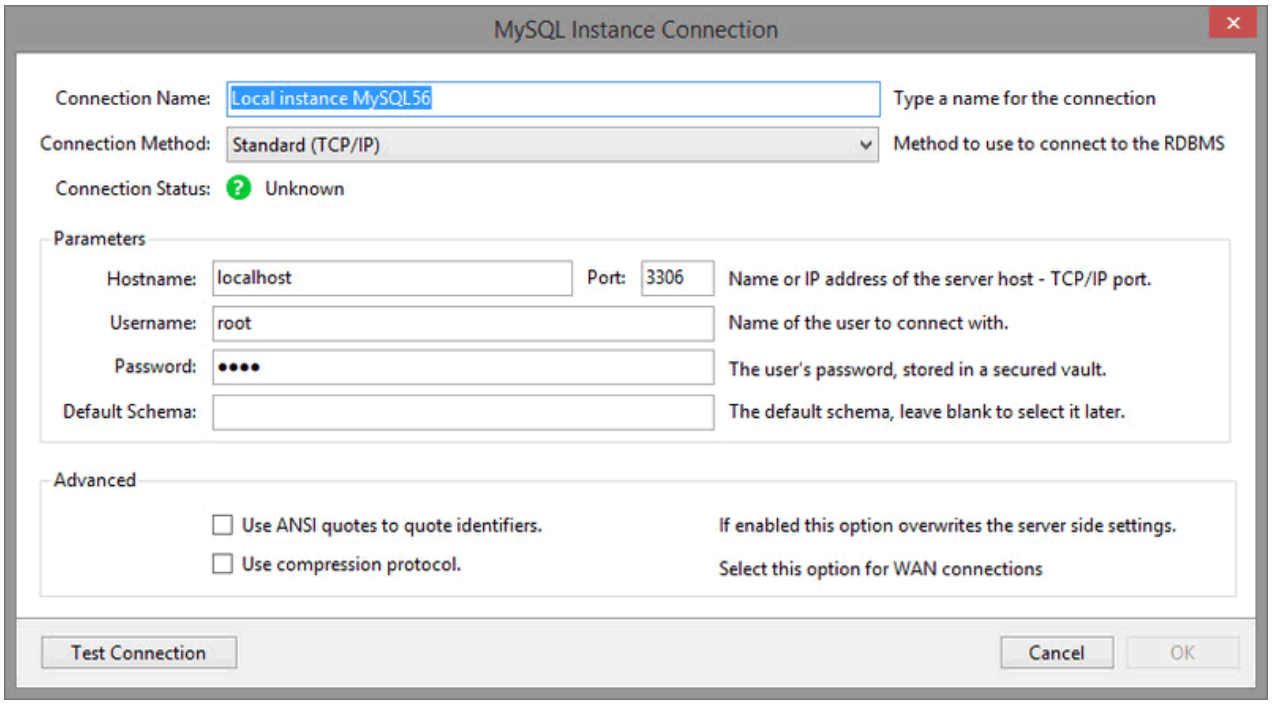
To edit a MySQL connection, right-click the connection you want to modify and select **Edit Connection** from the context menu, like so:

Figure 7.8 MySQL for Excel: Choosing a MySQL Connection to Edit



The MySQL connection edit dialog is similar to the edit dialog in MySQL Workbench. Configure the changes and click **OK** to save your changes:

Figure 7.9 MySQL for Excel: Editing a MySQL Connection



Editing MySQL Connections in MySQL Workbench

Optionally, you can edit your MySQL for Excel MySQL connections using MySQL Workbench. To do this, open MySQL Workbench, edit a MySQL connection, and then refresh the connection list in MySQL for Excel.

For information about editing MySQL connections in MySQL Workbench, see the MySQL Workbench documentation titled MySQL Connections.

Delete MySQL Connections

MySQL connections can be deleted from MySQL for Excel or MySQL Workbench.

> **Note**
>
> MySQL connections cannot be deleted if MySQL Workbench is open. To remove connections, you must first close MySQL Workbench.

To delete an edit or import connection from MySQL for Excel to a particular Excel worksheet, click **Options**, **Manage Stored Import and Edit Connection**, check the desired worksheets, and then click **Apply** to execute the delete action.

Chapter 8 What Is New In MySQL for Excel

Table of Contents

This section summarizes how MySQL for Excel progressed with each minor and major release.

8.1 What Is New In MySQL for Excel 1.3

Most of the new features added to MySQL for Excel 1.3.x involve improvements to the **Data Import** functionality.

- You can now refresh imported data from the source MySQL database by clicking **Refresh** from the context-menu, or **Refresh All** from the navigation menu. These check for changes in the source MySQL database and update your imported MySQL data accordingly.

  Use case: A colleague sends you a MySQL Excel spreadsheet with data exported from a MySQL database. You open the file several days later, and worry that the data is outdated so you click **Refresh**.

- A new **Refresh To Defaults** button was added to the options pages. It changes each option to the default value, and you then confirm (or cancel) the application of these changes.

- Enabling the new **Add Summary Fields for Numeric Columns** option adds a summary field to the end of each numeric column in Excel. From here, you choose the desired function for the column, such as total or average.

- You may now import data from multiple objects in a single operation. Use **Control** or **Shift** to select multiple objects (tables and/or views) from the MySQL for Excel panel, and click **Import** to open the new dialog for selecting additional objects that have direct relationships to the objects you selected. Each object opens in its own Worksheet.

  From this new dialog, you may also generate a **Relationships** model in Excel. This functionality requires Excel 2013 or higher, or Excel 2010 with the PowerPivot add-in.

- A new **Create a PivotTable with the Imported Data** option was added. This creates a Pivot Table in Excel.

- All options now have descriptive tooltips. Hover over an option/preference to view helpful information about its use.

- You may now specify a collation for created schemas. The collation type defaults to "Server Default." These statements can be reviewed before execution.

- All MySQL data types are now available when performing **Data Export** operations. By default, only the most commonly used data types are listed, which was only behavior in previous versions of MySQL for Excel. You may still type in a type instead of selecting it.

8.2 What Is New In MySQL for Excel 1.2

- `Edit Connections`: MySQL connections can now be edited from within the MySQL for Excel plugin by right-clicking and choosing **Edit Connection**. Before, these connections could only be edited with MySQL Workbench.

- `Optimistic Updates`: Previously, only "Pessimistic Updates" were used, which means that pressing **Commit Changes** would overwrite changes performed outside of MySQL for Excel for the edited cells.

  Both options remain available today, and optimistic updates are enabled by default. This update type can be set either as a preference, or toggled per session.

- The **Append Data** dialog will now notify you of incompatible types (with visual warnings) when mapping source Excel columns to target MySQL columns.

  If a mismatch is discovered, then the column in the source grid that contains the mapped Excel data turns red, and selecting this column displays a warning with text explaining that the source data is not suitable for the mapped target column's data type.

- New preview preferences allow you to enable one of the following three options:

  - **Preview SQL statements before they are sent to the Server**: View (and optionally) edit the MySQL UPDATE/INSERT statements before they are committed.

  - **Show executed SQL statements along with the results**: View the statements after they are committed, which is the current behavior.

  - **Do not show the MySQL statements**: Only show summary information, such as number of affected rows, and not MySQL statements. This is enabled by default.

- Create Table: The **Data Export** feature now has the option to only create the table without inserting the data.

  To execute, toggle the **Export Data** button to **Create Table**, and then click.

- The selected schema name is now displayed on top of the MySQL for Excel Database Object Selection panel.

- The **Advanced Options** dialogs opened from the Import, Export and Append Data windows now immediately apply the option changes, when before the **Advanced Options** dialog had to be reopened before the changes could be previewed.

- **Edit Data** sessions can now be saved: Using the new **Edit Session** preferences, these sessions were automatically closed after closing an Excel workbook. This data, such as the Workbench connection ID, MySQL schema, and MySQL table name, can now be preserved if the Excel workbook is saved to disk, and available when the Excel workbook is reopened.

- Excel tables are automatically created for any data imported from MySQL to an Excel worksheet, with a name like "Schema.DB-Object-name". The DB object name can be a MySQL table, view, or stored procedure. Options for this feature are listed under **Import Data**, **Advanced Options**. The newly created Excel tables can be referenced for data analysis in Pivot Tables or reports.

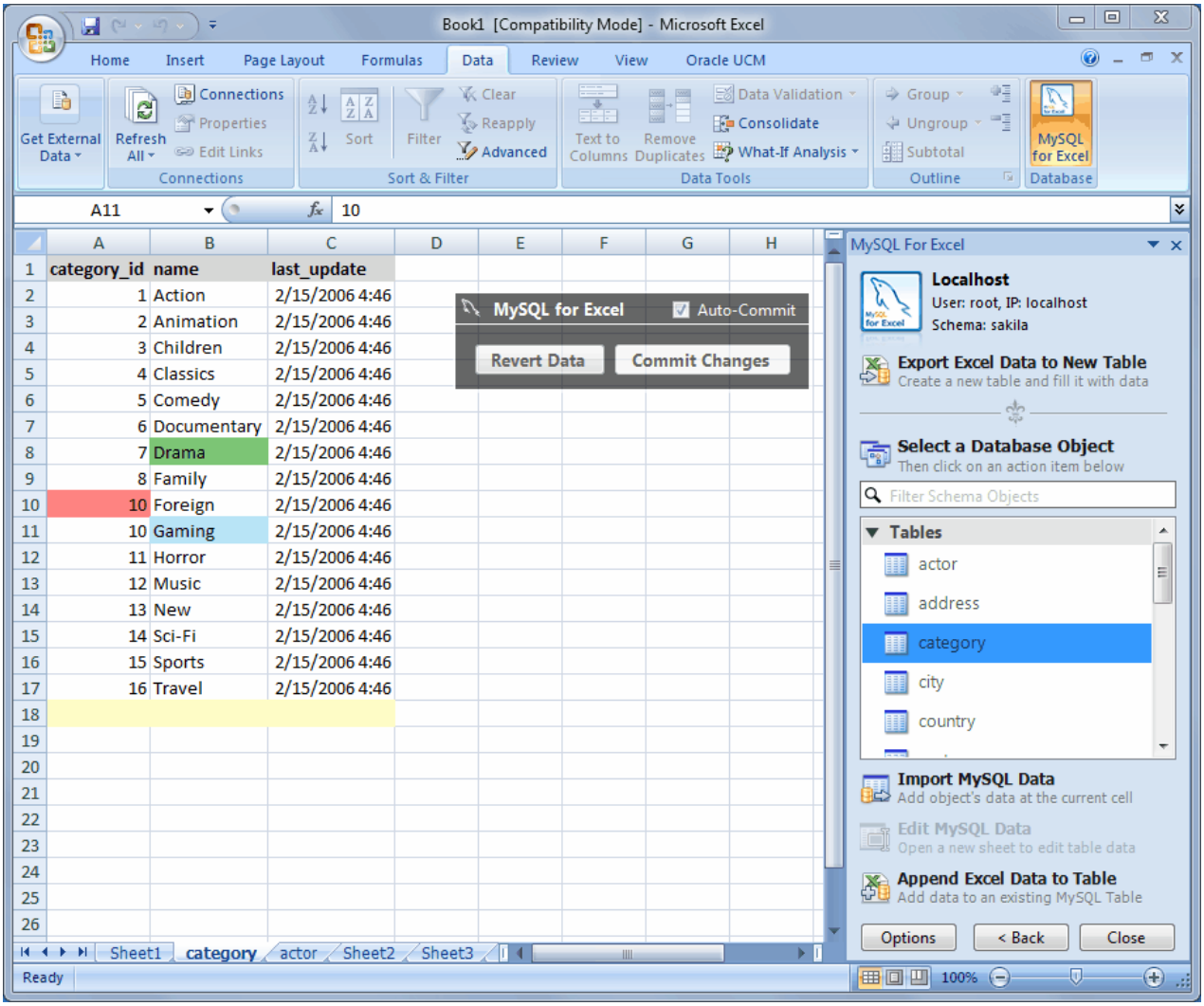
Chapter 9 Edit MySQL Data in Excel

MySQL for Excel enables you to load and edit MySQL data directly from Microsoft Excel. Changes are immediately committed if the **Auto-Commit** option is enabled, or done manually by pressing **Commit Changes**.

The example below uses the `category` table of the example `sakila` database, but the screen will look the same for any table. Within MySQL for Excel, **Open a MySQL Connection**, click the `sakila` schema, **Next**, select the `category` table, click **Edit MySQL Data**, then choose **Import** to import the data into a new Microsoft Excel worksheet for editing.

> **Note**
>
> For additional information about the importing procedure, see Chapter 10, *Import MySQL Data into Excel*.

Figure 9.1 Editing table data with MySQL for Excel



The background color represents the status of each cell, and there are four distinct colors that are used while editing table data:

> **Note**
>
> The Green and Blue colors were switched in MySQL for Excel 1.2.0.

Table 9.1 Background cell colors

Color	Description
White	Default color for all cells. This is either the original data, or the data after **Refresh from DB** is clicked.
Green	Cells that were committed with success.
Blue	Cells that were modified but have not yet been committed.
Red	Cells that generated an error when a commit was attempted. An error dialog is also displayed while the commit is attempted.
Orange	Cells that had a commit attempted, but the commit failed due to detected changes from external sources. For example, a different user made a change to a field after it was imported into Excel. This is a feature of Optimistic Updates.
Yellow	Cells that accept new data. Data entered here is inserted into the MySQL table.

In our example, the green "Drama" field was changed and then committed first, then the blue "Gaming" field was changed but not committed, and then **Auto-Commit** was enabled before changing the "9" to a "10" in column 10, which generated an error because this commit would have added a duplicate value as primary key.

Chapter 10 Import MySQL Data into Excel

Table of Contents

Data can be imported from MySQL into a Microsoft Excel spreadsheet by using the **Import MySQL Data** option after selecting either a table, view, or procedure to import.

10.1 Choosing Columns To Export

By default, all columns are selected and will be imported. Specific columns may be selected (or unselected) using the standard Microsoft Windows method of either **Control** + `Mouse click` to toggle the selection of individual columns, or **Shift** + `Mouse click` to select a range of columns.

The background color of a column shows the status of each column. The color white means that the column has been selected, and therefore it will be imported. Conversely, a gray background means that the column will not be imported.

Right-clicking anywhere in the preview grid opens a context-menu with either a `Select None` or `Select All` option, depending on the current status.

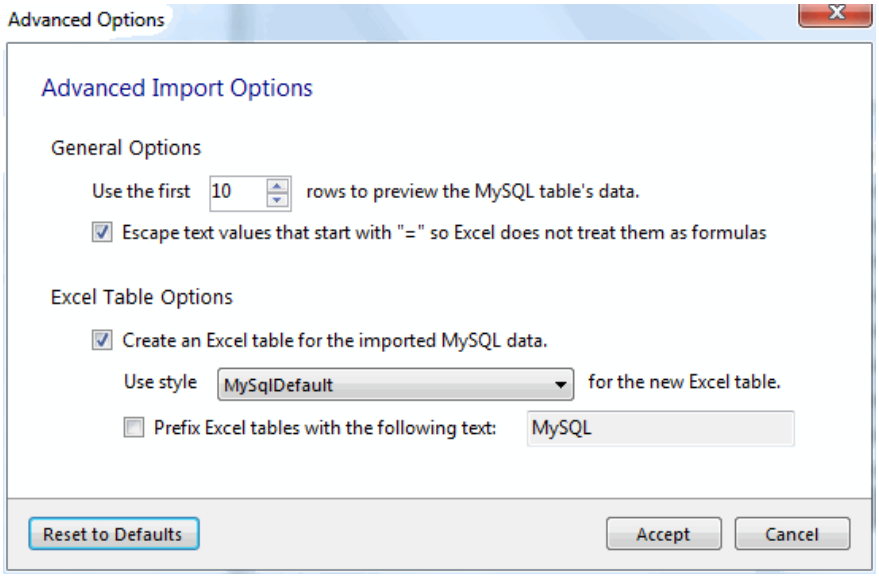
10.2 Importing a Table

The dialog while importing a table includes the following options:

- [ ] **Include Column Names as Headers**: Enabled by default, this inserts the column names at the top of the Microsoft Excel spreadsheet as a "headers" row.
- [ ] **Limit to ___ Rows and Start with Row ___**: Disabled by default, this limits the range of imported data. The `Limit to` option defaults to `1`, and defines the number of rows to import. The `Start with Row` option defaults to `1` (the first row), and defines where the import begins. Each option has a maximum value of COUNT(rows) in the table.
- [ ] **Add Summary Fields**: Disabled by default, this option adds a summary field to each column. For additional information, see Section 10.5, “Adding Summary Fields”.

10.3 Import: Advanced Options

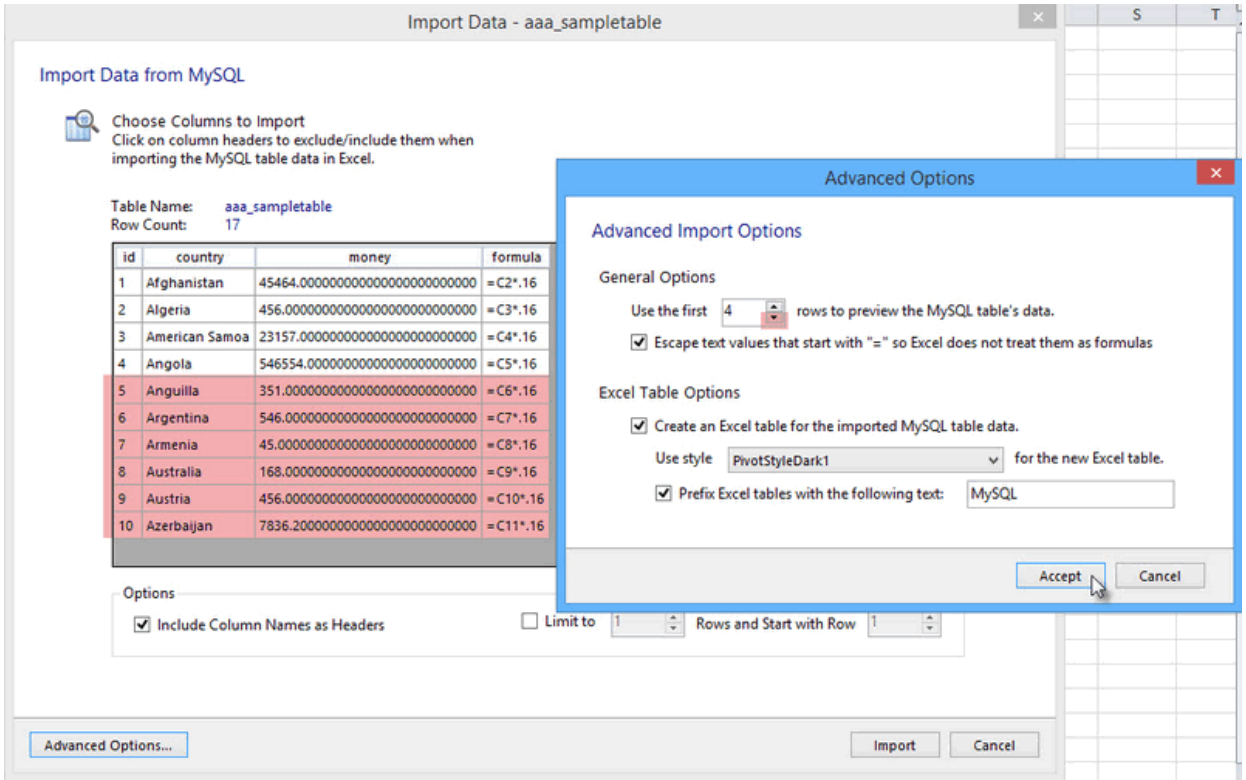
Figure 10.1 Importing table data with MySQL for Excel: Advanced options



General Options:

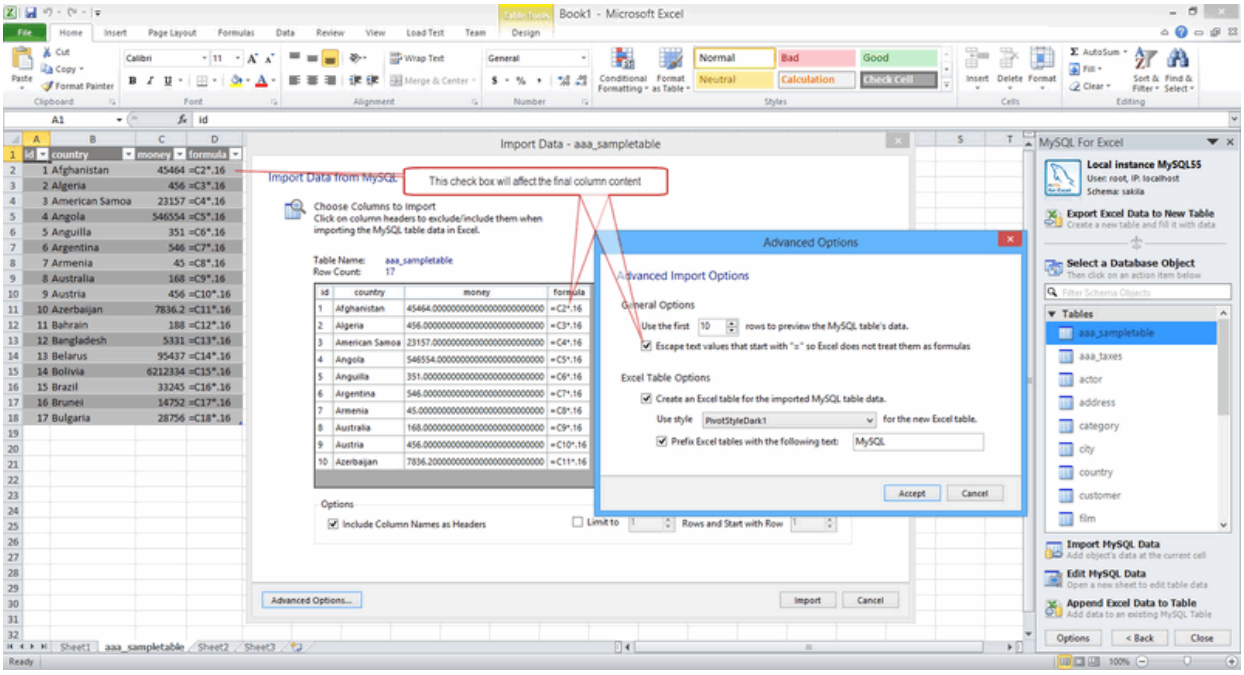
- Use the first [*10*] rows to preview the MySQL tables data. This affects the preview step in the import process, and defaults to 10.

Figure 10.2 MySQL for Excel: Preview



- [ ] Escape text values that start with "=" so Excel does not treat them as formulas, and is enabled by default. This option may not reflect any differences in the preview because it is only applied after the data is imported into the Excel Worksheet.

Figure 10.3 MySQL for Excel: Escape "=" (formulas)

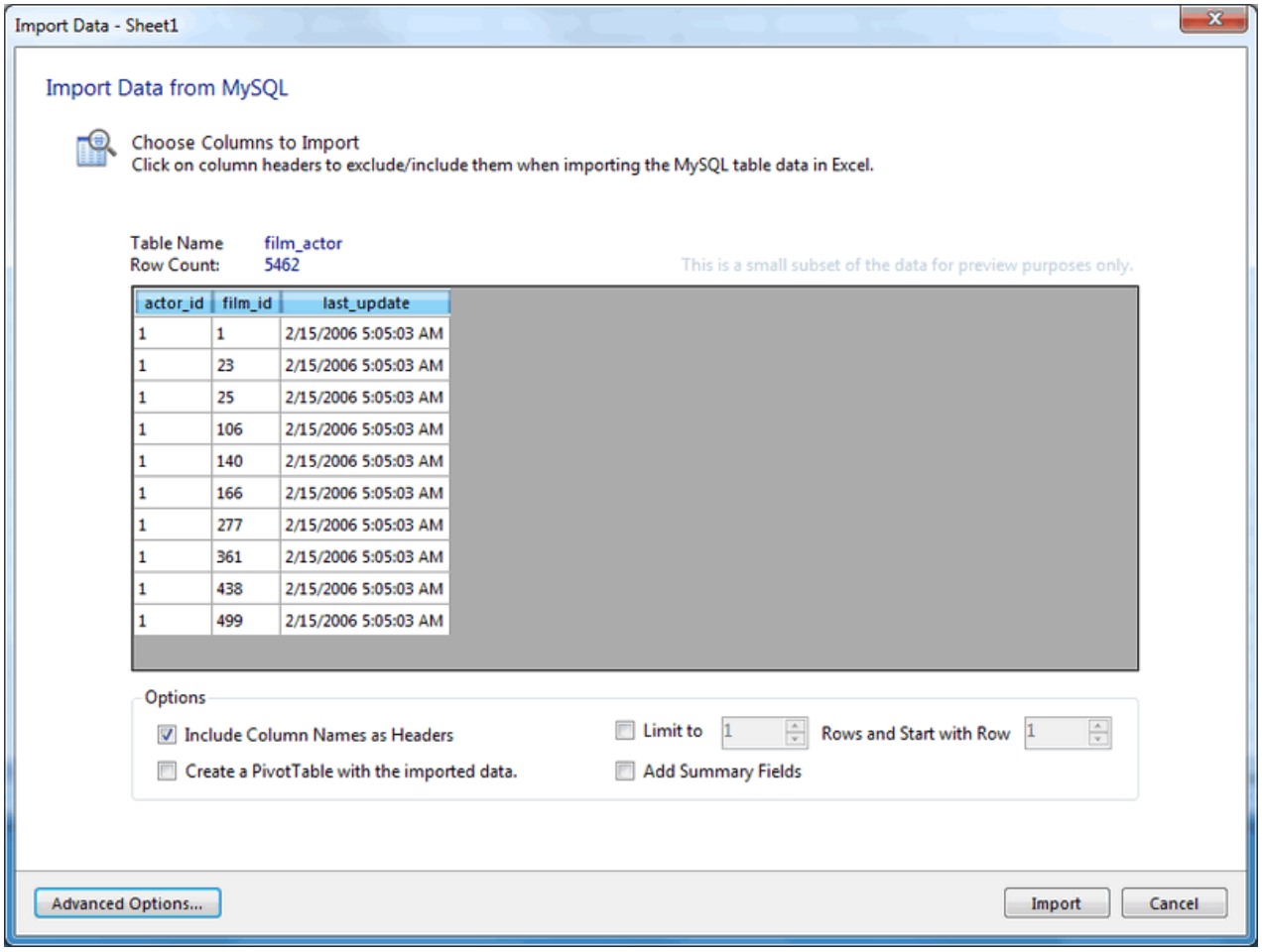


Excel Table Options:

- [ ] Create an Excel table for the imported MySQL table data. Enabled by default.
- Use style [ ] for the new Excel table. Defaults to `MySqlDefault`.
- [ ] Prefix Excel tables with the following text: _____. Disabled by default.

Importing a table displays a dialog similar to the following:

Figure 10.4 Importing table data with MySQL for Excel



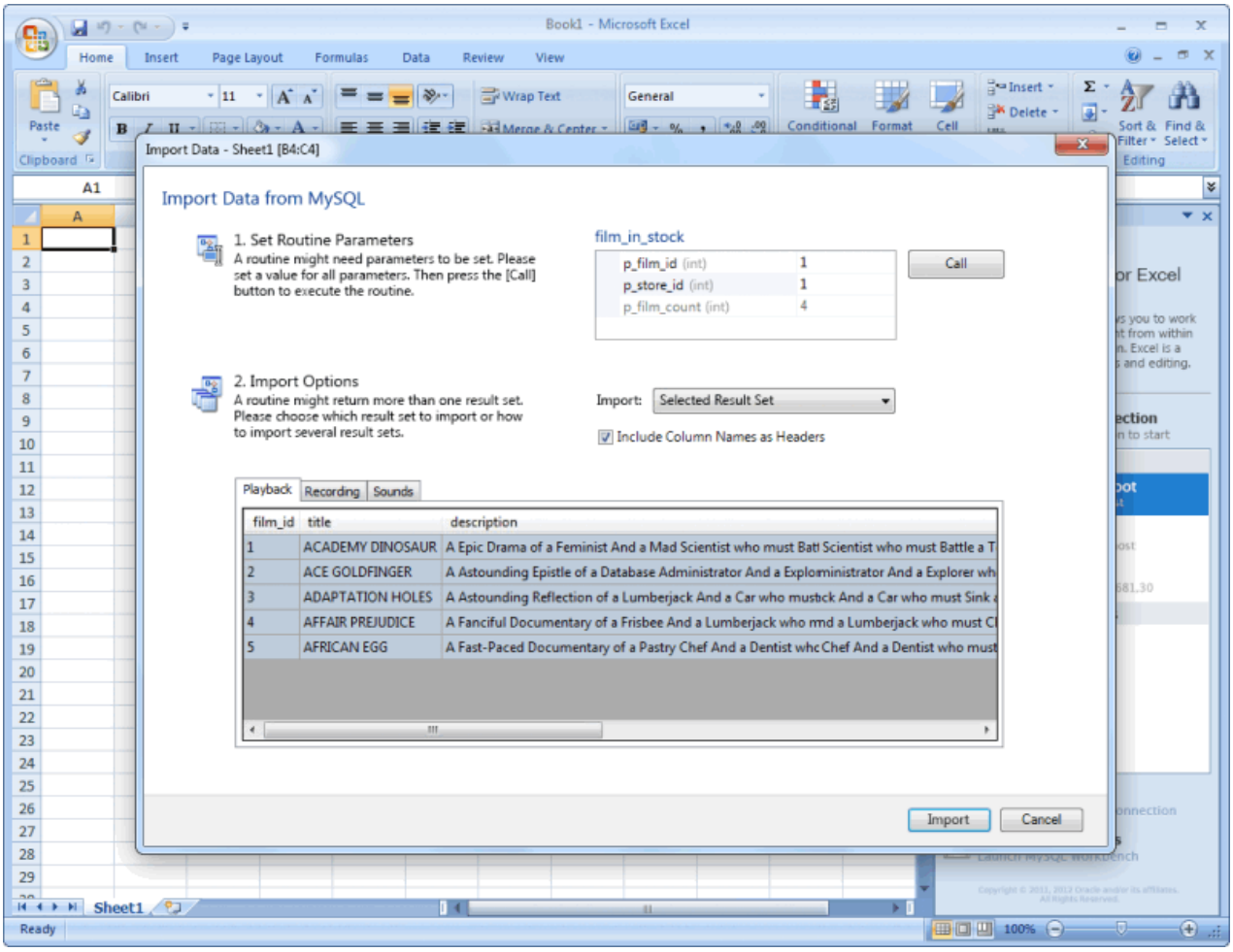
10.4 Importing a View or Procedure

Importing a view or procedure displays a similar dialogue, but with the following options:

- **Include Column Names as Headers**: Enabled by default, this will insert the column names at the top of the Excel spreadsheet as a "headers" row.
- **Import**: Because a procedure might return multiple result sets, the import options include:
  - **Selected Result Set**: Imports the selected tab sheet. This is the default behavior.
  - **All Result Sets - Arranged Horizontally**: Imports all result sets into the Excel Worksheet horizontally, and inserts one empty column between each result set.
  - **All Result Sets - Arranged Vertically**: Imports all result sets into the Excel Worksheet vertically, and inserts one empty row between each result set.

For example, a dialogue like the following is displayed after importing a procedure and pressing the **Call** button to invoke the stored procedure:

Figure 10.5 Importing called stored procedure data with MySQL for Excel



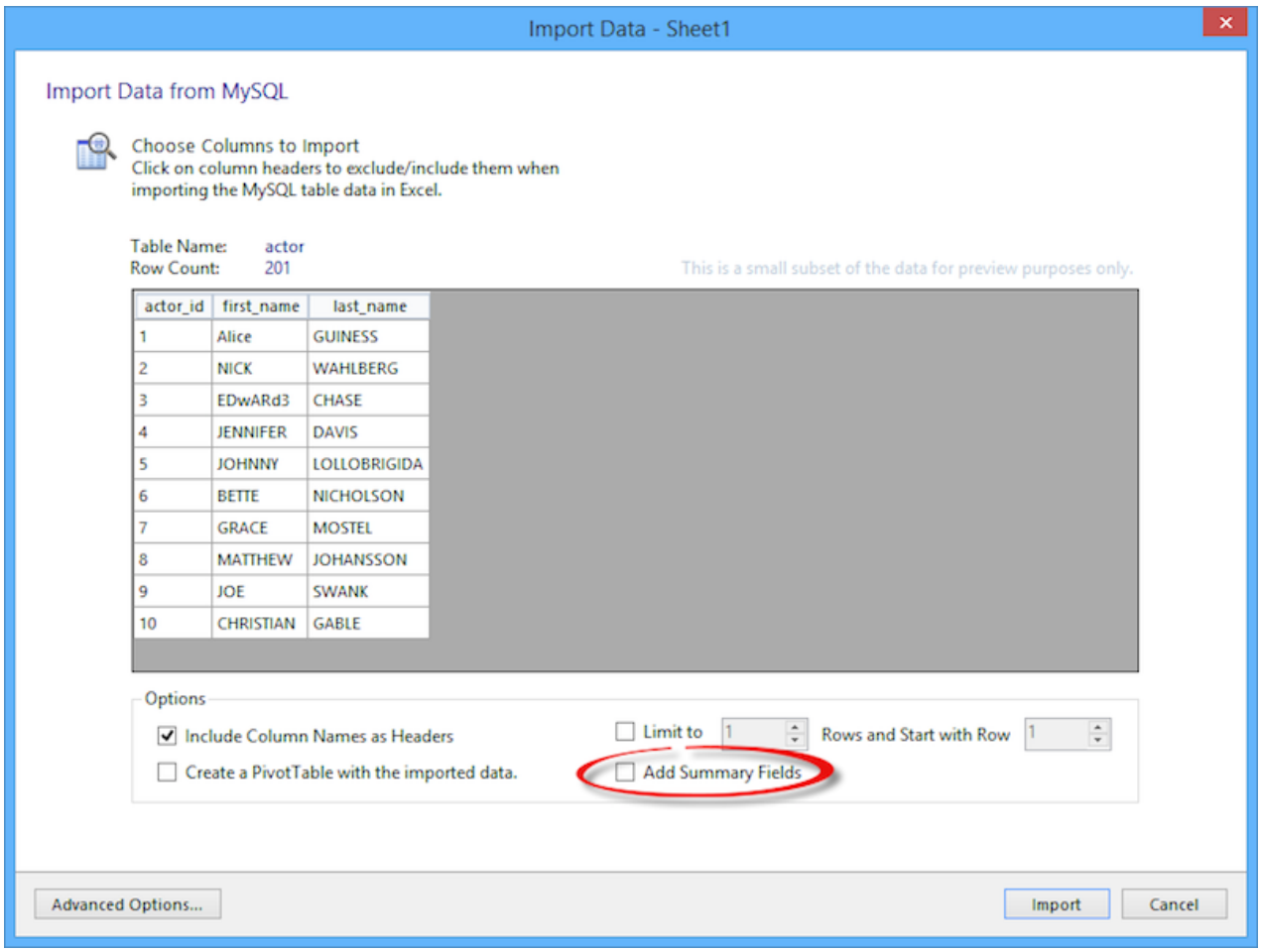
10.5 Adding Summary Fields

Summary fields are calculated fields, and this option adds summary related functions to each of the imported columns. These fields are added to the last row of the imported table data, and the dropdown of choices includes Average, Sum, Min, and Max.

> **Note**
>
> This feature was added in MySQL for Excel 1.3.0.

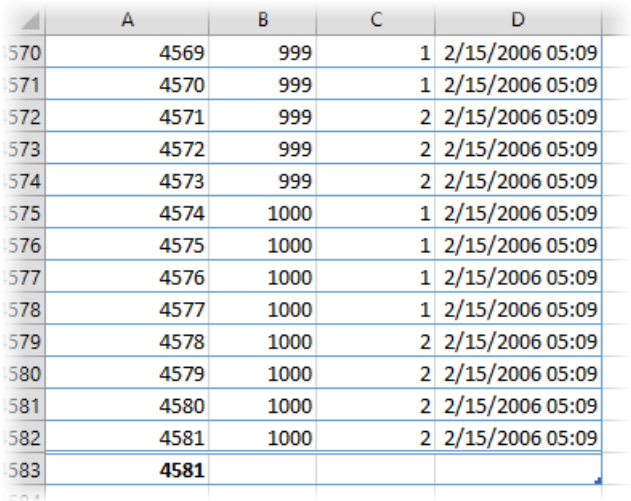
The **Add Summary Fields** option (disabled by default) is listed on the import dialog:

Figure 10.6 The 'Add Summary Fields' option



Enabling this option adds a row of summary fields for the appropriate columns in your imported data. Notice the newly created row on the bottom:

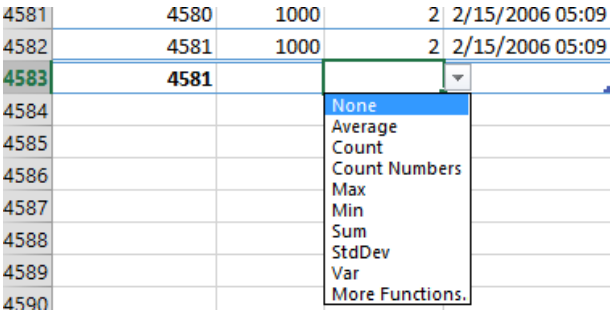
Figure 10.7 The new 'Add Summary Fields': the new row



	A	B	C	D
570	4569	999	1	2/15/2006 05:09
571	4570	999	1	2/15/2006 05:09
572	4571	999	2	2/15/2006 05:09
573	4572	999	2	2/15/2006 05:09
574	4573	999	2	2/15/2006 05:09
575	4574	1000	1	2/15/2006 05:09
576	4575	1000	1	2/15/2006 05:09
577	4576	1000	1	2/15/2006 05:09
578	4577	1000	1	2/15/2006 05:09
579	4578	1000	2	2/15/2006 05:09
580	4579	1000	2	2/15/2006 05:09
581	4580	1000	2	2/15/2006 05:09
582	4581	1000	2	2/15/2006 05:09
583	**4581**			

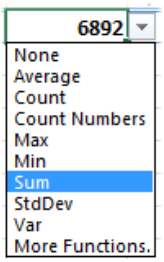
Select the row to reveal a down arrow, and click it to display a set of summary options:

Figure 10.8 The 'Add Summary Fields' row: choices



For example, choosing **Sum**:

Figure 10.9 The 'Add Summary Fields' row: sum example



Adjust each summary field accordingly.

10.6 Creating PivotTables

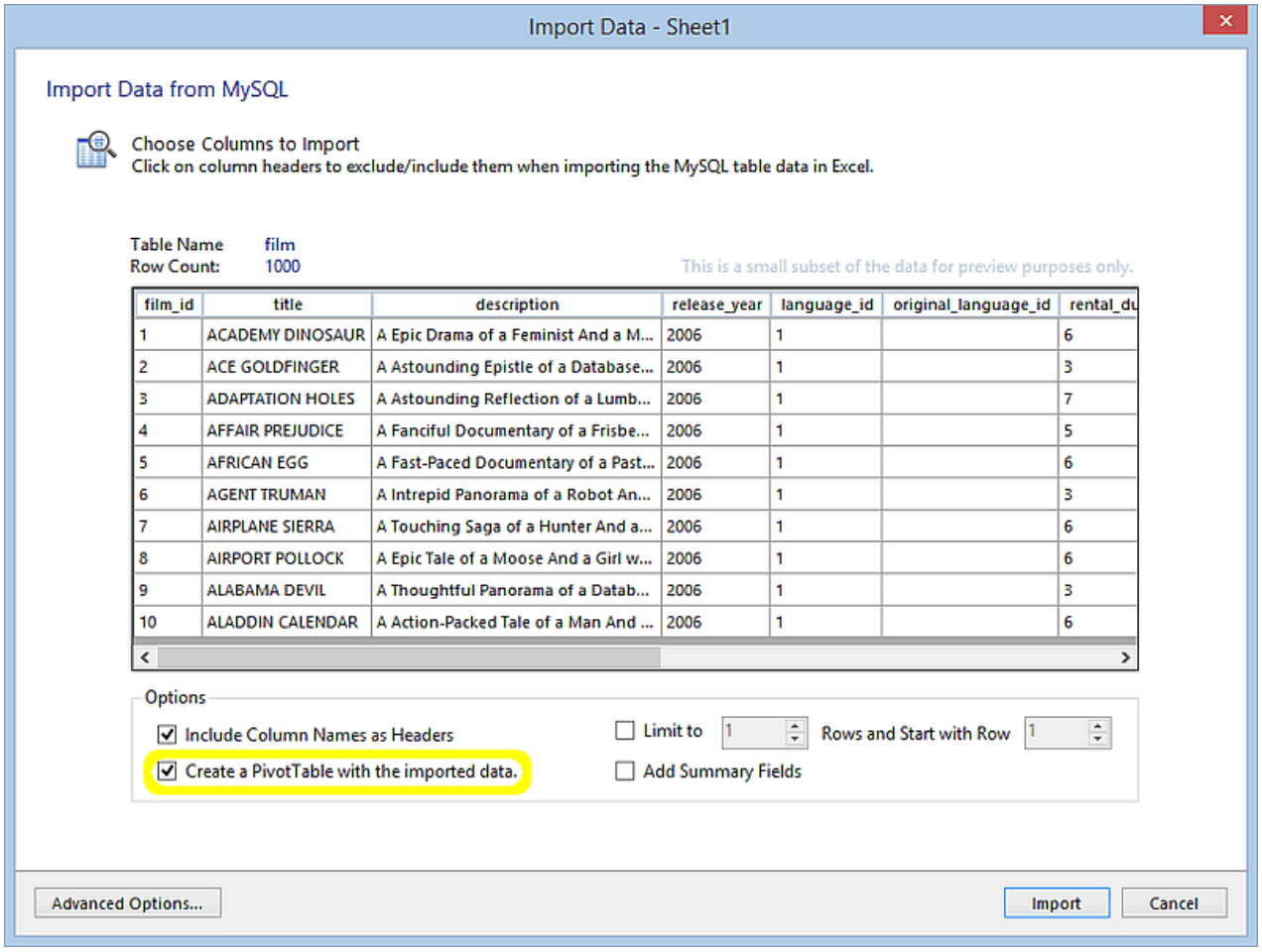
A PivotTable can be created from imported MySQL tables, views, stored procedures, or the entire Excel Data Model.

> **Note**
>
> This feature was added in MySQL for Excel 1.3.0.

An Excel PivotTable report summarizes and provides a visual representation of data in many different ways. It is a native Excel feature, see PivotTable reports 101 for additional information about Excel PivotTables.

Our example covers a simple use case where an empty PivotTable is created from an imported MySQL table. This example uses the "film" table of the "Sakila" database. To create the PivotTable, select the "film" table from the database object's selection panel and then click **Import MySQL Data**. On the **Import Data** dialog, check the **Create a PivotTable** before pressing **OK** to execute the operation.

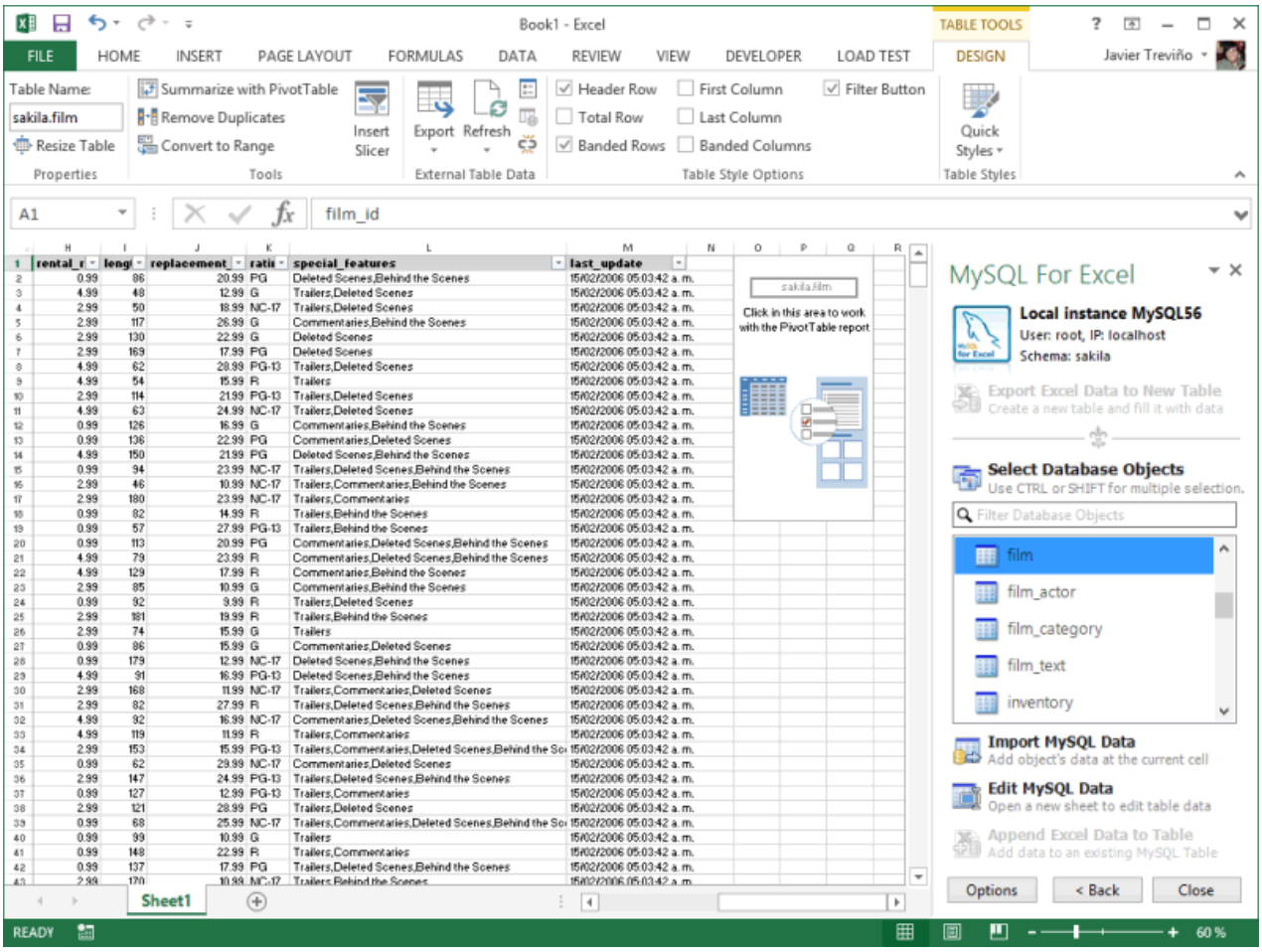
Figure 10.10 Option: Create a PivotTable with the imported data



When the **Create a PivotTable with the imported data** option is checked, an empty PivotTable (or a PivotTable placeholder) is inserted just to the right of the imported data. The PivotTable name follows the same naming rules used for Excel tables created from tje imported data, but PivotTables can be created with or without enabling the **Create an Excel table for the imported MySQL data** advanced option. That means a PivotTable can be created from an imported Excel range (if the aforementioned advanced option is off), or from an imported Excel table (if the option is on).

Click **Import** to dump the film table's data to an active Excel Spreadsheet, and this also creates a PivotTable for that data.

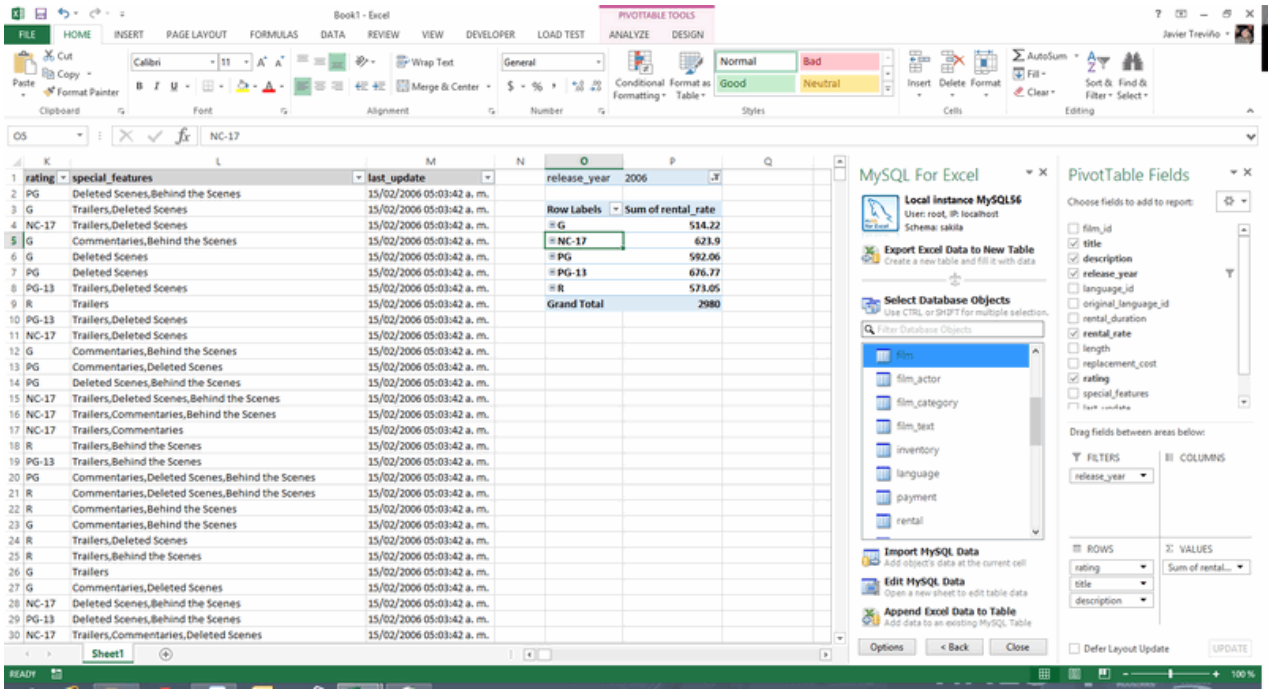
Figure 10.11 PivotTable Example: Empty PivotTable



Clicking the PivotTable opens a PivotTable Fields panel to next to the MySQL for Excel panel, and from here you can select fields you want to summarize in the PivotTable report. Drag and drop fields from the list to any of the FILTERS, COLUMNS, ROWS, or VALUES areas, depending on the visualizations you want in the report. The report is completely dynamic, meaning that you can change the views by moving fields around the areas until you see the visualization you need for your PivotTable report.

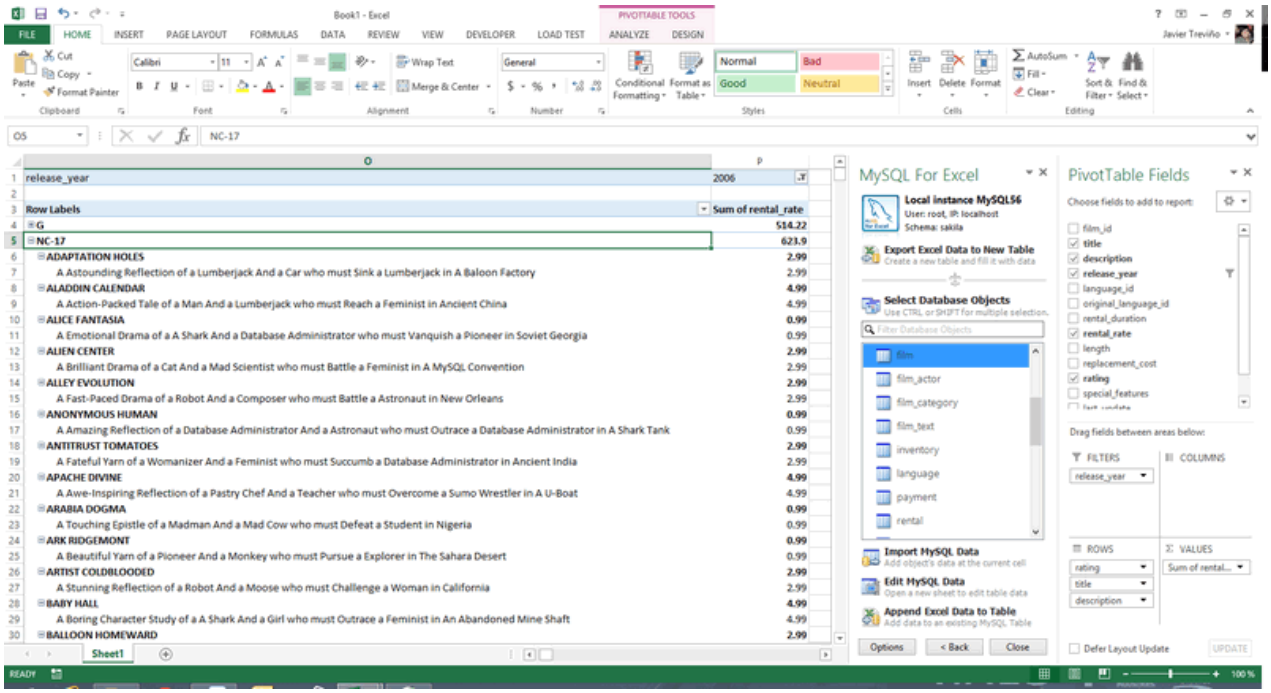
Below is an example PivotTable report using the sakila.film table we imported in our previous example. This report includes a filter by release_year, and it summarizes the rental_rate values while also grouping the data by values in the rating column.

Figure 10.12 PivotTable Example: Film Ratings



Expanding one of the groups reveals its values from the title and description columns.

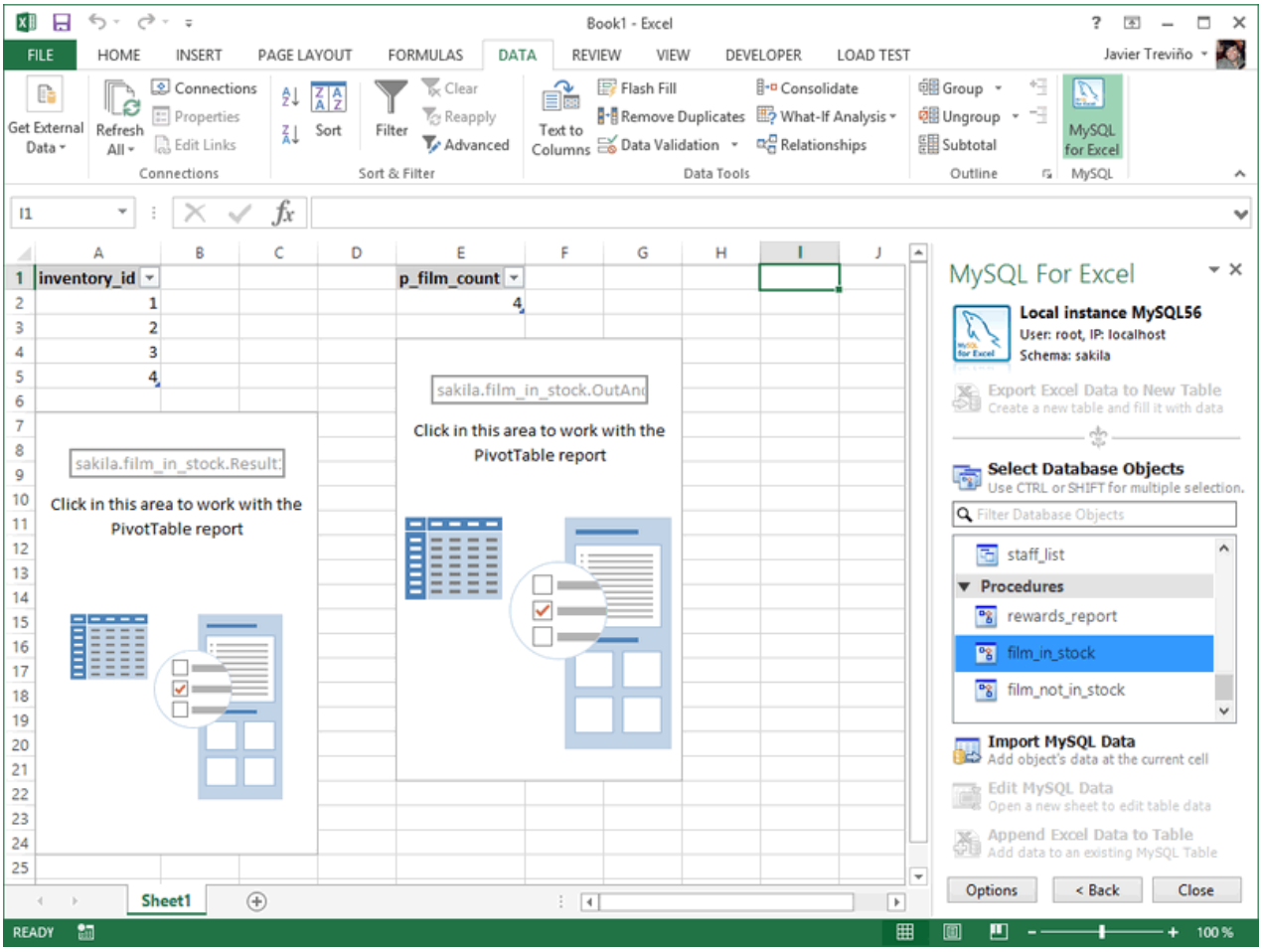
Figure 10.13 PivotTable Example: Expanded Group



We can also do this with data coming from a MySQL view or stored procedures. The only difference is that for stored procedures we can create a PivotTable for each of the imported result sets returned by the procedure's call. Take the following screenshot as an example where we have the film_in_stock stored procedure selected, we configured its input parameter values, and we called the procedure. You can see

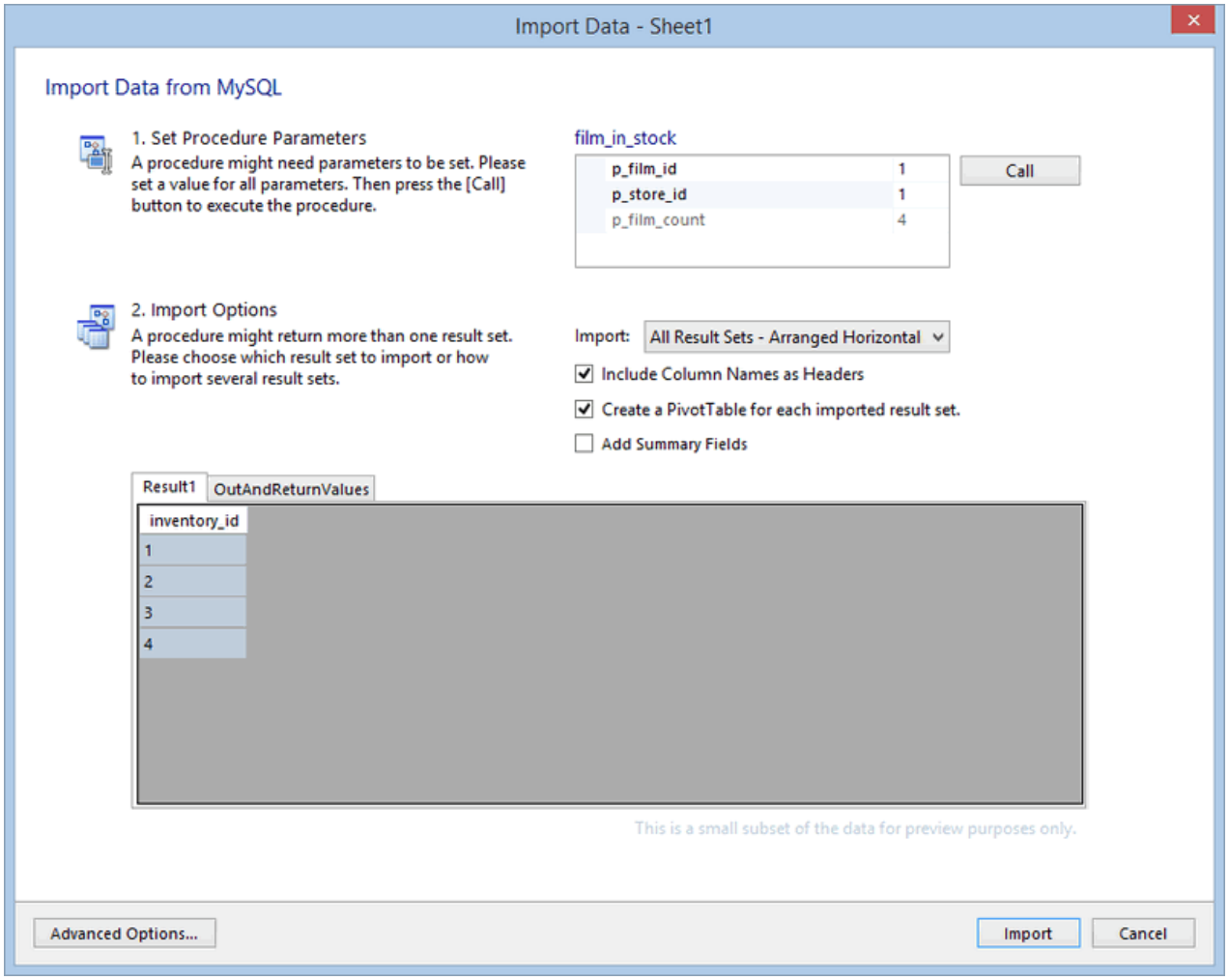
the procedure returned one result set (Result1) and the OutAndReturnValues table (always present if the procedure has output parameters or a return value).

Figure 10.14 PivotTable Example: Stored Procedure



In our example, we selected to import **All Result Sets - Arranged Horizontally**. Because the **Create a PivotTable with the imported data** option was also checked, a PivotTable was created for each returned result set.

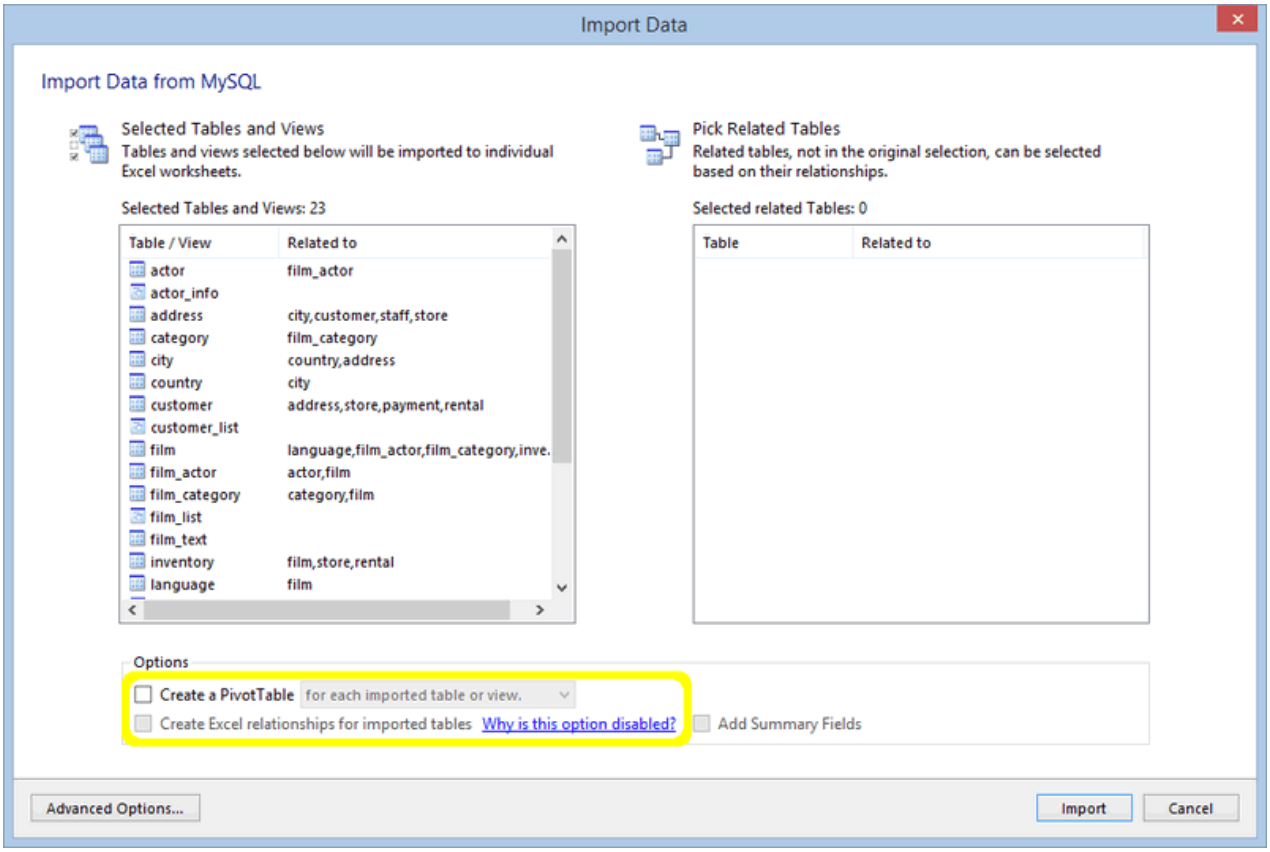
Figure 10.15 PivotTable Example: Arranged Horizontal



An important use case for PivotTables is when we create it for multiple related tables as typically a single table does not contain all of the data needed by a PivotTables report. You can create a single PivotTable tied to the data in the current Excel Data Model that contains fields from several related tables. That way you can use the data in a single report for an entire MySQL schema if needed. However, you can only do this in Excel 2013 (and later) where the Excel Data Model is available.

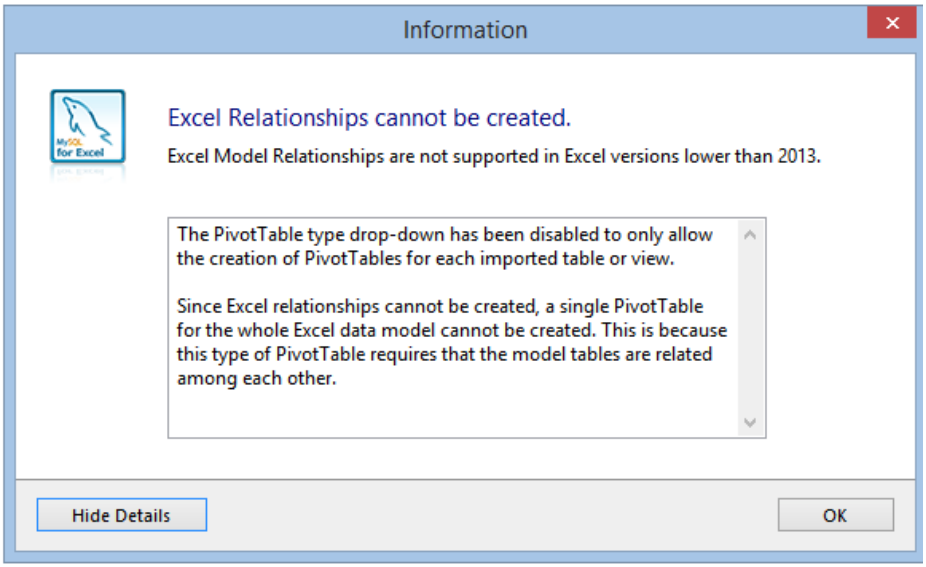
In Excel versions before Excel 2013, only a PivotTable for each imported table or view can be created. This is because a single PivotTable for the entire Excel Data Model requires that the tables are related to each other. If Excel relationships cannot be created, then this type of PivotTable cannot be created. So in these cases, the Import Data dialog looks like the following sample screenshot:

Figure 10.16 Disabled 'Create Excel relationships' option before Excel 2013



Clicking **Why is this option disabled?** displays an information dialog with an explanation of the disabled controls.

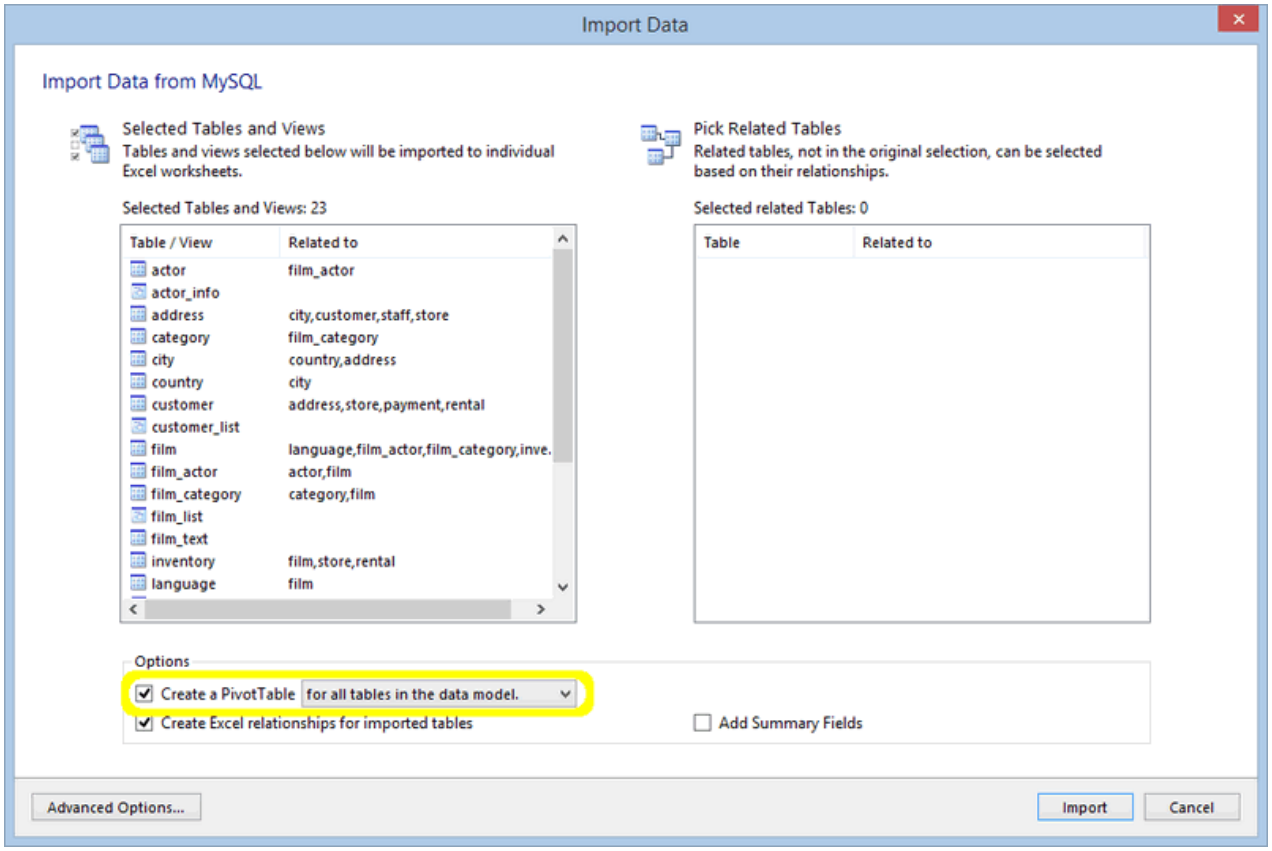
Figure 10.17 Disabled 'Create Excel relationships' option description



Our next example uses all tables in our schema. You can manually choose each table or use **Control** + **A** in the database objects list to select them all. When clicking **Import Multiple Tables and Views**, the

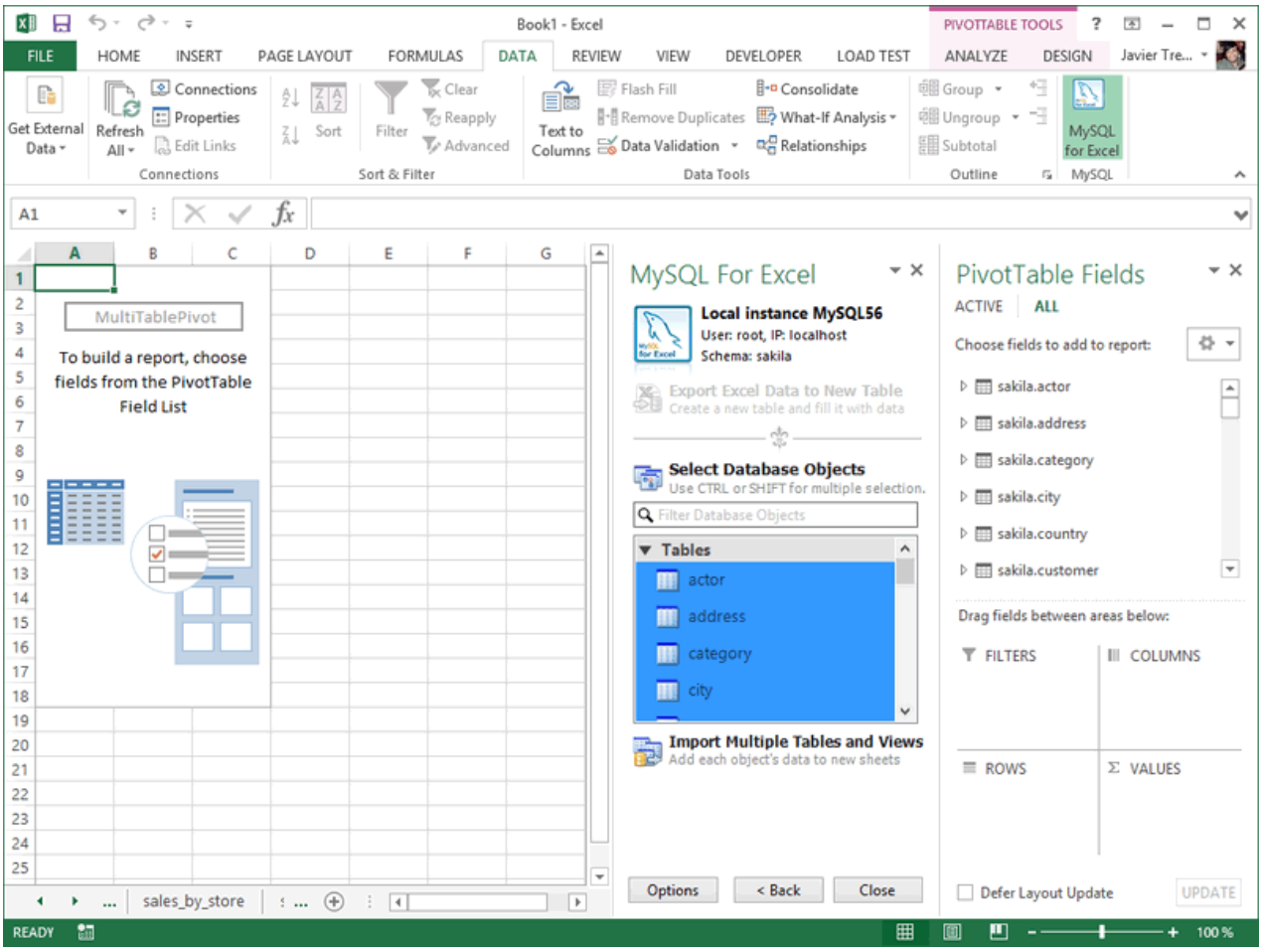
Import Data dialog appears as shown below. We need to check the **Create a PivotTable** option, which by default its drop-down is set to **for all the tables in the data model**. Keep that value.

Figure 10.18 Importing All Tables and Views



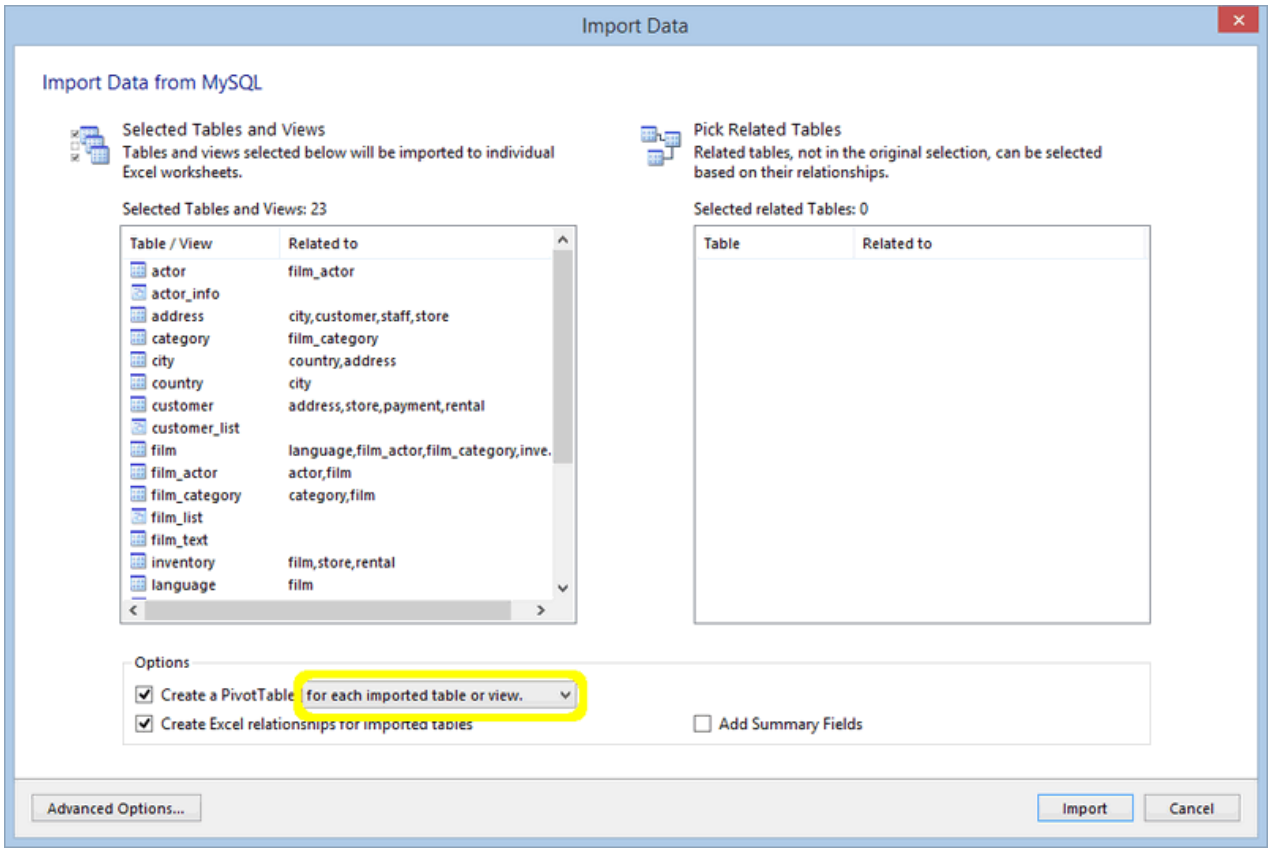
When clicking **Import**, the data in all of the selected tables is imported to Excel, its Data Model and Excel relationships are created, and a new worksheet is created that contains a PivotTable with all of the tables that were imported. This is demonstrated in the screenshot below, and notice that all tables are listed in the **PivotTable Fields** panel.

Figure 10.19 Importing All Tables and Views: Listing



You can also configure the **Create a PivotTable** drop-down list the for each imported table or view, which in turn will create a PivotTable for each of the imported tables or views, as opposed to creating a single PivotTable for all of them.

Figure 10.20 Importing Each Imported Table or View



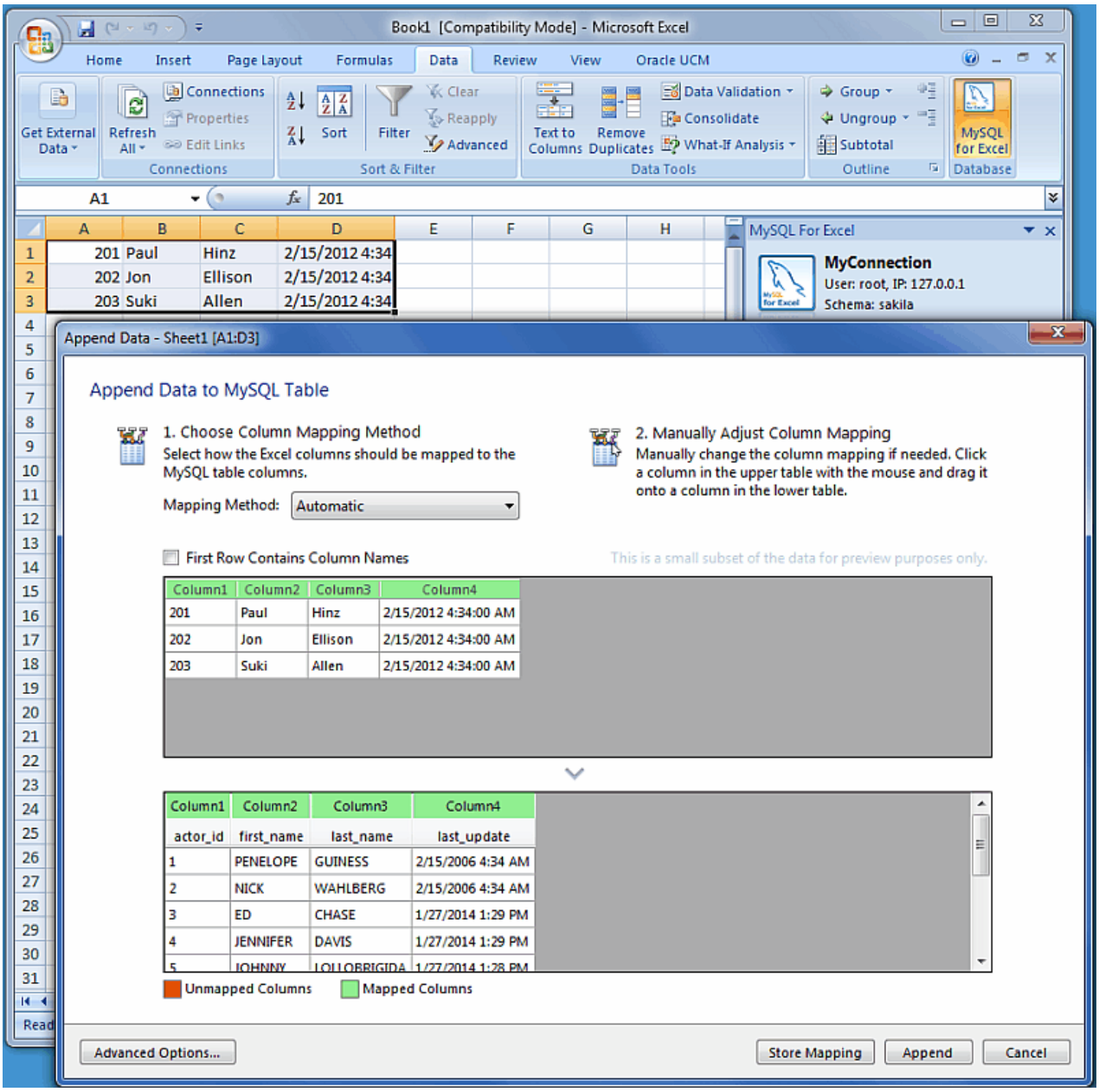
Chapter 11 Append Excel Data into MySQL

Data from a Microsoft Excel spreadsheet can be appended to a MySQL database table by using the **Append Excel MySQL Data to Table** option.

Column Mappings

Mapping the Excel columns to the MySQL columns can be executed automatically (default), manually, or by using a stored mapping routine. An automatic mapping routine is the default, and can be can be tweaked if every column cannot be matched automatically. The following screenshot shows two columns of Excel data, and the preview dialog after choosing **Append Excel Data to Table**:

Figure 11.1 Appending Excel data to MySQL (Automatic mapping)



General Mapping Information

It is common to tweak the column mappings. A few notes about the manual mapping process:

- Manual mapping is performed by dragging a column from the upper source grid (Excel spreadsheet) and dropping it into the lower target column MySQL table grid. Click anywhere within the column to initiate this dragging routine.
- The color of the header field for each column defines the current mapping status of the column. The colors include:
  - **Green**: A source column is mapped to a target column.
  - **Red**: A target column is not mapped.
  - **Gray**: A source column is not mapped.
- A source column may be mapped to multiple target columns, although this action generates a warning dialog.
- Right-clicking a target column shows a context menu with options to either **Remove Column Mapping** for a single column, or to **Clear All Mappings** for all columns. Dragging a target column outside of the grid removes the mapping.

Mapping Methods

The three mapping methods are described below:

- **Automatic**: The automatic mapping method attempts to match the Excel source column names with the MySQL target table column names. It is then possible to manually tweak the mapping afterwards.

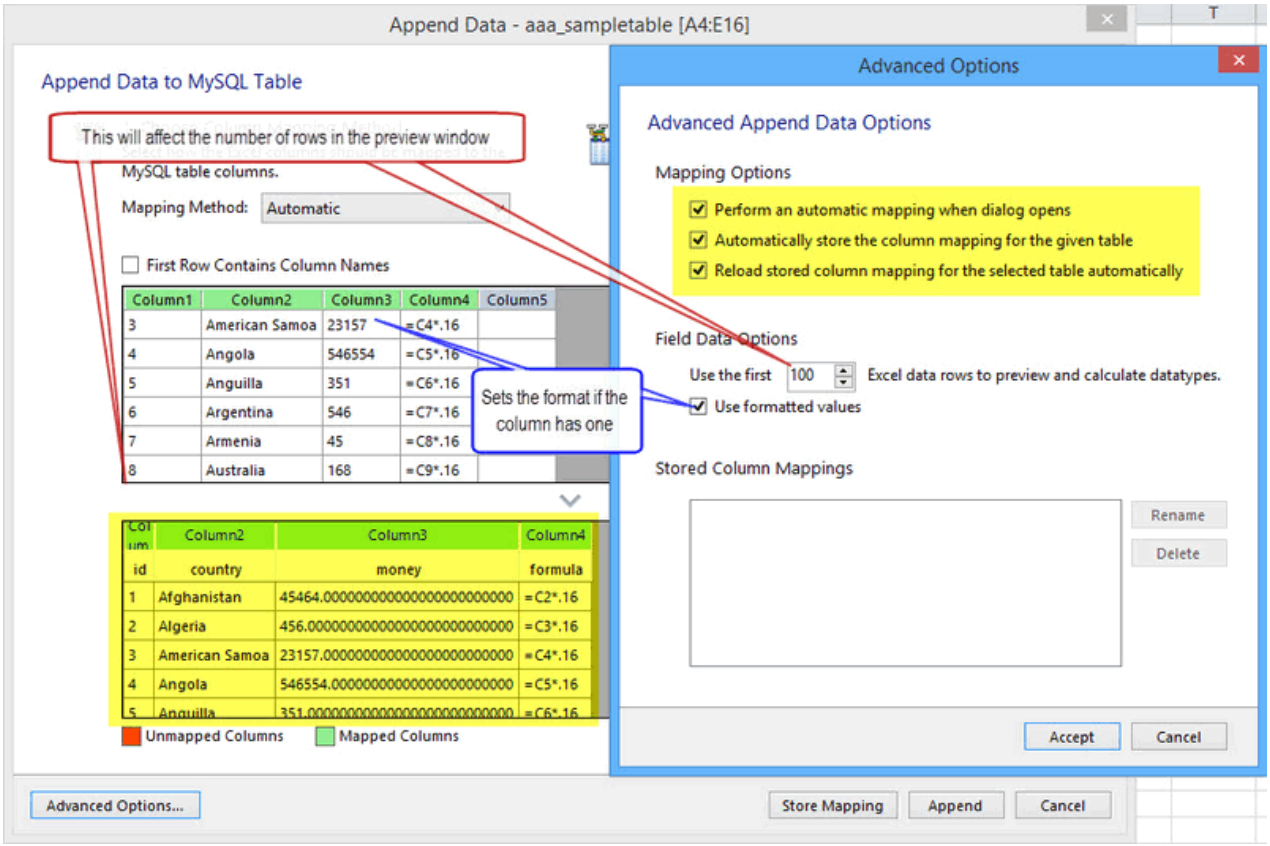
  If the automatic process finds zero columns to match, then a simple 1 to 1 matching routine is attempted. Meaning, SourceColumn #1 to TargetColumn #1, SourceColumn #2 to TargetColumn #2, and so on.
- **Manual**: The source column names are manually dragged (matched) with the target column names. Manual dragging can also be performed after the **Automatic** method is selected.
- **Stored**: Manual mapping styles may be saved using the **Store Mapping** button, which will also prompt for a name and then save it using a "*name* (dbname.tablename)" naming scheme. The saved mapping style will then be available alongside the **Automatic** and **Manual** options.

  Stored mappings may be deleted or renamed within the **Advanced Options** dialog.

Append: Advanced Options

There are several advanced options that are configured and stored between sessions for each Excel user. The advanced options are:

Figure 11.2 Appending Excel data to MySQL (Advanced Options)



The advanced **Mapping Options**:

- `Perform an automatic mapping when dialog opens`: Automatically attempt to map the target and source when the **Append Data** dialog is opened. This feature is enabled by default.
- `Automatically store the column mapping for the given table`: Stores each mapping routine after executing the **Append** operation. The mapping routine is saved using the "tablenameMapping (dbname.tablename)" format. This may also be performed manually using the **Store Mapping** button. It is enabled by default, and this feature was added in MySQL for Excel 1.1.0.
- `Reload stored column mapping for the selected table automatically`: If a stored mapping routine exists that matches all column names in the source grid with the target grid, then it is automatically be loaded. This is enabled by default, and this feature was added in MySQL for Excel 1.1.0.

The advanced **Field Data Options**:

- **Use the first** *`100`* (default) Excel data rows to preview and calculate data types. This determines the number of rows that the preview displays, and the values that affect the automatic mapping feature.
- **Use formatted values**: The data from Excel is treated as `Text`, `Double`, or `Date`. This is enabled by default. When disabled, data is never treated as a `Date` type, so for example, this means that a date would be represented as a number.

The advanced **SQL Queries Options**:

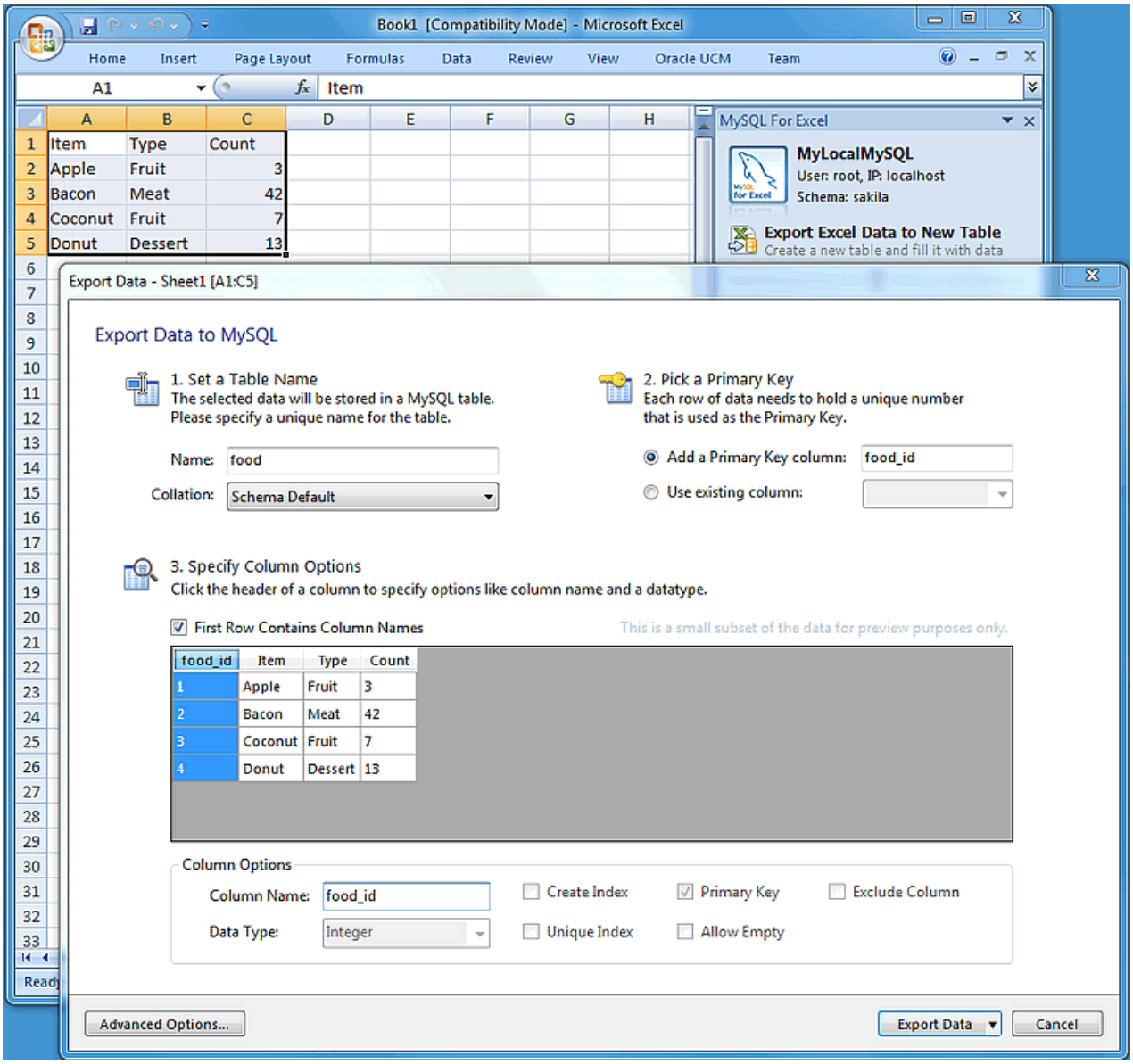
- `Disable table indexes to speed-up rows insertion`: This option is disabled by default, since you must make sure that if unique indexes are present, that the data mapped to that column does not contain duplicate data. This option was added in MySQL for Excel 1.2.1.

The **Stored Column Mappings** is a list of saved column mappings that were saved with the "Automatically store the column mapping for the given table" feature, or manually with the **Store Mapping** option.

Chapter 12 Export Excel Data into MySQL

Data from a Microsoft Excel spreadsheet can be exported to a new MySQL database table by using the **Export Excel Data to New Table** option. Exporting data looks like so:

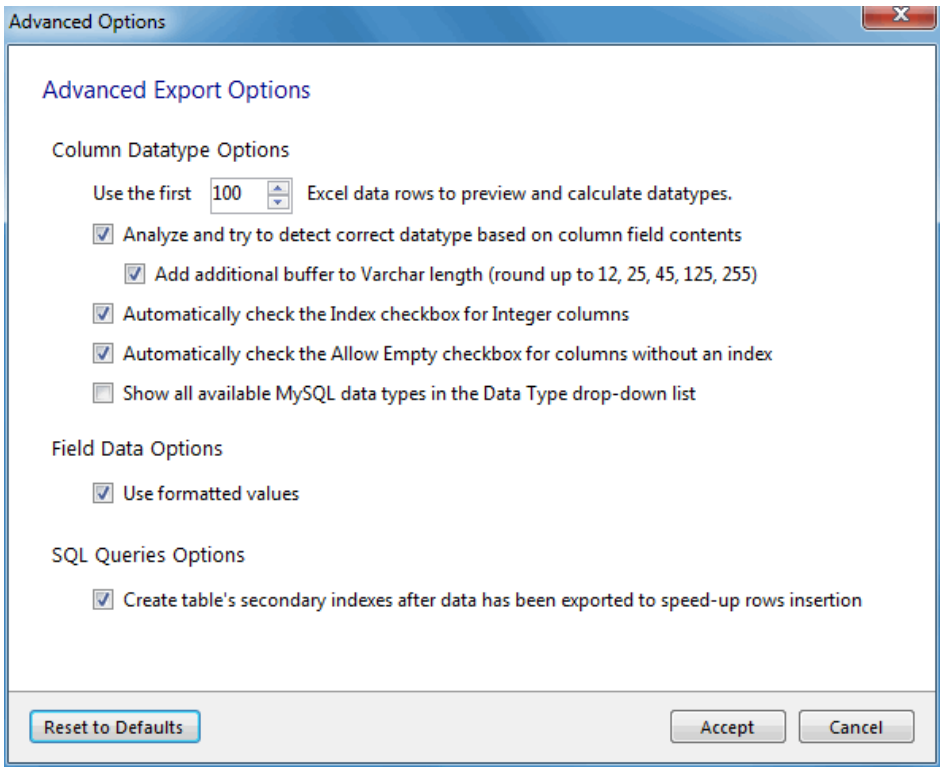
Figure 12.1 Exporting Excel data to MySQL



Advanced Export options

Several advanced options enables you to tweak the exported data. The advanced options dialog looks like so:

Figure 12.2 Exporting Excel data to MySQL (Advanced options)



- **Column Datatype Options**:
  - *Use the first `100` (default) Excel data rows to preview and calculate data types*: This determines the number of rows that the preview displays, and the values that affect the automatic mapping feature.
  - *Analyze and try to detect correct datatype based on column field contents*: Attempts to analyze the data and determine the data type for the column. The column type is defined as VARCHAR if it contains multiple types.
  - *Add additional buffer to VARCHAR length (round up to 12, 25, 45, 125, 255)*: When the data type is automatically detected and is set to VARCHAR, then it calculates the maximum length for all rows within the column, and rounds up the maximum length to one of the defined lengths above.

    If disabled, then the VARCHAR length is set to the length of the longest entry in the Excel spreadsheet.
  - *Automatically check the Index checkbox for Integer columns*: If enabled (default), columns with an Integer data type will have the **Create Index** option enabled by default.
  - *Automatically check the Allow Empty checkbox for columns without an index*: If enabled (default), columns without the **Create Index** checkbox checked will automatically enable the **Allow Empty** configuration option.
  - *Show all available MySQL data types in the Data Type drop-down list*: By default, only the most commonly used data types are displayed. Enable (disabled by default) this setting to see a list of all MySQL data types.

    Note

    This option was added in MySQL for Excel 1.3.0

- **Field Data options**:
  - *Use formatted values*: When enabled (default), the data from Excel is treated as `Text`, `Double`, or `Date`. When disabled, data is never treated as a `Date` type, so for example this means that a date would be represented as a number.
- **Other options**:
  - *Create table's secondary indexes after data has been exported to speed-up rows insertion*: This saves disk I/O for bulk inserts (thousands of rows) since re-indexing will not happen every time a row is inserted, but only once at the end of the data insertion. This option is enabled by default.

    > **Note**
    >
    > This option was added in MySQL for Excel 1.2.1.

  - > **Note**
    >
    > The following option was `Removed` in MySQL for Excel 1.2.1. Now, the default behavior is to always remove empty columns from the calculations.

    Remove columns that contain no data, otherwise flag them as "Excluded": If enabled, columns without data in Excel are removed and not shown in the preview panel. If disabled (default), these columns will exist but have the **Exclude Column** option checked.

Additional Notes

- Entering "0" into a date column.

  Entering the value "0" into an Excel date column will convert the value to "12/30/1899" in MySQL. This is because Excel first translates the value to the minimum date in Excel, which is "1/0/1900", because dates are internally stored in Excel as numbers (the days that have passed since "1/0/1900". However, because "1/0/1900" is not a valid date, the committed value to MySQL will change to "12/30/1899" because the .NET MySQL connector automatically converts "1/0/1900" to "12/30/1899", which is the closest valid date.

Chapter 13 MySQL for Excel Frequently Asked Questions

Questions

Questions and Answers

13.1: When I'm using Excel to edit data from a live MySQL database, will my changes overwrite changes made by other users? For example, maybe they used MySQL Workbench to edit the same data at the same time.

The optimistic updates feature checks for external edits and notifies you of their existence before committing any changes. If an external edit is discovered, you can then choose whether or not to overwrite their changes. This option is enabled by default and can be disabled (to use pessimistic updates). Disabling this option means external changes will always be overwritten. In other words, the choice is yours.

13.2: I installed the MySQL for Excel plugin, but can't find it in Microsoft Excel. How do I start it?

The MySQL for Excel plugin is automatically added to Microsoft Excel's data menu when it is installed. Look for the MySQL for Excel icon, by default it will be listed on the right side of the main menu.

If it's not there, then you might have to reinstall the plugin. But before doing so, first check if it's listed under "Add/Remove Programs" in Microsoft Windows. If not, then it has not been installed. Next, check the Excel Add-Ins list. For Office 2007 this is found by clicking the Office logo in Excel (top left corner), click **Excel Options**, then select **Add-Ins**. Is MySQL for Excel listed as a COM Add-in? If so, then consider filing a bug report (bugs.mysql.com), or attempt to reinstall the plugin.

13.3: I click on Edit Data and after importing the table data into Excel, I can't sort or move columns around. How can I do that?

In order to maintain the mapping of rows and columns in the Excel Worksheet against the rows and columns in the MySQL table, no alteration is permitted on the worksheet (i.e. sorting, deleting rows, deleting columns). If you need to alter the data there you can do that by right-clicking the **Edit Data** window and selecting **Exit Edit Mode**.

13.4: When editing a MySQL table's data, the Excel worksheet where the data is dumped is protected. How can unprotect it?

The Excel worksheet is protected to not allow alterations to the order of rows and columns. The password used for the protection is a GUID auto-generated at runtime so that the protection is not violated in any way. If you wish to unprotect the worksheet to manipulate your data, you can do that by right-clicking the **Edit Data** window and selecting **Exit Edit Mode**.

13.5: The MySQL Workbench connections that use SSH tunneling appear grayed out (disabled) in MySQL for Excel. How can I use a SSH connection?

This is a known limitation of MySQL for Excel. MySQL for Excel uses MySQL Connector/NET to connect and communicate with MySQL databases. Connector/NET does not have SSH support, so the behavior will change if Connector/NET supports it in the future.