

# MySQL Connector/ODBC

---

# MySQL Connector/ODBC

## Abstract

This manual describes MySQL Connector/ODBC.

Document generated on: 2009-06-02 (revision: 15165)

Copyright © 1997-2008 MySQL AB, 2009 Sun Microsystems, Inc. All rights reserved. U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements. Use is subject to license terms. Sun, Sun Microsystems, the Sun logo, Java, Solaris, StarOffice, MySQL Enterprise Monitor 2.0, MySQL logo™ and MySQL™ are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Copyright © 1997-2008 MySQL AB, 2009 Sun Microsystems, Inc. Tous droits réservés. L'utilisation est soumise aux termes du contrat de licence. Sun, Sun Microsystems, le logo Sun, Java, Solaris, StarOffice, MySQL Enterprise Monitor 2.0, MySQL logo™ et MySQL™ sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

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

For more information on the terms of this license, for details on how the MySQL documentation is built and produced, or if you are interested in doing a translation, please contact the [Documentation Team](#).

For additional licensing information, including licenses for libraries used by MySQL, see [Preface, Notes, Licenses](#).

If you want 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, CHM, and PDF formats, see [MySQL Documentation Library](#).

---

---

---

---

# MySQL Connector/ODBC

The MySQL Connector/ODBC is the name for the family of MySQL ODBC drivers (previously called MyODBC drivers) that provide access to a MySQL database using the industry standard Open Database Connectivity (ODBC) API. This reference covers Connector/ODBC 3.51 and Connector/ODBC 5.1. Both releases provide an ODBC compliant interface to MySQL Server.

MySQL Connector/ODBC provides both driver-manager based and native interfaces to the MySQL database, which full support for MySQL functionality, including stored procedures, transactions and, with Connector/ODBC 5.1, full Unicode compliance.

For more information on the ODBC API standard and how to use it, refer to <http://support.microsoft.com/kb/110093>.

The application development part of this reference assumes a good working knowledge of C, general DBMS knowledge, and finally, but not least, familiarity with MySQL. For more information about MySQL functionality and its syntax, refer to <http://dev.mysql.com/doc/>.

Typically, you need to install Connector/ODBC only on Windows machines. For Unix and Mac OS X you can use the native MySQL network or named pipe to communicate with your MySQL database. You may need Connector/ODBC for Unix or Mac OS X if you have an application that requires an ODBC interface to communicate with the database. Applications that require ODBC to communicate with MySQL include ColdFusion, Microsoft Office, and Filemaker Pro.

## Key topics:

- For help installing Connector/ODBC see [Chapter 2, \*Connector/ODBC Installation\*](#).
- For information on the configuration options, see [Section 3.2, “Connector/ODBC Connection Parameters”](#).
- For more information on connecting to a MySQL database from a Windows host using Connector/ODBC see [Section 4.2, “Step-by-step Guide to Connecting to a MySQL Database through Connector/ODBC”](#).
- If you want to use Microsoft Access as an interface to a MySQL database using Connector/ODBC see [Section 4.4, “Using Connector/ODBC with Microsoft Access”](#).
- General tips on using Connector/ODBC, including obtaining the last auto-increment ID see [Section 6.1, “Connector/ODBC General Functionality”](#).
- For tips and common questions on using Connector/ODBC with specific application see [Section 6.2, “Connector/ODBC Application Specific Tips”](#).
- For a general list of Frequently Asked Questions see [Section 6.3, “Connector/ODBC Errors and Resolutions \(FAQ\)”](#).
- Additional support when using Connector/ODBC is available, see [Chapter 7, \*Connector/ODBC Support\*](#).

### MySQL Enterprise

MySQL Enterprise subscribers will find more information about MySQL and ODBC in the Knowledge Base articles about [ODBC](#). Access to the MySQL Knowledge Base collection of articles is one of the advantages of subscribing to MySQL Enterprise. For more information, see <http://www.mysql.com/products/enterprise/advisors.html>.

---

# Chapter 1. Connector/ODBC Introduction

ODBC (Open Database Connectivity) provides a way for client programs to access a wide range of databases or data sources. ODBC is a standardized API that allows connections to SQL database servers. It was developed according to the specifications of the SQL Access Group and defines a set of function calls, error codes, and data types that can be used to develop database-independent applications. ODBC usually is used when database independence or simultaneous access to different data sources is required.

For more information about ODBC, refer to <http://support.microsoft.com/kb/110093>.

## 1.1. Connector/ODBC Roadmap

Connector/ODBC 5.1 is currently in development and will be a complete implementation of the ODBC Core interface, plus more Level 1 and Level 2 functionality of the ODBC specification than that currently supported by Connector/ODBC 3.51.

The following functionality was added or changed as part of 5.1:

- Add support for SQL\_NUMERIC\_STRUCT: [MSDN Article 714556](#).
- Replace installer library with new implementation (from v5 tree).
- Implement native Windows setup library.
- Implement SQLCancel() ([Bug#15601](#)): [MSDN Article 714112](#).

The following functionality will be added in a version after 5.1:

- Implement native Mac OS X setup library.
- Replace OPTIONS flags with individual DSN settings (but support OPTIONS for backwards-compatibility).
- Fix support for SQLBIGINT ([Bug#28887](#)): [MSDN Article 714121](#).
- Make diagnostics support standards-compliant: [MSDN Article 711021](#).
- Add support for SQL\_ATTR\_METADATA\_ID: [MSDN Article 716447](#).
- Implement SQLBrowseConnect(): [MSDN Article 714565](#), [MSDN Article 712446](#).
- Implement arrays of parameters: [MSDN Article 711818](#).

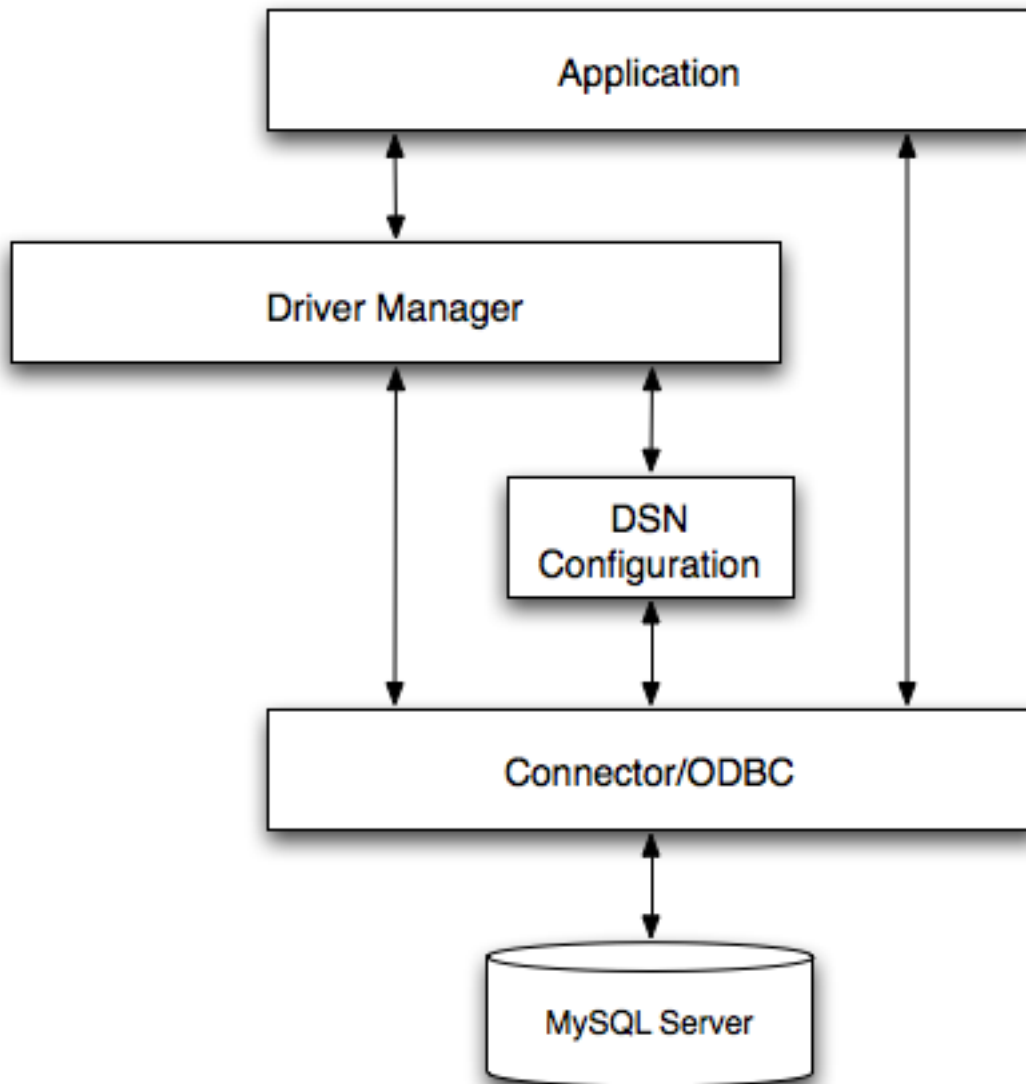
## 1.2. General Information About ODBC and Connector/ODBC

Open Database Connectivity (ODBC) is a widely accepted application-programming interface (API) for database access. It is based on the Call-Level Interface (CLI) specifications from X/Open and ISO/IEC for database APIs and uses Structured Query Language (SQL) as its database access language.

A survey of ODBC functions supported by Connector/ODBC is given at [Section 5.1, “Connector/ODBC API Reference”](#). For general information about ODBC, see <http://support.microsoft.com/kb/110093>.

### 1.2.1. Connector/ODBC Architecture

The Connector/ODBC architecture is based on five components, as shown in the following diagram:



- **Application:**

The Application uses the ODBC API to access the data from the MySQL server. The ODBC API in turn uses the communicates with the Driver Manager. The Application communicates with the Driver Manager using the standard ODBC calls. The Application does not care where the data is stored, how it is stored, or even how the system is configured to access the data. It needs to know only the Data Source Name (DSN).

A number of tasks are common to all applications, no matter how they use ODBC. These tasks are:

- Selecting the MySQL server and connecting to it
- Submitting SQL statements for execution
- Retrieving results (if any)
- Processing errors
- Committing or rolling back the transaction enclosing the SQL statement
- Disconnecting from the MySQL server

Because most data access work is done with SQL, the primary tasks for applications that use ODBC are submitting SQL statements and retrieving any results generated by those statements.

- **Driver manager:**

The Driver Manager is a library that manages communication between application and driver or drivers. It performs the following tasks:

- Resolves Data Source Names (DSN). The DSN is a configuration string that identifies a given database driver, database, database host and optionally authentication information that enables an ODBC application to connect to a database using a standardized reference.

Because the database connectivity information is identified by the DSN, any ODBC compliant application can connect to the data source using the same DSN reference. This eliminates the need to separately configure each application that needs access to a given database; instead you instruct the application to use a pre-configured DSN.

- Loading and unloading of the driver required to access a specific database as defined within the DSN. For example, if you have configured a DSN that connects to a MySQL database then the driver manager will load the Connector/ODBC driver to enable the ODBC API to communicate with the MySQL host.
- Processes ODBC function calls or passes them to the driver for processing.

- **Connector/ODBC Driver:**

The Connector/ODBC driver is a library that implements the functions supported by the ODBC API. It processes ODBC function calls, submits SQL requests to MySQL server, and returns results back to the application. If necessary, the driver modifies an application's request so that the request conforms to syntax supported by MySQL.

- **DSN Configuration:**

The ODBC configuration file stores the driver and database information required to connect to the server. It is used by the Driver Manager to determine which driver to be loaded according to the definition in the DSN. The driver uses this to read connection parameters based on the DSN specified. For more information, [Chapter 3, Connector/ODBC Configuration](#).

- **MySQL Server:**

The MySQL database where the information is stored. The database is used as the source of the data (during queries) and the destination for data (during inserts and updates).

## 1.2.2. ODBC Driver Managers

An ODBC Driver Manager is a library that manages communication between the ODBC-aware application and any drivers. Its main functionality includes:

- Resolving Data Source Names (DSN).
- Driver loading and unloading.
- Processing ODBC function calls or passing them to the driver.

Both Windows and Mac OS X include ODBC driver managers with the operating system. Most ODBC Driver Manager implementations also include an administration application that makes the configuration of DSN and drivers easier. Examples and information on these managers, including Unix ODBC driver managers are listed below:

- Microsoft Windows ODBC Driver Manager (`odbc32.dll`), <http://support.microsoft.com/kb/110093>.
- Mac OS X includes [ODBC Administrator](#), a GUI application that provides a simpler configuration mechanism for the Unix iODBC Driver Manager. You can configure DSN and driver information either through ODBC Administrator or through the iODBC configuration files. This also means that you can test ODBC Administrator configurations using the `iodbctest` command. <http://www.apple.com>.
- [unixODBC Driver Manager for Unix](#) (`libodbc.so`). See <http://www.unixodbc.org>, for more information. The `unixODBC` Driver Manager includes the Connector/ODBC driver 3.51 in the installation package, starting with version `unixODBC 2.1.2`.
- [iODBC ODBC Driver Manager for Unix](#) (`libiodbc.so`), see <http://www.iodbc.org>, for more information.

---

## Chapter 2. Connector/ODBC Installation

You can install the Connector/ODBC drivers using two different methods, a binary installation and a source installation. The binary installation is the easiest and most straightforward method of installation. Using the source installation methods should only be necessary on platforms where a binary installation package is not available, or in situations where you want to customize or modify the installation process or Connector/ODBC drivers before installation.

### Where to Get Connector/ODBC

Sun Microsystems, Inc distributes its MySQL products under the General Public License (GPL). You can get a copy of the latest version of Connector/ODBC binaries and sources from our Web site at <http://dev.mysql.com/downloads/>.

For more information about Connector/ODBC, visit <http://www.mysql.com/products/myodbc/>.

For more information about licensing, visit <http://www.mysql.com/company/legal/licensing/>.

### Supported Platforms

Connector/ODBC can be used on all major platforms supported by MySQL. You can install it on:

- Windows 95, 98, Me, NT, 2000, XP, and 2003
- All Unix-like Operating Systems, including: AIX, Amiga, BSDI, DEC, FreeBSD, HP-UX 10/11, Linux, NetBSD, OpenBSD, OS/2, SGI Irix, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64 Unix
- Mac OS X and Mac OS X Server

Using a binary distribution offers the most straightforward method for installing Connector/ODBC. If you want more control over the driver, the installation location and or to customize elements of the driver you will need to build and install from the source.

If a binary distribution is not available for a particular platform build the driver from the original source code. You can contribute the binaries you create to MySQL by sending a mail message to [<myodbc@lists.mysql.com>](mailto:myodbc@lists.mysql.com), so that it becomes available for other users.

#### Note

On all non-Windows platforms except Mac OS X, the driver is built against `unixODBC` and is expecting a 2-byte `SQLWCHAR`, not 4 bytes as `iODBC` is using. For this reason, the binaries are **only** compatible with `unixODBC` and you will need to recompile the driver against `iODBC` if you wish to use them together. For further information see [Section 1.2.2, “ODBC Driver Managers”](#).

For further instructions:

Platform	Binary	Source
Windows	<a href="#">Installation Instructions</a>	<a href="#">Build Instructions</a>
Unix/Linux	<a href="#">Installation Instructions</a>	<a href="#">Build Instructions</a>
Mac OS X	<a href="#">Installation Instructions</a>	

## 2.1. Installing Connector/ODBC from a Binary Distribution on Windows

Before installing the Connector/ODBC drivers on Windows you should ensure that your Microsoft Data Access Components (MDAC) are up to date. You can obtain the latest version from the [Microsoft Data Access and Storage](#) Web site.

There are three available distribution types to use when installing for Windows. The contents in each case are identical, it is only the installation method which is different.

- Zipped installer consists of a Zipped package containing a standalone installation application. To install from this package, you must unzip the installer, and then run the installation application. See [Section 2.1.1, “Installing the Windows Connector/ODBC Driver using an installer”](#) to complete the installation.
- MSI installer, an installation file that can be used with the installer included in Windows 2000, Windows XP and Windows Server 2003. See [Section 2.1.1, “Installing the Windows Connector/ODBC Driver using an installer”](#) to complete the installa-



tion.

- Zipped DLL package, containing the DLL files that need must be manually installed. See [Section 2.1.2, “Installing the Windows Connector/ODBC Driver using the Zipped DLL package”](#) to complete the installation.

### Note

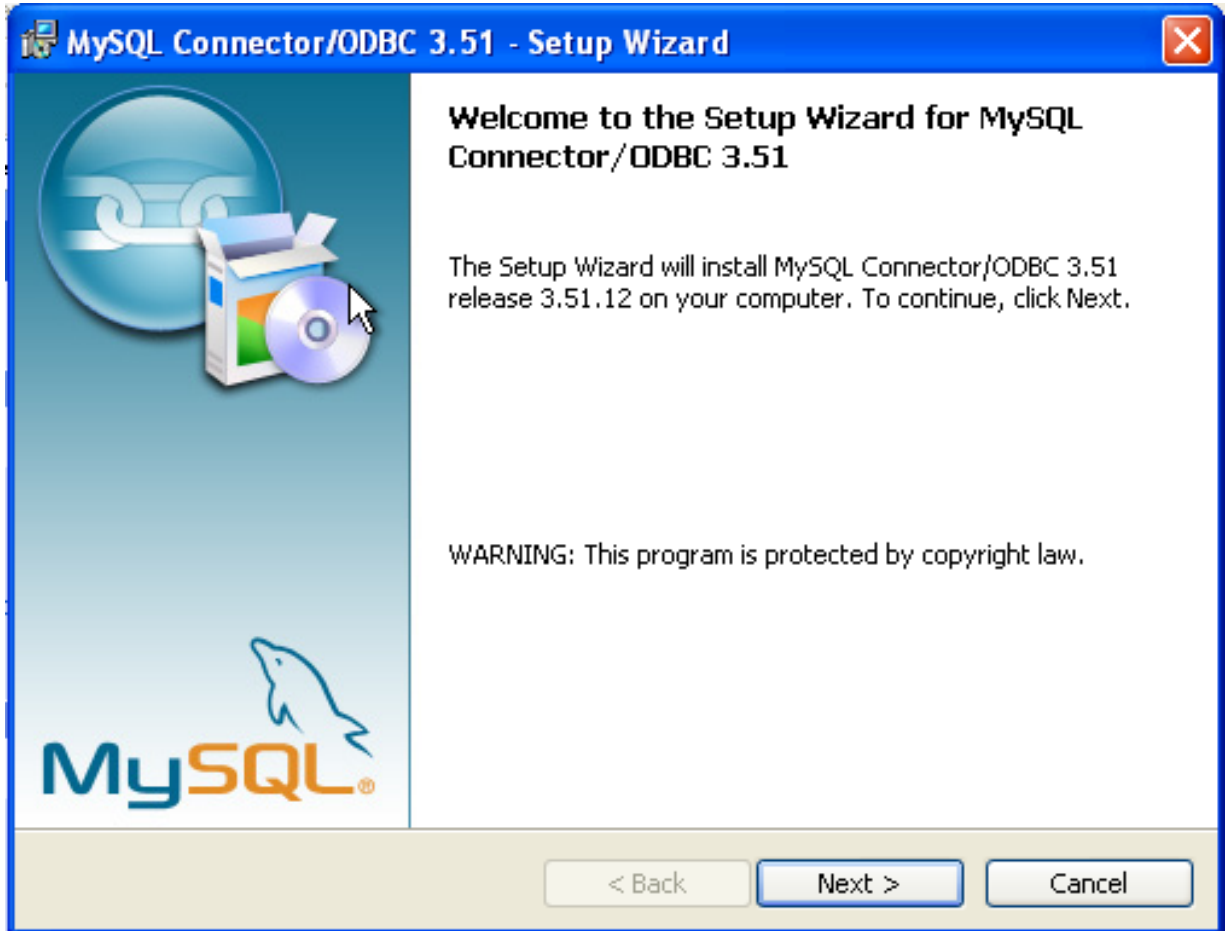
An OLEDB/ODBC driver for Windows 64-bit is available from [Microsoft Downloads](#).

## 2.1.1. Installing the Windows Connector/ODBC Driver using an installer

The installer packages offer a very simple method for installing the Connector/ODBC drivers. If you have downloaded the zipped installer then you must extract the installer application. The basic installation process is identical for both installers.

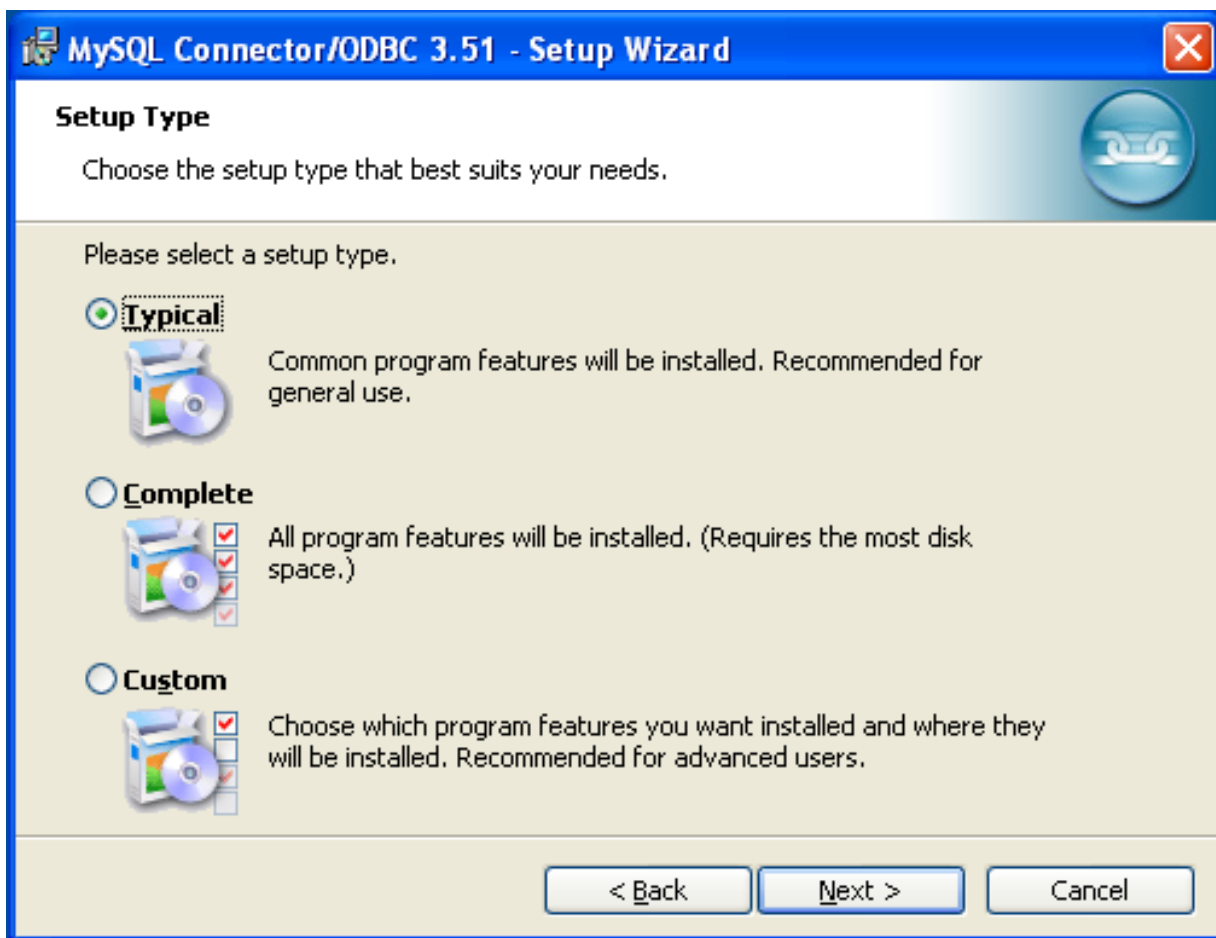
You should follow these steps to complete the installation:

1. Double click on the standalone installer that you extracted, or the MSI file you downloaded.
2. The MySQL Connector/ODBC 3.51 - Setup Wizard will start. Click the NEXT button to begin the installation process.

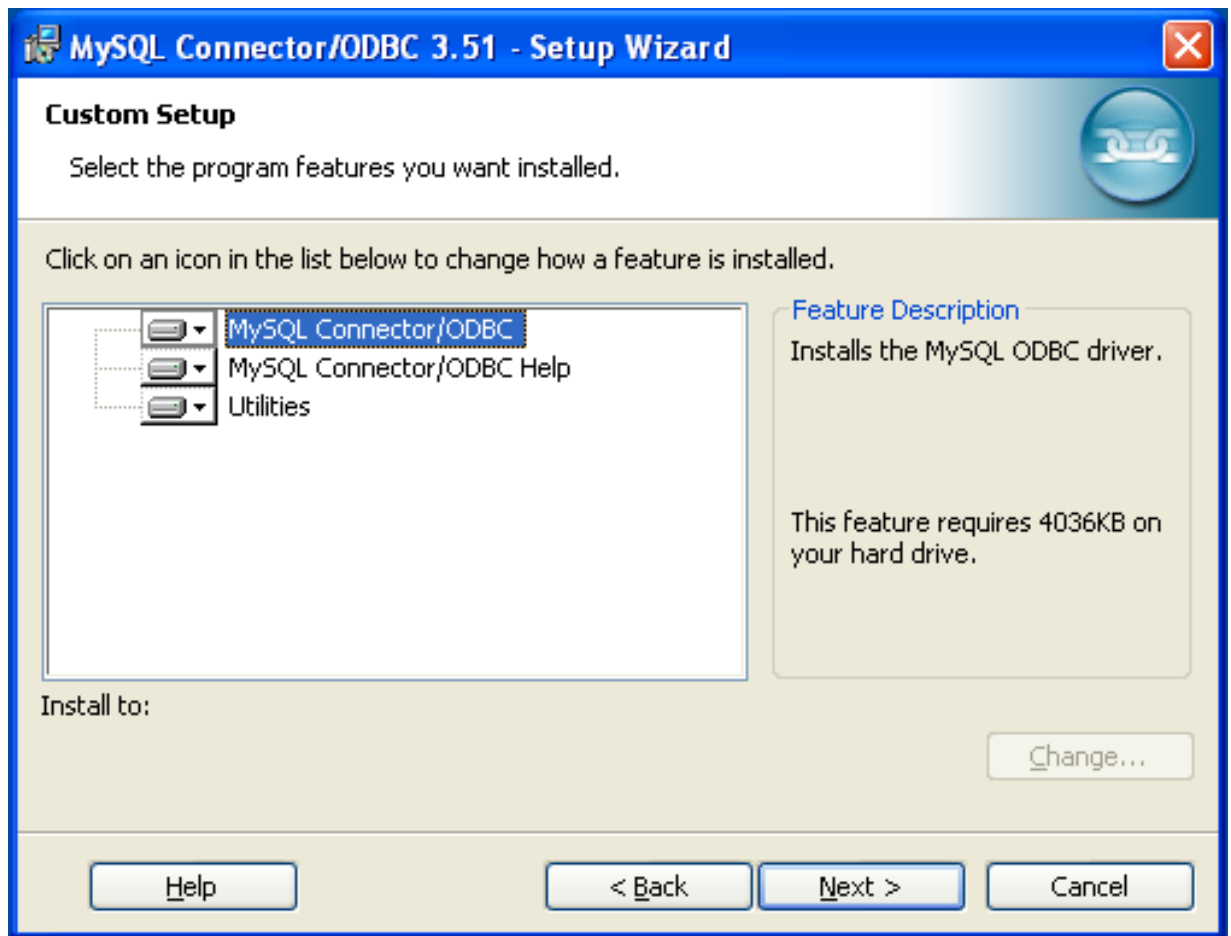


3. You will need to choose the installation type. The Typical installation provides the standard files you will need to connect to a MySQL database using ODBC. The Complete option installs all the available files, including debug and utility components. It is recommended you choose one of these two options to complete the installation. If choose one of these methods, click NEXT and then proceed to step 5.

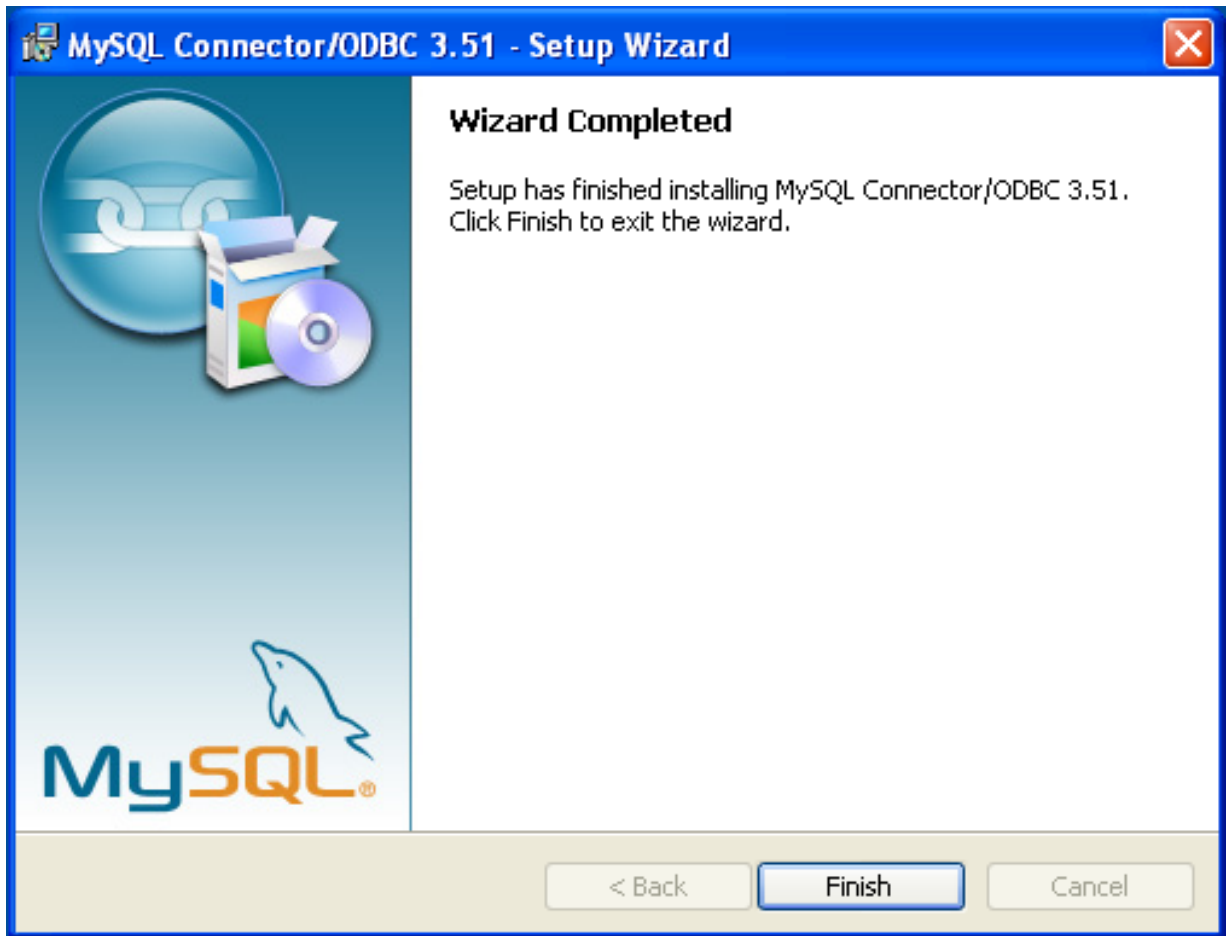
You may also choose a Custom installation, which enables you to select the individual components that you want to install. You have chosen this method, click NEXT and then proceed to step 4.



4. If you have chosen a custom installation, use the pop-ups to select which components to install and then click NEXT to install the necessary files.



5. Once the files have copied to your machine, the installation is complete. Click FINISH to exit the installer.



Now the installation is complete, you can continue to configure your ODBC connections using [Chapter 3, Connector/ODBC Configuration](#).

## 2.1.2. Installing the Windows Connector/ODBC Driver using the Zipped DLL package

If you have downloaded the Zipped DLL package then you must install the individual files required for Connector/ODBC operation manually. Once you have unzipped the installation files, you can either perform this operation by hand, executing each statement individually, or you can use the included Batch file to perform an installation to the default locations.

To install using the Batch file:

1. Unzip the Connector/ODBC Zipped DLL package.
2. Open a Command Prompt.
3. Change to the directory created when you unzipped the Connector/ODBC Zipped DLL package.
4. Run `Install.bat`:

```
C:\> Install.bat
```

This will copy the necessary files into the default location, and then register the Connector/ODBC driver with the Windows ODBC manager.

If you want to copy the files to an alternative location - for example, to run or test different versions of the Connector/ODBC driver on the same machine, then you must copy the files by hand. It is however not recommended to install these files in a non-standard location. To copy the files by hand to the default installation location use the following steps:

1. Unzip the Connector/ODBC Zipped DLL package.
2. Open a Command Prompt.
3. Change to the directory created when you unzipped the Connector/ODBC Zipped DLL package.
4. Copy the library files to a suitable directory. The default is to copy them into the default Windows system directory `\Windows\System32`:

```
C:\> copy lib\myodbc3S.dll \Windows\System32
C:\> copy lib\myodbc3S.lib \Windows\System32
C:\> copy lib\myodbc3.dll \Windows\System32
C:\> copy lib\myodbc3.lib \Windows\System32
```

5. Copy the Connector/ODBC tools. These must be placed into a directory that is in the system `PATH`. The default is to install these into the Windows system directory `\Windows\System32`:

```
C:\> copy bin\myodbc3i.exe \Windows\System32
C:\> copy bin\myodbc3m.exe \Windows\System32
C:\> copy bin\myodbc3c.exe \Windows\System32
```

6. Optionally copy the help files. For these files to be accessible through the help system, they must be installed in the Windows system directory:

```
C:\> copy doc\*.hlp \Windows\System32
```

7. Finally, you must register the Connector/ODBC driver with the ODBC manager:

```
C:\> myodbc3i -a -d -t"MySQL ODBC 3.51 Driver;\
DRIVER=myodbc3.dll;SETUP=myodbc3S.dll"
```

You must change the references to the DLL files and command location in the above statement if you have not installed these files into the default location.

## 2.2. Installing Connector/ODBC from a Binary Distribution on Unix

There are two methods available for installing Connector/ODBC on Unix from a binary distribution. For most Unix environments you will need to use the tarball distribution. For Linux systems, there is also an RPM distribution available.

### Note

To install Connector/ODBC 5.1 on Unix you require unixODBC 2.2.12 or later to be installed.

### 2.2.1. Installing Connector/ODBC from a Binary Tarball Distribution

To install the driver from a tarball distribution (`.tar.gz` file), download the latest version of the driver for your operating system and follow these steps that demonstrate the process using the Linux version of the tarball:

```
shell> su root
shell> gunzip mysql-connector-odbc-3.51.11-i686-pc-linux.tar.gz
shell> tar xvf mysql-connector-odbc-3.51.11-i686-pc-linux.tar
shell> cd mysql-connector-odbc-3.51.11-i686-pc-linux
```

Read the installation instructions in the `INSTALL` file and execute these commands.

Then proceed on to [Section 3.5, “Configuring a Connector/ODBC DSN on Unix”](#), to configure the DSN for Connector/ODBC. For more information, refer to the `INSTALL` file that comes with your distribution.

### 2.2.2. Installing Connector/ODBC from an RPM Distribution

To install or upgrade Connector/ODBC from an RPM distribution on Linux, simply download the RPM distribution of the latest version of Connector/ODBC and follow the instructions below. Use `su root` to become `root`, then install the RPM file.

If you are installing for the first time:

```
shell> su root
shell> rpm -ivh mysql-connector-odbc-3.51.12.i386.rpm
```

If the driver exists, upgrade it like this:

```
shell> su root
shell> rpm -Uvh mysql-connector-odbc-3.51.12.i386.rpm
```

If there is any dependency error for MySQL client library, `libmysqlclient`, simply ignore it by supplying the `--nodeps` option, and then make sure the MySQL client shared library is in the path or set through `LD_LIBRARY_PATH`.

This installs the driver libraries and related documents to `/usr/local/lib` and `/usr/share/doc/MyODBC`, respectively. Proceed onto [Section 3.5, “Configuring a Connector/ODBC DSN on Unix”](#).

To **uninstall** the driver, become `root` and execute an `rpm` command:

```
shell> su root
shell> rpm -e mysql-connector-odbc
```

## 2.3. Installing Connector/ODBC from a Binary Distribution on Mac OS X

Mac OS X is based on the FreeBSD operating system, and you can normally use the MySQL network port for connecting to MySQL servers on other hosts. Installing the Connector/ODBC driver enables you to connect to MySQL databases on any platform through the ODBC interface. You should only need to install the Connector/ODBC driver when your application requires an ODBC interface. Applications that require or can use ODBC (and therefore the Connector/ODBC driver) include ColdFusion, Filemaker Pro, 4th Dimension and many other applications.

Mac OS X includes its own ODBC manager, based on the `iODBC` manager. Mac OS X includes an administration tool that provides easier administration of ODBC drivers and configuration, updating the underlying `iODBC` configuration files.

The method for installing Connector/ODBC on Mac OS X depends on the version on Connector/ODBC you are using. For Connector/ODBC 3.51.14 and later, the package is provided as a compressed tar archive that you must manually install. For Connector/ODBC 3.51.13 and earlier the software was provided on a compressed disk image (`.dmg`) file and included an installer.

In either case, the driver is designed to work with the `iODBC` driver manager included with Mac OS X.

To install Connector/ODBC 3.51.14 and later:

1. Download the installation file. Note that versions are available for both PowerPC and Intel platforms.
2. Extract the archive:

```
shell> tar zxf mysql-connector-odbc-3.51.16-osx10.4-x86-32bit.tar.gz
```

3. The directory created will contain two subdirectories, `lib` and `bin`. You need to copy these to a suitable location such as `/usr/local`:

```
shell> cp bin/* /usr/local/bin
shell> cp lib/* /usr/local/lib
```

4. Finally, you must register the driver with `iODBC` using the `myodbc3i` tool you just installed:

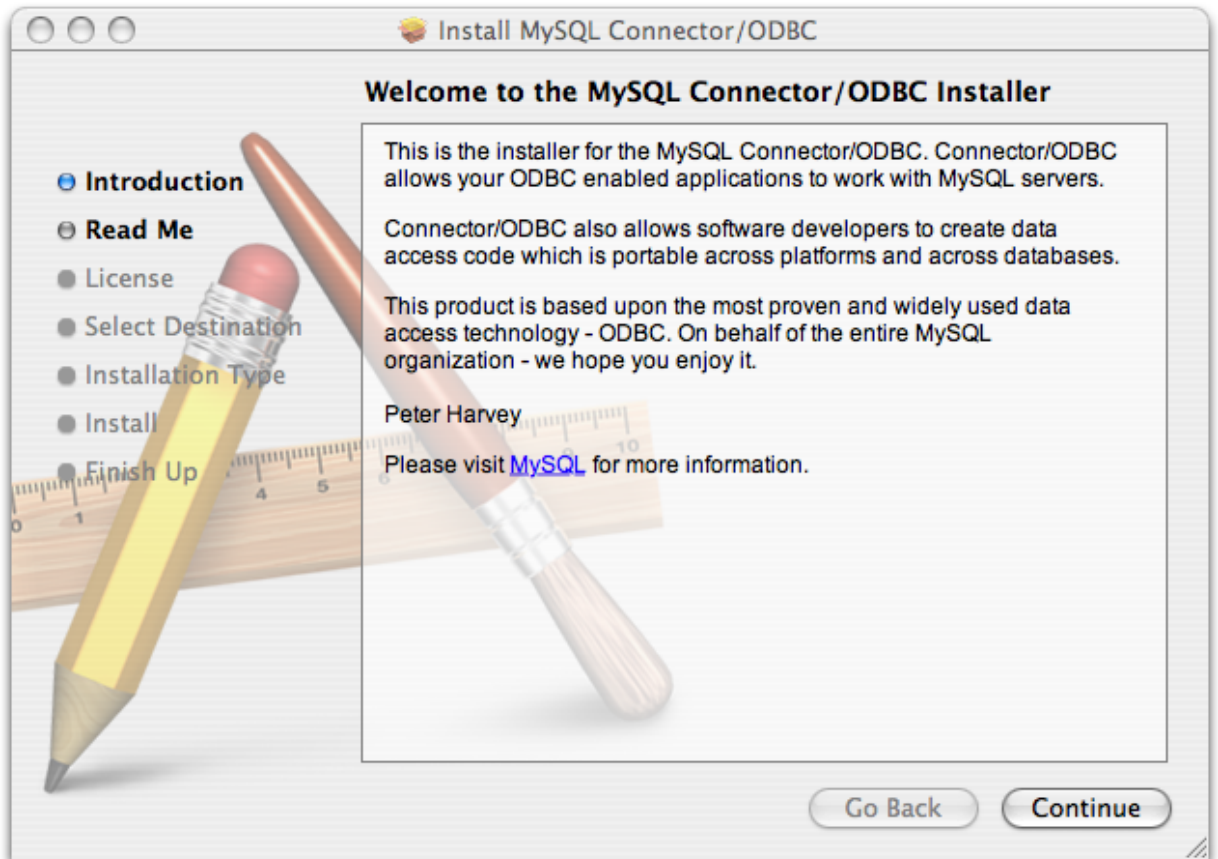
```
shell> myodbc3i -a -d -t"MySQL ODBC 3.51 Driver;Driver=/usr/local/lib/libmyodbc3.so;Setup=/usr/local/lib/libmyodbc3i"
```

You can verify the installed drivers either by using the ODBC Administrator application or the `myodbc3i` utility:

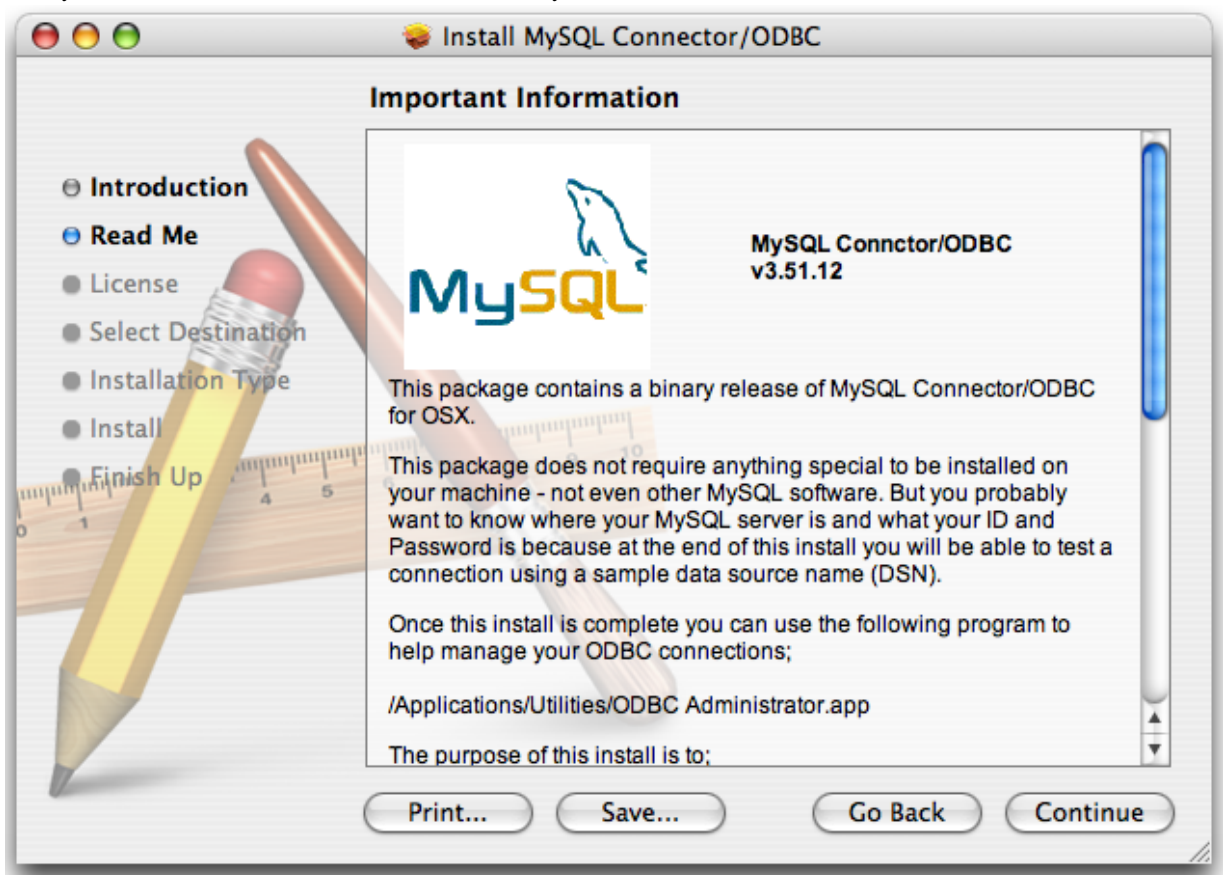
```
shell> myodbc3i -q -d
```

To install Connector/ODBC 3.51.13 and earlier, follow these steps:

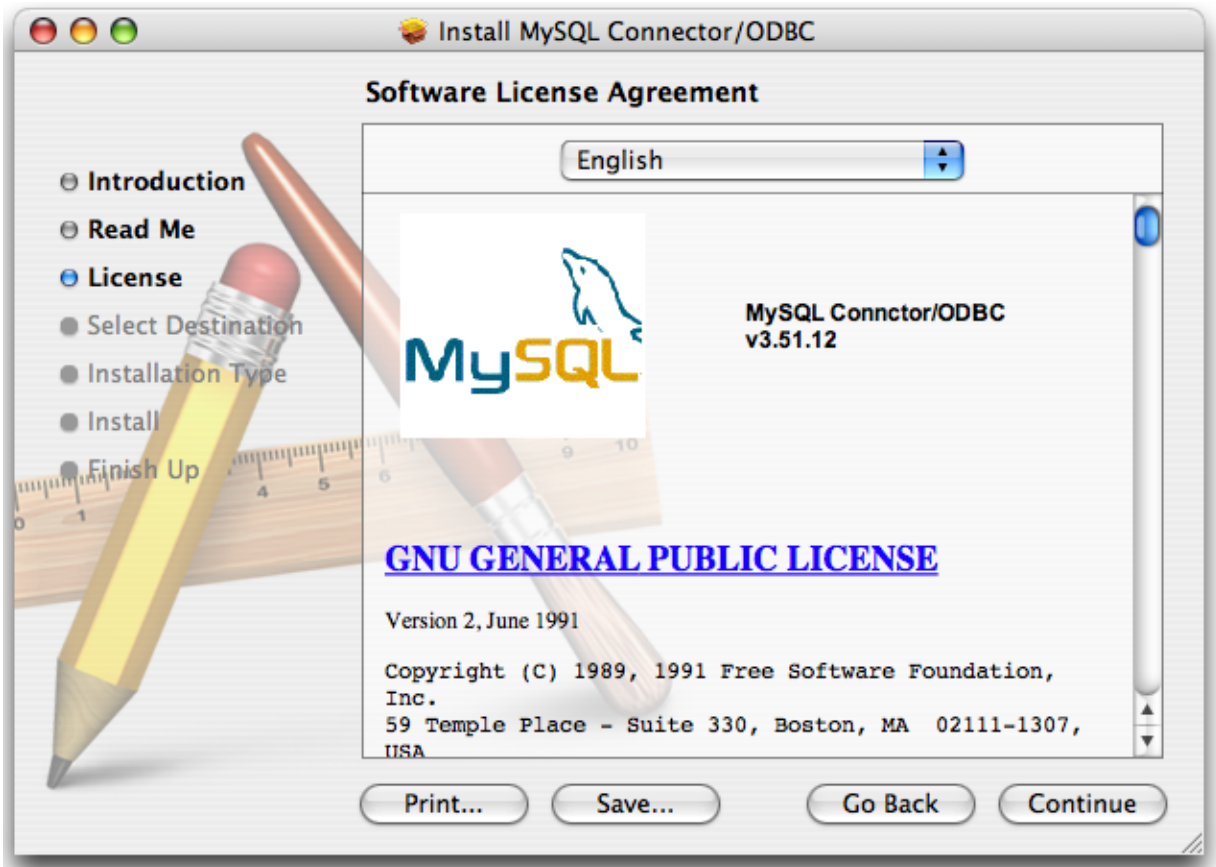
1. Download the file to your computer and double-click on the downloaded image file.
2. Within the disk image you will find an installer package (with the `.pkg` extension). Double click on this file to start the Mac OS X installer.
3. You will be presented with the installer welcome message. Click the CONTINUE button to begin the installation process.



4. Please take the time to read the Important Information as it contains guidance on how to complete the installation process. Once you have read the notice and collected the necessary information, click CONTINUE.

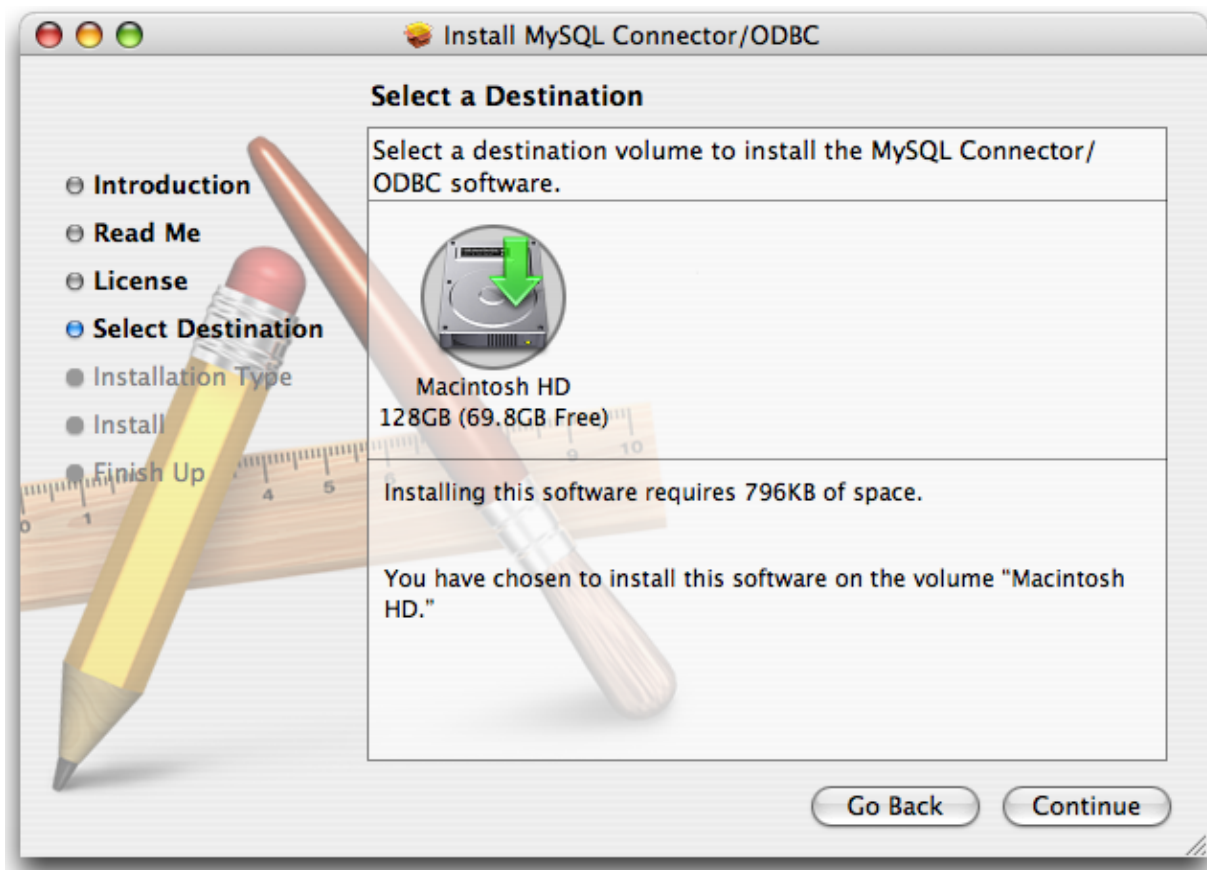


- Connector/ODBC drivers are made available under the GNU General Public License. Please read the license if you are not familiar with it before continuing installation. Click CONTINUE to approve the license (you will be asked to confirm that decision) and continue the installation.

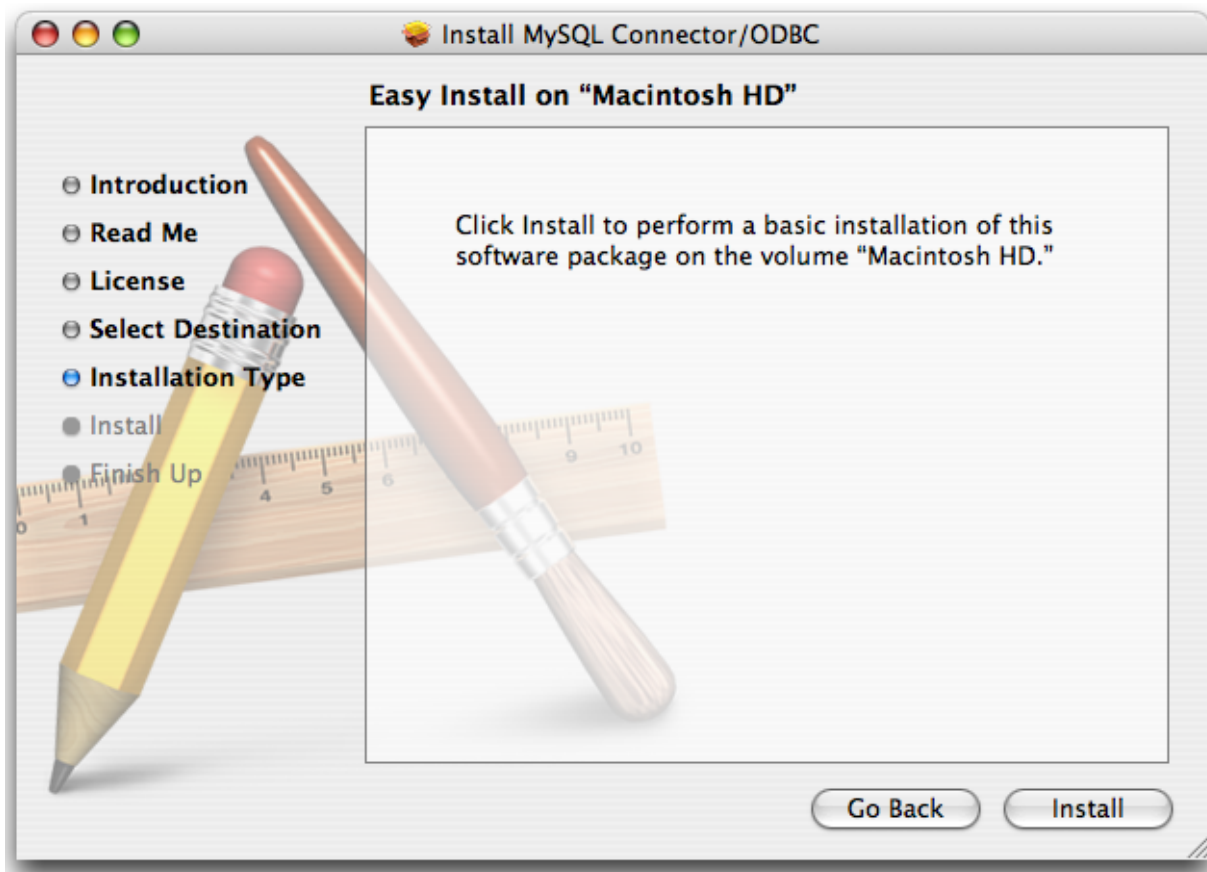


- Choose a location to install the Connector/ODBC drivers and the ODBC Administrator application. You must install the files onto a drive with an operating system and you may be limited in the choices available. Select the drive you want to use, and then click CONTINUE.

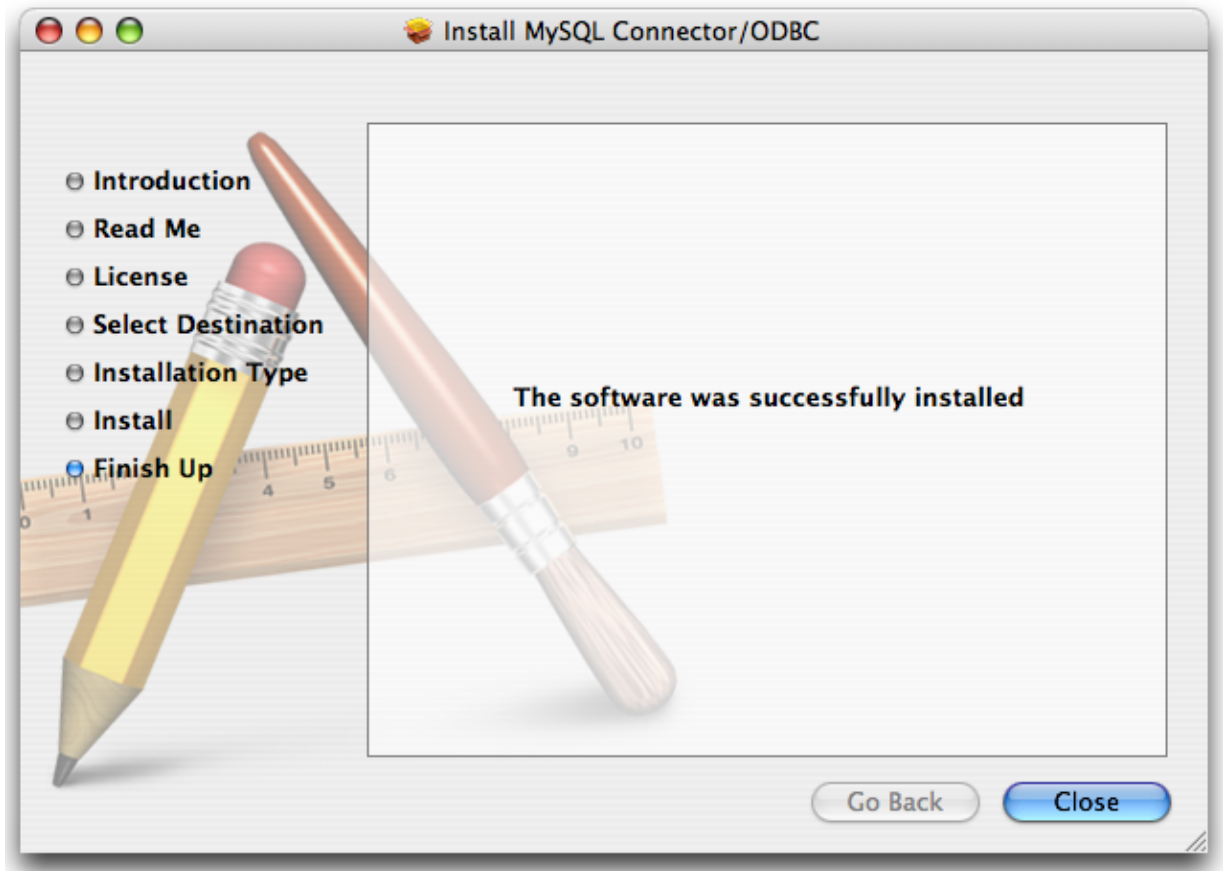




7. The installer will automatically select the files that need to be installed on your machine. Click **INSTALL** to continue. The installer will copy the necessary files to your machine. A progress bar will be shown indicating the installation progress.



8. When installation has been completed you will get a window like the one shown below. Click CLOSE to close and quit the installer.



## 2.4. Installing Connector/ODBC from a Source Distribution on Windows

You should only need to install Connector/ODBC from source on Windows if you want to change or modify the source or installation. If you are unsure whether to install from source, please use the binary installation detailed in [Section 2.1, “Installing Connector/ODBC from a Binary Distribution on Windows”](#).

Installing Connector/ODBC from source on Windows requires a number of different tools and packages:

- MDAC, Microsoft Data Access SDK from <http://support.microsoft.com/kb/110093>.
- Suitable C compiler, such as Microsoft Visual C++ or the C compiler included with Microsoft Visual Studio.
- Compatible `make` tool. Microsoft's `nmake` is used in the examples in this section.
- MySQL client libraries and include files from MySQL 4.0.0 or higher. (Preferably MySQL 4.0.16 or higher). This is required because Connector/ODBC uses new calls and structures that exist only starting from this version of the library. To get the client libraries and include files, visit <http://dev.mysql.com/downloads/>.

### 2.4.1. Building Connector/ODBC 3.51

Connector/ODBC source distributions include `Makefiles` that require the `nmake` or other `make` utility. In the distribution, you can find `Makefile` for building the release version and `Makefile_debug` for building debugging versions of the driver libraries and DLLs.

To build the driver, use this procedure:

1. Download and extract the sources to a folder, then change directory into that folder. The following command assumes the

folder is named `myodbc3-src`:

```
C:\> cd myodbc3-src
```

2. Edit `Makefile` to specify the correct path for the MySQL client libraries and header files. Then use the following commands to build and install the release version:

```
C:\> nmake -f Makefile
C:\> nmake -f Makefile install
```

`nmake -f Makefile` builds the release version of the driver and places the binaries in subdirectory called `Release`.

`nmake -f Makefile install` installs (copies) the driver DLLs and libraries (`myodbc3.dll`, `myodbc3.lib`) to your system directory.

3. To build the debug version, use `Makefile_Debug` rather than `Makefile`, as shown below:

```
C:\> nmake -f Makefile_debug
C:\> nmake -f Makefile_debug install
```

4. You can clean and rebuild the driver by using:

```
C:\> nmake -f Makefile clean
C:\> nmake -f Makefile install
```

### Note

- Make sure to specify the correct MySQL client libraries and header files path in the Makefiles (set the `MYSQL_LIB_PATH` and `MYSQL_INCLUDE_PATH` variables). The default header file path is assumed to be `C:\mysql\include`. The default library path is assumed to be `C:\mysql\lib\opt` for release DLLs and `C:\mysql\lib\debug` for debug versions.
- For the complete usage of `nmake`, visit [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv\\_vcce4/html/evgrfRunningNMAKE.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vcce4/html/evgrfRunningNMAKE.asp).
- If you are using the Subversion tree for compiling, all Windows-specific `Makefiles` are named as `Win_Makefile*`.

## 2.4.2. Testing

After the driver libraries are copied/installed to the system directory, you can test whether the libraries are properly built by using the samples provided in the `samples` subdirectory:

```
C:\> cd samples
C:\> nmake -f Makefile all
```

## 2.5. Installing Connector/ODBC from a Source Distribution on Unix

You need the following tools to build MySQL from source on Unix:

- A working ANSI C++ compiler. `gcc` 2.95.2 or later, SGI C++, and SunPro C++ are some of the compilers that are known to work.
- A good `make` program. GNU `make` is always recommended and is sometimes required.
- MySQL client libraries and include files from MySQL 4.0.0 or higher. (Preferably MySQL 4.0.16 or higher). This is required because Connector/ODBC uses new calls and structures that exist only starting from this version of the library. To get the client libraries and include files, visit <http://dev.mysql.com/downloads/>.

If you have built your own MySQL server and/or client libraries from source then you must have used the `-enable-thread-safe-client` option to `configure` when the libraries were built.

You should also ensure that the `libmysqlclient` library were built and installed as a shared library.

- A compatible ODBC manager must be installed. Connector/ODBC is known to work with the `iODBC` and `unixODBC` man-

agers. See [Section 1.2.2, “ODBC Driver Managers”](#), for more information.

- If you are using a character set that isn't compiled into the MySQL client library then you need to install the MySQL character definitions from the `charsets` directory into `$SHAREDIR` (by default, `/usr/local/mysql/share/mysql/charsets`). These should be in place if you have installed the MySQL server on the same machine. See [Character Set Support](#), for more information on character set support.

Once you have all the required files, unpack the source files to a separate directory, you then have to run `configure` and build the library using `make`.

## 2.5.1. Typical `configure` Options

The `configure` script gives you a great deal of control over how you configure your Connector/ODBC build. Typically you do this using options on the `configure` command line. You can also affect `configure` using certain environment variables. For a list of options and environment variables supported by `configure`, run this command:

```
shell> ./configure --help
```

Some of the more commonly used `configure` options are described here:

1. To compile Connector/ODBC, you need to supply the MySQL client include and library files path using the `--with-mysql-path=DIR` option, where `DIR` is the directory where MySQL is installed.

MySQL compile options can be determined by running `DIR/bin/mysql_config`.

2. Supply the standard header and library files path for your ODBC Driver Manager (`iODBC` or `unixODBC`).

- If you are using `iODBC` and `iODBC` is not installed in its default location (`/usr/local`), you might have to use the `--with-iodbc=DIR` option, where `DIR` is the directory where `iODBC` is installed.

If the `iODBC` headers do not reside in `DIR/include`, you can use the `--with-iodbc-includes=INCDIR` option to specify their location.

The applies to libraries. If they are not in `DIR/lib`, you can use the `--with-iodbc-libs=LIBDIR` option.

- If you are using `unixODBC`, use the `--with-unixODBC=DIR` option (case sensitive) to make `configure` look for `unixODBC` instead of `iODBC` by default, `DIR` is the directory where `unixODBC` is installed.

If the `unixODBC` headers and libraries aren't located in `DIR/include` and `DIR/lib`, use the `--with-unixODBC-includes=INCDIR` and `--with-unixODBC-libs=LIBDIR` options.

3. You might want to specify an installation prefix other than `/usr/local`. For example, to install the Connector/ODBC drivers in `/usr/local/odbc/lib`, use the `--prefix=/usr/local/odbc` option.

The final configuration command looks something like this:

```
shell> ./configure --prefix=/usr/local \
  --with-iodbc=/usr/local \
  --with-mysql-path=/usr/local/mysql
```

## 2.5.2. Additional `configure` Options

There are a number of other options that you need, or want, to set when configuring the Connector/ODBC driver before it is built.

- To link the driver with MySQL thread safe client libraries `libmysqlclient_r.so` or `libmysqlclient_r.a`, you must specify the following `configure` option:

```
--enable-thread-safe
```

and can be disabled (default) using

```
--disable-thread-safe
```

This option enables the building of the driver thread-safe library `libmyodbc3_r.so` from by linking with MySQL thread-safe client library `libmysqlclient_r.so` (The extensions are OS dependent).

If the compilation with the thread-safe option fails, it may be because the correct thread-libraries on the system could not be located. You should set the value of `LIBS` to point to the correct thread library for your system.

```
LIBS="-lpthread" ./configure ..
```

- You can enable or disable the shared and static versions of Connector/ODBC using these options:

```
--enable-shared[=yes/no]
--disable-shared
--enable-static[=yes/no]
--disable-static
```

- By default, all the binary distributions are built as non-debugging versions (configured with `--without-debug`).

To enable debugging information, build the driver from source distribution and use the `--with-debug` option when you run `configure`.

- This option is available only for source trees that have been obtained from the Subversion repository. This option does not apply to the packaged source distributions.

By default, the driver is built with the `--without-docs` option. If you would like the documentation to be built, then execute `configure` with:

```
--with-docs
```

## 2.5.3. Building and Compilation

To build the driver libraries, you have to just execute `make`.

```
shell> make
```

If any errors occur, correct them and continue the build process. If you aren't able to build, then send a detailed email to [<myodbc@lists.mysql.com>](mailto:myodbc@lists.mysql.com) for further assistance.

## 2.5.4. Building Shared Libraries

On most platforms, MySQL does not build or support `.so` (shared) client libraries by default. This is based on our experience of problems when building shared libraries.

In cases like this, you have to download the MySQL distribution and configure it with these options:

```
--without-server --enable-shared
```

To build shared driver libraries, you must specify the `--enable-shared` option for `configure`. By default, `configure` does not enable this option.

If you have configured with the `--disable-shared` option, you can build the `.so` file from the static libraries using the following commands:

```
shell> cd mysql-connector-odbc-3.51.01
shell> make
shell> cd driver
shell> CC=/usr/bin/gcc \
    $CC -bundle -flat_namespace -undefined error \
    -o .libs/libmyodbc3-3.51.01.so \
    catalog.o connect.o cursor.o dll.o error.o execute.o \
    handle.o info.o misc.o myodbc3.o options.o prepare.o \
    results.o transact.o utility.o \
    -L/usr/local/mysql/lib/mysql/ \
    -L/usr/local/iodbc/lib/ \
    -lz -lc -lmysqlclient -liodbcinst
```

Make sure to change `-liodbcinst` to `-lodbcinst` if you are using `unixODBC` instead of `iODBC`, and configure the library paths accordingly.

This builds and places the `libmyodbc3-3.51.01.so` file in the `.libs` directory. Copy this file to the Connector/ODBC library installation directory (`/usr/local/lib` (or the `lib` directory under the installation directory that you supplied with the `-prefix`)).

```
shell> cd .libs
```

```
shell> cp libmyodbc3-3.51.01.so /usr/local/lib
shell> cd /usr/local/lib
shell> ln -s libmyodbc3-3.51.01.so libmyodbc3.so
```

To build the thread-safe driver library:

```
shell> CC=/usr/bin/gcc \
$CC -bundle -flat_namespace -undefined error
-o .libs/libmyodbc3_r-3.51.01.so
catalog.o connect.o cursor.o dll.o error.o execute.o
handle.o info.o misc.o myodbc3.o options.o prepare.o
results.o transact.o utility.o
-L/usr/local/mysql/lib/mysql/
-L/usr/local/iodbc/lib/
-lz -lc -lmysqlclient_r -liodbcinst
```

## 2.5.5. Installing Driver Libraries

To install the driver libraries, execute the following command:

```
shell> make install
```

That command installs one of the following sets of libraries:

For Connector/ODBC 3.51:

- `libmyodbc3.so`
- `libmyodbc3-3.51.01.so`, where 3.51.01 is the version of the driver
- `libmyodbc3.a`

For thread-safe Connector/ODBC 3.51:

- `libmyodbc3_r.so`
- `libmyodbc3-3_r.51.01.so`
- `libmyodbc3_r.a`

For more information on build process, refer to the `INSTALL` file that comes with the source distribution. Note that if you are trying to use the `make` from Sun, you may end up with errors. On the other hand, GNU `gmake` should work fine on all platforms.

## 2.5.6. Testing Connector/ODBC on Unix

To run the basic samples provided in the distribution with the libraries that you built, use the following command:

```
shell> make test
```

Before running the tests, create the DSN 'myodbc3' in `odbc.ini` and set the environment variable `ODBCINI` to the correct `odbc.ini` file; and MySQL server is running. You can find a sample `odbc.ini` with the driver distribution.

You can even modify the `samples/run-samples` script to pass the desired DSN, UID, and PASSWORD values as the command-line arguments to each sample.

## 2.5.7. Building Connector/ODBC from Source on Mac OS X

To build the driver on Mac OS X (Darwin), make use of the following `configure` example:

```
shell> ./configure --prefix=/usr/local
--with-unixODBC=/usr/local
--with-mysql-path=/usr/local/mysql
--disable-shared
--enable-gui=no
--host=powerpc-apple
```

The command assumes that the `unixODBC` and MySQL are installed in the default locations. If not, configure accordingly.

On Mac OS X, `--enable-shared` builds `.dylib` files by default. You can build `.so` files like this:

```
shell> make
shell> cd driver
shell> CC=/usr/bin/gcc \
    $CC -bundle -flat_namespace -undefined error
    -o .libs/libmyodbc3-3.51.01.so *.o
    -L/usr/local/mysql/lib/
    -L/usr/local/iodbc/lib
    -liodbcinst -lmysqlclient -lz -lc
```

To build the thread-safe driver library:

```
shell> CC=/usr/bin/gcc \
    $CC -bundle -flat_namespace -undefined error
    -o .libs/libmyodbc3-3.51.01.so *.o
    -L/usr/local/mysql/lib/
    -L/usr/local/iodbc/lib
    -liodbcinst -lmysqlclient_r -lz -lc -lpthread
```

Make sure to change the `-liodbcinst` to `-lodbcinst` in case of using `unixODBC` instead of `iODBC` and configure the libraries path accordingly.

In Apple's version of GCC, both `cc` and `gcc` are actually symbolic links to `gcc3`.

Copy this library to the `$prefix/lib` directory and symlink to `libmyodbc3.so`.

You can cross-check the output shared-library properties using this command:

```
shell> otool -LD .libs/libmyodbc3-3.51.01.so
```

## 2.5.8. Building Connector/ODBC from Source on HP-UX

To build the driver on HP-UX 10.x or 11.x, make use of the following `configure` example:

If using `cc`:

```
shell> CC="cc" \
    CFLAGS="+z" \
    LDFLAGS="-Wl,+b:-Wl,+s" \
    ./configure --prefix=/usr/local
    --with-unixodbc=/usr/local
    --with-mysql-path=/usr/local/mysql/lib/mysql
    --enable-shared
    --enable-thread-safe
```

If using `gcc`:

```
shell> CC="gcc" \
    LDFLAGS="-Wl,+b:-Wl,+s" \
    ./configure --prefix=/usr/local
    --with-unixodbc=/usr/local
    --with-mysql-path=/usr/local/mysql
    --enable-shared
    --enable-thread-safe
```

Once the driver is built, cross-check its attributes using `chatr .libs/libmyodbc3.sl` to determine whether you need to have set the MySQL client library path using the `SHLIB_PATH` environment variable. For static versions, ignore all shared-library options and run `configure` with the `--disable-shared` option.

## 2.5.9. Building Connector/ODBC from Source on AIX

To build the driver on AIX, make use of the following `configure` example:

```
shell> ./configure --prefix=/usr/local
    --with-unixodbc=/usr/local
    --with-mysql-path=/usr/local/mysql
    --disable-shared
    --enable-thread-safe
```

### Note

For more information about how to build and set up the static and shared libraries across the different platforms refer to 'Using static and shared libraries across platforms'.

## 2.6. Installing Connector/ODBC from the Development Source Tree

**Caution**

You should read this section only if you are interested in helping us test our new code. If you just want to get MySQL Connector/ODBC up and running on your system, you should use a standard release distribution.

To be able to access the Connector/ODBC source tree, you must have Subversion installed. Subversion is freely available from <http://subversion.tigris.org/>.

To build from the source trees, you need the following tools:

- autoconf 2.52 (or newer)
- automake 1.4 (or newer)
- libtool 1.4 (or newer)
- m4

The most recent development source tree is available from our public Subversion trees at <http://dev.mysql.com/tech-resources/sources.html>.

To checkout out the Connector/ODBC sources, change to the directory where you want the copy of the Connector/ODBC tree to be stored, then use the following command:

```
shell> svn co http://svn.mysql.com/svnpublic/connector-odbc3
```

You should now have a copy of the entire Connector/ODBC source tree in the directory `connector-odbc3`. To build from this source tree on Unix or Linux follow these steps:

```
shell> cd connector-odbc3
shell> aclocal
shell> autoheader
shell> autoconf
shell> automake;
shell> ./configure # Add your favorite options here
shell> make
```

For more information on how to build, refer to the `INSTALL` file located in the same directory. For more information on options to `configure`, see [Section 2.5.1, “Typical configure Options”](#)

When the build is done, run `make install` to install the Connector/ODBC 3.51 driver on your system.

If you have gotten to the `make` stage and the distribution does not compile, please report it to [<myodbc@lists.mysql.com>](mailto:myodbc@lists.mysql.com).

On Windows, make use of Windows Makefiles `WIN-Makefile` and `WIN-Makefile_debug` in building the driver. For more information, see [Section 2.4, “Installing Connector/ODBC from a Source Distribution on Windows”](#).

After the initial checkout operation to get the source tree, you should run `svn update` periodically update your source according to the latest version.



---

## Chapter 3. Connector/ODBC Configuration

Before you connect to a MySQL database using the Connector/ODBC driver you must configure an ODBC *Data Source Name*. The DSN associates the various configuration parameters required to communicate with a database to a specific name. You use the DSN in an application to communicate with the database, rather than specifying individual parameters within the application itself. DSN information can be user specific, system specific, or provided in a special file. ODBC data source names are configured in different ways, depending on your platform and ODBC driver.

### 3.1. Data Source Names

A Data Source Name associates the configuration parameters for communicating with a specific database. Generally a DSN consists of the following parameters:

- Name
- Host Name
- Database Name
- Login
- Password

In addition, different ODBC drivers, including Connector/ODBC, may accept additional driver-specific options and parameters.

There are three types of DSN:

- A *System DSN* is a global DSN definition that is available to any user and application on a particular system. A System DSN can normally only be configured by a systems administrator, or by a user who has specific permissions that let them create System DSNs.
- A *User DSN* is specific to an individual user, and can be used to store database connectivity information that the user regularly uses.
- A *File DSN* uses a simple file to define the DSN configuration. File DSNs can be shared between users and machines and are therefore more practical when installing or deploying DSN information as part of an application across many machines.

DSN information is stored in different locations depending on your platform and environment.

### 3.2. Connector/ODBC Connection Parameters

You can specify the parameters in the following tables for Connector/ODBC when configuring a DSN. Users on Windows can use the Options and Advanced panels when configuring a DSN to set these parameters; see the table for information on which options relate to which fields and checkboxes. On Unix and Mac OS X, use the parameter name and value as the keyword/value pair in the DSN configuration. Alternatively, you can set these parameters within the `InConnectionString` argument in the `SQLDriverConnect()` call.

Parameter	Default Value	Comment
<code>user</code>	ODBC	The user name used to connect to MySQL.
<code>uid</code>	ODBC	Synonymous with <code>user</code> . Added in 3.51.16.
<code>server</code>	<code>localhost</code>	The host name of the MySQL server.
<code>database</code>		The default database.
<code>option</code>	0	Options that specify how Connector/ODBC should work. See below.
<code>port</code>	3306	The TCP/IP port to use if <code>server</code> is not <code>localhost</code> .
<code>initstmt</code>		Initial statement. A statement to execute when connecting to MySQL. In version 3.51 the parameter is called <code>stmt</code> . Note, the driver supports the initial statement being executed only at the time of the initial connection.
<code>password</code>		The password for the <code>user</code> account on <code>server</code> .
<code>pwd</code>		Synonymous with <code>password</code> . Added in 3.51.16.
<code>socket</code>		The Unix socket file or Windows named pipe to connect to if <code>server</code> is <code>local-</code>

Parameter	Default Value	Comment
		<code>host</code> .
<code>sslca</code>		The path to a file with a list of trust SSL CAs. Added in 3.51.16.
<code>sslcapath</code>		The path to a directory that contains trusted SSL CA certificates in PEM format. Added in 3.51.16.
<code>sslcert</code>		The name of the SSL certificate file to use for establishing a secure connection. Added in 3.51.16.
<code>sslcipher</code>		A list of allowable ciphers to use for SSL encryption. The cipher list has the same format as the <code>openssl ciphers</code> command. Added in 3.51.16.
<code>sslkey</code>		The name of the SSL key file to use for establishing a secure connection. Added in 3.51.16.
<code>charset</code>		The character set to use for the connection. Added in 3.51.17.
<code>sslverify</code>		If set to 1, the SSL certificate will be verified when used with the MySQL connection. If not set, then the default behaviour is to ignore SSL certificate verification.
<code>readtimeout</code>		The timeout in seconds for attempts to read from the server. Each attempt uses this timeout value and there are retries if necessary, so the total effective timeout value is three times the option value. You can set the value so that a lost connection can be detected earlier than the TCP/IP <code>Close_Wait_Timeout</code> value of 10 minutes. This option works only for TCP/IP connections, and only for Windows prior to MySQL 5.1.12. Corresponds to the <code>MYSQL_OPT_READ_TIMEOUT</code> option of the MySQL Client Library. This option was added in Connector/ODBC 3.51.27.
<code>writetimeout</code>		The timeout in seconds for attempts to write to the server. Each attempt uses this timeout value and there are retries if necessary, so the total effective timeout value is <code>net_retry_count</code> times the option value. This option works only for TCP/IP connections, and only for Windows prior to MySQL 5.1.12. Corresponds to the <code>MYSQL_OPT_WRITE_TIMEOUT</code> option of the MySQL Client Library. This option was added in Connector/ODBC 3.51.27.

## Note

The SSL configuration parameters can also be automatically loaded from a `my.ini` or `my.cnf` file.

The `option` argument is used to tell Connector/ODBC that the client isn't 100% ODBC compliant. On Windows, you normally select options by toggling the checkboxes in the connection screen, but you can also select them in the `option` argument. The following options are listed in the order in which they appear in the Connector/ODBC connect screen.

Value	Flagname	GUI Option	Description
1	<code>FLAG_FIELD_LENGTH</code>	Do not Optimize Column Width	The client cannot handle that Connector/ODBC returns the real width of a column. This option was removed in 3.51.18.
2	<code>FLAG_FOUND_ROWS</code>	Return Matching Rows	The client cannot handle that MySQL returns the true value of affected rows. If this flag is set, MySQL returns "found rows" instead. You must have MySQL 3.21.14 or newer to get this to work.
4	<code>FLAG_DEBUG</code>	Trace Driver Calls To myodbc.log	Make a debug log in <code>C:\myodbc.log</code> on Windows, or <code>/tmp/myodbc.log</code> on Unix variants. This option was removed in Connector/ODBC 3.51.18.
8	<code>FLAG_BIG_PACKETS</code>	Allow Big Results	Do not set any packet limit for results and bind parameters. Without this option, parameter binding will be truncated to 255 characters.
16	<code>FLAG_NO_PROMPT</code>	Do not Prompt Upon Connect	Do not prompt for questions even if driver would like to prompt.
32	<code>FLAG_DYNAMIC_CURSOR</code>	Enable Dynamic Cursor	Enable or disable the dynamic cursor support.
64	<code>FLAG_NO_SCHEMA</code>	Ignore # in Table Name	Ignore use of database name in <code>db_name.tbl_name.col_name</code> .
128	<code>FLAG_NO_DEFAULT_CURSOR</code>	User Manager Cursors	Force use of ODBC manager cursors (experimental).
256	<code>FLAG_NO_LOCALE</code>	Do not Use Set Locale	Disable the use of extended fetch (experimental).
512	<code>FLAG_PAD_SPACE</code>	Pad Char To Full Length	Pad <code>CHAR</code> columns to full column length.

1024	FLAG_FULL_COLUMN_NAMES	Return Table Names for SQLDescribeCol	SQLDescribeCol() returns fully qualified column names.
2048	FLAG_COMPRESSED_PROTO	Use Compressed Protocol	Use the compressed client/server protocol.
4096	FLAG_IGNORE_SPACE	Ignore Space After Function Names	Tell server to ignore space after function name and before "(" (needed by PowerBuilder). This makes all function names keywords.
8192	FLAG_NAMED_PIPE	Force Use of Named Pipes	Connect with named pipes to a mysqld server running on NT.
16384	FLAG_NO_BIGINT	Change BIGINT Columns to Int	Change BIGINT columns to INT columns (some applications cannot handle BIGINT).
32768	FLAG_NO_CATALOG	No Catalog	Forces results from the catalog functions, such as SQLTables, to always return NULL and the driver to report that catalogs are not supported.
65536	FLAG_USE_MYCNF	Read Options From my.cnf	Read parameters from the [client] and [odbc] groups from my.cnf.
131072	FLAG_SAFE	Safe	Add some extra safety checks.
262144	FLAG_NO_TRANSACTIONS	Disable transactions	Disable transactions.
524288	FLAG_LOG_QUERY	Save queries to myodbc.sql	Enable query logging to c:\myodbc.sql(/tmp/myodbc.sql) file. (Enabled only in debug mode.)
1048576	FLAG_NO_CACHE	Do not Cache Result (forward only cursors)	Do not cache the results locally in the driver, instead read from server (mysql_use_result()). This works only for forward-only cursors. This option is very important in dealing with large tables when you do not want the driver to cache the entire result set.
2097152	FLAG_FORWARD_CURSOR	Force Use Of Forward Only Cursors	Force the use of Forward-only cursor type. In case of applications setting the default static/dynamic cursor type, and one wants the driver to use non-cache result sets, then this option ensures the forward-only cursor behavior.
4194304	FLAG_AUTO_RECONNECT	Enable auto-reconnect.	Enables auto-reconnection functionality. You should not use this option with transactions, since a auto reconnection during a incomplete transaction may cause corruption. Note that an auto-reconnected connection will not inherit the same settings and environment as the original. This option was added in Connector/ODBC 3.51.13.
8388608	FLAG_AUTO_IS_NULL	Flag Auto Is Null	<p>When FLAG_AUTO_IS_NULL is set, the driver does not change the default value of sql_auto_is_null, leaving it at 1, so you get the MySQL default, not the SQL standard behavior.</p> <p>When FLAG_AUTO_IS_NULL is not set, the driver changes the default value of SQL_AUTO_IS_NULL to 0 after connecting, so you get the SQL standard, not the MySQL default behaviour.</p> <p>Thus, omitting the flag disables the compatibility option and forces SQL standard behaviour.</p> <p>See IS NULL. This option was added in Connector/ODBC 3.51.13.</p>
16777216	FLAG_ZERO_DATE_TO_MIN	Flag Zero Date to Min	Translates zero dates (XXXX-00-00) into the minimum date values supported by ODBC, XXXX-01-01. This resolves an issue where some statements will not work because the date returned and the minimum ODBC date value are incompatible. This option was added in Connector/ODBC 3.51.17.
33554432	FLAG_MIN_DATE_TO_ZERO	Flag Min Date to Zero	Translates the minimum ODBC date value (XXXX-01-01) to the zero date format supported by MySQL (XXXX-00-00). This resolves an issue where some statements will not work because the date returned and the minimum ODBC date value are incompatible. This option was added in Connector/ODBC 3.51.17.

67108864	FLAG_MULTI_STATEMENTS	Allow multiple statements	Enables support for batched statements. This option was added in Connector/ODBC 3.51.18.
134217728	FLAG_COLUMN_SIZE_S32	Limit column size to 32-bit value	Limits the column size to a signed 32-bit value to prevent problems with larger column sizes in applications that do not support them. This option is automatically enabled when working with ADO applications. This option was added in Connector/ODBC 3.51.22.
268435456	FLAG_NO_BINARY_RESULT	Always handle binary function results as character data	When set this option disables charset 63 for columns with an empty <code>org_table</code> . This option was added in Connector/ODBC 3.51.26.

To select multiple options, add together their values. For example, setting `option` to 12 (4+8) gives you debugging without packet limits.

The following table shows some recommended `option` values for various configurations.

Configuration	Option Value
Microsoft Access, Visual Basic	3
Driver trace generation (Debug mode)	4
Microsoft Access (with improved DELETE queries)	35
Large tables with too many rows	2049
Sybase PowerBuilder	135168
Query log generation (Debug mode)	524288
Generate driver trace as well as query log (Debug mode)	524292
Large tables with no-cache results	3145731

### 3.3. Configuring a Connector/ODBC DSN on Windows

The `ODBC Data Source Administrator` within Windows enables you to create DSNs, check driver installation and configure ODBC systems such as tracing (used for debugging) and connection pooling.

Different editions and versions of Windows store the `ODBC Data Source Administrator` in different locations depending on the version of Windows that you are using.

To open the `ODBC Data Source Administrator` in Windows Server 2003:

#### Tip

Because it is possible to create DSN using either the 32-bit or 64-bit driver, but using the same DNS identifier, it is advisable to include the driver being used within the DSN identifier. This will help you to identify the DSN when using it from applications such as Excel that are only compatible with the 32-bit driver. For example, you might add `Using32bitCOBDC` to the DSN identifier for the 32-bit interface and `Using64bitCOBDC` for those using the 64-bit Connector/ODBC driver.

1. On the `Start` menu, choose `Administrative Tools`, and then click `Data Sources (ODBC)`.

To open the `ODBC Data Source Administrator` in Windows 2000 Server or Windows 2000 Professional:

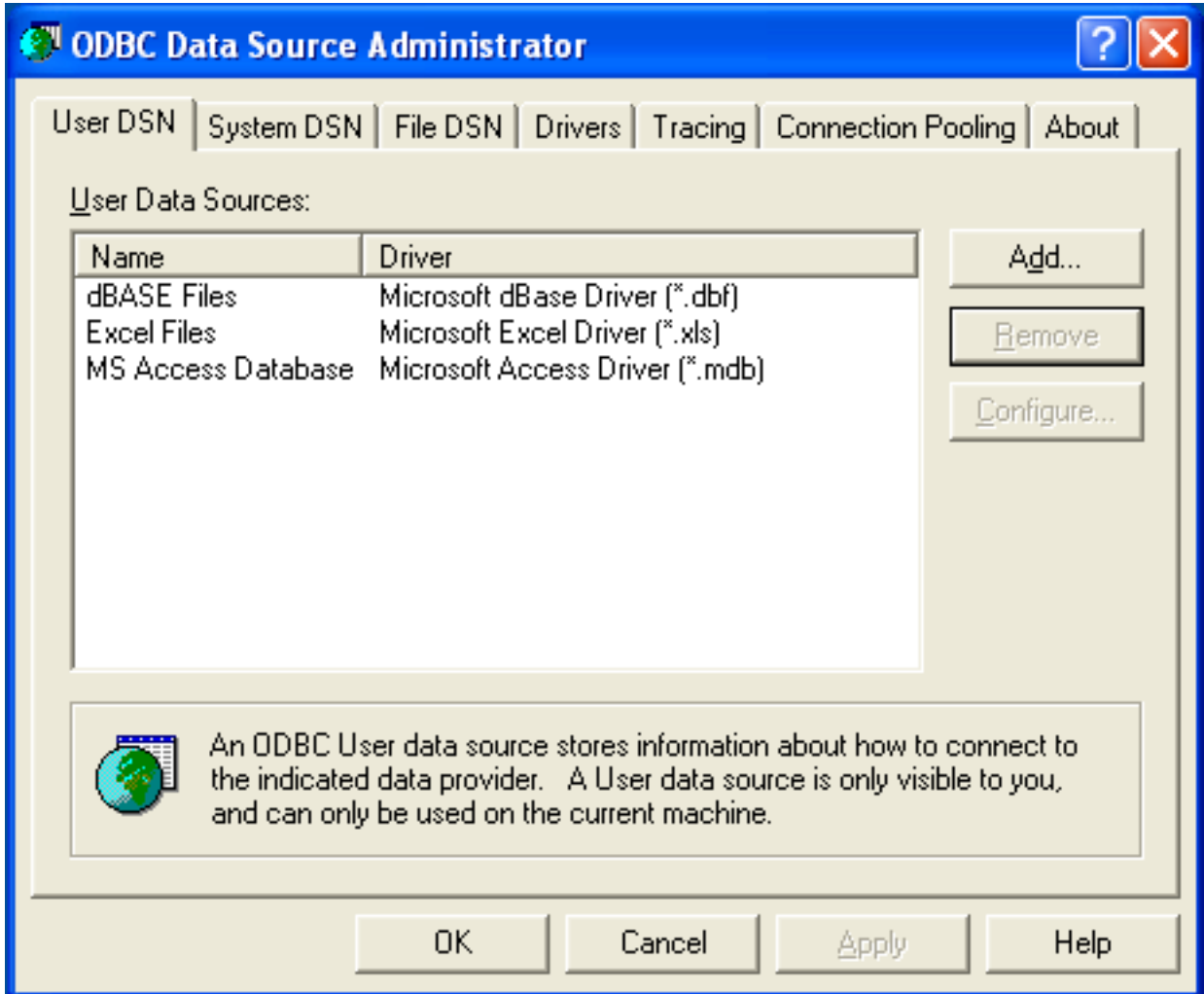
1. On the `Start` menu, choose `Settings`, and then click `Control Panel`.
2. In `Control Panel`, click `Administrative Tools`.
3. In `Administrative Tools`, click `Data Sources (ODBC)`.

To open the `ODBC Data Source Administrator` on Windows XP:

1. On the `Start` menu, click `Control Panel`.

2. In the [Control Panel](#) when in [Category View](#) click [Performance and Maintenance](#) and then click [Administrative Tools](#).. If you are viewing the [Control Panel](#) in [Classic View](#), click [Administrative Tools](#).
3. In [Administrative Tools](#), click [Data Sources \(ODBC\)](#).

Irrespective of your Windows version, you should be presented the [ODBC Data Source Administrator](#) window:



Within Windows XP, you can add the [Administrative Tools](#) folder to your [START](#) menu to make it easier to locate the [ODBC Data Source Administrator](#). To do this:

1. Right click on the [START](#) menu.
2. Select [Properties](#).
3. Click [CUSTOMIZE...](#)
4. Select the [ADVANCED](#) tab.
5. Within [Start menu items](#), within the [System Administrative Tools](#) section, select [Display on the All Programs menu](#).

Within both Windows Server 2003 and Windows XP you may want to permanently add the [ODBC Data Source Administrator](#) to your [START](#) menu. To do this, locate the [Data Sources \(ODBC\)](#) icon using the methods shown, then right-click on the icon and then choose [PIN TO START MENU](#).

The interfaces for the 3.51 and 5.1 versions of the Connector/ODBC driver are different, although the fields and information that you need to enter remain the same.

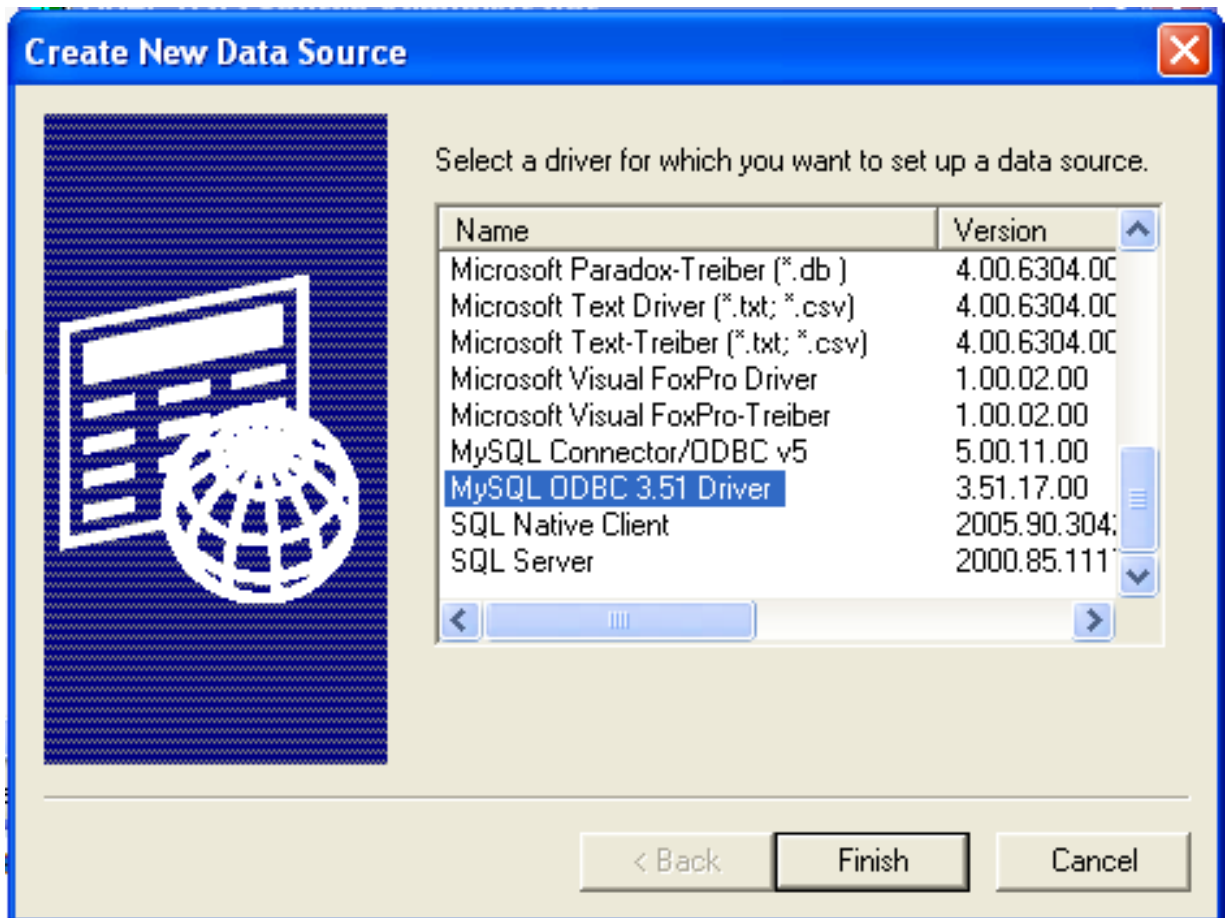
To configure a DSN using Connector/ODBC 3.51.x or Connector/ODBC 5.1.0, see [Section 3.3.1, “Configuring a Connector/ODBC 3.51 DSN on Windows”](#).

To configure a DSN using Connector/ODBC 5.1.1 or later, see [Section 3.3.2, “Configuring a Connector/ODBC 5.1 DSN on Windows”](#).

### 3.3.1. Configuring a Connector/ODBC 3.51 DSN on Windows

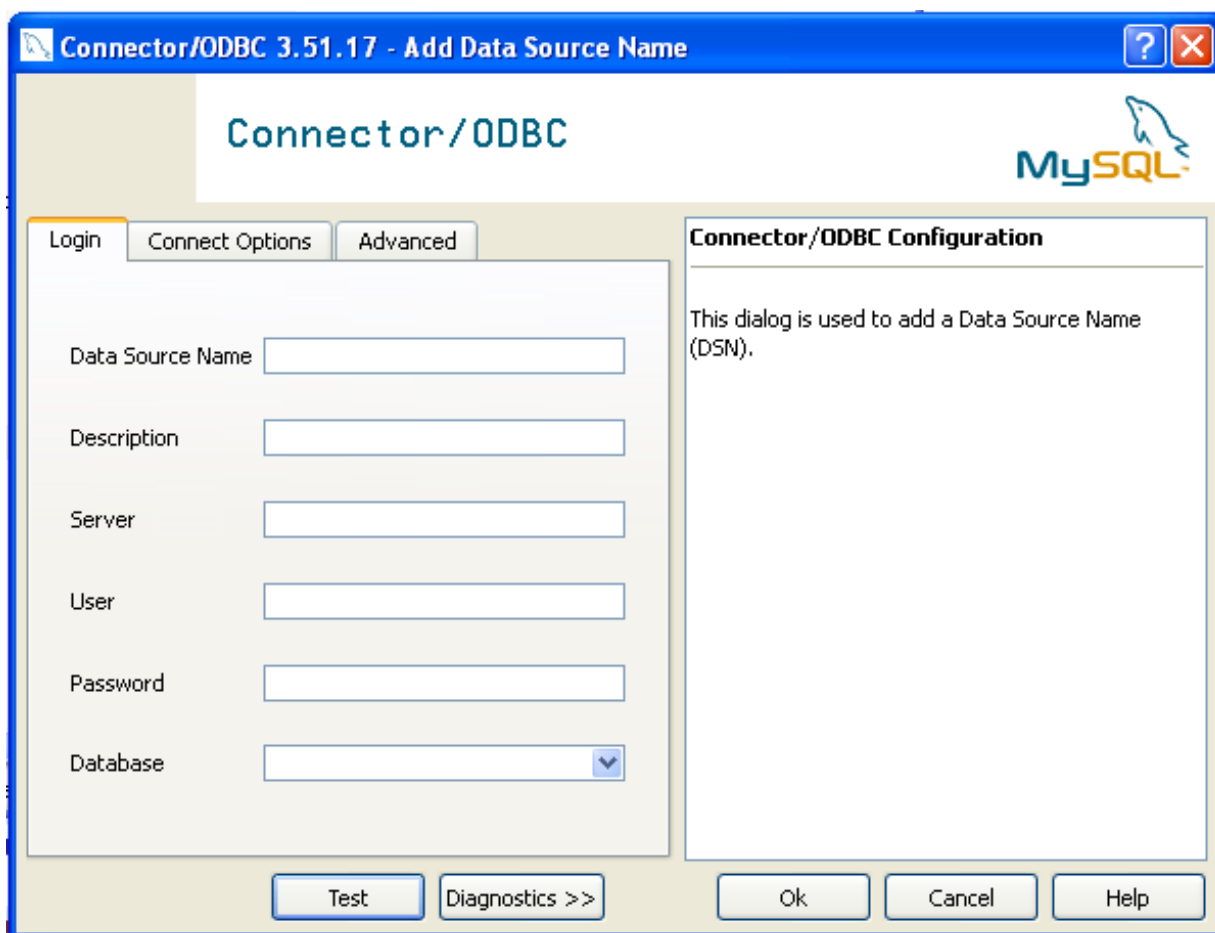
To add and configure a new Connector/ODBC data source on Windows, use the [ODBC Data Source Administrator](#):

1. Open the [ODBC Data Source Administrator](#).
2. To create a System DSN (which will be available to all users), select the [System DSN](#) tab. To create a User DSN, which will be unique only to the current user, click the [ADD...](#) button.
3. You will need to select the ODBC driver for this DSN.



Select [MySQL ODBC 3.51 Driver](#), then click [FINISH](#).

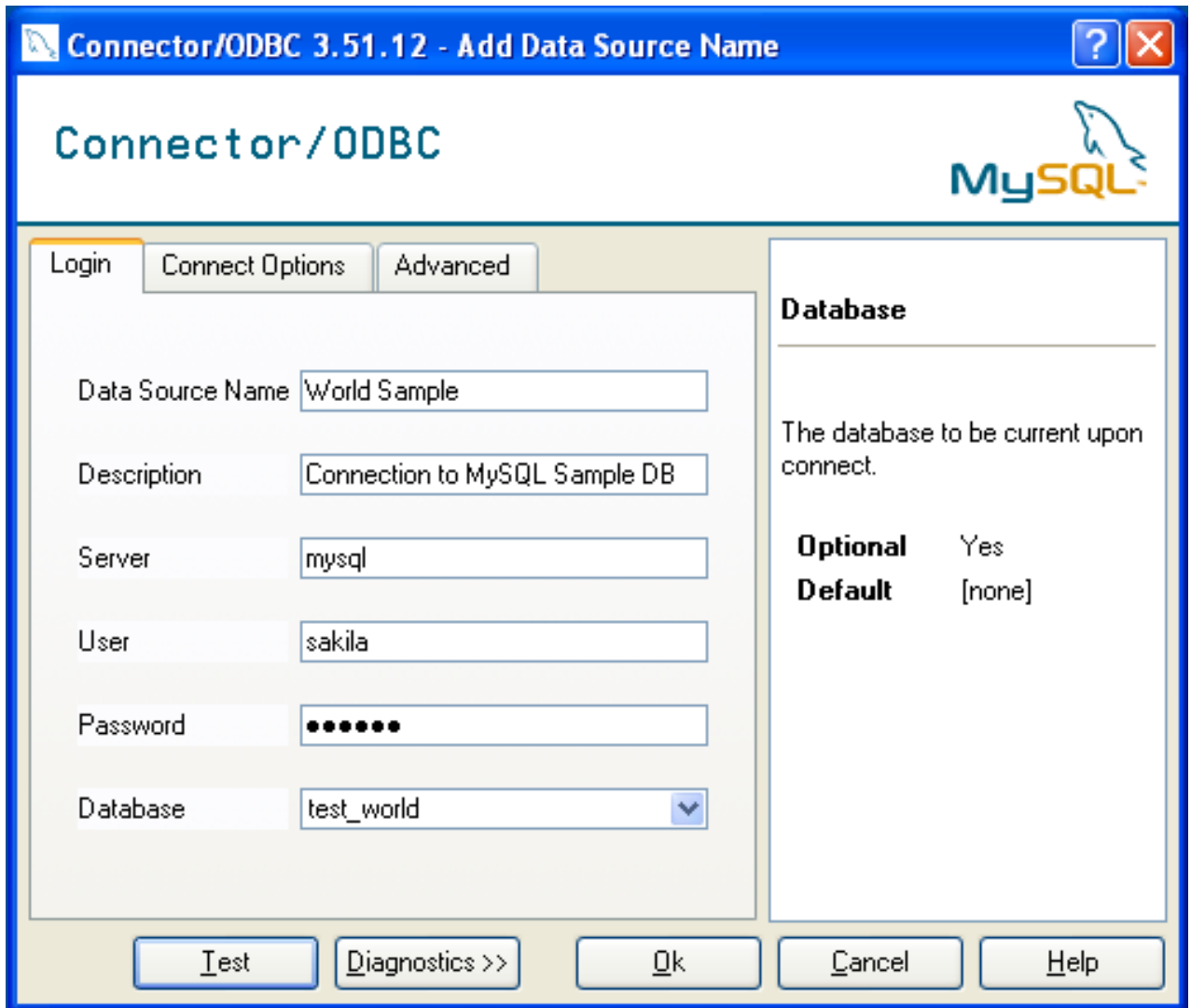
4. You now need to configure the specific fields for the DSN you are creating through the [Add Data Source Name](#) dialog.



In the **DATA SOURCE NAME** box, enter the name of the data source you want to access. It can be any valid name that you choose.

5. In the **DESCRIPTION** box, enter some text to help identify the connection.
6. In the **SERVER** field, enter the name of the MySQL server host that you want to access. By default, it is `localhost`.
7. In the **USER** field, enter the user name to use for this connection.
8. In the **PASSWORD** field, enter the corresponding password for this connection.
9. The **DATABASE** pop-up should automatically populate with the list of databases that the user has permissions to access.
10. Click OK to save the DSN.

A completed DSN configuration may look like this:

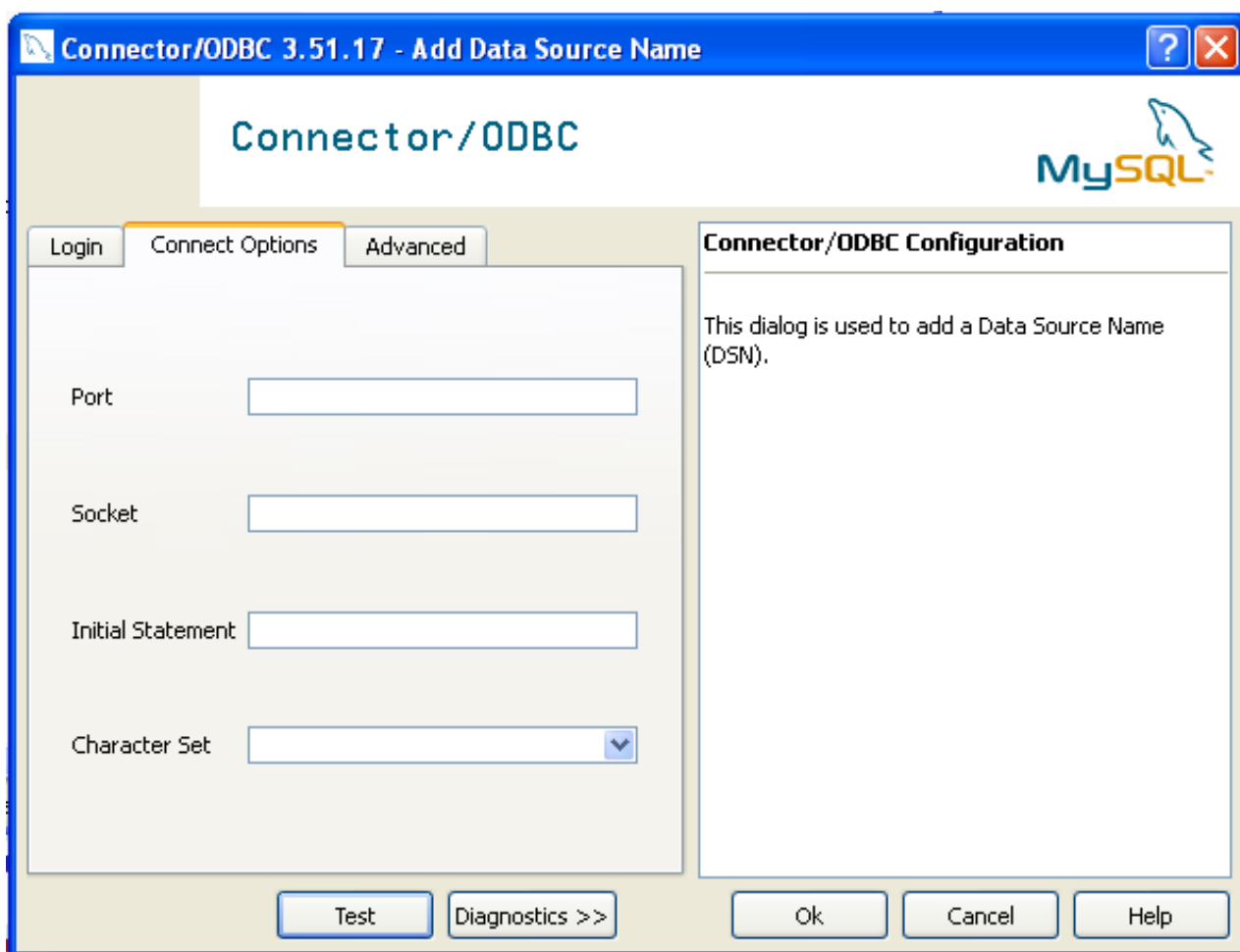


You can verify the connection using the parameters you have entered by clicking the TEST button. If the connection could be made successfully, you will be notified with a `Success; connection was made!` dialog.

If the connection failed, you can obtain more information on the test and why it may have failed by clicking the DIAGNOSTICS... button to show additional error messages.

You can configure a number of options for a specific DSN by using either the CONNECT OPTIONS or ADVANCED tabs in the DSN configuration dialog.

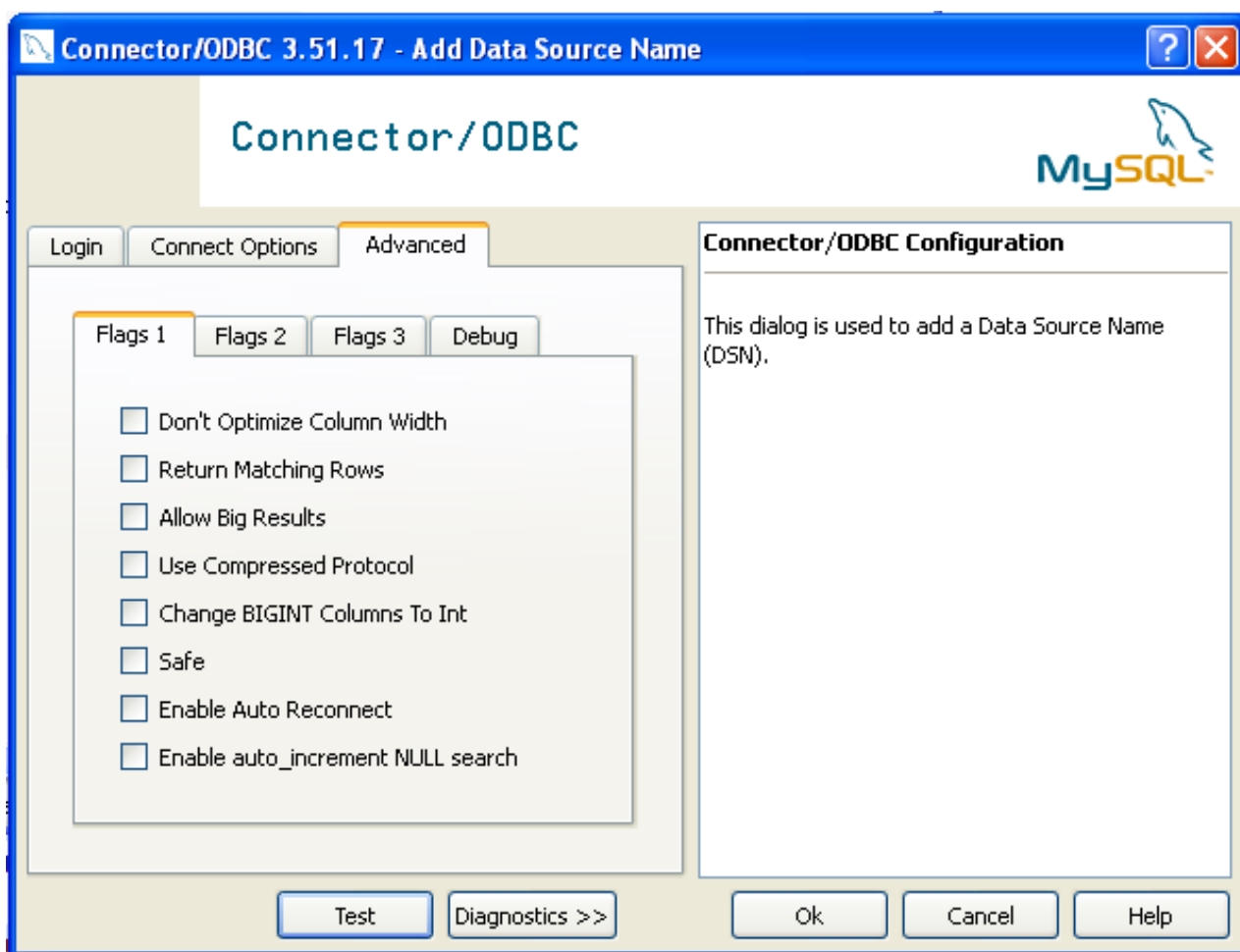




The three options you can configure are:

- **PORT** sets the TCP/IP port number to use when communicating with MySQL. Communication with MySQL uses port 3306 by default. If your server is configured to use a different TCP/IP port, you must specify that port number here.
- **SOCKET** sets the name or location of a specific socket or Windows pipe to use when communicating with MySQL.
- **INITIAL STATEMENT** defines an SQL statement that will be executed when the connection to MySQL is opened. You can use this to set MySQL options for your connection, such as disabling autocommit.
- **CHARACTER SET** is a pop-up list from which you can select the default character set to be used with this connection. The Character Set option was added in 3.5.17.

The **ADVANCED** tab enables you to configure Connector/ODBC connection parameters. Refer to [Section 3.2, “Connector/ODBC Connection Parameters”](#), for information about the meaning of these options.

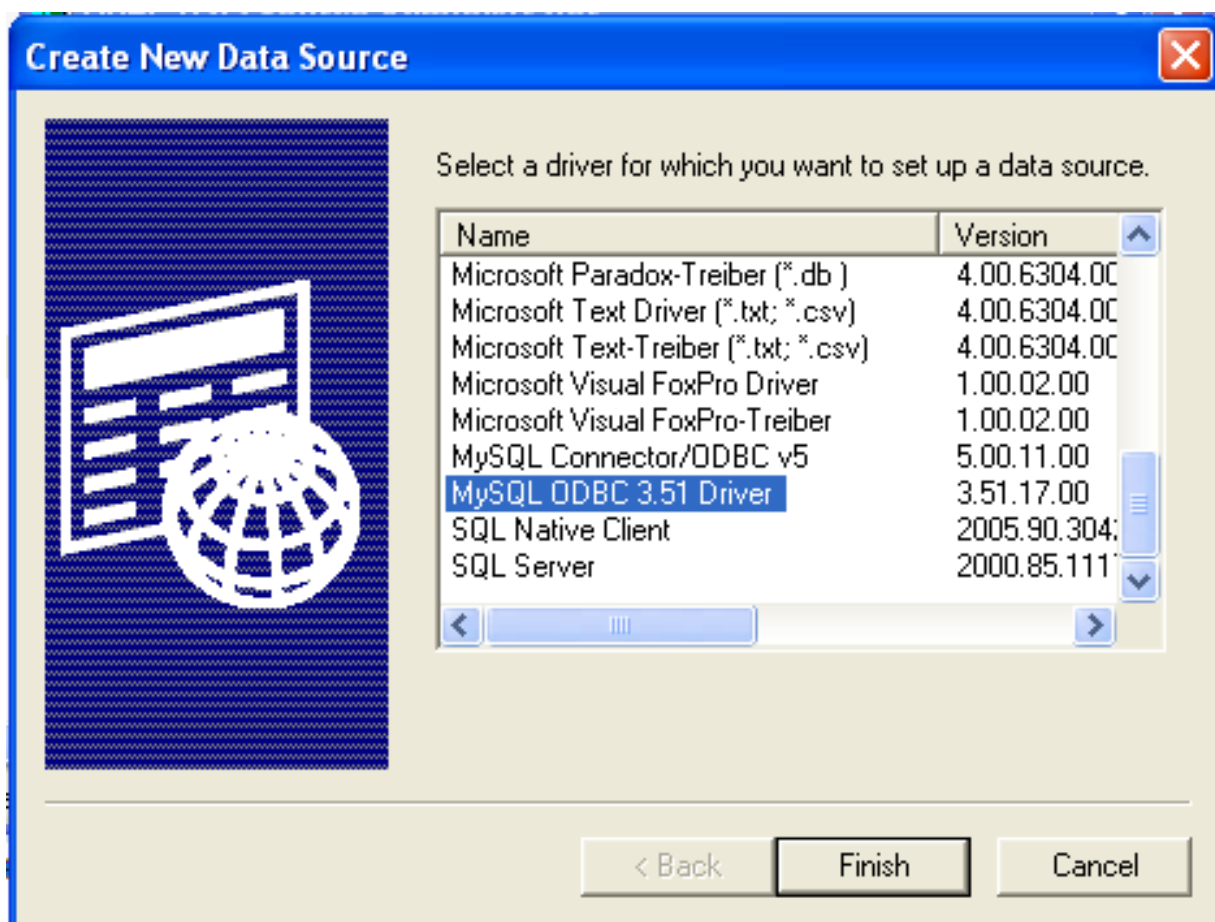


### 3.3.2. Configuring a Connector/ODBC 5.1 DSN on Windows

The DSN configuration when using Connector/ODBC 5.1.1 and later has a slightly different layout. Also, due to the native Unicode support within Connector/ODBC 5.1, you no longer need to specify the initial character set to be used with your connection.

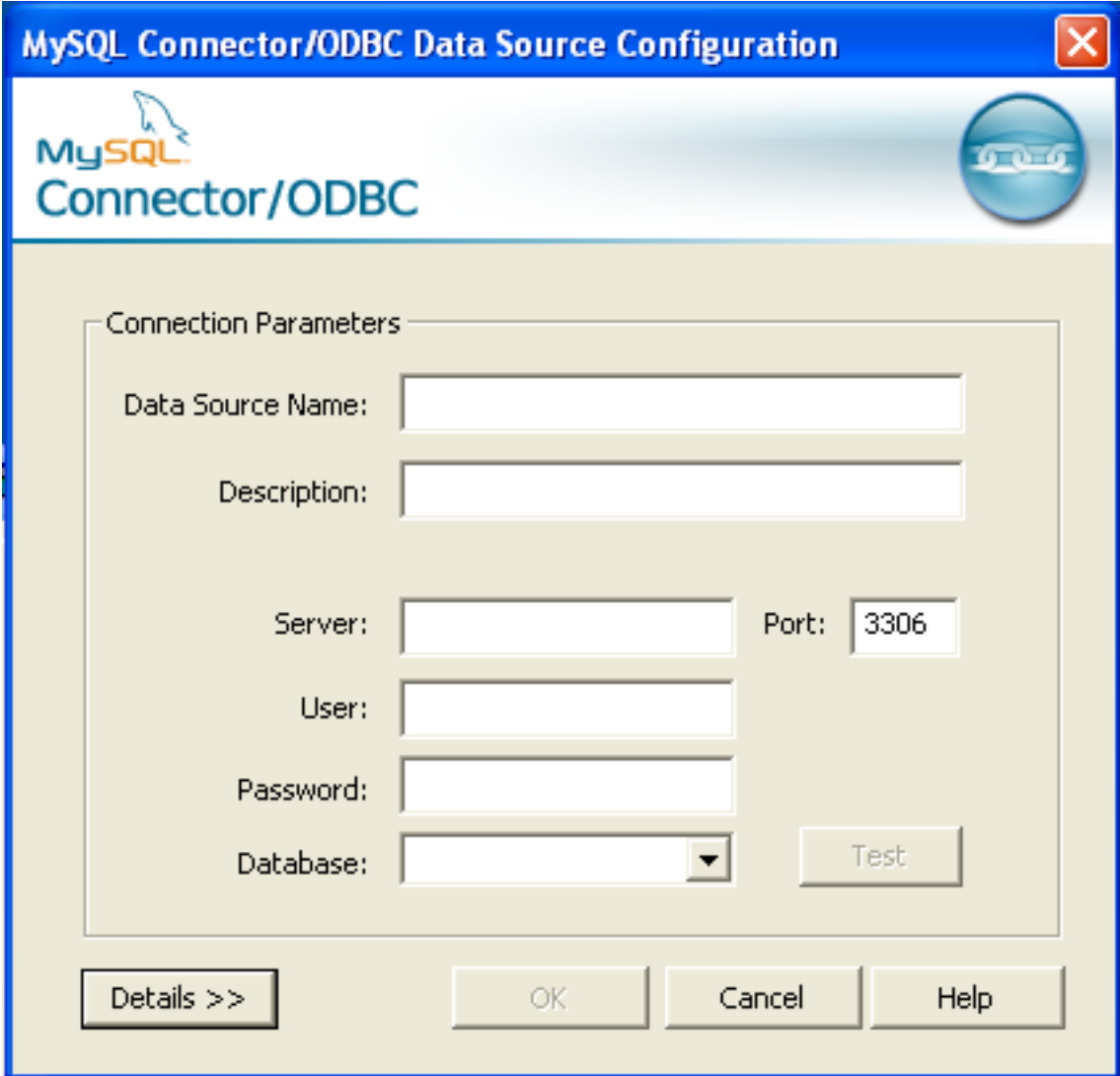
To configure a DSN using the Connector/ODBC 5.1.1 or later driver:

1. Open the [ODBC Data Source Administrator](#).
2. To create a System DSN (which will be available to all users) , select the **SYSTEM DSN** tab. To create a User DSN, which will be unique only to the current user, click the **ADD...** button.
3. You will need to select the ODBC driver for this DSN.



Select [MySQL ODBC 3.51 Driver](#), then click FINISH.

4. You now need to configure the specific fields for the DSN you are creating through the [Connection Parameters](#) dialog.



MySQL Connector/ODBC Data Source Configuration

MySQL Connector/ODBC

Connection Parameters

Data Source Name:

Description:

Server:  Port:

User:

Password:

Database:

In the **DATA SOURCE NAME** box, enter the name of the data source you want to access. It can be any valid name that you choose.

5. In the **DESCRIPTION** box, enter some text to help identify the connection.
6. In the **SERVER** field, enter the name of the MySQL server host that you want to access. By default, it is `localhost`.
7. In the **USER** field, enter the user name to use for this connection.
8. In the **PASSWORD** field, enter the corresponding password for this connection.
9. The **DATABASE** pop-up should automatically populate with the list of databases that the user has permissions to access.
10. To communicate over a different TCP/IP port than the default (3306), change the value of the **PORT**.
11. Click OK to save the DSN.

You can verify the connection using the parameters you have entered by clicking the **TEST** button. If the connection could be made successfully, you will be notified with a `Success; connection was made!` dialog.

You can configure a number of options for a specific DSN by using the **DETAILS** button.

MySQL Connector/ODBC Data Source Configuration

MySQL Connector/ODBC

Connection Parameters

Data Source Name:

Description:

Server:  Port:

User:

Password:

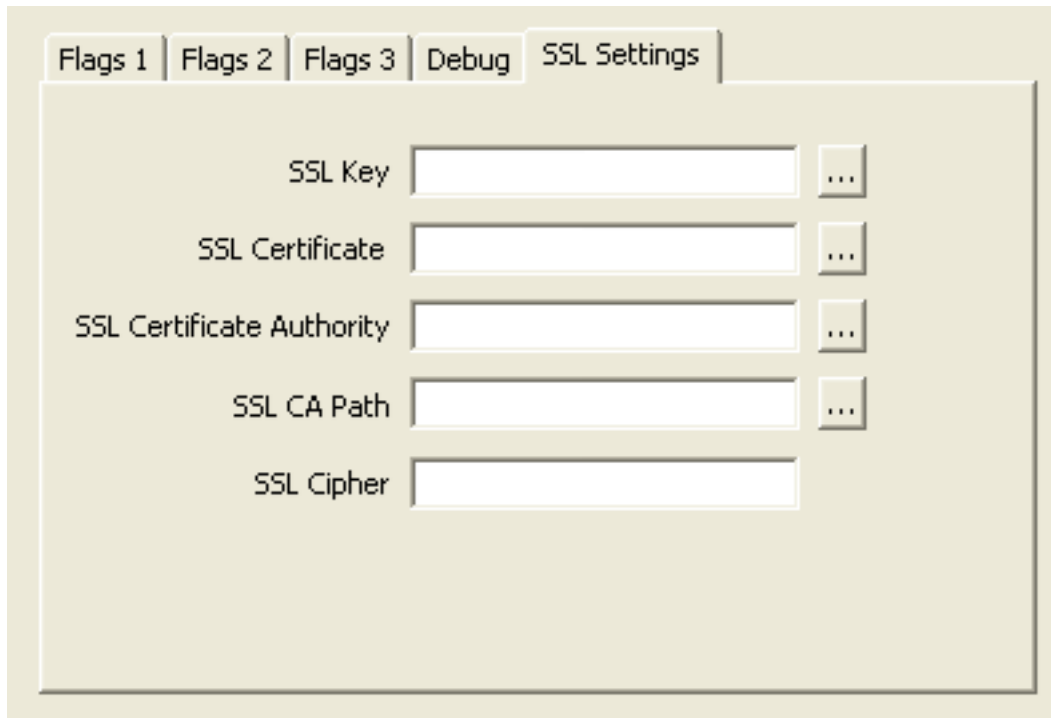
Database:

Flags 1 | Flags 2 | Flags 3 | Debug | SSL Settings

- Return matched rows instead of affected rows
- Allow big result sets
- Use compression
- Treat BIGINT columns as INT columns
- Enable safe options (see documentation)
- Enable automatic reconnect
- Enable SQL\_AUTO\_IS\_NULL

The **DETAILS** button opens a tabbed display which allows you to set additional options:

- **FLAGS 1**, **FLAGS 2**, and **FLAGS 3** enable you to select the additional flags for the DSN connection. For more information on these flags, see [Section 3.2, “Connector/ODBC Connection Parameters”](#).
- **DEBUG** allows you to enable ODBC debugging to record the queries you execute through the DSN to the `myodbc.sql` file. For more information, see [Section 3.8, “Getting an ODBC Trace File”](#).
- **SSL SETTINGS** configures the additional options required for using the Secure Sockets Layer (SSL) when communicating with MySQL server. Note that you must have enabled SSL and configured the MySQL server with suitable certificates to communicate over SSL.



The **ADVANCED** tab enables you to configure Connector/ODBC connection parameters. Refer to [Section 3.2, “Connector/ODBC Connection Parameters”](#), for information about the meaning of these options.

### 3.3.3. Errors and Debugging

This section answers Connector/ODBC connection-related questions.

- **While configuring a Connector/ODBC DSN, a [Could Not Load Translator or Setup Library](#) error occurs**

For more information, refer to [MS KnowledgeBase Article\(Q260558\)](#). Also, make sure you have the latest valid `ct13d32.dll` in your system directory.

- On Windows, the default `myodbc3.dll` is compiled for optimal performance. If you want to debug Connector/ODBC 3.51 (for example, to enable tracing), you should instead use `myodbc3d.dll`. To install this file, copy `myodbc3d.dll` over the installed `myodbc3.dll` file. Make sure to revert back to the release version of the driver DLL once you are done with the debugging because the debug version may cause performance issues. Note that the `myodbc3d.dll` isn't included in Connector/ODBC 3.51.07 through 3.51.11. If you are using one of these versions, you should copy that DLL from a previous version (for example, 3.51.06).

## 3.4. Configuring a Connector/ODBC DSN on Mac OS X

To configure a DSN on Mac OS X you can either use the `myodbc3i` utility, edit the `odbc.ini` file within the `Library/ODBC` directory of the user or the should use the ODBC Administrator. If you have Mac OS X 10.2 or earlier, refer to [Section 3.5, “Configuring a Connector/ODBC DSN on Unix”](#). Select whether you want to create a User DSN or a System DSN. If you want to add a System DSN, you may need to authenticate with the system. You must click the padlock and enter a user and password with administrator privileges.

For correct operation of ODBC Administrator, you should ensure that the `/Library/ODBC/odbc.ini` file used to set up ODBC connectivity and DSNs are writable by the `admin` group. If this file is not writable by this group then the ODBC Administrator may fail, or may appear to have worked but not generated the correct entry.

### Warning

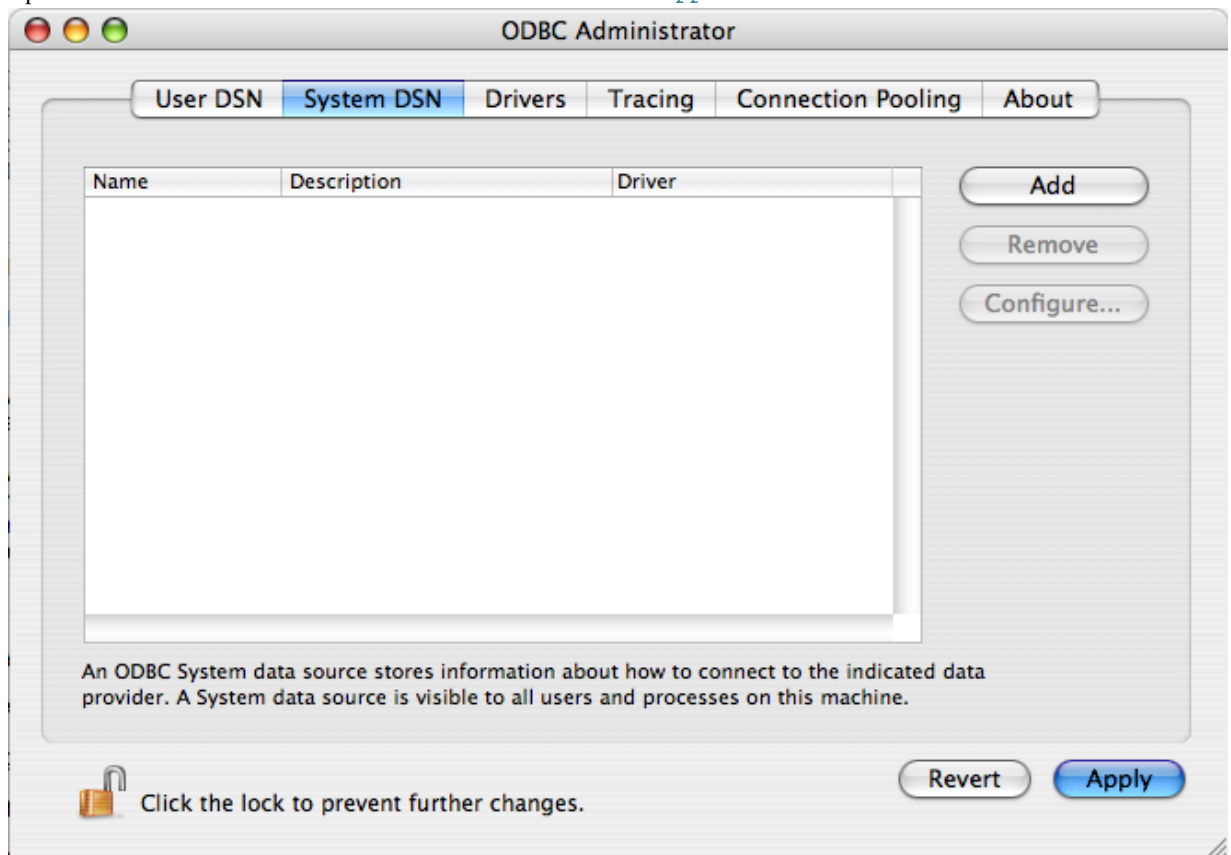
There are known issues with the OS X ODBC Administrator and Connector/ODBC that may prevent you from creating a DSN using this method. In this case you should use the command-line or edit the `odbc.ini` file directly. Note that existing DSNs or those that you create via the `myodbc3i` or `myodbc-installertool` can still be checked and edited using ODBC Administrator.

To create a DSN using the `myodbc3i` utility, you need only specify the DSN type and the DSN connection string. For example:

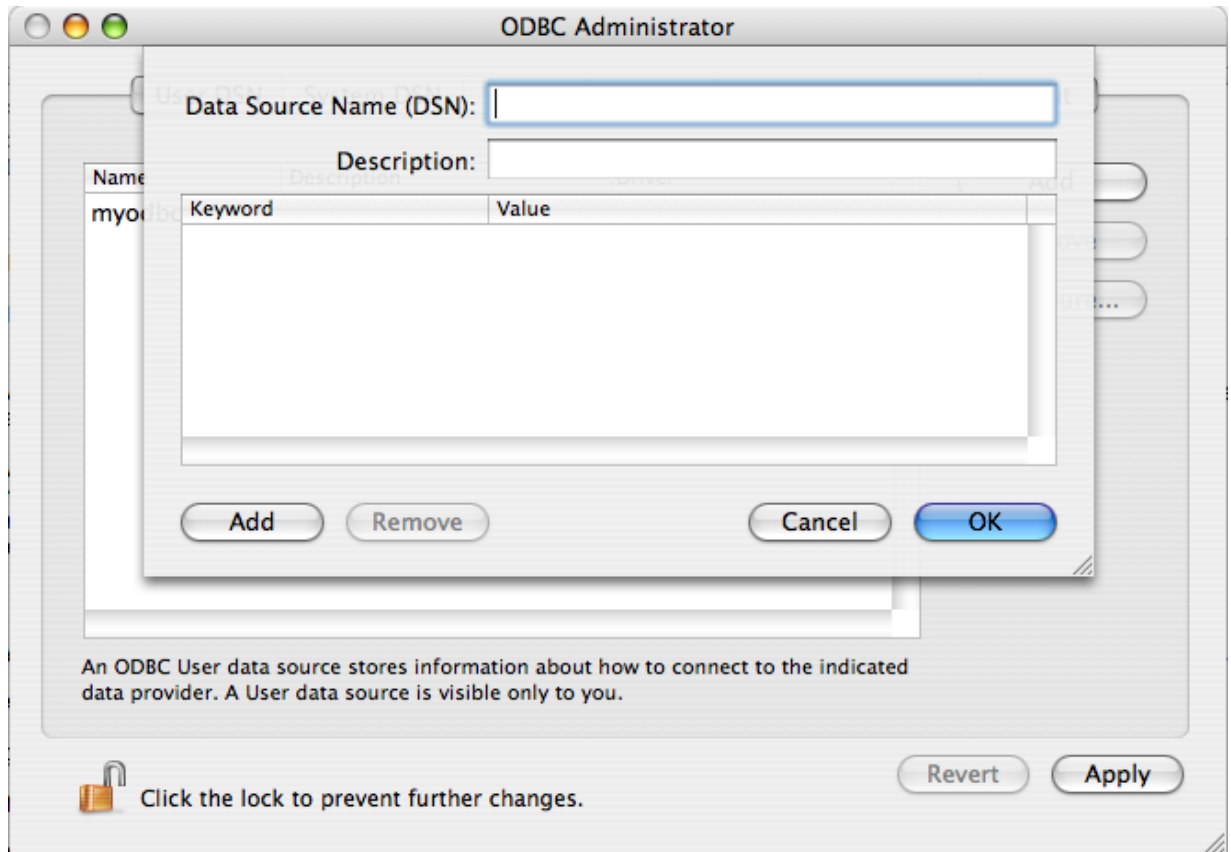
```
shell> myodbc3i -a -s -t "DSN=mydb;DRIVER=MySQL ODBC 3.51 Driver;SERVER=mysql;USER=username;PASSWORD=pass"
```

To use ODBC Administrator:

1. Open the ODBC Administrator from the `Utilities` folder in the `Applications` folder.

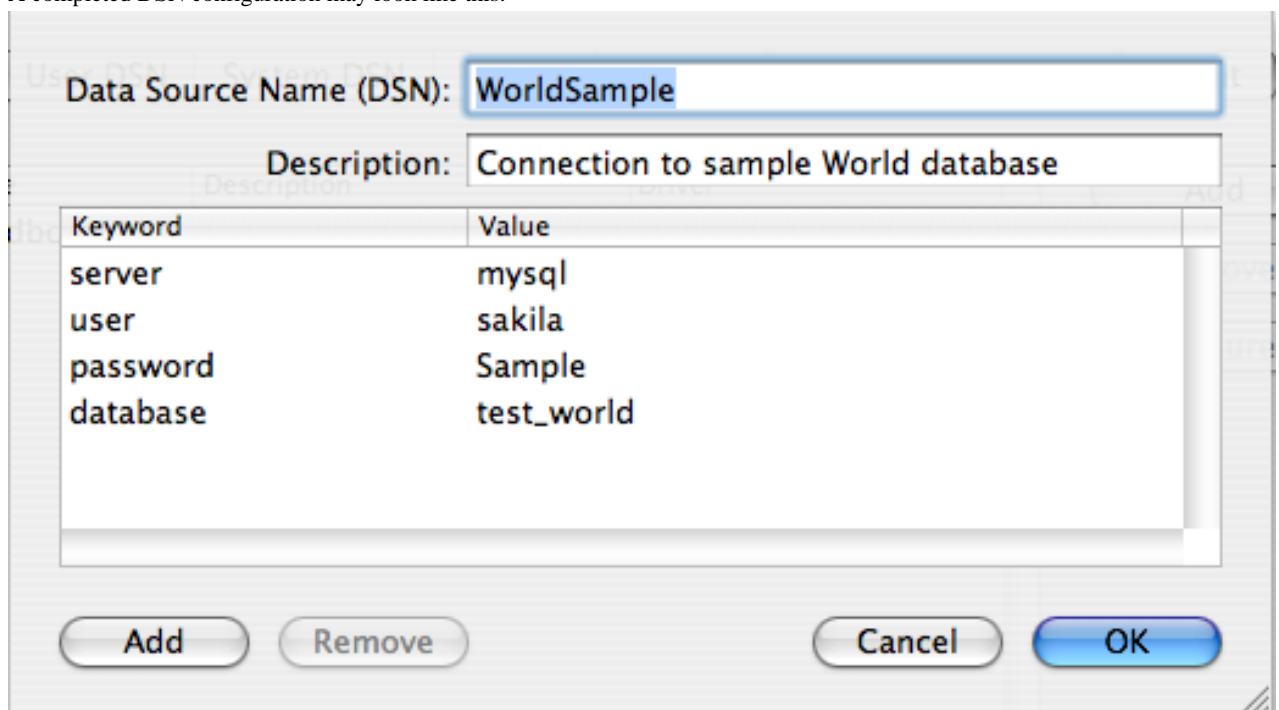


2. On the User DSN or System DSN panel, click ADD.
3. Select the Connector/ODBC driver and click OK.
4. You will be presented with the `Data Source Name` dialog. Enter The `Data Source Name` and an optional `Description` for the DSN.



5. Click ADD to add a new keyword/value pair to the panel. You should configure at least four pairs to specify the `server`, `username`, `password` and `database` connection parameters. See Section 3.2, “Connector/ODBC Connection Parameters”.
6. Click OK to add the DSN to the list of configured data source names.

A completed DSN configuration may look like this:





You can configure additional ODBC options to your DSN by adding further keyword/value pairs and setting the corresponding values. See [Section 3.2, “Connector/ODBC Connection Parameters”](#).

## 3.5. Configuring a Connector/ODBC DSN on Unix

On [Unix](#), you configure DSN entries directly in the `odbc.ini` file. Here is a typical `odbc.ini` file that configures `myodbc3` as the DSN name for Connector/ODBC 3.51:

```
;
; odbc.ini configuration for Connector/ODBC and Connector/ODBC 3.51 drivers
;
[ODBC Data Sources]
myodbc3      = MyODBC 3.51 Driver DSN
[myodbc3]
Driver       = /usr/local/lib/libmyodbc3.so
Description  = Connector/ODBC 3.51 Driver DSN
SERVER      = localhost
PORT        =
USER        = root
Password    =
Database    = test
OPTION      = 3
SOCKET      =
[Default]
Driver       = /usr/local/lib/libmyodbc3.so
Description  = Connector/ODBC 3.51 Driver DSN
SERVER      = localhost
PORT        =
USER        = root
Password    =
Database    = test
OPTION      = 3
SOCKET      =
```

Refer to the [Section 3.2, “Connector/ODBC Connection Parameters”](#), for the list of connection parameters that can be supplied.

### Note

If you are using `unixODBC`, you can use the following tools to set up the DSN:

- ODBCConfig GUI tool([HOWTO: ODBCConfig](#))
- `odbcinst`

In some cases when using `unixODBC`, you might get this error:

```
Data source name not found and no default driver specified
```

If this happens, make sure the `ODBCINI` and `ODBCSYSINI` environment variables are pointing to the right `odbc.ini` file. For example, if your `odbc.ini` file is located in `/usr/local/etc`, set the environment variables like this:

```
export ODBCINI=/usr/local/etc/odbc.ini
export ODBCSYSINI=/usr/local/etc
```

## 3.6. Connecting Without a Predefined DSN

You can connect to the MySQL server using `SQLDriverConnect`, by specifying the `DRIVER` name field. Here are the connection strings for Connector/ODBC using DSN-Less connections:

### For Connector/ODBC 3.51:

```
ConnectionString = "DRIVER={MySQL ODBC 3.51 Driver};\
SERVER=localhost;\
DATABASE=test;\
USER=venu;\
PASSWORD=venu;\
OPTION=3;"
```

If your programming language converts backslash followed by whitespace to a space, it is preferable to specify the connection string as a single long string, or to use a concatenation of multiple strings that does not add spaces in between. For example:

```
ConnectionString = "DRIVER={MySQL ODBC 3.51 Driver};"
                  "SERVER=localhost;"
                  "DATABASE=test;"
                  "USER=venu;"
```

```
"PASSWORD=venu; "  
"OPTION=3; "
```

**Note.** Note that on Mac OS X you may need to specify the full path to the Connector/ODBC driver library.

Refer to the [Section 3.2, “Connector/ODBC Connection Parameters”](#), for the list of connection parameters that can be supplied.

## 3.7. ODBC Connection Pooling

Connection pooling enables the ODBC driver to re-use existing connections to a given database from a pool of connections, instead of opening a new connection each time the database is accessed. By enabling connection pooling you can improve the overall performance of your application by lowering the time taken to open a connection to a database in the connection pool.

For more information about connection pooling: <http://support.microsoft.com/default.aspx?scid=kb;EN-US;q169470>.

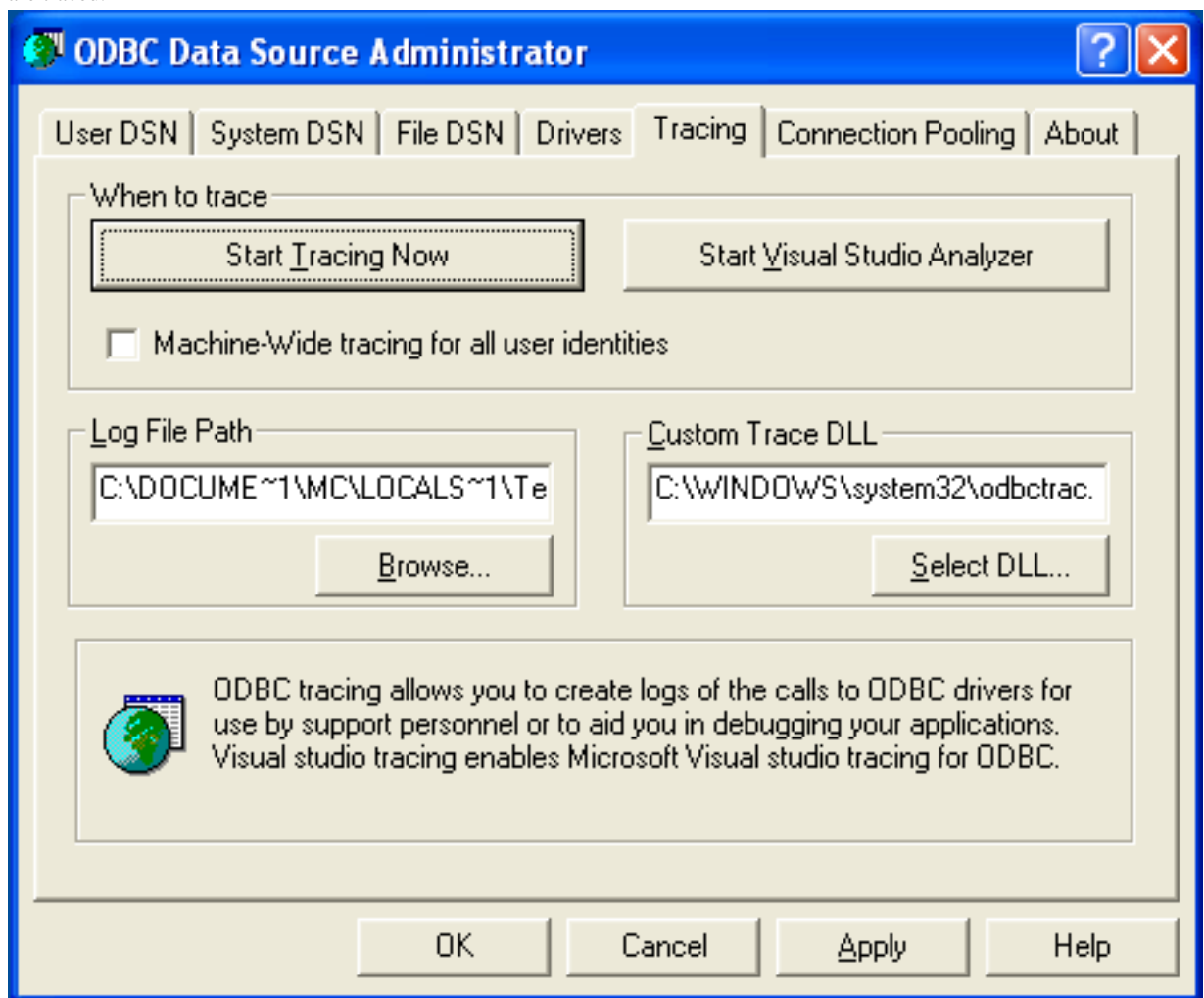
## 3.8. Getting an ODBC Trace File

If you encounter difficulties or problems with Connector/ODBC, you should start by making a log file from the [ODBC Manager](#) and Connector/ODBC. This is called *tracing*, and is enabled through the ODBC Manager. The procedure for this differs for Windows, Mac OS X and Unix.

### 3.8.1. Enabling ODBC Tracing on Windows

To enable the trace option on Windows:

1. The [Tracing](#) tab of the ODBC Data Source Administrator dialog box enables you to configure the way ODBC function calls are traced.

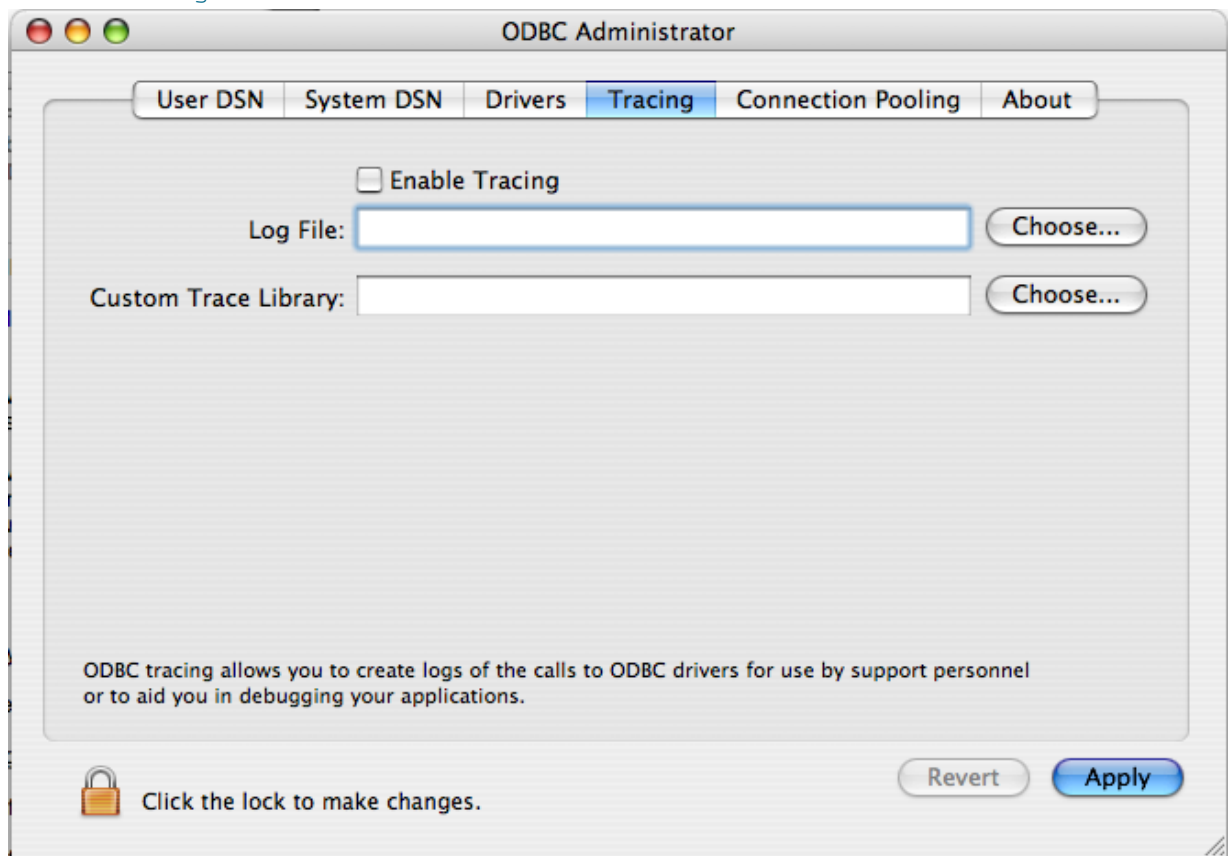


2. When you activate tracing from the [Tracing](#) tab, the [Driver Manager](#) logs all ODBC function calls for all subsequently run applications.
3. ODBC function calls from applications running before tracing is activated are not logged. ODBC function calls are recorded in a log file you specify.
4. Tracing ceases only after you click [Stop Tracing Now](#). Remember that while tracing is on, the log file continues to increase in size and that tracing affects the performance of all your ODBC applications.

### 3.8.2. Enabling ODBC Tracing on Mac OS X

To enable the trace option on Mac OS X 10.3 or later you should use the [Tracing](#) tab within ODBC Administrator .

1. Open the ODBC Administrator.
2. Select the [Tracing](#) tab.



3. Select the [Enable Tracing](#) checkbox.
4. Enter the location where you want to save the Tracing log. If you want to append information to an existing log file, click the CHOOSE... button.

### 3.8.3. Enabling ODBC Tracing on Unix

To enable the trace option on Mac OS X 10.2 (or earlier) or Unix you must add the [trace](#) option to the ODBC configuration:

1. On Unix, you need to explicitly set the [Trace](#) option in the [ODBC.INI](#) file.

Set the tracing [ON](#) or [OFF](#) by using [TraceFile](#) and [Trace](#) parameters in [odbc.ini](#) as shown below:

```
TraceFile = /tmp/odbc.trace
Trace     = 1
```

`TraceFile` specifies the name and full path of the trace file and `Trace` is set to `ON` or `OFF`. You can also use `1` or `YES` for `ON` and `0` or `NO` for `OFF`. If you are using `ODBCConfig` from `unixODBC`, then follow the instructions for tracing `unixODBC` calls at [HOWTO-ODBCConfig](#).

### 3.8.4. Enabling a Connector/ODBC Log

To generate a Connector/ODBC log, do the following:

1. Within Windows, enable the `Trace Connector/ODBC` option flag in the Connector/ODBC connect/configure screen. The log is written to file `C:\myodbc.log`. If the trace option is not remembered when you are going back to the above screen, it means that you are not using the `myodbcd.dll` driver, see [Section 3.3.3, “Errors and Debugging”](#).

On Mac OS X, Unix, or if you are using DSN-Less connection, then you need to supply `OPTION=4` in the connection string or set the corresponding keyword/value pair in the DSN.

2. Start your application and try to get it to fail. Then check the Connector/ODBC trace file to find out what could be wrong.

If you need help determining what is wrong, see [Section 7.1, “Connector/ODBC Community Support”](#).

---

## Chapter 4. Connector/ODBC Examples

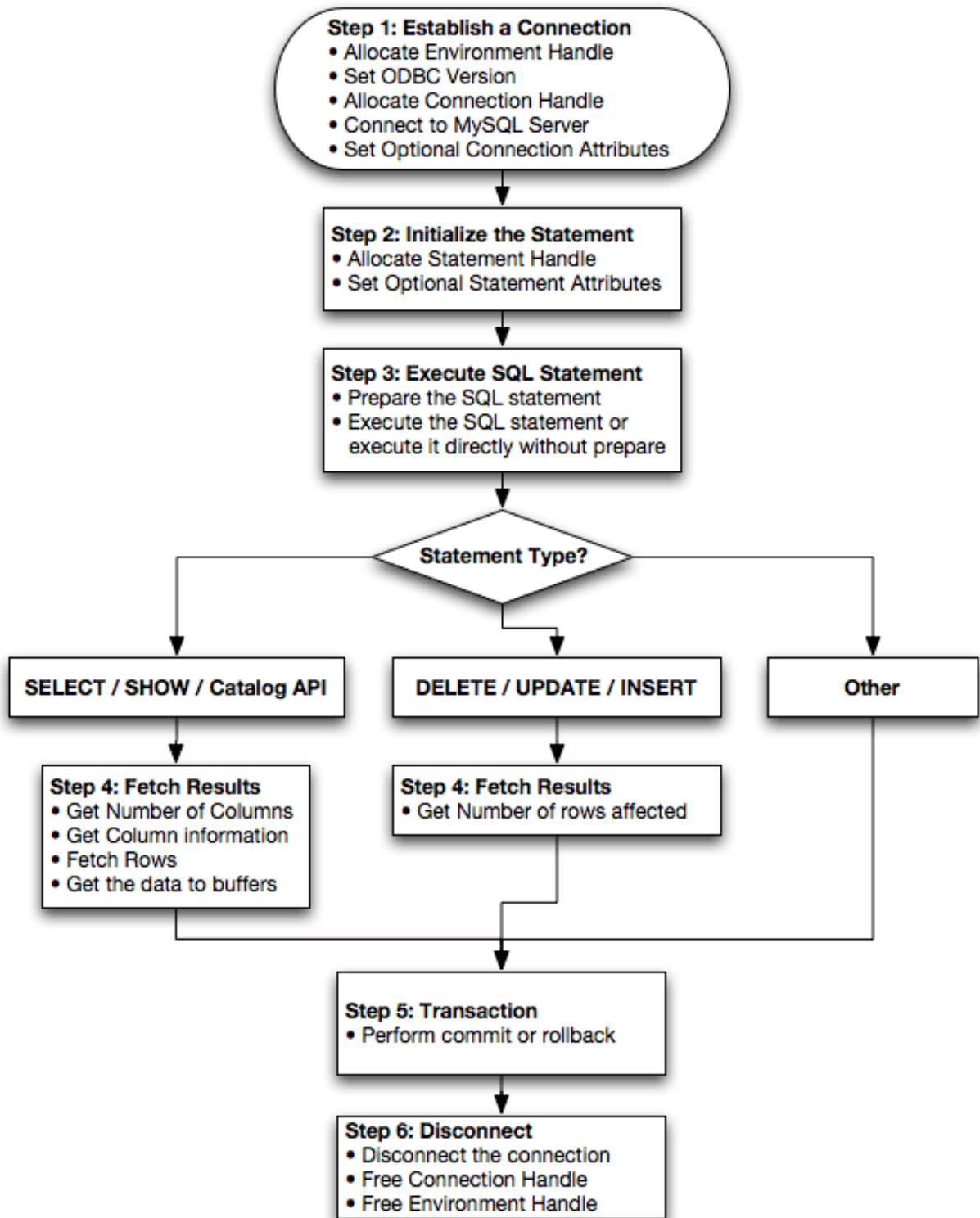
Once you have configured a DSN to provide access to a database, how you access and use that connection is dependent on the application or programming language. As ODBC is a standardized interface, any application or language that supports ODBC can use the DSN and connect to the configured database.

### 4.1. Basic Connector/ODBC Application Steps

Interacting with a MySQL server from an applications using the Connector/ODBC typically involves the following operations:

- Configure the Connector/ODBC DSN
- Connect to MySQL server
- Initialization operations
- Execute SQL statements
- Retrieve results
- Perform Transactions
- Disconnect from the server

Most applications use some variation of these steps. The basic application steps are shown in the following diagram:



## 4.2. Step-by-step Guide to Connecting to a MySQL Database through Connector/ODBC

A typical installation situation where you would install Connector/ODBC is when you want to access a database on a Linux or Unix host from a Windows machine.

As an example of the process required to set up access between two machines, the steps below take you through the basic steps. These instructions assume that you want to connect to system ALPHA from system BETA with a user name and password of `my-`

`user` and `mypassword`.

On system ALPHA (the MySQL server) follow these steps:

1. Start the MySQL server.
2. Use `GRANT` to set up an account with a user name of `myuser` that can connect from system BETA using a password of `myuser` to the database `test`:

```
GRANT ALL ON test.* to 'myuser'@'BETA' IDENTIFIED BY 'mypassword';
```

For more information about MySQL privileges, refer to [MySQL User Account Management](#).

On system BETA (the Connector/ODBC client), follow these steps:

1. Configure a Connector/ODBC DSN using parameters that match the server, database and authentication information that you have just configured on system ALPHA.

Parameter	Value	Comment
DSN	remote_test	A name to identify the connection.
SERVER	ALPHA	The address of the remote server.
DATABASE	test	The name of the default database.
USER	myuser	The user name configured for access to this database.
PASSWORD	mypassword	The password for <code>myuser</code> .

2. Using an ODBC-capable application, such as Microsoft Office, connect to the MySQL server using the DSN you have just created. If the connection fails, use tracing to examine the connection process. See [Section 3.8, “Getting an ODBC Trace File”](#), for more information.

## 4.3. Connector/ODBC and Third-Party ODBC Tools

Once you have configured your Connector/ODBC DSN, you can access your MySQL database through any application that supports the ODBC interface, including programming languages and third-party applications. This section contains guides and help on using Connector/ODBC with various ODBC-compatible tools and applications, including Microsoft Word, Microsoft Excel and Adobe/Macromedia ColdFusion.

Connector/ODBC has been tested with the following applications.

Publisher	Application	Notes
Adobe	ColdFusion	Formerly Macromedia ColdFusion
Borland	C++ Builder	
	Builder 4	
	Delphi	
Business Objects	Crystal Reports	
Claris	Filemaker Pro	
Corel	Paradox	
Computer Associates	Visual Objects	Also known as CAVO
	AllFusion ERwin Data Modeler	
Gupta	Team Developer	Previously known as Centura Team Developer; Gupta SQL/Windows
Gensym	G2-ODBC Bridge	
Inline	iHTML	
Lotus	Notes	Versions 4.5 and 4.6
Microsoft	Access	
	Excel	

	Visio Enterprise	
	Visual C++	
	Visual Basic	
	ODBC.NET	Using C#, Visual Basic, C++
	FoxPro	
	Visual Interdev	
OpenOffice.org	OpenOffice.org	
Perl	DBD::ODBC	
Pervasive Software	DataJunction	
Sambar Technologies	Sambar Server	
SPSS	SPSS	
SoftVelocity	Clarion	
SQLExpress	SQLExpress for Xbase++	
Sun	StarOffice	
SunSystems	Vision	
Sybase	PowerBuilder	
	PowerDesigner	
theKompany.com	Data Architect	

If you know of any other applications that work with Connector/ODBC, please send mail to [myodbc@lists.mysql.com](mailto:myodbc@lists.mysql.com) about them.

## 4.4. Using Connector/ODBC with Microsoft Access

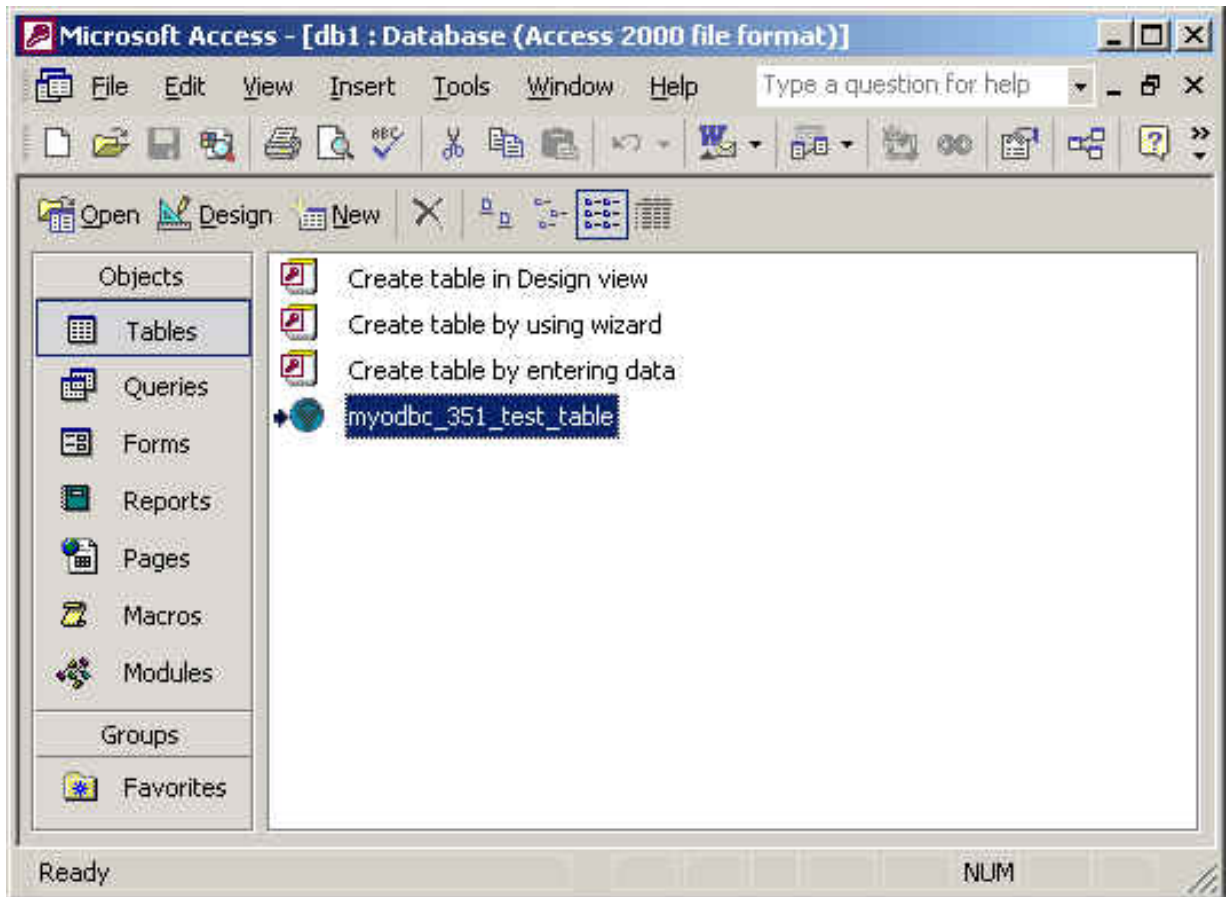
You can use MySQL database with Microsoft Access using Connector/ODBC. The MySQL database can be used as an import source, an export source, or as a linked table for direct use within an Access application, so you can use Access as the front-end interface to a MySQL database.

### 4.4.1. Exporting Access Data to MySQL

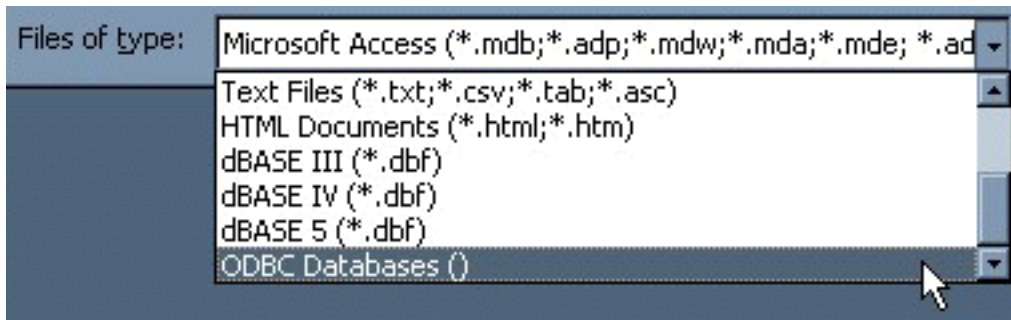
To export a table of data from an Access database to MySQL, follow these instructions:

1. When you open an Access database or an Access project, a Database window appears. It displays shortcuts for creating new database objects and opening existing objects.





2. Click the name of the [table](#) or [query](#) you want to export, and then in the **File** menu, select **Export**.
3. In the **Export Object Type Object name To** dialog box, in the **Save As Type** box, select **ODBC Databases ()** as shown here:



4. In the **Export** dialog box, enter a name for the file (or use the suggested name), and then select **OK**.
5. The **Select Data Source** dialog box is displayed; it lists the defined data sources for any ODBC drivers installed on your computer. Click either the **File Data Source** or **Machine Data Source** tab, and then double-click the **Connector/ODBC** or **Connector/ODBC 3.51** data source that you want to export to. To define a new data source for **Connector/ODBC**, please [Section 3.3, "Configuring a Connector/ODBC DSN on Windows"](#).

#### Note

Ensure that the information that you are exporting to the MySQL table is valid for the corresponding MySQL data types. Values that are outside of the supported range of the MySQL data type but valid within Access may trigger an "overflow" error during the export.

Microsoft Access connects to the MySQL Server through this data source and exports new tables and or data.

## 4.4.2. Importing MySQL Data to Access

To import a table or tables from MySQL to Access, follow these instructions:

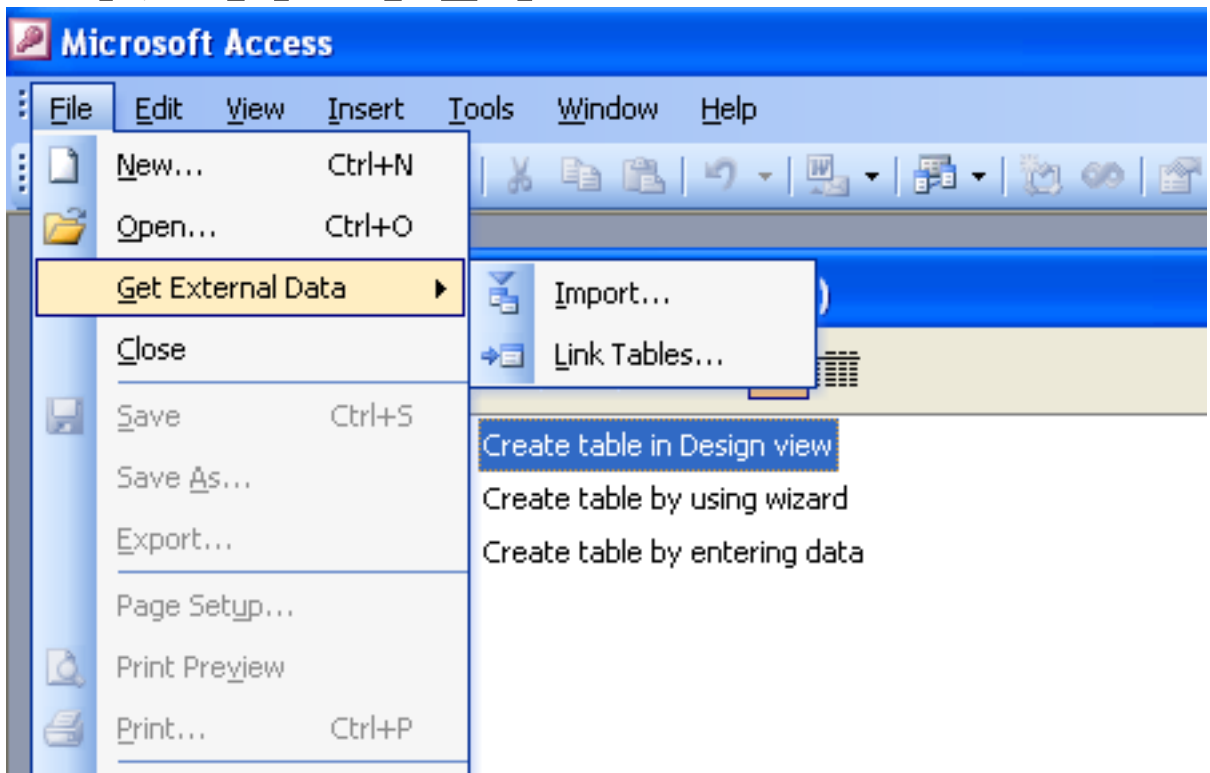
1. Open a database, or switch to the Database window for the open database.
2. To import tables, on the **File** menu, point to **Get External Data**, and then click **Import**.
3. In the **Import** dialog box, in the Files Of Type box, select **ODBC DATABASES ()**. The Select Data Source dialog box lists the defined data sources **THE SELECT DATA SOURCE** dialog box is displayed; it lists the defined data source names.
4. If the ODBC data source that you selected requires you to log on, enter your login ID and password (additional information might also be required), and then click **OK**.
5. Microsoft Access connects to the MySQL server through **ODBC data source** and displays the list of tables that you can **import**.
6. Click each table that you want to **import**, and then click **OK**.

### 4.4.3. Using Microsoft Access as a Front-end to MySQL

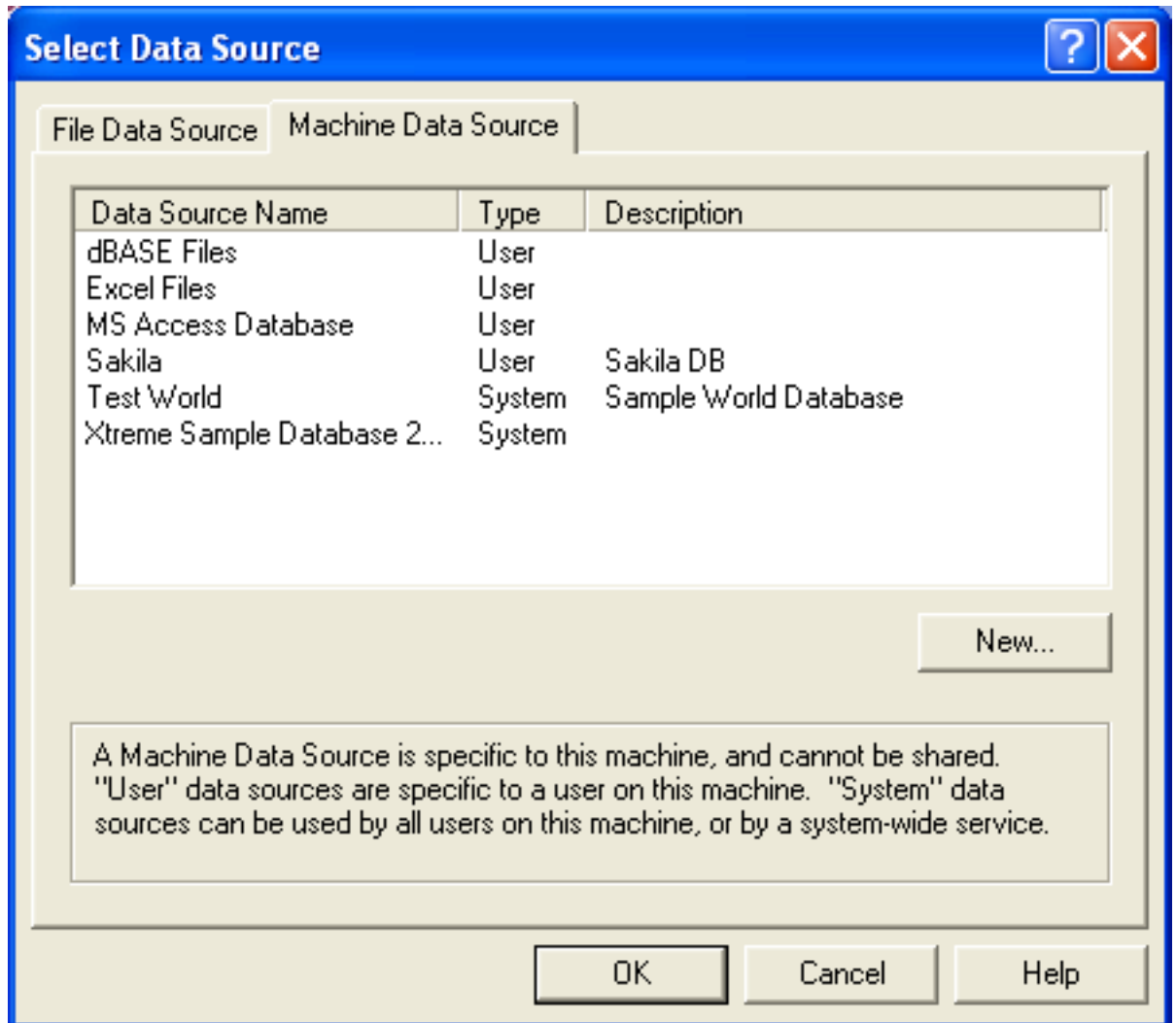
You can use Microsoft Access as a front end to a MySQL database by linking tables within your Microsoft Access database to tables that exist within your MySQL database. When a query is requested on a table within Access, ODBC is used to execute the queries on the MySQL database instead.

**To create a linked table:**

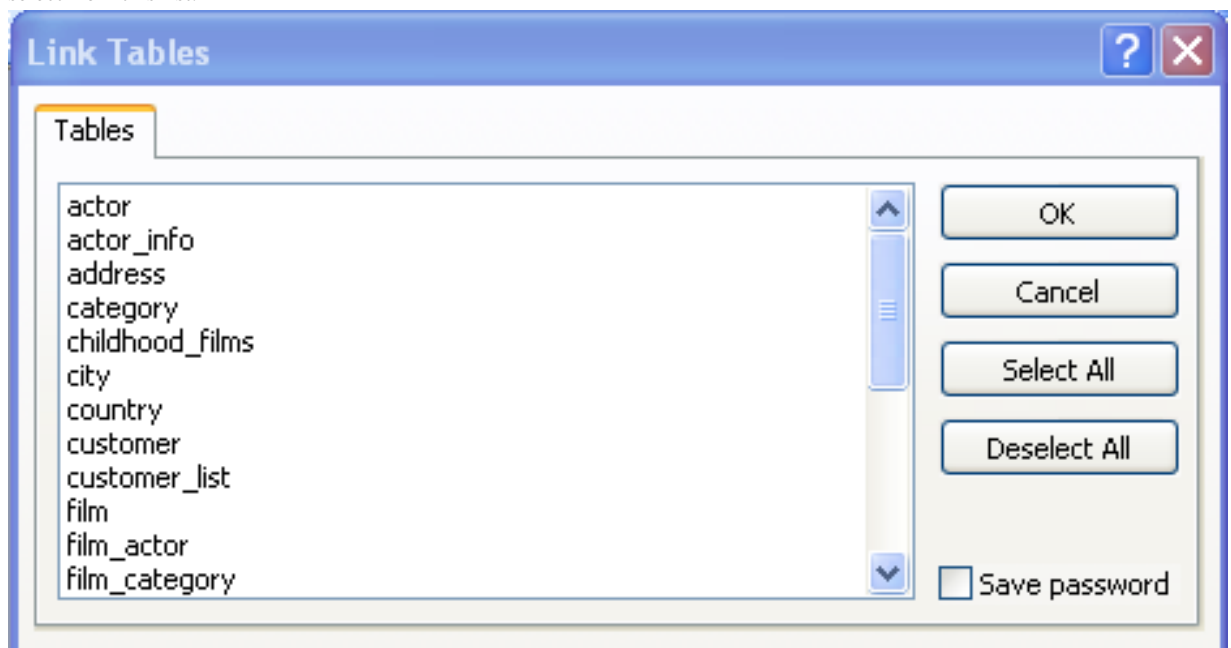
1. Open the Access database that you want to link to MySQL.
2. From the **FILE**, choose **GET EXTERNAL DATA->LINK TABLES**.



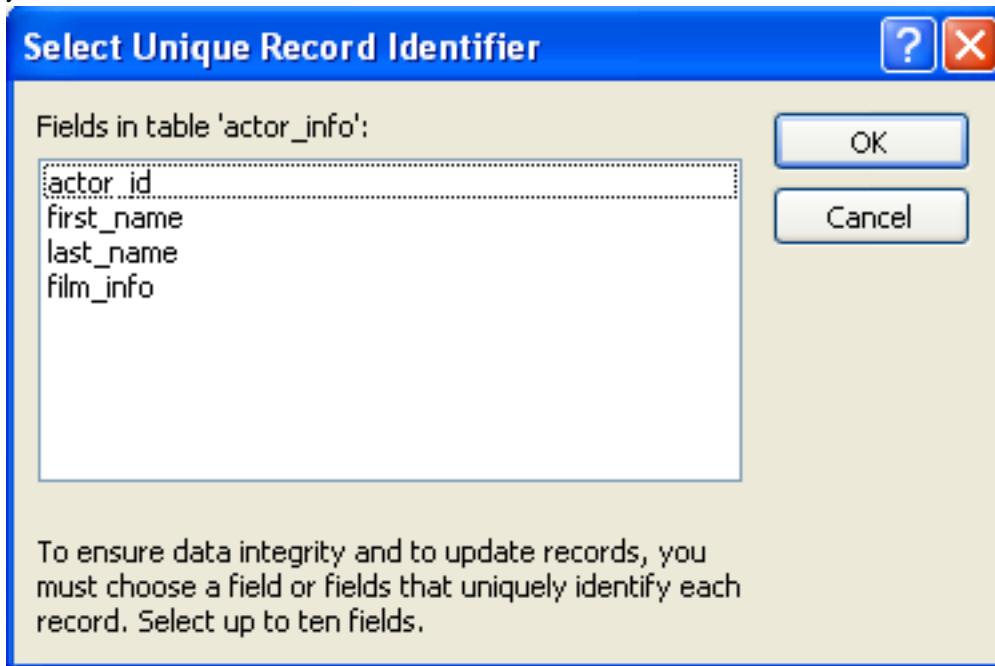
3. From the browser, choose **ODBC DATABASES ()** from the **FILES OF TYPE** pop-up.
4. In the **SELECT DATA SOURCE** window, choose an existing DSN, either from a **FILE DATA SOURCE** or **MACHINE DATA SOURCE**. You can also create a new DSN using the **NEW...** button. For more information on creating a DSN see [Section 3.3](#), "Configuring a Connector/ODBC DSN on Windows".



5. In the **LINK TABLES** dialog, select one or more tables from the MySQL database. A link will be created to each table that you select from this list.



- If Microsoft Access is unable to determine the unique record identifier for a table automatically then it may ask you to confirm the column, or combination of columns, to be used to uniquely identify each row from the source table. Select the columns you want to use and click OK.



Once the process has been completed, you can now build interfaces and queries to the linked tables just as you would for any Access database.

Use the following procedure to view or to refresh links when the structure or location of a linked table has changed. The Linked Table Manager lists the paths to all currently linked tables.

**To view or refresh links:**

- Open the database that contains links to MySQL tables.
- On the **Tools** menu, point to **Add-ins (Database Utilities)** in Access 2000 or newer, and then click **Linked Table Manager**.
- Select the check box for the tables whose links you want to refresh.
- Click OK to refresh the links.

Microsoft Access confirms a successful refresh or, if the table wasn't found, displays the **Select New Location of <table name>** dialog box in which you can specify its the table's new location. If several selected tables have moved to the new location that you specify, the Linked Table Manager searches that location for all selected tables, and updates all links in one step.

**To change the path for a set of linked tables:**

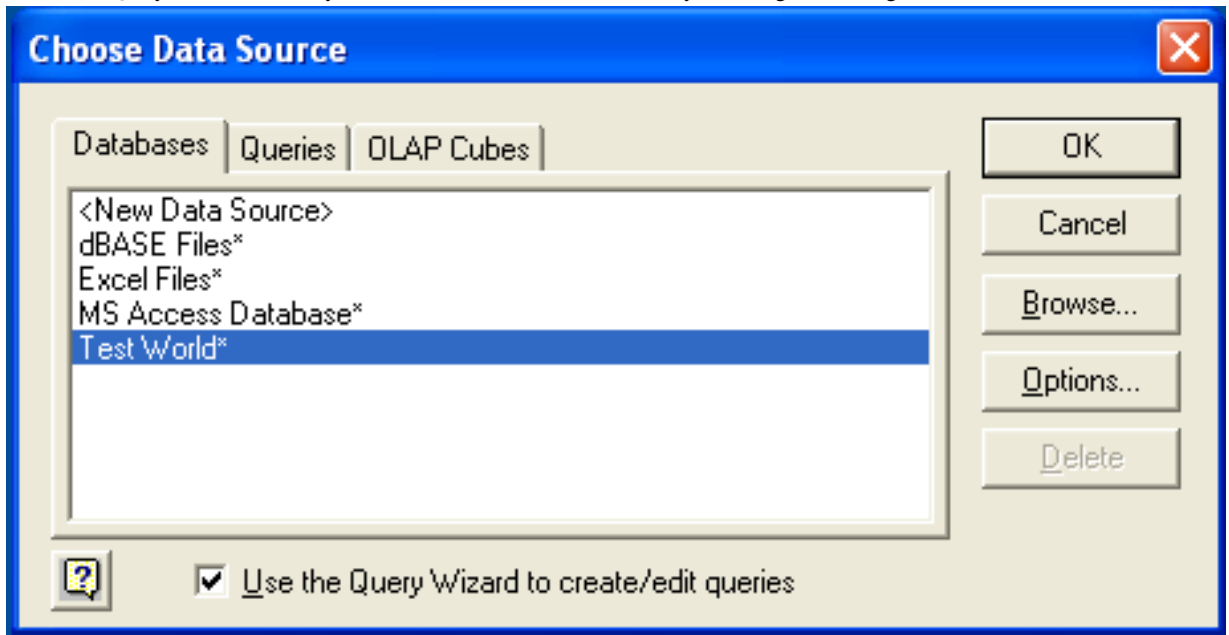
- Open the database that contains links to tables.
- On the **Tools** menu, point to **Add-ins (Database Utilities)** in Access 2000 or newer, and then click **Linked Table Manager**.
- Select the **Always Prompt For A New Location** check box.
- Select the check box for the tables whose links you want to change, and then click **OK**.
- In the **Select New Location of <table name>** dialog box, specify the new location, click **Open**, and then click **OK**.

## 4.5. Using Connector/ODBC with Microsoft Word or Excel

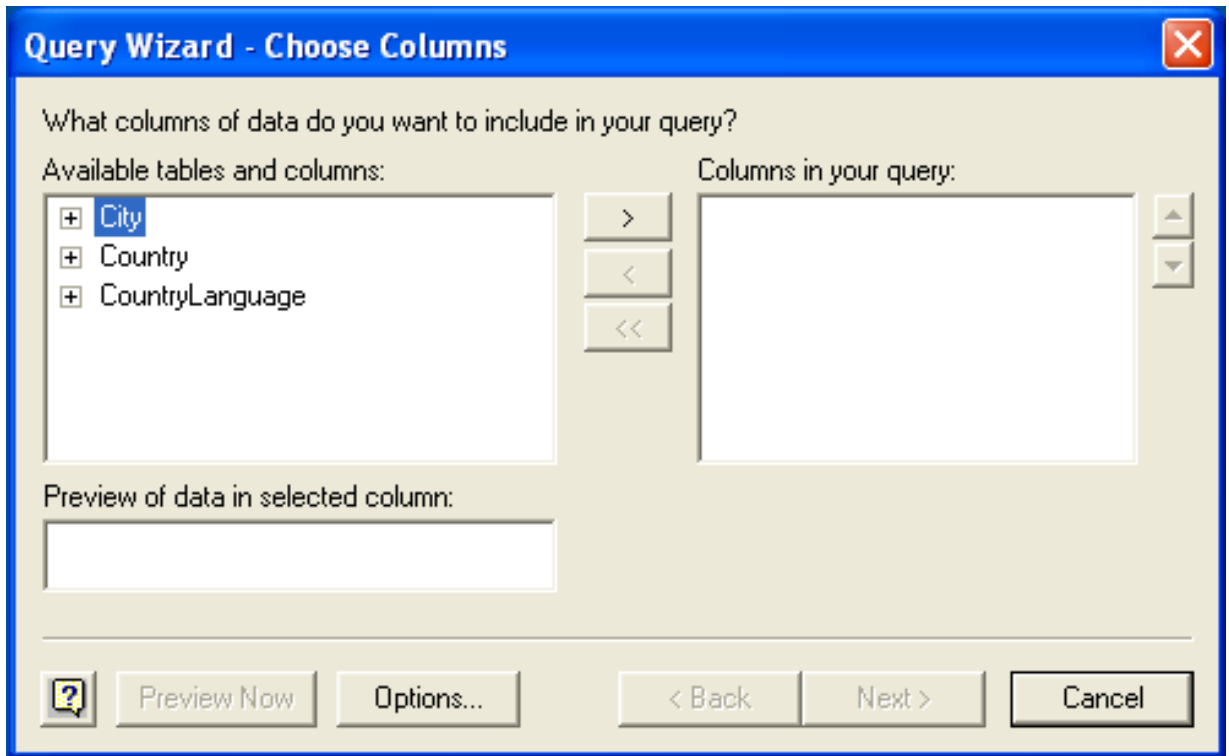
You can use Microsoft Word and Microsoft Excel to access information from a MySQL database using Connector/ODBC. Within Microsoft Word, this facility is most useful when importing data for mailmerge, or for tables and data to be included in reports. Within Microsoft Excel, you can execute queries on your MySQL server and import the data directly into an Excel Worksheet, presenting the data as a series of rows and columns.

With both applications, data is accessed and imported into the application using Microsoft Query, which enables you to execute a query through an ODBC source. You use Microsoft Query to build the SQL statement to be executed, selecting the tables, fields, selection criteria and sort order. For example, to insert information from a table in the World test database into an Excel spreadsheet, using the DSN samples shown in [Chapter 3, Connector/ODBC Configuration](#):

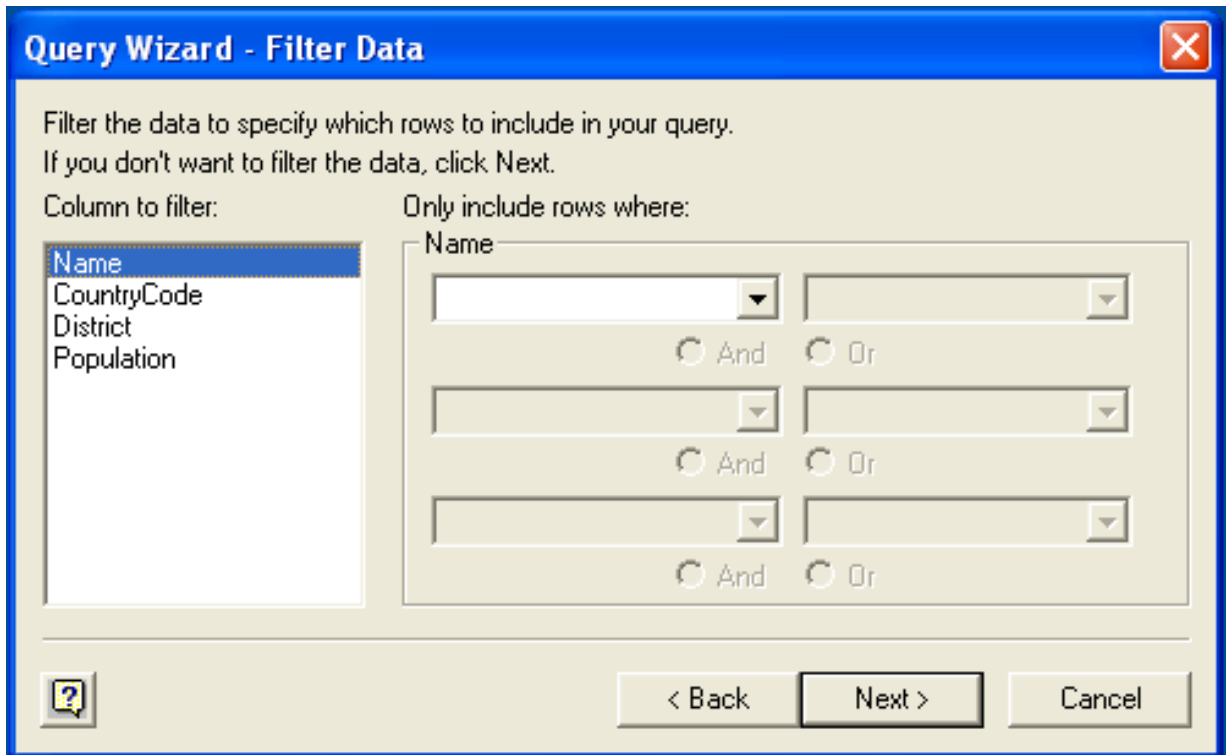
1. Create a new Worksheet.
2. From the **Data** menu, choose **Import External Data**, and then select **New Database Query**.
3. Microsoft Query will start. First, you need to choose the data source, by selecting an existing Data Source Name.



4. Within the **Query Wizard**, you must choose the columns that you want to import. The list of tables available to the user configured through the DSN is shown on the left, the columns that will be added to your query are shown on the right. The columns you choose are equivalent to those in the first section of a **SELECT** query. Click **NEXT** to continue.



5. You can filter rows from the query (the equivalent of a `WHERE` clause) using the `Filter Data` dialog. Click `NEXT` to continue.



6. Select an (optional) sort order for the data. This is equivalent to using a `ORDER BY` clause in your SQL query. You can select up to three fields for sorting the information returned by the query. Click `NEXT` to continue.

**Query Wizard - Sort Order**

Specify how you want your data sorted.  
If you don't want to sort the data, click Next.

Sort by  
Name  Ascending  Descending

Then by  Ascending  Descending

Then by  Ascending  Descending

7. Select the destination for your query. You can select to return the data Microsoft Excel, where you can choose a worksheet and cell where the data will be inserted; you can continue to view the query and results within Microsoft Query, where you can edit the SQL query and further filter and sort the information returned; or you can create an OLAP Cube from the query, which can then be used directly within Microsoft Excel. Click FINISH.

**Query Wizard - Finish**

What would you like to do next?

Return Data to Microsoft Office Excel

View data or edit query in Microsoft Query

Create an OLAP Cube from this query

The same process can be used to import data into a Word document, where the data will be inserted as a table. This can be used for mail merge purposes (where the field data is read from a Word table), or where you want to include data and reports within a report or other document.

## 4.6. Using Connector/ODBC with Crystal Reports

Crystal Reports can use an ODBC DSN to connect to a database from which you to extract data and information for reporting purposes.

### Note

There is a known issue with certain versions of Crystal Reports where the application is unable to open and browse tables and fields through an ODBC connection. Before using Crystal Reports with MySQL, please ensure that you have update to the latest version, including any outstanding service packs and hotfixes. For more information on this issue, see the [Business\) Objects Knowledgebase](#) for more information.

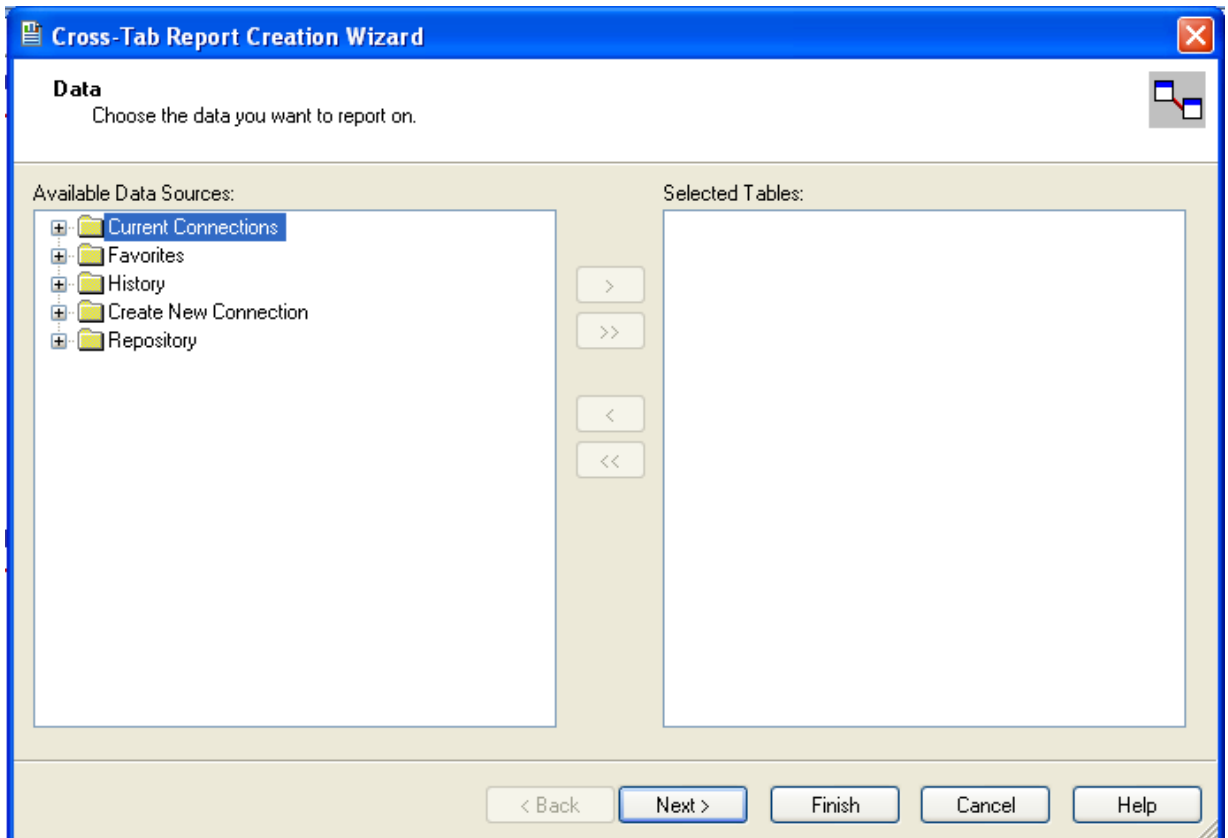
For example, to create a simple crosstab report within Crystal Reports XI, you should follow these steps:

1. Create a DSN using the [Data Sources \(ODBC\)](#) tool. You can either specify a complete database, including user name and password, or you can build a basic DSN and use Crystal Reports to set the user name and password.

For the purposes of this example, a DSN that provides a connection to an instance of the MySQL Sakila sample database has been created.

2. Open Crystal Reports and create a new project, or an open an existing reporting project into which you want to insert data from your MySQL data source.
3. Start the Cross-Tab Report Wizard, either by clicking on the option on the Start Page. Expand the **CREATE NEW CONNECTION** folder, then expand the **ODBC (RDO)** folder to obtain a list of ODBC data sources.

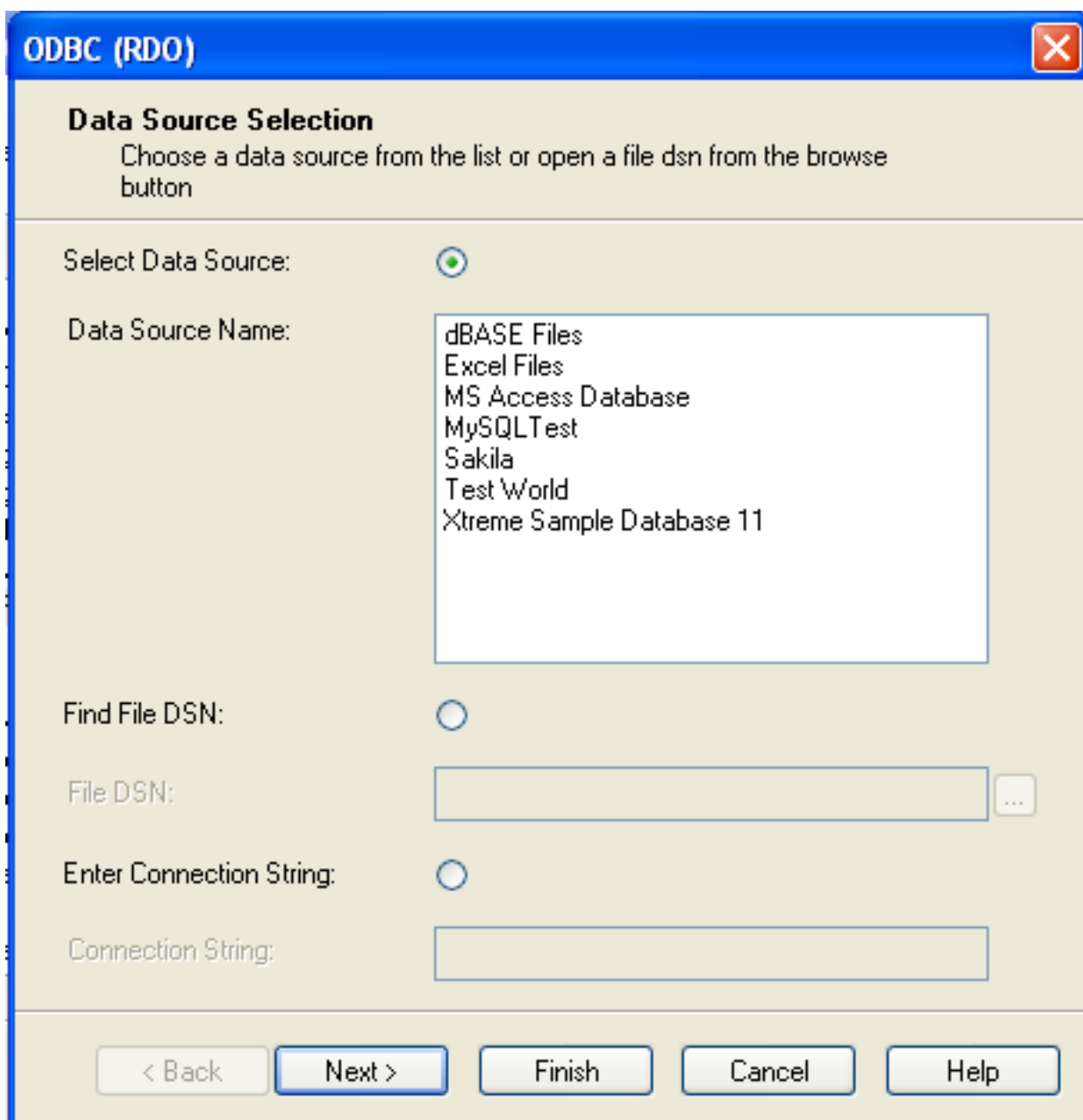
You will be asked to select a data source.



4. When you first expand the **ODBC (RDO)** folder you will be presented the Data Source Selection screen. From here you can select either a pre-configured DSN, open a file-based DSN or enter and manual connection string. For this example, the **SAKILA** DSN will be used.

If the DSN contains a user name/password combination, or you want to use different authentication credentials, click **NEXT** to enter the user name and password that you want to use. Otherwise, click **FINISH** to continue the data source selection wizard.

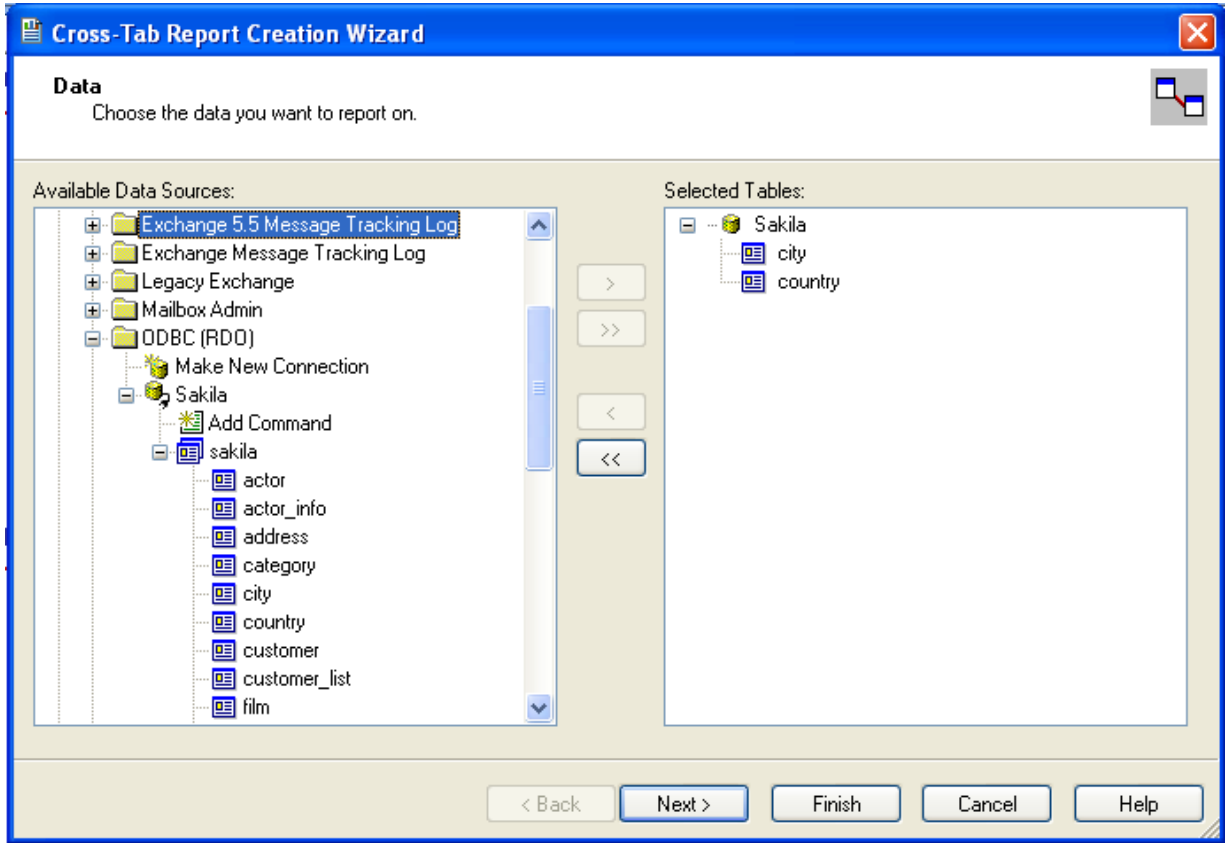




5. You will be returned the Cross-Tab Report Creation Wizard. You now need to select the database and tables that you want to include in your report. For our example, we will expand the selected Sakila database. Click the `city` table and use the `>` button to add the table to the report. Then repeat the action with the `country` table. Alternatively you can select multiple tables and add them to the report.

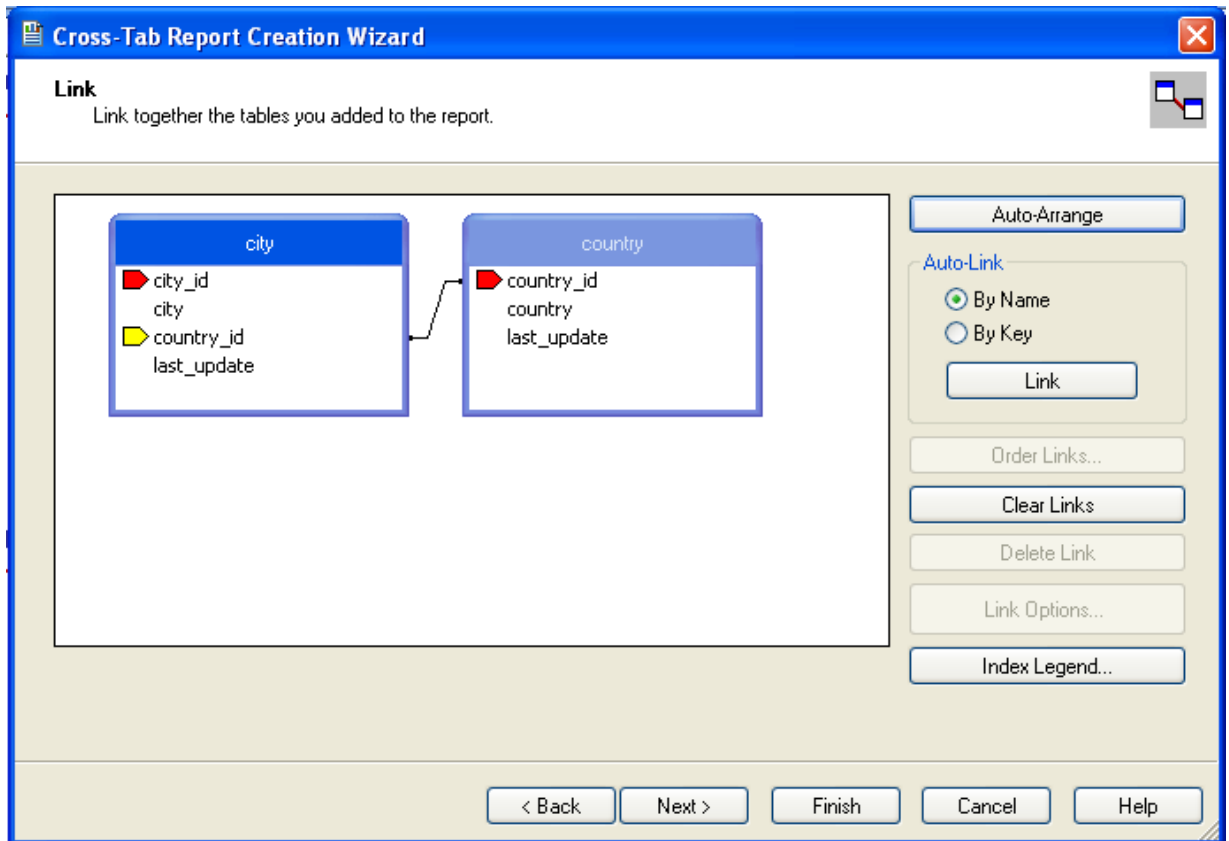
Finally, you can select the parent **SAKILA** resource and add of the tables to the report.

Once you have selected the tables you want to include, click NEXT to continue.



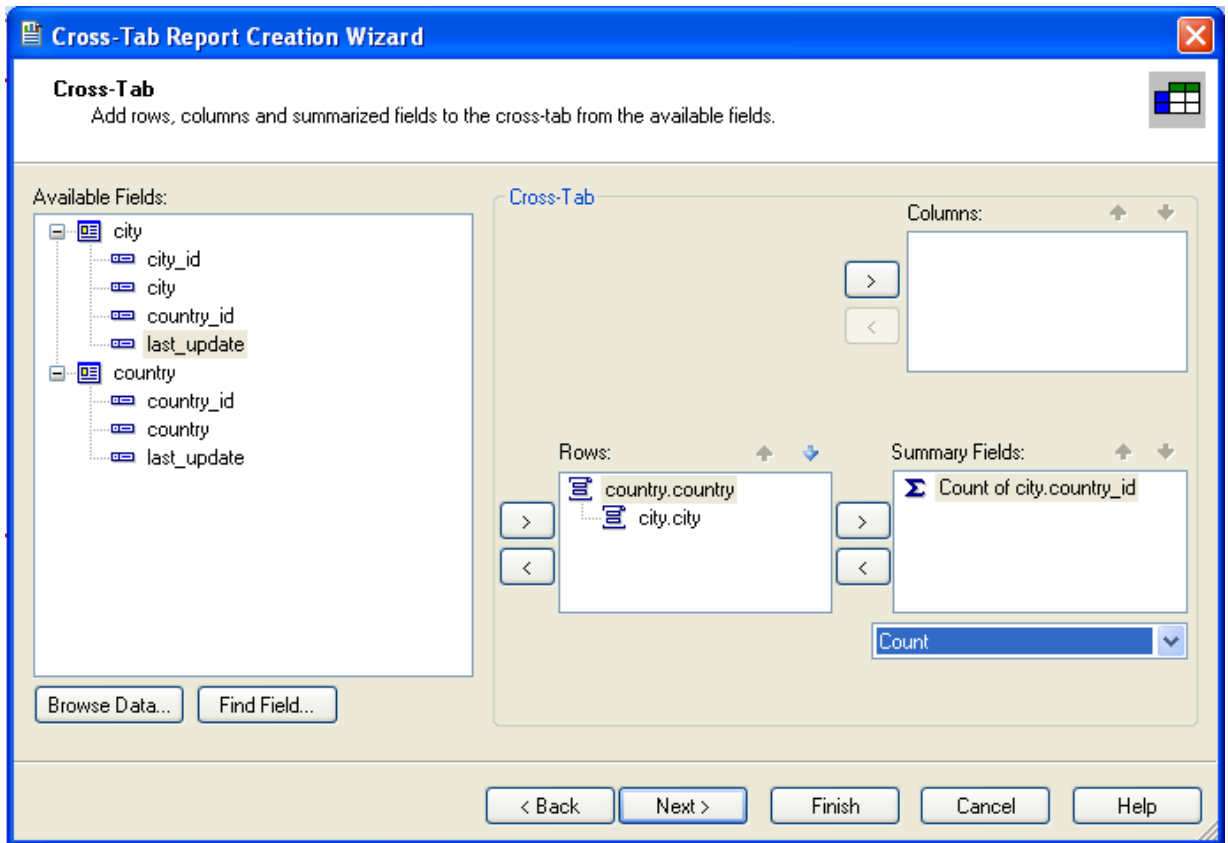
6. Crystal Reports will now read the table definitions and automatically identify the links between the tables. The identification of links between tables enables Crystal Reports to automatically lookup and summarize information based on all the tables in the database according to your query. If Crystal Reports is unable to perform the linking itself, you can manually create the links between fields in the tables you have selected.

Click NEXT to continue the process.



7. You can now select the columns and rows that you wish to include within the Cross-Tab report. Drag and drop or use the > buttons to add fields to each area of the report. In the example shown, we will report on cities, organized by country, incorporating a count of the number of cities within each country. If you want to browse the data, select a field and click the BROWSE DATA... button.

Click NEXT to create a graph of the results. Since we are not creating a graph from this data, click FINISH to generate the report.



8. The finished report will be shown, a sample of the output from the Sakila sample database is shown below.

		<b>Total</b>
<b>Total</b>		600
<b>Afghanistan</b>	<b>Total</b>	1
	<b>Kabul</b>	1
<b>Algeria</b>	<b>Total</b>	3
	<b>Batna</b>	1
	<b>Bchar</b>	1
	<b>Skikda</b>	1
<b>American Samoa</b>	<b>Total</b>	1
	<b>Tafuna</b>	1
<b>Angola</b>	<b>Total</b>	2
	<b>Benguela</b>	1
	<b>Namibe</b>	1
<b>Anguilla</b>	<b>Total</b>	1
	<b>South Hill</b>	1
<b>Argentina</b>	<b>Total</b>	13
	<b>Almirante Brow</b>	1

Once the ODBC connection has been opened within Crystal Reports, you can browse and add any fields within the available tables into your reports.

## 4.7. Connector/ODBC Programming

With a suitable ODBC Manager and the Connector/ODBC driver installed, any programming language or environment that can support ODBC should be able to connect to a MySQL database through Connector/ODBC.

This includes, but is certainly not limited to, Microsoft support languages (including Visual Basic, C# and interfaces such as ODBC.NET), Perl (through the DBI module, and the DBD::ODBC driver).

### 4.7.1. Using Connector/ODBC with Visual Basic Using ADO, DAO and RDO

This section contains simple examples of the use of MySQL ODBC 3.51 Driver with ADO, DAO and RDO.

#### 4.7.1.1. ADO: `rs.addNew`, `rs.delete`, and `rs.update`

The following ADO (ActiveX Data Objects) example creates a table `my_ado` and demonstrates the use of `rs.addNew`, `rs.delete`, and `rs.update`.

```
Private Sub myodbc_ado_Click()
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim fld As ADODB.Field
Dim sql As String
'connect to MySQL server using MySQL ODBC 3.51 Driver
Set conn = New ADODB.Connection
conn.ConnectionString = "DRIVER={MySQL ODBC 3.51 Driver};" _
& "SERVER=localhost;" _
& "DATABASE=test;" _
& "UID=venu;PWD=venu; OPTION=3"
conn.Open
'create table
conn.Execute "DROP TABLE IF EXISTS my_ado"
conn.Execute "CREATE TABLE my_ado(id int not null primary key, name varchar(20)," _
& "txt text, dt date, tm time, ts timestamp)"
'direct insert
conn.Execute "INSERT INTO my_ado(id,name,txt) values(1,100,'venu')"
conn.Execute "INSERT INTO my_ado(id,name,txt) values(2,200,'MySQL')"
conn.Execute "INSERT INTO my_ado(id,name,txt) values(3,300,'Delete')"
Set rs = New ADODB.Recordset
rs.CursorLocation = adUseServer
'fetch the initial table ..
rs.Open "SELECT * FROM my_ado", conn
Debug.Print rs.RecordCount
rs.MoveFirst
Debug.Print String(50, "-") & "Initial my_ado Result Set " & String(50, "-")
For Each fld In rs.Fields
Debug.Print fld.Name,
Next
Debug.Print
Do Until rs.EOF
For Each fld In rs.Fields
Debug.Print fld.Value,
Next
rs.MoveNext
Debug.Print
Loop
rs.Close
'rs insert
rs.Open "select * from my_ado", conn, adOpenDynamic, adLockOptimistic
rs.AddNew
rs!Name = "Monty"
rs!txt = "Insert row"
rs.Update
rs.Close
'rs update
rs.Open "SELECT * FROM my_ado"
rs!Name = "update"
rs!txt = "updated-row"
rs.Update
rs.Close
'rs update second time..
rs.Open "SELECT * FROM my_ado"
rs!Name = "update"
rs!txt = "updated-second-time"
rs.Update
rs.Close
'rs delete
rs.Open "SELECT * FROM my_ado"
rs.MoveNext
rs.MoveNext
rs.Delete
rs.Close
'fetch the updated table ..
rs.Open "SELECT * FROM my_ado", conn
Debug.Print rs.RecordCount
rs.MoveFirst
Debug.Print String(50, "-") & "Updated my_ado Result Set " & String(50, "-")
For Each fld In rs.Fields
Debug.Print fld.Name,
Next
Debug.Print
Do Until rs.EOF
For Each fld In rs.Fields
Debug.Print fld.Value,
Next
rs.MoveNext
Debug.Print
Loop
rs.Close
conn.Close
End Sub
```

#### 4.7.1.2. DAO: `rs.addNew`, `rs.update`, and Scrolling

The following DAO (Data Access Objects) example creates a table `my_dao` and demonstrates the use of `rs.addNew`, `rs.update`, and result set scrolling.

```
Private Sub myodbc_dao_Click()
Dim ws As Workspace
Dim conn As Connection
Dim queryDef As queryDef
Dim str As String
'connect to MySQL using MySQL ODBC 3.51 Driver
Set ws = DBEngine.CreateWorkspace("", "venu", "venu", dbUseODBC)
str = "odbc;DRIVER={MySQL ODBC 3.51 Driver};_"
& "SERVER=localhost;_"
& "DATABASE=test;_"
& "UID=venu;PWD=venu; OPTION=3"
Set conn = ws.OpenConnection("test", dbDriverNoPrompt, False, str)
'Create table my_dao
Set queryDef = conn.CreateQueryDef("", "drop table if exists my_dao")
queryDef.Execute
Set queryDef = conn.CreateQueryDef("", "create table my_dao(Id INT AUTO_INCREMENT PRIMARY KEY, " _
& "Ts TIMESTAMP(14) NOT NULL, Name varchar(20), Id2 INT)")
queryDef.Execute
'Insert new records using rs.addNew
Set rs = conn.OpenRecordset("my_dao")
Dim i As Integer
For i = 10 To 15
rs.AddNew
rs!Name = "insert record" & i
rs!Id2 = i
rs.Update
Next i
rs.Close
'rs update..
Set rs = conn.OpenRecordset("my_dao")
rs.Edit
rs!Name = "updated-string"
rs.Update
rs.Close
'fetch the table back...
Set rs = conn.OpenRecordset("my_dao", dbOpenDynamic)
str = "Results:"
rs.MoveFirst
While Not rs.EOF
str = " " & rs!Id & " , " & rs!Name & " , " & rs!Ts & " , " & rs!Id2
Debug.Print "DATA:" & str
rs.MoveNext
Wend
'rs Scrolling
rs.MoveFirst
str = " FIRST ROW: " & rs!Id & " , " & rs!Name & " , " & rs!Ts & " , " & rs!Id2
Debug.Print str
rs.MoveLast
str = " LAST ROW: " & rs!Id & " , " & rs!Name & " , " & rs!Ts & " , " & rs!Id2
Debug.Print str
rs.MovePrevious
str = " LAST-1 ROW: " & rs!Id & " , " & rs!Name & " , " & rs!Ts & " , " & rs!Id2
Debug.Print str
'free all resources
rs.Close
queryDef.Close
conn.Close
ws.Close
End Sub
```

#### 4.7.1.3. RDO: `rs.addNew` and `rs.update`

The following RDO (Remote Data Objects) example creates a table `my_rdo` and demonstrates the use of `rs.addNew` and `rs.update`.

```
Dim rs As rdoResultset
Dim cn As New rdoConnection
Dim cl As rdoColumn
Dim SQL As String
'cn.Connect = "DSN=test;"
cn.Connect = "DRIVER={MySQL ODBC 3.51 Driver};_"
& "SERVER=localhost;_"
& "DATABASE=test;_"
& "UID=venu;PWD=venu; OPTION=3"
cn.CursorDriver = rdUseOdbc
cn.EstablishConnection rdDriverPrompt
'drop table my_rdo
SQL = "drop table if exists my_rdo"
cn.Execute SQL, rdExecDirect
'create table my_rdo
SQL = "create table my_rdo(id int, name varchar(20))"
cn.Execute SQL, rdExecDirect
'insert - direct
SQL = "insert into my_rdo values (100,'venu')"
cn.Execute SQL, rdExecDirect
SQL = "insert into my_rdo values (200,'MySQL')"
cn.Execute SQL, rdExecDirect
'rs insert
SQL = "select * from my_rdo"
```

```

Set rs = cn.OpenResultset(SQL, rdOpenStatic, rdConcurRowVer, rdExecDirect)
rs.AddNew
rs!id = 300
rs!Name = "Insert1"
rs.Update
rs.Close
'rs insert
SQL = "select * from my_rdo"
Set rs = cn.OpenResultset(SQL, rdOpenStatic, rdConcurRowVer, rdExecDirect)
rs.AddNew
rs!id = 400
rs!Name = "Insert 2"
rs.Update
rs.Close
'rs update
SQL = "select * from my_rdo"
Set rs = cn.OpenResultset(SQL, rdOpenStatic, rdConcurRowVer, rdExecDirect)
rs.Edit
rs!id = 999
rs!Name = "updated"
rs.Update
rs.Close
'fetch back...
SQL = "select * from my_rdo"
Set rs = cn.OpenResultset(SQL, rdOpenStatic, rdConcurRowVer, rdExecDirect)
Do Until rs.EOF
For Each cl In rs.rdoColumns
Debug.Print cl.Value,
Next
rs.MoveNext
Debug.Print
Loop
Debug.Print "Row count="; rs.RowCount
'close
rs.Close
cn.Close
End Sub

```

## 4.7.2. Using Connector/ODBC with .NET

This section contains simple examples that demonstrate the use of Connector/ODBC drivers with ODBC.NET.

### 4.7.2.1. Using Connector/ODBC with ODBC.NET and C# (C sharp)

The following sample creates a table `my_odbc_net` and demonstrates its use in C#.

```

/**
 * @sample      : mycon.cs
 * @purpose     : Demo sample for ODBC.NET using Connector/ODBC
 * @author      : Venu, <myodbc@lists.mysql.com>
 *
 * (C) Copyright MySQL AB, 1995-2006
 *
 */
/* build command
 *
 * csc /t:exe
 *      /out:mycon.exe mycon.cs
 *      /r:Microsoft.Data.Odbc.dll
 */
using Console = System.Console;
using Microsoft.Data.Odbc;
namespace myodbc3
{
    class mycon
    {
        static void Main(string[] args)
        {
            try
            {
                //Connection string for Connector/ODBC 3.51
                string MyConString = "DRIVER={MySQL ODBC 3.51 Driver};" +
                    "SERVER=localhost;" +
                    "DATABASE=test;" +
                    "UID=venu;" +
                    "PASSWORD=venu;" +
                    "OPTION=3";

                //Connect to MySQL using Connector/ODBC
                OdbcConnection MyConnection = new OdbcConnection(MyConString);
                MyConnection.Open();
                Console.WriteLine("\n !!! success, connected successfully !!!\n");
                //Display connection information
                Console.WriteLine("Connection Information:");
                Console.WriteLine("\tConnection String:" +
                    MyConnection.ConnectionString);
                Console.WriteLine("\tConnection Timeout:" +
                    MyConnection.ConnectionTimeout);
                Console.WriteLine("\tDatabase:" +
                    MyConnection.Database);
                Console.WriteLine("\tDataSource:" +
                    MyConnection.DataSource);
            }
            catch { }
        }
    }
}

```





```

' /out:myvb.exe
' /r:Microsoft.Data.Odbc.dll
' /r:System.dll
' /r:System.Data.dll
Imports Microsoft.Data.Odbc
Imports System
Module myvb
Sub Main()
Try
'Connector/ODBC 3.51 connection string
Dim MyConString As String = "DRIVER={MySQL ODBC 3.51 Driver};" & _
"SERVER=localhost;" & _
"DATABASE=test;" & _
"UID=venu;" & _
"PASSWORD=venu;" & _
"OPTION=3;"
'Connection
Dim MyConnection As New OdbcConnection(MyConString)
MyConnection.Open()
Console.WriteLine("Connection State:" & MyConnection.State.ToString)
'Drop
Console.WriteLine("Dropping table")
Dim MyCommand As New OdbcCommand()
MyCommand.Connection = MyConnection
MyCommand.CommandText = "DROP TABLE IF EXISTS my_vb_net"
MyCommand.ExecuteNonQuery()
'Create
Console.WriteLine("Creating...")
MyCommand.CommandText = "CREATE TABLE my_vb_net(id int, name varchar(30))"
MyCommand.ExecuteNonQuery()
'Insert
MyCommand.CommandText = "INSERT INTO my_vb_net VALUES(10,'venu')"
Console.WriteLine("INSERT, Total rows affected:" & _
MyCommand.ExecuteNonQuery())
'Insert
MyCommand.CommandText = "INSERT INTO my_vb_net VALUES(20,'mysql')"
Console.WriteLine("INSERT, Total rows affected:" & _
MyCommand.ExecuteNonQuery())
'Insert
MyCommand.CommandText = "INSERT INTO my_vb_net VALUES(20,'mysql')"
Console.WriteLine("INSERT, Total rows affected:" & _
MyCommand.ExecuteNonQuery())
'Insert
MyCommand.CommandText = "INSERT INTO my_vb_net(id) VALUES(30)"
Console.WriteLine("INSERT, Total rows affected:" & _
MyCommand.ExecuteNonQuery())
'Update
MyCommand.CommandText = "UPDATE my_vb_net SET id=999 WHERE id=20"
Console.WriteLine("Update, Total rows affected:" & _
MyCommand.ExecuteNonQuery())
'COUNT(*)
MyCommand.CommandText = "SELECT COUNT(*) as TRows FROM my_vb_net"
Console.WriteLine("Total Rows:" & MyCommand.ExecuteScalar())
'Select
Console.WriteLine("Select * FROM my_vb_net")
MyCommand.CommandText = "SELECT * FROM my_vb_net"
Dim MyDataReader As OdbcDataReader
MyDataReader = MyCommand.ExecuteReader
While MyDataReader.Read
If MyDataReader("name") Is DBNull.Value Then
Console.WriteLine("id = " & _
CStr(MyDataReader("id")) & " name = " & _
"NULL")
Else
Console.WriteLine("id = " & _
CStr(MyDataReader("id")) & " name = " & _
CStr(MyDataReader("name")))
End If
End While
'Catch ODBC Exception
Catch MyOdbcException As OdbcException
Dim i As Integer
Console.WriteLine(MyOdbcException.ToString)
'Catch program exception
Catch MyException As Exception
Console.WriteLine(MyException.ToString)
End Try
End Sub

```

---

# Chapter 5. Connector/ODBC Reference

This section provides reference material for the Connector/ODBC API, showing supported functions and methods, supported MySQL column types and the corresponding native type in Connector/ODBC, and the error codes returned by Connector/ODBC when a fault occurs.

## 5.1. Connector/ODBC API Reference

This section summarizes ODBC routines, categorized by functionality.

For the complete ODBC API reference, please refer to the ODBC Programmer's Reference at <http://msdn.microsoft.com/en-us/library/ms714177.aspx>.

An application can call `SQLGetInfo` function to obtain conformance information about Connector/ODBC. To obtain information about support for a specific function in the driver, an application can call `SQLGetFunctions`.

### Note

For backward compatibility, the Connector/ODBC 3.51 driver supports all deprecated functions.

The following tables list Connector/ODBC API calls grouped by task:

#### Connecting to a data source

Function name	C/ODBC 3.51	Standard	Purpose
<code>SQLAllocHandle</code>	Yes	ISO 92	Obtains an environment, connection, statement, or descriptor handle.
<code>SQLConnect</code>	Yes	ISO 92	Connects to a specific driver by data source name, user ID, and password.
<code>SQLDriverConnect</code>	Yes	ODBC	Connects to a specific driver by connection string or requests that the Driver Manager and driver display connection dialog boxes for the user.
<code>SQLAllocEnv</code>	Yes	Deprecated	Obtains an environment handle allocated from driver.
<code>SQLAllocConnect</code>	Yes	Deprecated	Obtains a connection handle

#### Obtaining information about a driver and data source

Function name	C/ODBC 3.51	Standard	Purpose
<code>SQLDataSources</code>	No	ISO 92	Returns the list of available data sources, handled by the Driver Manager
<code>SQLDrivers</code>	No	ODBC	Returns the list of installed drivers and their attributes, handles by Driver Manager
<code>SQLGetInfo</code>	Yes	ISO 92	Returns information about a specific driver and data source.
<code>SQLGetFunctions</code>	Yes	ISO 92	Returns supported driver functions.
<code>SQLGetTypeInfo</code>	Yes	ISO 92	Returns information about supported data types.

#### Setting and retrieving driver attributes

Function name	C/ODBC 3.51	Standard	Purpose
<code>SQLSetConnectAttr</code>	Yes	ISO 92	Sets a connection attribute.
<code>SQLGetConnectAttr</code>	Yes	ISO 92	Returns the value of a connection attribute.
<code>SQLSetConnectOption</code>	Yes	Deprecated	Sets a connection option
<code>SQLGetConnectOption</code>	Yes	Deprecated	Returns the value of a connection option
<code>SQLSetEnvAttr</code>	Yes	ISO 92	Sets an environment attribute.
<code>SQLGetEnvAttr</code>	Yes	ISO 92	Returns the value of an environment attribute.

<a href="#">SQLSetStmtAttr</a>	Yes	ISO 92	Sets a statement attribute.
<a href="#">SQLGetStmtAttr</a>	Yes	ISO 92	Returns the value of a statement attribute.
<a href="#">SQLSetStmtOption</a>	Yes	Deprecated	Sets a statement option
<a href="#">SQLGetStmtOption</a>	Yes	Deprecated	Returns the value of a statement option

### Preparing SQL requests

Function name	C/ODBC 3.51	Standard	Purpose
<a href="#">SQLAllocStmt</a>	Yes	Deprecated	Allocates a statement handle
<a href="#">SQLPrepare</a>	Yes	ISO 92	Prepares an SQL statement for later execution.
<a href="#">SQLBindParameter</a>	Yes	ODBC	Assigns storage for a parameter in an SQL statement.
<a href="#">SQLGetCursorName</a>	Yes	ISO 92	Returns the cursor name associated with a statement handle.
<a href="#">SQLSetCursorName</a>	Yes	ISO 92	Specifies a cursor name.
<a href="#">SQLSetScrollOptions</a>	Yes	ODBC	Sets options that control cursor behavior.

### Submitting requests

Function name	C/ODBC 3.51	Standard	Purpose
<a href="#">SQLExecute</a>	Yes	ISO 92	Executes a prepared statement.
<a href="#">SQLExecDirect</a>	Yes	ISO 92	Executes a statement
<a href="#">SQLNativeSql</a>	Yes	ODBC	Returns the text of an SQL statement as translated by the driver.
<a href="#">SQLDescribeParam</a>	Yes	ODBC	Returns the description for a specific parameter in a statement.
<a href="#">SQLNumParams</a>	Yes	ISO 92	Returns the number of parameters in a statement.
<a href="#">SQLParamData</a>	Yes	ISO 92	Used in conjunction with <a href="#">SQLPutData</a> to supply parameter data at execution time. (Useful for long data values.)
<a href="#">SQLPutData</a>	Yes	ISO 92	Sends part or all of a data value for a parameter. (Useful for long data values.)

### Retrieving results and information about results

Function name	C/ODBC 3.51	Standard	Purpose
<a href="#">SQLRowCount</a>	Yes	ISO 92	Returns the number of rows affected by an insert, update, or delete request.
<a href="#">SQLNumResultCols</a>	Yes	ISO 92	Returns the number of columns in the result set.
<a href="#">SQLDescribeCol</a>	Yes	ISO 92	Describes a column in the result set.
<a href="#">SQLColAttribute</a>	Yes	ISO 92	Describes attributes of a column in the result set.
<a href="#">SQLColAttributes</a>	Yes	Deprecated	Describes attributes of a column in the result set.
<a href="#">SQLFetch</a>	Yes	ISO 92	Returns multiple result rows.
<a href="#">SQLFetchScroll</a>	Yes	ISO 92	Returns scrollable result rows.
<a href="#">SQLExtendedFetch</a>	Yes	Deprecated	Returns scrollable result rows.
<a href="#">SQLSetPos</a>	Yes	ODBC	Positions a cursor within a fetched block of data and allows an application to refresh data in the rowset or to update or delete data in the result set.
<a href="#">SQLBulkOperations</a>	Yes	ODBC	Performs bulk insertions and bulk bookmark operations, including update, delete, and fetch by bookmark.

### Retrieving error or diagnostic information

Function name	C/ODBC 3.51	Standard	Purpose
---------------	-------------	----------	---------

SQLError	Yes	Deprecated	Returns additional error or status information
SQLGetDiagField	Yes	ISO 92	Returns additional diagnostic information (a single field of the diagnostic data structure).
SQLGetDiagRec	Yes	ISO 92	Returns additional diagnostic information (multiple fields of the diagnostic data structure).

#### Obtaining information about the data source's system tables (catalog functions) item

Function name	C/ODBC 3.51	Standard	Purpose
SQLColumnPrivileges	Yes	ODBC	Returns a list of columns and associated privileges for one or more tables.
SQLColumns	Yes	X/Open	Returns the list of column names in specified tables.
SQLForeignKeys	Yes	ODBC	Returns a list of column names that make up foreign keys, if they exist for a specified table.
SQLPrimaryKeys	Yes	ODBC	Returns the list of column names that make up the primary key for a table.
SQLSpecialColumns	Yes	X/Open	Returns information about the optimal set of columns that uniquely identifies a row in a specified table, or the columns that are automatically updated when any value in the row is updated by a transaction.
SQLStatistics	Yes	ISO 92	Returns statistics about a single table and the list of indexes associated with the table.
SQLTablePrivileges	Yes	ODBC	Returns a list of tables and the privileges associated with each table.
SQLTables	Yes	X/Open	Returns the list of table names stored in a specific data source.

#### Performing transactions

Function name	C/ODBC 3.51	Standard	Purpose
SQLTransact	Yes	Deprecated	Commits or rolls back a transaction
SQLEndTran	Yes	ISO 92	Commits or rolls back a transaction.

#### Terminating a statement

Function name	C/ODBC 3.51	Standard	Purpose
SQLFreeStmt	Yes	ISO 92	Ends statement processing, discards pending results, and, optionally, frees all resources associated with the statement handle.
SQLCloseCursor	Yes	ISO 92	Closes a cursor that has been opened on a statement handle.
SQLCancel	Yes	ISO 92	Cancels an SQL statement.

#### Terminating a connection

Function name	C/ODBC 3.51	Standard	Purpose
SQLDisconnect	Yes	ISO 92	Closes the connection.
SQLFreeHandle	Yes	ISO 92	Releases an environment, connection, statement, or descriptor handle.
SQLFreeConnect	Yes	Deprecated	Releases connection handle
SQLFreeEnv	Yes	Deprecated	Releases an environment handle

## 5.2. Connector/ODBC Data Types

The following table illustrates how driver maps the server data types to default SQL and C data types.

Native Value	SQL Type	C Type
bigint unsigned	SQL_BIGINT	SQL_C_UBIGINT
bigint	SQL_BIGINT	SQL_C_SBIGINT
bit	SQL_BIT	SQL_C_BIT
bit	SQL_CHAR	SQL_C_CHAR
blob	SQL_LONGVARBINARY	SQL_C_BINARY
bool	SQL_CHAR	SQL_C_CHAR
char	SQL_CHAR	SQL_C_CHAR
date	SQL_DATE	SQL_C_DATE
datetime	SQL_TIMESTAMP	SQL_C_TIMESTAMP
decimal	SQL_DECIMAL	SQL_C_CHAR
double precision	SQL_DOUBLE	SQL_C_DOUBLE
double	SQL_FLOAT	SQL_C_DOUBLE
enum	SQL_VARCHAR	SQL_C_CHAR
float	SQL_REAL	SQL_C_FLOAT
int unsigned	SQL_INTEGER	SQL_C_ULONG
int	SQL_INTEGER	SQL_C_SLONG
integer unsigned	SQL_INTEGER	SQL_C_ULONG
integer	SQL_INTEGER	SQL_C_SLONG
long varbinary	SQL_LONGVARBINARY	SQL_C_BINARY
long varchar	SQL_LONGVARCHAR	SQL_C_CHAR
longblob	SQL_LONGVARBINARY	SQL_C_BINARY
longtext	SQL_LONGVARCHAR	SQL_C_CHAR
mediumblob	SQL_LONGVARBINARY	SQL_C_BINARY
mediumint unsigned	SQL_INTEGER	SQL_C_ULONG
mediumint	SQL_INTEGER	SQL_C_SLONG
mediumtext	SQL_LONGVARCHAR	SQL_C_CHAR
numeric	SQL_NUMERIC	SQL_C_CHAR
real	SQL_FLOAT	SQL_C_DOUBLE
set	SQL_VARCHAR	SQL_C_CHAR
smallint unsigned	SQL_SMALLINT	SQL_C_USHORT
smallint	SQL_SMALLINT	SQL_C_SSHORT
text	SQL_LONGVARCHAR	SQL_C_CHAR
time	SQL_TIME	SQL_C_TIME
timestamp	SQL_TIMESTAMP	SQL_C_TIMESTAMP
tinyblob	SQL_LONGVARBINARY	SQL_C_BINARY
tinyint unsigned	SQL_TINYINT	SQL_C_UTINYINT
tinyint	SQL_TINYINT	SQL_C_STINYINT
tinytext	SQL_LONGVARCHAR	SQL_C_CHAR
varchar	SQL_VARCHAR	SQL_C_CHAR
year	SQL_SMALLINT	SQL_C_SHORT

### 5.3. Connector/ODBC Error Codes

The following tables lists the error codes returned by the driver apart from the server errors.

Native Code	SQLSTATE 2	SQLSTATE 3	Error Message
-------------	------------	------------	---------------

500	01000	01000	General warning
501	01004	01004	String data, right truncated
502	01S02	01S02	Option value changed
503	01S03	01S03	No rows updated/deleted
504	01S04	01S04	More than one row updated/deleted
505	01S06	01S06	Attempt to fetch before the result set returned the first row set
506	07001	07002	<a href="#">SQLBindParameter</a> not used for all parameters
507	07005	07005	Prepared statement not a cursor-specification
508	07009	07009	Invalid descriptor index
509	08002	08002	Connection name in use
510	08003	08003	Connection does not exist
511	24000	24000	Invalid cursor state
512	25000	25000	Invalid transaction state
513	25S01	25S01	Transaction state unknown
514	34000	34000	Invalid cursor name
515	S1000	HY000	General driver defined error
516	S1001	HY001	Memory allocation error
517	S1002	HY002	Invalid column number
518	S1003	HY003	Invalid application buffer type
519	S1004	HY004	Invalid SQL data type
520	S1009	HY009	Invalid use of null pointer
521	S1010	HY010	Function sequence error
522	S1011	HY011	Attribute can not be set now
523	S1012	HY012	Invalid transaction operation code
524	S1013	HY013	Memory management error
525	S1015	HY015	No cursor name available
526	S1024	HY024	Invalid attribute value
527	S1090	HY090	Invalid string or buffer length
528	S1091	HY091	Invalid descriptor field identifier
529	S1092	HY092	Invalid attribute/option identifier
530	S1093	HY093	Invalid parameter number
531	S1095	HY095	Function type out of range
532	S1106	HY106	Fetch type out of range
533	S1117	HY117	Row value out of range
534	S1109	HY109	Invalid cursor position
535	S1C00	HYC00	Optional feature not implemented
0	21S01	21S01	Column count does not match value count
0	23000	23000	Integrity constraint violation
0	42000	42000	Syntax error or access violation
0	42S02	42S02	Base table or view not found
0	42S12	42S12	Index not found
0	42S21	42S21	Column already exists
0	42S22	42S22	Column not found
0	08S01	08S01	Communication link failure

---

## Chapter 6. Connector/ODBC Notes and Tips

Here are some common notes and tips for using Connector/ODBC within different environments, applications and tools. The notes provided here are based on the experiences of Connector/ODBC developers and users.

### 6.1. Connector/ODBC General Functionality

This section provides help with common queries and areas of functionality in MySQL and how to use them with Connector/ODBC.

#### 6.1.1. Obtaining Auto-Increment Values

Obtaining the value of column that uses `AUTO_INCREMENT` after an `INSERT` statement can be achieved in a number of different ways. To obtain the value immediately after an `INSERT`, use a `SELECT` query with the `LAST_INSERT_ID()` function.

For example, using Connector/ODBC you would execute two separate statements, the `INSERT` statement and the `SELECT` query to obtain the auto-increment value.

```
INSERT INTO tbl (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

If you do not require the value within your application, but do require the value as part of another `INSERT`, the entire process can be handled by executing the following statements:

```
INSERT INTO tbl (auto,text) VALUES(NULL,'text');
INSERT INTO tbl2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

Certain ODBC applications (including Delphi and Access) may have trouble obtaining the auto-increment value using the previous examples. In this case, try the following statement as an alternative:

```
SELECT * FROM tbl WHERE auto IS NULL;
```

See [How to Get the Unique ID for the Last Inserted Row](#).

#### 6.1.2. Dynamic Cursor Support

Support for the `dynamic cursor` is provided in Connector/ODBC 3.51, but dynamic cursors are not enabled by default. You can enable this function within Windows by selecting the `Enable Dynamic Cursor` checkbox within the ODBC Data Source Administrator.

On other platforms, you can enable the dynamic cursor by adding `32` to the `OPTION` value when creating the DSN.

#### 6.1.3. Connector/ODBC Performance

The Connector/ODBC driver has been optimized to provide very fast performance. If you experience problems with the performance of Connector/ODBC, or notice a large amount of disk activity for simple queries, there are a number of aspects you should check:

- Ensure that `ODBC Tracing` is not enabled. With tracing enabled, a lot of information is recorded in the tracing file by the ODBC Manager. You can check, and disable, tracing within Windows using the `TRACING` panel of the ODBC Data Source Administrator. Within Mac OS X, check the `TRACING` panel of ODBC Administrator. See [Section 3.8, "Getting an ODBC Trace File"](#).
- Make sure you are using the standard version of the driver, and not the debug version. The debug version includes additional checks and reporting measures.
- Disable the Connector/ODBC driver trace and query logs. These options are enabled for each DSN, so make sure to examine only the DSN that you are using in your application. Within Windows, you can disable the Connector/ODBC and query logs by modifying the DSN configuration. Within Mac OS X and Unix, ensure that the driver trace (option value 4) and query logging (option value 524288) are not enabled.

#### 6.1.4. Setting ODBC Query Timeout in Windows

For more information on how to set the query timeout on Microsoft Windows when executing queries through an ODBC connection, read the Microsoft knowledgebase document at <http://support.microsoft.com/default.aspx?scid=kb%3Ben-us%3B153756>.



## 6.2. Connector/ODBC Application Specific Tips

Most programs should work with Connector/ODBC, but for each of those listed here, there are specific notes and tips to improve or enhance the way you work with Connector/ODBC and these applications.

With all applications you should ensure that you are using the latest Connector/ODBC drivers, ODBC Manager and any supporting libraries and interfaces used by your application. For example, on Windows, using the latest version of Microsoft Data Access Components (MDAC) will improve the compatibility with ODBC in general, and with the Connector/ODBC driver.

### 6.2.1. Using Connector/ODBC with Microsoft Applications

The majority of Microsoft applications have been tested with Connector/ODBC, including Microsoft Office, Microsoft Access and the various programming languages supported within ASP and Microsoft Visual Studio.

#### 6.2.1.1. Microsoft Access

To improve the integration between Microsoft Access and MySQL through Connector/ODBC:

- For all versions of Access, you should enable the Connector/ODBC `Return matching rows` option. For Access 2.0, you should additionally enable the `Simulate ODBC 1.0` option.
- You should have a `TIMESTAMP` column in all tables that you want to be able to update. For maximum portability, do not use a length specification in the column declaration (which is unsupported within MySQL in versions earlier than 4.1).
- You should have a primary key in each MySQL table you want to use with Access. If not, new or updated rows may show up as `#DELETED#`.
- Use only `DOUBLE` float fields. Access fails when comparing with single-precision floats. The symptom usually is that new or updated rows may show up as `#DELETED#` or that you cannot find or update rows.
- If you are using Connector/ODBC to link to a table that has a `BIGINT` column, the results are displayed as `#DELETED#`. The work around solution is:
  - Have one more dummy column with `TIMESTAMP` as the data type.
  - Select the `Change BIGINT columns to INT` option in the connection dialog in ODBC DSN Administrator.
  - Delete the table link from Access and re-create it.

Old records may still display as `#DELETED#`, but newly added/updated records are displayed properly.

- If you still get the error `Another user has changed your data` after adding a `TIMESTAMP` column, the following trick may help you:

Do not use a `table` data sheet view. Instead, create a form with the fields you want, and use that `form` data sheet view. You should set the `DefaultValue` property for the `TIMESTAMP` column to `NOW( )`. It may be a good idea to hide the `TIMESTAMP` column from view so your users are not confused.

- In some cases, Access may generate SQL statements that MySQL cannot understand. You can fix this by selecting `"Query|SQLSpecific|Pass-Through"` from the Access menu.
- On Windows NT, Access reports `BLOB` columns as `OLE OBJECTS`. If you want to have `MEMO` columns instead, you should change `BLOB` columns to `TEXT` with `ALTER TABLE`.
- Access cannot always handle the MySQL `DATE` column properly. If you have a problem with these, change the columns to `DATETIME`.
- If you have in Access a column defined as `BYTE`, Access tries to export this as `TINYINT` instead of `TINYINT UNSIGNED`. This gives you problems if you have values larger than 127 in the column.
- If you have very large (long) tables in Access, it might take a very long time to open them. Or you might run low on virtual memory and eventually get an `ODBC Query Failed` error and the table cannot open. To deal with this, select the following options:
  - Return Matching Rows (2)
  - Allow BIG Results (8).

These add up to a value of 10 (`OPTION=10`).

Some external articles and tips that may be useful when using Access, ODBC and Connector/ODBC:

- Read [How to Trap ODBC Login Error Messages in Access](#)
- [Optimizing Access ODBC Applications](#)
  - [Optimizing for Client/Server Performance](#)
  - [Tips for Converting Applications to Using ODBCdirect](#)
  - [Tips for Optimizing Queries on Attached SQL Tables](#)
- For a list of tools that can be used with Access and ODBC data sources, refer to [converters](#) section for list of available tools.

#### MySQL Enterprise

MySQL Enterprise subscribers will find more information about using ODBC with Access in Knowledge Base articles such as [Use MySQL-Specific Syntax with Microsoft Access](#). To subscribe to MySQL Enterprise see <http://www.mysql.com/products/enterprise/advisors.html>.

### 6.2.1.2. Microsoft Excel and Column Types

If you have problems importing data into Microsoft Excel, particularly numerical, date, and time values, this is probably because of a bug in Excel, where the column type of the source data is used to determine the data type when that data is inserted into a cell within the worksheet. The result is that Excel incorrectly identifies the content and this affects both the display format and the data when it is used within calculations.

To address this issue, use the `CONCAT()` function in your queries. The use of `CONCAT()` forces Excel to treat the value as a string, which Excel will then parse and usually correctly identify the embedded information.

However, even with this option, some data may be incorrectly formatted, even though the source data remains unchanged. Use the [Format Cells](#) option within Excel to change the format of the displayed information.

### 6.2.1.3. Microsoft Visual Basic

To be able to update a table, you must define a primary key for the table.

Visual Basic with ADO cannot handle big integers. This means that some queries like `SHOW PROCESSLIST` do not work properly. The fix is to use `OPTION=16384` in the ODBC connect string or to select the [Change BIGINT columns to INT](#) option in the Connector/ODBC connect screen. You may also want to select the [Return matching rows](#) option.

#### MySQL Enterprise

MySQL Enterprise subscribers can find a discussion about using VBA in the Knowledge Base article, [MySQL-Specific Syntax with VBA](#). To subscribe to MySQL Enterprise see <http://www.mysql.com/products/enterprise/advisors.html>.

### 6.2.1.4. Microsoft Visual InterDev

If you have a `BIGINT` in your result, you may get the error `[Microsoft][ODBC Driver Manager] Driver does not support this parameter`. Try selecting the [Change BIGINT columns to INT](#) option in the Connector/ODBC connect screen.

### 6.2.1.5. Visual Objects

You should select the [Don't optimize column widths](#) option.

### 6.2.1.6. Microsoft ADO

When you are coding with the ADO API and Connector/ODBC, you need to pay attention to some default properties that aren't supported by the MySQL server. For example, using the [CursorLocation Property](#) as `adUseServer` returns a result of `-1` for the [RecordCount Property](#). To have the right value, you need to set this property to `adUseClient`, as shown in the VB code here:

```
Dim myconn As New ADODB.Connection
Dim myrs As New Recordset
Dim mySQL As String
Dim myrows As Long
myconn.Open "DSN=MyODBCsample"
mySQL = "SELECT * from user"
myrs.Source = mySQL
```

```
Set myrs.ActiveConnection = myconn
myrs.CursorLocation = adUseClient
myrs.Open
myrows = myrs.RecordCount
myrs.Close
myconn.Close
```

Another workaround is to use a `SELECT COUNT(*)` statement for a similar query to get the correct row count.

To find the number of rows affected by a specific SQL statement in ADO, use the `RecordsAffected` property in the ADO `execute` method. For more information on the usage of `execute` method, refer to <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdmthcnnextecute.asp>.

For information, see [ActiveX Data Objects\(ADO\) Frequently Asked Questions](#).

### 6.2.1.7. Using Connector/ODBC with Active Server Pages (ASP)

You should select the `Return matching rows` option in the DSN.

For more information about how to access MySQL via ASP using Connector/ODBC, refer to the following articles:

- [Using MyODBC To Access Your MySQL Database Via ASP](#)
- [ASP and MySQL at DWAM.NT](#)

A Frequently Asked Questions list for ASP can be found at <http://support.microsoft.com/default.aspx?scid=/Support/ActiveServer/faq/data/adofaq.asp>.

### 6.2.1.8. Using Connector/ODBC with Visual Basic (ADO, DAO and RDO) and ASP

Some articles that may help with Visual Basic and ASP:

- [MySQL BLOB columns and Visual Basic 6](#) by Mike Hillyer (<mike@openwin.org>).
- [How to map Visual basic data type to MySQL types](#) by Mike Hillyer (<mike@openwin.org>).

## 6.2.2. Using Connector/ODBC with Borland Applications

With all Borland applications where the Borland Database Engine (BDE) is used, follow these steps to improve compatibility:

- Update to BDE 3.2 or newer.
- Enable the `Don't optimize column widths` option in the DSN.
- Enabled the `Return matching rows` option in the DSN.

### 6.2.2.1. Using Connector/ODBC with Borland Builder 4

When you start a query, you can use the `Active` property or the `Open` method. Note that `Active` starts by automatically issuing a `SELECT * FROM ...` query. That may not be a good thing if your tables are large.

### 6.2.2.2. Using Connector/ODBC with Delphi

Also, here is some potentially useful Delphi code that sets up both an ODBC entry and a BDE entry for Connector/ODBC. The BDE entry requires a BDE Alias Editor that is free at a Delphi Super Page near you. (Thanks to Bryan Brunton <bryan@flesherfab.com> for this):

```
fReg:= TRegistry.Create;
fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
fReg.WriteString('Database', 'Documents');
fReg.WriteString('Description', '');
fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
fReg.WriteString('Flag', '1');
fReg.WriteString('Password', '');
fReg.WriteString('Port', '');
fReg.WriteString('Server', 'xmark');
fReg.WriteString('User', 'winuser');
fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
```

```
fReg.WriteString('DocumentsFab', 'MySQL');
fReg.CloseKey;
fReg.Free;
Memo1.Lines.Add('DATABASE NAME=');
Memo1.Lines.Add('USER NAME=');
Memo1.Lines.Add('ODBC DSN=DocumentsFab');
Memo1.Lines.Add('OPEN MODE=READ/WRITE');
Memo1.Lines.Add('BATCH COUNT=200');
Memo1.Lines.Add('LANGDRIVER=');
Memo1.Lines.Add('MAX ROWS=-1');
Memo1.Lines.Add('SCHEMA CACHE DIR=');
Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
Memo1.Lines.Add('SQLQRYMODE=');
Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
Memo1.Lines.Add('ENABLE BCD=FALSE');
Memo1.Lines.Add('ROWSET SIZE=20');
Memo1.Lines.Add('BLOBS TO CACHE=64');
Memo1.Lines.Add('BLOB SIZE=32');
AliasEditor.Add('DocumentsFab', 'MySQL', Memo1.Lines);
```

### 6.2.2.3. Using Connector/ODBC with C++ Builder

Tested with BDE 3.0. The only known problem is that when the table schema changes, query fields are not updated. BDE, however, does not seem to recognize primary keys, only the index named `PRIMARY`, although this has not been a problem.

## 6.2.3. Using Connector/ODBC with ColdFusion

The following information is taken from the ColdFusion documentation:

Use the following information to configure ColdFusion Server for Linux to use the `unixODBC` driver with Connector/ODBC for MySQL data sources. You can download Connector/ODBC at <http://dev.mysql.com/downloads/connector/odbc/>.

ColdFusion version 4.5.1 allows you to use the ColdFusion Administrator to add the MySQL data source. However, the driver is not included with ColdFusion version 4.5.1. Before the MySQL driver appears in the ODBC data sources drop-down list, you must build and copy the Connector/ODBC driver to `/opt/coldfusion/lib/libmyodbc.so`.

The Contrib directory contains the program `mydsn-xxx.zip` which allows you to build and remove the DSN registry file for the Connector/ODBC driver on ColdFusion applications.

For more information and guides on using ColdFusion and Connector/ODBC, see the following external sites:

- [Troubleshooting Data Sources and Database Connectivity for Unix Platforms](#).

### 6.2.4. Using Connector/ODBC with OpenOffice.org

Open Office (<http://www.openoffice.org>) [How-to: MySQL + OpenOffice](#). [How-to: OpenOffice + MyODBC + unixODBC](#).

### 6.2.5. Using Connector/ODBC with Sambar Server

Sambar Server (<http://www.sambarserver.info>) [How-to: MyODBC + SambarServer + MySQL](#).

### 6.2.6. Using Connector/ODBC with Pervasive Software DataJunction

You have to change it to output `VARCHAR` rather than `ENUM`, as it exports the latter in a manner that causes MySQL problems.

### 6.2.7. Using Connector/ODBC with SunSystems Vision

You should select the `Return matching rows` option.

## 6.3. Connector/ODBC Errors and Resolutions (FAQ)

The following section details some common errors and their suggested fix or alternative solution. If you are still experiencing problems, use the Connector/ODBC mailing list; see [Section 7.1, "Connector/ODBC Community Support"](#).

Many problems can be resolved by upgrading your Connector/ODBC drivers to the latest available release. On Windows, you should also make sure that you have the latest versions of the Microsoft Data Access Components (MDAC) installed.

### Questions

- [6.3.1](#): I have installed Connector/ODBC on Windows XP x64 Edition or Windows Server 2003 R2 x64. The installation completed successfully, but the Connector/ODBC driver does not appear in [ODBC Data Source Administrator](#).
- [6.3.2](#): When connecting or using the TEST button in [ODBC Data Source Administrator](#) I get error 10061 (Cannot connect to server)
- [6.3.3](#): The following error is reported when using transactions: [Transactions are not enabled](#)
- [6.3.4](#): Access reports records as [#DELETED#](#) when inserting or updating records in linked tables.
- [6.3.5](#): How do I handle Write Conflicts or Row Location errors?
- [6.3.6](#): Exporting data from Access 97 to MySQL reports a [Syntax Error](#).
- [6.3.7](#): Exporting data from Microsoft DTS to MySQL reports a [Syntax Error](#).
- [6.3.8](#): Using ODBC.NET with Connector/ODBC, while fetching empty string (0 length), it starts giving the SQL\_NO\_DATA exception.
- [6.3.9](#): Using `SELECT COUNT(*) FROM tbl_name` within Visual Basic and ASP returns an error.
- [6.3.10](#): Using the `AppendChunk()` or `GetChunk()` ADO methods, the [Multiple-step operation generated errors. Check each status value](#) error is returned.
- [6.3.11](#): Access Returns [Another user had modified the record that you have modified](#) while editing records on a Linked Table.
- [6.3.12](#): When linking an application directly to the Connector/ODBC library under Unix/Linux, the application crashes.
- [6.3.13](#): Applications in the Microsoft Office suite are unable to update tables that have [DATE](#) or [TIMESTAMP](#) columns.
- [6.3.14](#): When connecting Connector/ODBC 5.x (Beta) to a MySQL 4.x server, the error [1044 Access denied for user 'xxx'@'%' to database 'information\\_schema'](#) is returned.
- [6.3.15](#): When calling `SQLTables`, the error [S1T00](#) is returned, but I cannot find this in the list of error numbers for Connector/ODBC.
- [6.3.16](#): When linking to tables in Access 2000 and generating links to tables programmatically, rather than through the table designer interface, you may get errors about tables not existing.
- [6.3.17](#): When I try to use batched statements, the execution of the batched statements fails.
- [6.3.18](#): When connecting to a MySQL server using ADODB and Excel, occasionally the application fails to communicate with the server and the error [Got an error reading communication packets](#) appears in the error log.
- [6.3.19](#): When using some applications to access a MySQL server using C/ODBC and outer joins, an error is reported regarding the Outer Join Escape Sequence.
- [6.3.20](#): I can correctly store extended characters in the database (Hebrew/CJK) using C/ODBC 5.1, but when I retrieve the data, the text is not formatted correctly and I get garbled characters.
- [6.3.21](#): I have a duplicate MySQL Connector/ODBC entry within my **INSTALLED PROGRAMS** list, but I cannot delete one of them.
- [6.3.22](#): When submitting queries with parameter binding using `UPDATE`, my field values are being truncated to 255 characters.
- [6.3.23](#): Is it possible to disable data-at-execution using a flag?

### Questions and Answers

**6.3.1: I have installed Connector/ODBC on Windows XP x64 Edition or Windows Server 2003 R2 x64. The installation completed successfully, but the Connector/ODBC driver does not appear in ODBC Data Source Administrator.**

This is not a bug, but is related to the way Windows x64 editions operate with the ODBC driver. On Windows x64 editions, the Connector/ODBC driver is installed in the `%SystemRoot%\SysWOW64` folder. However, the default [ODBC Data Source Administrator](#) that is available through the [Administrative Tools](#) or [Control Panel](#) in Windows x64 Editions is located in the `%SystemRoot%\system32` folder, and only searches this folder for ODBC drivers.

On Windows x64 editions, you should use the ODBC administration tool located at `%SystemRoot%\SysWOW64\odbcad32.exe`, this will correctly locate the installed Connector/ODBC drivers and enable you

to create a Connector/ODBC DSN.

This issue was originally reported as [Bug#20301](#).

### 6.3.2: When connecting or using the TEST button in ODBC Data Source Administrator I get error 10061 (Cannot connect to server)

This error can be raised by a number of different issues, including server problems, network problems, and firewall and port blocking problems. For more information, see [Can't connect to \[local\] MySQL server](#).

### 6.3.3: The following error is reported when using transactions: Transactions are not enabled

This error indicates that you are trying to use transactions with a MySQL table that does not support transactions. Transactions are supported within MySQL when using the `InnoDB` database engine. In versions of MySQL before MySQL 5.1 you may also use the `BDB` engine.

You should check the following before continuing:

- Verify that your MySQL server supports a transactional database engine. Use `SHOW ENGINES` to obtain a list of the available engine types.
- Verify that the tables you are updating use a transaction database engine.
- Ensure that you have not enabled the `disable transactions` option in your DSN.

### 6.3.4: Access reports records as #DELETED# when inserting or updating records in linked tables.

If the inserted or updated records are shown as `#DELETED#` in the access, then:

- If you are using Access 2000, you should get and install the newest (version 2.6 or higher) Microsoft MDAC ([Microsoft Data Access Components](#)) from <http://support.microsoft.com/kb/110093>. This fixes a bug in Access that when you export data to MySQL, the table and column names aren't specified.

You should also get and apply the Microsoft Jet 4.0 Service Pack 5 (SP5) which can be found at <http://support.microsoft.com/default.aspx?scid=kb;EN-US;q239114>. This fixes some cases where columns are marked as `#DELETED#` in Access.

- For all versions of Access, you should enable the Connector/ODBC `Return matching rows` option. For Access 2.0, you should additionally enable the `Simulate ODBC 1.0` option.
- You should have a timestamp in all tables that you want to be able to update.
- You should have a primary key in the table. If not, new or updated rows may show up as `#DELETED#`.
- Use only `DOUBLE` float fields. Access fails when comparing with single-precision floats. The symptom usually is that new or updated rows may show up as `#DELETED#` or that you cannot find or update rows.
- If you are using Connector/ODBC to link to a table that has a `BIGINT` column, the results are displayed as `#DELETED`. The work around solution is:
  - Have one more dummy column with `TIMESTAMP` as the data type.
  - Select the `Change BIGINT columns to INT` option in the connection dialog in ODBC DSN Administrator.
  - Delete the table link from Access and re-create it.

Old records still display as `#DELETED#`, but newly added/updated records are displayed properly.

### 6.3.5: How do I handle Write Conflicts or Row Location errors?

If you see the following errors, select the `Return Matching Rows` option in the DSN configuration dialog, or specify `OPTION=2`, as the connection parameter:

```
Write Conflict. Another user has changed your data.
Row cannot be located for updating. Some values may have been changed
since it was last read.
```

### 6.3.6: Exporting data from Access 97 to MySQL reports a Syntax Error.



This error is specific to Access 97 and versions of Connector/ODBC earlier than 3.51.02. Update to the latest version of the Connector/ODBC driver to resolve this problem.

**6.3.7: Exporting data from Microsoft DTS to MySQL reports a `Syntax Error`.**

This error occurs only with MySQL tables using the `TEXT` or `VARCHAR` data types. You can fix this error by upgrading your Connector/ODBC driver to version 3.51.02 or higher.

**6.3.8: Using ODBC.NET with Connector/ODBC, while fetching empty string (0 length), it starts giving the `SQL_NO_DATA` exception.**

You can get the patch that addresses this problem from <http://support.microsoft.com/default.aspx?scid=kb;EN-US;q319243>.

**6.3.9: Using `SELECT COUNT(*) FROM tbl_name` within Visual Basic and ASP returns an error.**

This error occurs because the `COUNT(*)` expression is returning a `BIGINT`, and ADO cannot make sense of a number this big. Select the `Change BIGINT columns to INT` option (option value 16384).

**6.3.10: Using the `AppendChunk()` or `GetChunk()` ADO methods, the `Multiple-step operation generated errors. Check each status value error is returned`.**

The `GetChunk()` and `AppendChunk()` methods from ADO doesn't work as expected when the cursor location is specified as `adUseServer`. On the other hand, you can overcome this error by using `adUseClient`.

A simple example can be found from [http://www.dwam.net/iishelp/ado/docs/adomth02\\_4.htm](http://www.dwam.net/iishelp/ado/docs/adomth02_4.htm)

**6.3.11: Access Returns `Another user had modified the record that you have modified while editing records on a Linked Table`.**

In most cases, this can be solved by doing one of the following things:

- Add a primary key for the table if one doesn't exist.
- Add a timestamp column if one doesn't exist.
- Only use double-precision float fields. Some programs may fail when they compare single-precision floats.

If these strategies do not help, you should start by making a log file from the ODBC manager (the log you get when requesting logs from ODBCADMIN) and a Connector/ODBC log to help you figure out why things go wrong. For instructions, see [Section 3.8](#), "Getting an ODBC Trace File".

**6.3.12: When linking an application directly to the Connector/ODBC library under Unix/Linux, the application crashes.**

Connector/ODBC 3.51 under Unix/Linux is not compatible with direct application linking. You must use a driver manager, such as `iODBC` or `unixODBC` to connect to an ODBC source.

**6.3.13: Applications in the Microsoft Office suite are unable to update tables that have `DATE` or `TIMESTAMP` columns.**

This is a known issue with Connector/ODBC. You must ensure that the field has a default value (rather than `NULL` and that the default value is non-zero (i.e. the default value is not `0000-00-00 00:00:00`).

**6.3.14: When connecting Connector/ODBC 5.x (Beta) to a MySQL 4.x server, the error `1044 Access denied for user 'xxx'@'%' to database 'information_schema'` is returned.**

Connector/ODBC 5.x is designed to work with MySQL 5.0 or later, taking advantage of the `INFORMATION_SCHEMA` database to determine data definition information. Support for MySQL 4.1 is planned for the final release.

**6.3.15: When calling `SQLTables`, the error `S1T00` is returned, but I cannot find this in the list of error numbers for Connector/ODBC.**

The `S1T00` error indicates that a general timeout has occurred within the ODBC system and is not a MySQL error. Typically it indicates that the connection you are using is stale, the server is too busy to accept your request or that the server has gone away.

**6.3.16: When linking to tables in Access 2000 and generating links to tables programmatically, rather than through the table designer interface, you may get errors about tables not existing.**

There is a known issue with a specific version of the `msjet40.dll` that exhibits this issue. The version affected is 4.0.9025.0. Reverting to an older version will enable you to create the links. If you have recently updated your version, check your `WINDOWS` directory for the older version of the file and copy it to the drivers directory.

**6.3.17: When I try to use batched statements, the execution of the batched statements fails.**

Batched statement support was added in 3.51.18. Support for batched statements is not enabled by default. You must enable option `FLAG_MULTI_STATEMENTS`, value 67108864, or select the **ALLOW MULTIPLE STATEMENTS** flag within a GUI configuration.

**6.3.18: When connecting to a MySQL server using ADODB and Excel, occasionally the application fails to communicate with the server and the error `Got an error reading communication packets` appears in the error log.**

This error may be related to Keyboard Logger 1.1 from PanteraSoft.com, which is known to interfere with the network communication between MySQL Connector/ODBC and MySQL.

**6.3.19: When using some applications to access a MySQL server using C/ODBC and outer joins, an error is reported regarding the Outer Join Escape Sequence.**

This is a known issue with MySQL Connector/ODBC which is not correctly parsing the "Outer Join Escape Sequence", as per the specs at [Microsoft ODBC Specs](#). Currently, Connector/ODBC will return value > 0 when asked for `SQL_OJ_CAPABILITIES` even though no parsing takes place in the driver to handle the outer join escape sequence.

**6.3.20: I can correctly store extended characters in the database (Hebrew/CJK) using C/ODBC 5.1, but when I retrieve the data, the text is not formatted correctly and I get garbled characters.**

When using ASP and UTF8 characters you should add the following to your ASP files to ensure that the data returned is correctly encoded:

```
Response.CodePage = 65001
Response.CharSet = "utf-8"
```

**6.3.21: I have a duplicate MySQL Connector/ODBC entry within my INSTALLED PROGRAMS list, but I cannot delete one of them.**

This problem can occur when you upgrade an existing Connector/ODBC installation, rather than removing and then installing the updated version.

**Warning**

To fix the problem you should use any working uninstallers to remove existing installations and then may have to edit the contents of the registry. Make sure you have a backup of your registry information before attempting any editing of the registry contents.

**6.3.22: When submitting queries with parameter binding using `UPDATE`, my field values are being truncated to 255 characters.**

You should ensure that the `FLAG_BIG_PACKETS` option is set for your connection. This removes the 255 character limitation on bound parameters.

**6.3.23: Is it possible to disable data-at-execution using a flag?**

If you do not wish to use data-at-execution, simply remove the corresponding calls. For example:

```
SQLLEN ylen = SQL_LEN_DATA_AT_EXEC(10);
SQLBindCol(hstmt, 2, SQL_C_BINARY, buf, 10, &ylen);
```

Would become:

```
SQLBindCol(hstmt, 2, SQL_C_BINARY, buf, 10, NULL);
```

Note that in the call to `SQLBindCol()`, `&ylen` has been replaced by `NULL`.

For further information please refer to the [MSDN documentation](#) for `SQLBindCol()`.



---

## Chapter 7. Connector/ODBC Support

There are many different places where you can get support for using Connector/ODBC. You should always try the Connector/ODBC Mailing List or Connector/ODBC Forum. See [Section 7.1, “Connector/ODBC Community Support”](#), for help before reporting a specific bug or issue to MySQL.

### 7.1. Connector/ODBC Community Support

Sun Microsystems, Inc. provides assistance to the user community by means of its mailing lists. For Connector/ODBC-related issues, you can get help from experienced users by using the `<myodbc@lists.mysql.com>` mailing list. Archives are available online at <http://lists.mysql.com/myodbc>.

For information about subscribing to MySQL mailing lists or to browse list archives, visit <http://lists.mysql.com/>. See [MySQL Mailing Lists](#).

Community support from experienced users is also available through the [ODBC Forum](#). You may also find help from other users in the other MySQL Forums, located at <http://forums.mysql.com>. See [MySQL Community Support at the MySQL Forums](#).

### 7.2. How to Report Connector/ODBC Problems or Bugs

If you encounter difficulties or problems with Connector/ODBC, you should start by making a log file from the [ODBC Manager](#) (the log you get when requesting logs from `ODBC ADMIN`) and Connector/ODBC. The procedure for doing this is described in [Section 3.8, “Getting an ODBC Trace File”](#).

Check the Connector/ODBC trace file to find out what could be wrong. You should be able to determine what statements were issued by searching for the string `>mysql_real_query` in the `myodbc.log` file.

You should also try issuing the statements from the `mysql` client program or from `admindemo`. This helps you determine whether the error is in Connector/ODBC or MySQL.

If you find out something is wrong, please only send the relevant rows (maximum 40 rows) to the `myodbc` mailing list. See [MySQL Mailing Lists](#). Please never send the whole Connector/ODBC or ODBC log file!

You should ideally include the following information with the email:

- Operating system and version
- Connector/ODBC version
- ODBC Driver Manager type and version
- MySQL server version
- ODBC trace from Driver Manager
- Connector/ODBC log file from Connector/ODBC driver
- Simple reproducible sample

Remember that the more information you can supply to us, the more likely it is that we can fix the problem!

Also, before posting the bug, check the MyODBC mailing list archive at <http://lists.mysql.com/myodbc>.

If you are unable to find out what is wrong, the last option is to create an archive in `tar` or Zip format that contains a Connector/ODBC trace file, the ODBC log file, and a `README` file that explains the problem. You can send this to <ftp://ftp.mysql.com/pub/mysql/upload/>. Only MySQL engineers have access to the files you upload, and we are very discreet with the data.

If you can create a program that also demonstrates the problem, please include it in the archive as well.

If the program works with another SQL server, you should include an ODBC log file where you perform exactly the same SQL statements so that we can compare the results between the two systems.

Remember that the more information you can supply to us, the more likely it is that we can fix the problem.

### 7.3. How to Submit a Connector/ODBC Patch

You can send a patch or suggest a better solution for any existing code or problems by sending a mail message to [<myodbc@lists.mysql.com>](mailto:myodbc@lists.mysql.com).

## 7.4. Connector/ODBC Change History

The Connector/ODBC Change History (Changelog) is located with the main Changelog for MySQL. See [Appendix A, \*MySQL Connector/ODBC \(MyODBC\) Change History\*](#).

## 7.5. Credits

These are the developers that have worked on the Connector/ODBC and Connector/ODBC 3.51 Drivers from MySQL AB.

- Michael (Monty) Widenius
- Venu Anuganti
- Peter Harvey

---

# Appendix A. MySQL Connector/ODBC (MyODBC) Change History

## A.1. Changes in MySQL Connector/ODBC 5.1.6 (Not yet released)

Bugs fixed:

- Connector/ODBC failed to build with MySQL 5.1.30 due to incorrect use of the data type `bool`. (Bug#42120)
- Calling `SQLDescribeCol()` with a NULL buffer and nonzero buffer length caused a crash. (Bug#41942)
- MySQL Connector/ODBC updated some fields with random values, rather than with `NULL`. (Bug#41256)
- Calling `SQLDriverConnect()` with a `NULL` pointer for the output buffer caused a crash if `SQL_DRIVER_NOPROMPT` was also specified:

```
SQLDriverConnect(dbc, NULL, "DSN=myodbc5", SQL_NTS, NULL, 0, NULL, SQL_DRIVER_NOPROMPT)
```

(Bug#40316)

- Setting the ADO `Recordset` decimal field value to 44.56 resulted in an incorrect value of 445600.0000 being stored when the record set was updated with the `Update` method. (Bug#39961)
- The `SQLTablesW` API gave incorrect results. For example, table name and table type were returned as `NULL` rather than as the correct values. (Bug#39957)
- MyODBC would crash when a character set was being used on the server that was not supported in the client, for example `cp1251`:

```
[MySQL][ODBC 5.1 Driver][mysqld-5.0.27-community-nt]Restricted data type attribute violation
```

The fix causes MyODBC to return an error message instead of crashing. (Bug#39831)

- When the `SQLTables` method was called with `NULL` passed as the `tablename` parameter, only one row in the `resultset`, with table name of `NULL` was returned, instead of all tables for the given database. (Bug#39561)
- The `SQLGetInfo()` function returned 0 for `SQL_CATALOG_USAGE` information. (Bug#39560)
- MyODBC Driver 5.1.5 was not able to connect if the connection string parameters contained spaces or tab symbols. For example, if the `SERVER` parameter was specified as “SERVER= localhost” instead of “SERVER=localhost” the following error message will be displayed:

```
[MySQL][ODBC 5.1 Driver] Unknown MySQL server host ' localhost' (11001).
```

(Bug#39085)

- The pointer passed to the `SQLDriverConnect` method to retrieve the output connection string length was one greater than it should have been due to the inclusion of the `NULL` terminator. (Bug#38949)
- Data-at-execution parameters were not supported during positioned update. This meant updating a long text field with a cursor update would erroneously set the value to null. This would lead to the error `Column 'column_name' cannot be null` while updating the database, even when `column_name` had been assigned a valid non-null string. (Bug#37649)
- The `SQLDriverConnect` method truncated the `OutputConnectionString` parameter to 52 characters. (Bug#37278)
- The connection string option `Enable Auto-reconnect` did not work. When the connection failed, it could not be restored, and the errors generated were the same as if the option had not been selected. (Bug#37179)
- Insertion of data into a `LONGTEXT` table field did not work. If such an attempt was made the corresponding field would be found to be empty on examination, or contain random characters. (Bug#36071)
- No result record was returned for `SQLGetTypeInfo` for the `TIMESTAMP` data type. An application would receive the result `return code 100 (SQL_NO_DATA_FOUND)`. (Bug#30626)
- It was not possible to use Connector/ODBC to connect to a server using SSL. The following error was generated:

```
Runtime error '-2147467259 (80004005)':
```

```
[MySQL][ODBC 3.51 Driver]SSL connection error.
```

([Bug#29955](#))

- When the `recordSet.Update` function was called to update an `adLongVarChar` field, the field was updated but the recordset was immediately lost. This happened with driver cursors, whether the cursor was opened in optimistic or pessimistic mode.

When the next update was called the test code would exit with the following error:

```
-2147467259 : Query-based update failed because the row to update could not be found.
```

([Bug#26950](#))

## A.2. Changes in MySQL Connector/ODBC 5.1.5 (18 August 2008)

Bugs fixed:

- ODBC `TIMESTAMP` string format is not handled properly by the MyODBC driver. When passing a `TIMESTAMP` or `DATE` to MyODBC, in the ODBC format: {d <date>} or {ts <timestamp>}, the string that represents this is copied once into the SQL statement, and then added again, as an escaped string. ([Bug#37342](#))
- The connector failed to prompt for additional information required to create a DSN-less connection from an application such as Microsoft Excel. ([Bug#37254](#))
- `SQLDriverConnect` does not return `SQL_NO_DATA` on cancel. The ODBC documentation specifies that this method should return `SQL_NO_DATA` when the user cancels the dialog to connect. The connector, however, returns `SQL_ERROR`. ([Bug#36293](#))
- Assigning a string longer than 67 characters to the `TableType` parameter resulted in a buffer overrun when the `SQLTables()` function was called. ([Bug#36275](#))
- The ODBC connector randomly uses logon information stored in `odbc-profile`, or prompts the user for connection information and ignores any settings stored in `odbc-profile`. ([Bug#36203](#))
- After having successfully established a connection, a crash occurs when calling `SQLProcedures()` followed by `SQLFreeStmt()`, using the ODBC C API. ([Bug#36069](#))

## A.3. Changes in MySQL Connector/ODBC 5.1.4 (15 April 2008)

Bugs fixed:

- Wrong result obtained when using `sum()` on a `decimal(8,2)` field type. ([Bug#35920](#))
- The driver installer could not create a new DSN if many other drivers were already installed. ([Bug#35776](#))
- The `SQLColAttribute()` function returned `SQL_TRUE` when querying the `SQL_DESC_FIXED_PREC_SCALE` (`SQL_COLUMN_MONEY`) attribute of a `DECIMAL` column. Previously, the correct value of `SQL_FALSE` was returned; this is now again the case. ([Bug#35581](#))
- On Linux, `SQLGetDiagRec()` returned `SQL_SUCCESS` in cases when it should have returned `SQL_NO_DATA`. ([Bug#33910](#))
- The driver crashes ODBC Administrator on attempting to add a new DSN. ([Bug#32057](#))

## A.4. Changes in MySQL Connector/ODBC 5.1.3 (26 March 2008)

Platform specific notes:

- **Important Change:** You must uninstall previous 5.1.x editions of Connector/ODBC before installing the new version.

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- The installer for 64-bit Windows installs both the 32-bit and 64-bit driver. Please note that Microsoft does not yet supply a 64-bit bridge from ADO to ODBC.

Bugs fixed:

- **Important Change:** In previous versions, the SSL certificate would automatically be verified when used as part of the Connector/ODBC connection. The default mode is now to ignore the verification of certificates. To enforce verification of the SSL certificate during connection, use the `SSLVERIFY` DSN parameter, setting the value to 1. ([Bug#29955](#), [Bug#34648](#))
- Inserting characters to a UTF8 table using surrogate pairs would fail and insert invalid data. ([Bug#34672](#))
- Installation of Connector/ODBC would fail because it was unable to uninstall a previous installed version. The file being requested would match an older release version than any installed version of the connector. ([Bug#34522](#))
- Using `SqlGetData` in combination with `SQL_C_WCHAR` would return overlapping data. ([Bug#34429](#))
- Descriptor records were not cleared correctly when calling `SQLFreeStmt (SQL_UNBIND)`. ([Bug#34271](#))
- The dropdown selection for databases on a server when creating a DSN was too small. The list size now automatically adjusts up to a maximum size of 20 potential databases. ([Bug#33918](#))
- Microsoft Access would be unable to use `DBEngine.RegisterDatabase` to create a DSN using the Connector/ODBC driver. ([Bug#33825](#))
- Connector/ODBC erroneously reported that it supported the `CAST ( )` and `CONVERT ( )` ODBC functions for parsing values in SQL statements, which could lead to bad SQL generation during a query. ([Bug#33808](#))
- Using a linked table in Access 2003 where the table has a `BIGINT` column as the first column in the table, and is configured as the primary key, shows `#DELETED` for all rows of the table. ([Bug#24535](#))
- Updating a `RecordSet` when the query involves a `BLOB` field would fail. ([Bug#19065](#))

## A.5. Changes in MySQL Connector/ODBC 5.1.2 (13 February 2008)

MySQL Connector/ODBC 5.1.2-beta, a new version of the ODBC driver for the MySQL database management system, has been released. This release is the second beta (feature-complete) release of the new 5.1 series and is suitable for use with any MySQL server version since MySQL 4.1, including MySQL 5.0, 5.1, and 6.0. (It will not work with 4.0 or earlier releases.)

Keep in mind that this is a beta release, and as with any other pre-production release, caution should be taken when installing on production level systems or systems with critical data.

Platform specific notes:

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- The installer for 64-bit Windows installs both the 32-bit and 64-bit driver. Please note that Microsoft does not yet supply a 64-bit bridge from ADO to ODBC.
- Due to differences with the installation process used on Windows and potential registry corruption, it is recommended that uninstall any existing versions of Connector/ODBC 5.1.x before upgrading.

See also [Bug#34571](#).

Functionality added or changed:

- Explicit descriptors are implemented. ([Bug#32064](#))

- A full implementation of `SQLForeignKeys` based on the information available from `INFORMATION_SCHEMA` in 5.0 and later versions of the server has been implemented.
- Changed `SQL_ATTR_PARAMSET_SIZE` to return an error until support for it is implemented.
- Disabled `MYSQL_OPT_SSL_VERIFY_SERVER_CERT` when using an SSL connection.
- `SQLForeignKeys` uses `INFORMATION_SCHEMA` when it is available on the server, which allows more complete information to be returned.

Bugs fixed:

- The `SSLCIPHER` option would be incorrectly recorded within the SSL configuration on Windows. ([Bug#33897](#))
- Within the GUI interface, when connecting to a MySQL server on a non-standard port, the connection test within the GUI would fail. The issue was related to incorrect parsing of numeric values within the DSN when the option was not configured as the last parameter within the DSN. ([Bug#33822](#))
- Specifying a non-existent database name within the GUI dialog would result in an empty list, not an error. ([Bug#33615](#))
- When deleting rows from a static cursor, the cursor position would be incorrectly reported. ([Bug#33388](#))
- `SQLGetInfo()` reported characters for `SQL_SPECIAL_CHARACTERS` that were not encoded correctly. ([Bug#33130](#))
- Retrieving data from a `BLOB` column would fail within `SQLGetData` when the target data type was `SQL_C_WCHAR` due to incorrect handling of the character buffer. ([Bug#32684](#))
- Renaming an existing DSN entry would create a new entry with the new name without deleting the old entry. ([Bug#31165](#))
- Reading a `TEXT` column that had been used to store UTF8 data would result in the wrong information being returned during a query. ([Bug#28617](#))
- `SQLForeignKeys` would return an empty string for the schema columns instead of `NULL`. ([Bug#19923](#))
- When accessing column data, `FLAG_COLUMN_SIZE_S32` did not limit the octet length or display size reported for fields, causing problems with Microsoft Visual FoxPro.

The list of ODBC functions that could have caused failures in Microsoft software when retrieving the length of `LONGBLOB` or `LONGTEXT` columns includes:

- `SQLColumns`
- `SQLColAttribute`
- `SQLColAttributes`
- `SQLDescribeCol`
- `SQLSpecialColumns` (theoretically can have the same problem) ([Bug#12805](#), [Bug#30890](#))
- Dynamic cursors on statements with parameters were not supported. ([Bug#11846](#))
- Evaluating a simple numeric expression when using the OLEDB for ODBC provider and ADO would return an error, instead of the result. ([Bug#10128](#))
- Adding or updating a row using `SQLSetPos()` on a result set with aliased columns would fail. ([Bug#6157](#))

## A.6. Changes in MySQL Connector/ODBC 5.1.1 (13 December 2007)

MySQL Connector/ODBC 5.1.1-beta, a new version of the ODBC driver for the MySQL database management system, has been released. This release is the first beta (feature-complete) release of the new 5.1 series and is suitable for use with any MySQL server version since MySQL 4.1, including MySQL 5.0, 5.1, and 6.0. (It will not work with 4.0 or earlier releases.)

Keep in mind that this is a beta release, and as with any other pre-production release, caution should be taken when installing on production level systems or systems with critical data.

Includes changes from [Connector/ODBC 3.51.21](#) and [3.51.22](#).

Built using MySQL 5.0.52.

Platform specific notes:

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- The installer for 64-bit Windows installs both the 32-bit and 64-bit driver. Please note that Microsoft does not yet supply a 64-bit bridge from ADO to ODBC.
- Due to differences with the installation process used on Windows and potential registry corruption, it is recommended that un-install any existing versions of Connector/ODBC 5.1.x before upgrading.

See also [Bug#34571](#).

Functionality added or changed:

- **Incompatible Change:** Replaced myodbc3i (now myodbc-installer) with Connector/ODBC 5.0 version.
- **Incompatible Change:** Removed monitor (myodbc3m) and dsn-editor (myodbc3c).
- **Incompatible Change:** Disallow `SET NAMES` in initial statement and in executed statements.
- A wrapper for the `SQLGetPrivateProfileStringW()` function, which is required for Unicode support, has been created. This function is missing from the unixODBC driver manager. ([Bug#32685](#))
- Added MSI installer for Windows 64-bit. ([Bug#31510](#))
- Implemented support for `SQLCancel()`. ([Bug#15601](#))
- Added support for `SQL_NUMERIC_STRUCT`. ([Bug#3028](#), [Bug#24920](#))
- Removed non-threadsafe configuration of the driver. The driver is now always built against the threadsafe version of libmysql.
- Implemented native Windows setup library
- Replaced the internal library which handles creation and loading of DSN information. The new library, which was originally a part of Connector/ODBC 5.0, supports Unicode option values.
- The Windows installer now places files in a subdirectory of the `Program Files` directory instead of the Windows system directory.

Bugs fixed:

- The `SET NAMES` statement has been disabled because it causes problems in the ODBC driver when determining the current client character set. ([Bug#32596](#))
- `SQLDescribeColW` returned UTF-8 column as `SQL_VARCHAR` instead of `SQL_WVARCHAR`. ([Bug#32161](#))
- ADO was unable to open record set using dynamic cursor. ([Bug#32014](#))
- ADO applications would not open a `RecordSet` that contained a `DECIMAL` field. ([Bug#31720](#))
- Memory usage would increase considerably. ([Bug#31115](#))
- SQL statements are limited to 64KB. ([Bug#30983](#), [Bug#30984](#))
- `SQLSetPos` with `SQL_DELETE` advances dynamic cursor incorrectly. ([Bug#29765](#))
- Using an ODBC prepared statement with bound columns would produce an empty result set when called immediately after inserting a row into a table. ([Bug#29239](#))
- ADO Not possible to update a client side cursor. ([Bug#27961](#))
- Recordset `Update()` fails when using `adUseClient` cursor. ([Bug#26985](#))

- Connector/ODBC would fail to connect to the server if the password contained certain characters, including the semicolon and other punctuation marks. ([Bug#16178](#))
- Fixed `SQL_ATTR_PARAM_BIND_OFFSET`, and fixed row offsets to work with updatable cursors.
- `SQLSetConnectAttr()` did not clear previous errors, possibly confusing `SQLError()`.
- `SQLError()` incorrectly cleared the error information, making it unavailable from subsequent calls to `SQLGetDiagRec()`.
- NULL pointers passed to `SQLGetInfo()` could result in a crash.
- `SQL_ODBC_SQL_CONFORMANCE` was not handled by `SQLGetInfo()`.
- `SQLCopyDesc()` did not correctly copy all records.
- Diagnostics were not correctly cleared on connection and environment handles.

## A.7. Changes in MySQL Connector/ODBC 5.1.0 (10 September 2007)

This release is the first of the new 5.1 series and is suitable for use with any MySQL server version since MySQL 4.1, including MySQL 5.0, 5.1, and 6.0. (It will not work with 4.0 or earlier releases.)

Keep in mind that this is an alpha release, and as with any other pre-production release, caution should be taken when installing on production level systems or systems with critical data. Not all of the features planned for the final Connector/ODBC 5.1 release are implemented.

Functionality is based on Connector/ODBC 3.51.20.

Platform specific notes:

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- There are no installer packages for Microsoft Windows x64 Edition.
- Due to differences with the installation process used on Windows and potential registry corruption, it is recommended that un-install any existing versions of Connector/ODBC 5.1.x before upgrading.

See also [Bug#34571](#).

Functionality added or changed:

- Added support for Unicode functions (`SQLConnectW`, etc).
- Added descriptor support (`SQLGetDescField`, `SQLGetDescRec`, etc).
- Added support for `SQL_C_WCHAR`.

## A.8. Changes in MySQL Connector/ODBC 5.0.12 (Never released)

### Note

Development on Connector/ODBC 5.0.x has ceased. New features and functionality will be incorporated into Connector/ODBC 5.1. See [Section 1.1](#), “Connector/ODBC Roadmap”.

Bugs fixed:

- Inserting NULL values into a DATETIME column from Access reports an error. ([Bug#27896](#))
- Tables with TEXT columns would be incorrectly identified, returning an Unknown SQL type - 65535 error. ([Bug#20127](#))



## A.9. Changes in MySQL Connector/ODBC 5.0.11 (31 January 2007)

Functionality added or changed:

- Added support for ODBC v2 statement options using attributes.
- Driver now builds and is partially tested under Linux with the iODBC driver manager.

Bugs fixed:

- Connection string parsing for DSN-less connections could fail to identify some parameters. ([Bug#25316](#))
- Updates of `MEMO` or `TEXT` columns from within Microsoft Access would fail. ([Bug#25263](#))
- Transaction support has been added and tested. ([Bug#25045](#))
- Internal function, `my_setpos_delete_ignore()` could cause a crash. ([Bug#22796](#))
- Fixed occasional mis-handling of the `SQL_NUMERIC_C` type.
- Fixed the binding of certain integer types.

## A.10. Changes in MySQL Connector/ODBC 5.0.10 (14 December 2006)

Connector/ODBC 5.0.10 is the sixth BETA release.

Functionality added or changed:

- Significant performance improvement when retrieving large text fields in pieces using `SQLGetData()` with a buffer smaller than the whole data. Mainly used in Access when fetching very large text fields. ([Bug#24876](#))
- Added initial unicode support in data and metadata. ([Bug#24837](#))
- Added initial support for removing braces when calling stored procedures and retrieving result sets from procedure calls. ([Bug#24485](#))
- Added loose handling of retrieving some diagnostic data. ([Bug#15782](#))
- Added wide-string type info for `SQLGetTypeInfo()`.

Bugs fixed:

- Editing DSN no longer crashes ODBC data source administrator. ([Bug#24675](#))
- String query parameters are now escaped correctly. ([Bug#19078](#))

## A.11. Changes in MySQL Connector/ODBC 5.0.9 (22 November 2006)

Connector/ODBC 5.0.9 is the fifth BETA release.

This is an implementation and testing release, and is not designed for use within a production environment.

Functionality added or changed:

- Added support for column binding as `SQL_NUMERIC_STRUCT`.
- Added recognition of `SQL_C_SHORT` and `SQL_C_TINYINT` as C types.

Bugs fixed:

- Fixed wildcard handling of and listing of catalogs and tables in [SQLTables](#).
- Added limit of display size when requested via [SQLColAttribute/SQL\\_DESC\\_DISPLAY\\_SIZE](#).
- Fixed buffer length return for [SQLDriverConnect](#).
- ODBC v2 behaviour in driver now supports ODBC v3 date/time types (since [DriverManager](#) maps them).
- Catch use of [SQL\\_ATTR\\_PARAMSET\\_SIZE](#) and report error until we fully support.
- Fixed statistics to fail if it couldn't be completed.
- Corrected retrieval multiple field types bit and blob/text.
- Fixed [SQLGetData](#) to clear the NULL indicator correctly during multiple calls.

## A.12. Changes in MySQL Connector/ODBC 5.0.8 (17 November 2006)

Connector/ODBC 5.0.8 is the fourth BETA release.

This is an implementation and testing release, and is not designed for use within a production environment.

Functionality added or changed:

- Also made [SQL\\_DESC\\_NAME](#) only fill in the name if there was a data pointer given, otherwise just the length.
- Fixed display size to be length if max length isn't available.
- Made distinction between [CHAR/BINARY](#) (and VAR versions).
- Wildcards now support escaped chars and underscore matching (needed to link tables with underscores in access).

Bugs fixed:

- Fixed binding using [SQL\\_C\\_LONG](#).
- Fixed using wrong pointer for [SQL\\_MAX\\_DRIVER\\_CONNECTIONS](#) in [SQLGetInfo](#).
- Set default return to [SQL\\_SUCCESS](#) if nothing is done for [SQLSpecialColumns](#).
- Fixed [MDiagnostic](#) to use correct v2/v3 error codes.
- Allow [SQLDescribeCol](#) to be called to retrieve the length of the column name, but not the name itself.
- Length now used when handling bind parameter (needed in particular for [SQL\\_WCHAR](#)) - this enables updating char data in MS Access.
- Updated retrieval of descriptor fields to use the right pointer types.
- Fixed handling of numeric pointers in [SQLColAttribute](#).
- Fixed type returned for [MYSQL\\_TYPE\\_LONG](#) to [SQL\\_INTEGER](#) instead of [SQL\\_TINYINT](#).
- Fix size return from [SQLDescribeCol](#).
- Fixed string length to chars, not bytes, returned by [SQLGetDiagRec](#).

## A.13. Changes in MySQL Connector/ODBC 5.0.7 (08 November 2006)

Connector/ODBC 5.0.7 is the third BETA release.

This is an implementation and testing release, and is not designed for use within a production environment.

Functionality added or changed:

- Added support for `SQLStatistics` to `MYODBCShell`.
- Improved trace/log.

Bugs fixed:

- `SQLBindParameter` now handles `SQL_C_DEFAULT`.
- Corrected incorrect column index within `SQLStatistics`. Many more tables can now be linked into MS Access.
- Fixed `SQLDescribeCol` returning column name length in bytes rather than chars.

## A.14. Changes in MySQL Connector/ODBC 5.0.6 (03 November 2006)

Connector/ODBC 5.0.6 is the second BETA release.

This is an implementation and testing release, and is not designed for use within a production environment.

Features, limitations and notes on this release

- Connector/ODBC supports both `User` and `System` DSNs.
- Installation is provided in the form of a standard Microsoft System Installer (MSI).
- You no longer have to have Connector/ODBC 3.51 installed before installing this version.

Bugs fixed:

- You no longer have to have Connector/ODBC 3.51 installed before installing this version.
- Connector/ODBC supports both `User` and `System` DSNs.
- Installation is provided in the form of a standard Microsoft System Installer (MSI).

## A.15. Changes in MySQL Connector/ODBC 5.0.5 (17 October 2006)

Connector/ODBC 5.0.5 is the first BETA release.

This is an implementation and testing release, and is not designed for use within a production environment.

You no longer have to have Connector/ODBC 3.51 installed before installing this version.

Bugs fixed:

- You no longer have to have Connector/ODBC 3.51 installed before installing this version.

## A.16. Changes in Connector/ODBC 5.0.3 (Connector/ODBC 5.0 Alpha 3) (20 June 2006)

This is an implementation and testing release, and is not designed for use within a production environment.

Features, limitations and notes on this release:

- The following ODBC API functions have been added in this release:
  - `SQLBindParameter`
  - `SQLBindCol`

## A.17. Changes in Connector/ODBC 5.0.2 (Never released)

Connector/ODBC 5.0.2 was an internal implementation and testing release.

## A.18. Changes in Connector/ODBC 5.0.1 (Connector/ODBC 5.0 Alpha 2) (05 June 2006)

Features, limitations and notes on this release:

- Connector/ODBC 5.0 is Unicode aware.
- Connector/ODBC is currently limited to basic applications. ADO applications and Microsoft Office are not supported.
- Connector/ODBC must be used with a Driver Manager.
- The following ODBC API functions are implemented:
  - `SQLAllocHandle`
  - `SQLCloseCursor`
  - `SQLColAttribute`
  - `SQLColumns`
  - `SQLConnect`
  - `SQLCopyDesc`
  - `SQLDisconnect`
  - `SQLExecDirect`
  - `SQLExecute`
  - `SQLFetch`
  - `SQLFreeHandle`
  - `SQLFreeStmt`
  - `SQLGetConnectAttr`
  - `SQLGetData`
  - `SQLGetDescField`
  - `SQLGetDescRec`
  - `SQLGetDiagField`
  - `SQLGetDiagRec`
  - `SQLGetEnvAttr`
  - `SQLGetFunctions`
  - `SQLGetStmtAttr`
  - `SQLGetTypeInfo`
  - `SQLNumResultCols`
  - `SQLPrepare`
  - `SQLRowcount`
  - `SQLTables`

The following ODBC API function are implemented, but not yet support all the available attributes/options:

- [SQLSetConnectAttr](#)
- [SQLSetDescField](#)
- [SQLSetDescRec](#)
- [SQLSetEnvAttr](#)
- [SQLSetStmtAttr](#)

## A.19. Changes in MySQL Connector/ODBC 3.51.27 (20 November 2008)

Bugs fixed:

- The client program hung when the network connection to the server was interrupted. ([Bug#40407](#))
- The connection string option `Enable Auto-reconnect` did not work. When the connection failed, it could not be restored, and the errors generated were the same as if the option had not been selected. ([Bug#37179](#))
- It was not possible to use Connector/ODBC to connect to a server using SSL. The following error was generated:

```
Runtime error '-2147467259 (80004005)':  
[MySQL][ODBC 3.51 Driver]SSL connection error.
```

([Bug#29955](#))

## A.20. Changes in MySQL Connector/ODBC 3.51.26 (07 July 2008)

Functionality added or changed:

- There is a new connection option, `FLAG_NO_BINARY_RESULT`. When set this option disables charset 63 for columns with an empty `org_table`. ([Bug#29402](#))

Bugs fixed:

- When an `ADOConnection` is created and attempts to open a schema with `ADOConnection.OpenSchema` an access violation occurs in `myodbc3.dll`. ([Bug#30770](#))
- When `SHOW CREATE TABLE` was invoked and then the field values read, the result was truncated and unusable if the table had many rows and indexes. ([Bug#24131](#))

## A.21. Changes in MySQL Connector/ODBC 3.51.25 (11 April 2008)

Bugs fixed:

- The `SQLColAttribute()` function returned `SQL_TRUE` when querying the `SQL_DESC_FIXED_PREC_SCALE` (`SQL_COLUMN_MONEY`) attribute of a `DECIMAL` column. Previously, the correct value of `SQL_FALSE` was returned; this is now again the case. ([Bug#35581](#))
- The driver crashes ODBC Administrator on attempting to add a new DSN. ([Bug#32057](#))
- When accessing column data, `FLAG_COLUMN_SIZE_S32` did not limit the octet length or display size reported for fields, causing problems with Microsoft Visual FoxPro.

The list of ODBC functions that could have caused failures in Microsoft software when retrieving the length of `LONGBLOB` or `LONGTEXT` columns includes:

- [SQLColumns](#)

- `SQLColAttribute`
- `SQLColAttributes`
- `SQLDescribeCol`
- `SQLSpecialColumns` (theoretically can have the same problem) ([Bug#12805](#), [Bug#30890](#))

## A.22. Changes in MySQL Connector/ODBC 3.51.24 (14 March 2008)

Bugs fixed:

- **Security Enhancement:** Accessing a parameter with the type of `SQL_C_CHAR`, but with a numeric type and a length of zero, the parameter marker would get stripped from the query. In addition, an SQL injection was possible if the parameter value had a nonzero length and was not numeric, the text would be inserted verbatim. ([Bug#34575](#))
- **Important Change:** In previous versions, the SSL certificate would automatically be verified when used as part of the Connector/ODBC connection. The default mode is now to ignore the verification of certificates. To enforce verification of the SSL certificate during connection, use the `SSLVERIFY` DSN parameter, setting the value to 1. ([Bug#29955](#), [Bug#34648](#))
- When using ADO, the count of parameters in a query would always return zero. ([Bug#33298](#))
- Using tables with a single quote or other non-standard characters in the table or column names through ODBC would fail. ([Bug#32989](#))
- When using Crystal Reports, table and column names would be truncated to 21 characters, and truncated columns in tables where the truncated name was the duplicated would lead to only a single column being displayed. ([Bug#32864](#))
- `SQLExtendedFetch()` and `SQLFetchScroll()` ignored the rowset size if the `Don't cache result` DSN option was set. ([Bug#32420](#))
- When using the ODBC `SQL_TXN_READ_COMMITTED` option, 'dirty' records would be read from tables as if the option had not been applied. ([Bug#31959](#))
- When creating a System DSN using the ODBC Administrator on Mac OS X, a User DSN would be created instead. The root cause is a problem with the iODBC driver manager used on Mac OS X. The fix works around this issue.

### Note

ODBC Administrator may still be unable to register a System DSN unless the `/Library/ODBC/odbc.ini` file has the correct permissions. You should ensure that the file is writable by the `admin` group.

([Bug#31495](#))

- Calling `SQLFetch` or `SQLFetchScroll` would return negative data lengths when using `SQL_C_WCHAR`. ([Bug#31220](#))
- `SQLSetParam()` caused memory allocation errors due to driver manager's mapping of deprecated functions (buffer length - 1). ([Bug#29871](#))
- Static cursor was unable to be used through ADO when dynamic cursors were enabled. ([Bug#27351](#))
- Using `connection.Execute` to create a record set based on a table without declaring the cmd option as `adCmdTable` will fail when communicating with versions of MySQL 5.0.37 and higher. The issue is related to the way that `SQLSTATE` is returned when ADO tries to confirm the existence of the target object. ([Bug#27158](#))
- Updating a `RecordSet` when the query involves a `BLOB` field would fail. ([Bug#19065](#))
- With some connections to MySQL databases using Connector/ODBC, the connection would mistakenly report 'user cancelled' for accesses to the database information. ([Bug#16653](#))

## A.23. Changes in MySQL Connector/ODBC 3.51.23 (09 January 2008)

Platform specific notes:

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- There are no installer packages for Microsoft Windows x64 Edition.

Bugs fixed:

- Connector/ODBC would incorrectly return `SQL_SUCCESS` when checking for distributed transaction support. ([Bug#32727](#))
- When using unixODBC or directly linked applications where the thread level is set to less than 3 (within `odbcinst.ini`), a thread synchronization issue would lead to an application crash. This was because `SQLAllocStmt()` and `SQLFreeStmt()` did not synchronize access to the list of statements associated with a connection. ([Bug#32587](#))
- Cleaning up environment handles in multithread environments could result in a five (or more) second delay. ([Bug#32366](#))
- Renaming an existing DSN entry would create a new entry with the new name without deleting the old entry. ([Bug#31165](#))
- Setting the default database using the `DefaultDatabase` property of an ADO `Connection` object would fail with the error `Provider does not support this property`. The `SQLGetInfo()` returned the wrong value for `SQL_DATABASE_NAME` when no database was selected. ([Bug#3780](#))

## A.24. Changes in MySQL Connector/ODBC 3.51.22 (13 November 2007)

Functionality added or changed:

- The workaround for this bug was removed due to the fixes in MySQL Server 5.0.48 and 5.1.21.

This regression was introduced by [Bug#10491](#).

Bugs fixed:

- The `English` locale would be used when formatting floating point values. The `C` locale is now used for these values. ([Bug#32294](#))
- When accessing information about supported operations, the driver would return incorrect information about the support for `UNION`. ([Bug#32253](#))
- Unsigned integer values greater than the maximum value of a signed integer would be handled incorrectly. ([Bug#32171](#))
- The wrong result was returned by `SQLGetData()` when the data was an empty string and a zero-sized buffer was specified. ([Bug#30958](#))
- Added the `FLAG_COLUMN_SIZE_S32` option to limit the reported column size to a signed 32-bit integer. This option is automatically enabled for ADO applications to provide a work around for a bug in ADO. ([Bug#13776](#))

## A.25. Changes in MySQL Connector/ODBC 3.51.21 (08 October 2007)

Bugs fixed:

- When using a rowset/cursor and add a new row with a number of fields, subsequent rows with fewer fields will include the original fields from the previous row in the final `INSERT` statement. ([Bug#31246](#))
- Uninitiated memory could be used when C/ODBC internally calls `SQLGetFunctions()`. ([Bug#31055](#))
- The wrong `SQL_DESC_LITERAL_PREFIX` would be returned for date/time types. ([Bug#31009](#))
- The wrong `COLUMN_SIZE` would be returned by `SQLGetTypeInfo` for the `TIME` columns (`SQL_TYPE_TIME`). ([Bug#30939](#))

- Clicking outside the character set selection box when configuring a new DSN could cause the wrong character set to be selected. ([Bug#30568](#))
- Not specifying a user in the DSN dialog would raise a warning even though the parameter is optional. ([Bug#30499](#))
- `SQLSetParam()` caused memory allocation errors due to driver manager's mapping of deprecated functions (buffer length - 1). ([Bug#29871](#))
- When using ADO, a column marked as `AUTO_INCREMENT` could incorrectly report that the column allowed `NULL` values. This was due to an issue with `NULLABLE` and `IS_NULLABLE` return values from the call to `SQLColumns()`. ([Bug#26108](#))
- Connector/ODBC would return the wrong error code when the server disconnects the active connection because the configured `wait_timeout` has expired. Previously it would return `HY000`. Connector/ODBC now correctly returns an `SQLSTATE` of `08S01`. ([Bug#3456](#))

## A.26. Changes in MySQL Connector/ODBC 3.51.20 (10 September 2007)

Bugs fixed:

- Using `FLAG_NO_PROMPT` doesn't suppress the dialogs normally handled by `SQLDriverConnect`. ([Bug#30840](#))
- The specified length of the user name and authentication parameters to `SQLConnect()` were not being honored. ([Bug#30774](#))
- The wrong column size was returned for binary data. ([Bug#30547](#))
- `SQLGetData()` will now always return `SQL_NO_DATA_FOUND` on second call when no data left, even if requested size is 0. ([Bug#30520](#))
- `SQLGetConnectAttr()` did not reflect the connection state correctly. ([Bug#14639](#))
- Removed checkbox in setup dialog for `FLAG_FIELD_LENGTH` (identified as `Don't Optimize Column Width` within the GUI dialog), which was removed from the driver in 3.51.18.

## A.27. Changes in MySQL Connector/ODBC 3.51.19 (10 August 2007)

Connector/ODBC 3.51.19 fixes a specific issue with the 3.51.18 release. For a list of changes in the 3.51.18 release, see [Section A.28, "Changes in MySQL Connector/ODBC 3.51.18 \(08 August 2007\)"](#).

Functionality added or changed:

- Because of [Bug#10491](#) in the server, character string results were sometimes incorrectly identified as `SQL_VARBINARY`. Until this server bug is corrected, the driver will identify all variable-length strings as `SQL_VARCHAR`.

## A.28. Changes in MySQL Connector/ODBC 3.51.18 (08 August 2007)

Platform specific notes:

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- Binary packages for Sun Solaris are now available as `PKG` packages.
- Binary packages as disk images with installers are now available for Mac OS X.
- A binary package without an installer is available for Microsoft Windows x64 Edition. There are no installer packages for Microsoft Windows x64 Edition.

Functionality added or changed:



- **Incompatible Change:** The `FLAG_DEBUG` option was removed.
- When connecting to a specific database when using a DSN, the system tables from the `mysql` database are no longer also available. Previously, tables from the `mysql` database (catalog) were listed as `SYSTEM TABLES` by `SQLTables()` even when a different catalog was being queried. (Bug#28662)
- Installed for Mac OS X has been re-instated. The installer registers the driver at a system (not user) level and makes it possible to create both user and system DSNs using the Connector/ODBC driver. The installer also fixes the situation where the necessary drivers would be installed local to the user, not globally. (Bug#15326, Bug#10444)
- Connector/ODBC now supports batched statements. In order to enable cached statement support you must switch enable the batched statement option (`FLAG_MULTI_STATEMENTS`, 67108864, or `ALLOW MULTIPLE STATEMENTS` within a GUI configuration). Be aware that batched statements create an increased chance of SQL injection attacks and you must ensure that your application protects against this scenario. (Bug#7445)
- The `SQL_ATTR_ROW_BIND_OFFSET_PTR` is now supported for row bind offsets. (Bug#6741)
- The `TRACE` and `TRACEFILE` DSN options have been removed. Use the ODBC driver manager trace options instead.

#### Bugs fixed:

- When using a table with multiple `TIMESTAMP` columns, the final `TIMESTAMP` column within the table definition would not be updateable. Note that there is still a limitation in MySQL server regarding multiple `TIMESTAMP` columns. (Bug#9927) (Bug#30081)
- Fixed an issue where the `myodbc3i` would update the user ODBC configuration file (`~/Library/ODBC/odbcinst.ini`) instead of the system `/Library/ODBC/odbcinst.ini`. This was caused because `myodbc3i` was not honoring the `s` and `u` modifiers for the `-d` command-line option. (Bug#29964)
- Getting table metadata (through the `SQLColumns()`) would fail, returning a bad table definition to calling applications. (Bug#29888)
- `DATETIME` column types would return `FALSE` in place of `SQL_SUCCESS` when requesting the column type information. (Bug#28657)
- The `SQL_COLUMN_TYPE`, `SQL_COLUMN_DISPLAY` and `SQL_COLUMN_PRECISION` values would be returned incorrectly by `SQLColumns()`, `SQLDescribeCol()` and `SQLColAttribute()` when accessing character columns, especially those generated through `concat()`. The lengths returned should now conform to the ODBC specification. The `FLAG_FIELD_LENGTH` option no longer has any affect on the results returned. (Bug#27862)
- Obtaining the length of a column when using a character set for the connection of `utf8` would result in the length being returned incorrectly. (Bug#19345)
- The `SQLColumns()` function could return incorrect information about `TIMESTAMP` columns, indicating that the field was not nullable. (Bug#14414)
- The `SQLColumns()` function could return incorrect information about `AUTO_INCREMENT` columns, indicating that the field was not nullable. (Bug#14407)
- A binary package without an installer is available for Microsoft Windows x64 Edition. There are no installer packages for Microsoft Windows x64 Edition.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- `BIT(n)` columns are now treated as `SQL_BIT` data where `n = 1` and binary data where `n > 1`.
- The wrong value from `SQL_DESC_LITERAL_SUFFIX` was returned for binary fields.
- The `SQL_DATETIME_SUB` column in `SQLColumns()` was not correctly set for date and time types.
- The value for `SQL_DESC_FIXED_PREC_SCALE` was not returned correctly for values in MySQL 5.0 and later.
- The wrong value for `SQL_DESC_TYPE` was returned for date and time types.
- `SQLConnect()` and `SQLDriverConnect()` were rewritten to eliminate duplicate code and ensure all options were supported using both connection methods. `SQLDriverConnect()` now only requires the setup library to be present when the call requires it.

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- Binary packages as disk images with installers are now available for Mac OS X.
- Binary packages for Sun Solaris are now available as `PKG` packages.
- The wrong value for `DECIMAL_DIGITS` in `SQLColumns()` was reported for `FLOAT` and `DOUBLE` fields, as well as the wrong value for the scale parameter to `SQLDescribeCol()`, and the `SQL_DESC_SCALE` attribute from `SQLColAttribute()`.
- The `SQL_DATA_TYPE` column in `SQLColumns()` results did not report the correct value for date and time types.

## A.29. Changes in MySQL Connector/ODBC 3.51.17 (14 July 2007)

Platform specific notes:

- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- Binary packages for Sun Solaris are now available as `PKG` packages.
- Binary packages as disk images with installers are now available for Mac OS X.
- A binary package without an installer is available for Microsoft Windows x64 Edition. There are no installer packages for Microsoft Windows x64 Edition.

Functionality added or changed:

- It is now possible to specify a different character set as part of the DSN or connection string. This must be used instead of the `SET NAMES` statement. You can also configure the character set value from the GUI configuration. ([Bug#9498](#), [Bug#6667](#))
- Fixed calling convention ptr and wrong free in `myodbc3i`, and fixed the null terminating (was only one, not two) when writing DSN to string.
- Dis-allow NULL ptr for null indicator when calling `SQLGetData()` if value is null. Now returns `SQL_ERROR` w/state 22002.
- The setup library has been split into its own RPM package, to allow installing the driver itself with no GUI dependencies.

Bugs fixed:

- `myodbc3i` did not correctly format driver info, which could cause the installation to fail. ([Bug#29709](#))
- Connector/ODBC crashed with Crystal Reports due to a rproblem with `SQLProcedures()`. ([Bug#28316](#))
- Fixed a problem where the GUI would crash when configuring or removing a System or User DSN. ([Bug#27315](#))
- Fixed error handling of out-of-memory and bad connections in catalog functions. This might raise errors in code paths that had ignored them in the past. ([Bug#26934](#))
- For a stored procedure that returns multiple result sets, Connector/ODBC returned only the first result set. ([Bug#16817](#))
- Calling `SQLGetDiagField` with `RecNumber 0`, `DiagIdentifier NOT 0` returned `SQL_ERROR`, preventing access to diagnostic header fields. ([Bug#16224](#))
- Added a new DSN option (`FLAG_ZERO_DATE_TO_MIN`) to retrieve `XXXX-00-00` dates as the minimum allowed ODBC date (`XXXX-01-01`). Added another option (`FLAG_MIN_DATE_TO_ZERO`) to mirror this but for bound parameters. `FLAG_MIN_DATE_TO_ZERO` only changes `0000-01-01` to `0000-00-00`. ([Bug#13766](#))
- If there was more than one unique key on a table, the correct fields were not used in handling `SQLSetPos()`. ([Bug#10563](#))
- When inserting a large `BLOB` field, Connector/ODBC would crash due to a memory allocation error. ([Bug#10562](#))
- The driver was using `mysql_odbc_escape_string()`, which does not handle the `NO_BACKSLASH_ESCAPES` SQL

mode. Now it uses `mysql_real_escape_string()`, which does. (Bug#9498)

- `SQLColumns()` did not handle many of its parameters correctly, which could lead to incorrect results. The table name argument was not handled as a pattern value, and most arguments were not escaped correctly when they contained non-alphanumeric characters. (Bug#8860)
- There are no binary packages for Microsoft Windows x64 Edition.
- There is no binary package for Mac OS X on 64-bit PowerPC because Apple does not currently provide a 64-bit PowerPC version of iODBC.
- Correctly return error if `SQLBindCol` is called with an invalid column.
- Fixed possible crash if `SQLBindCol()` was not called before `SQLSetPos()`.
- The Mac OS X binary packages are only provided as tarballs, there is no installer.
- The binary packages for Sun Solaris are only provided as tarballs, not the PKG format.
- The HP-UX 11.23 IA64 binary package does not include the GUI bits because of problems building Qt on that platform.

## A.30. Changes in MySQL Connector/ODBC 3.51.16 (14 June 2007)

Functionality added or changed:

- Connector/ODBC now supports using SSL for communication. This is not yet exposed in the setup GUI, but must be enabled through configuration files or the DSN. (Bug#12918)

Bugs fixed:

- Calls to `SQLNativeSql()` could cause stack corruption due to an incorrect pointer cast. (Bug#28758)
- Using cursors on results sets with multi-column keys could select the wrong value. (Bug#28255)
- `SQLForeignKeys` does not escape `_` and `%` in the table name arguments. (Bug#27723)
- When using stored procedures, making a `SELECT` or second stored procedure call after an initial stored procedure call, the second statement will fail. (Bug#27544)
- `SQLTables()` did not distinguish tables from views. (Bug#23031)
- Data in `TEXT` columns would fail to be read correctly. (Bug#16917)
- Specifying strings as parameters using the `adBSTR` or `adVarChar` types, (`SQL_WVARCHAR` and `SQL_WLONGVARCHAR`) would be incorrectly quoted. (Bug#16235)
- `SQL_WVARCHAR` and `SQL_WLONGVARCHAR` parameters were not properly quoted and escaped. (Bug#16235)
- Using `BETWEEN` with date values, the wrong results could be returned. (Bug#15773)
- When using the `Don't Cache Results` (option value 1048576) with Microsoft Access, the connection will fail using DAO/VisualBasic. (Bug#4657)
- Return values from `SQLTables()` may be truncated. (Bugs #22797)

## A.31. Changes in MySQL Connector/ODBC 3.51.15 (07 May 2007)

Bugs fixed:

- Connector/ODBC would incorrectly claim to support `SQLProcedureColumns` (by returning true when queried about `SQLPROCEDURECOLUMNS` with `SQLGetFunctions`), but this functionality is not supported. (Bug#27591)
- An incorrect transaction isolation level may not be returned when accessing the connection attributes. (Bug#27589)

- Adding a new DSN with the `myodbc3i` utility under AIX would fail. ([Bug#27220](#))
- When inserting data using bulk statements (through `SQLBulkOperations`), the indicators for all rows within the insert would not updated correctly. ([Bug#24306](#))
- Using `SQLProcedures` does not return the database name within the returned resultset. ([Bug#23033](#))
- The `SQLTransact()` function did not support an empty connection handle. ([Bug#21588](#))
- Using `SQLDriverConnect` instead of `SQLConnect` could cause later operations to fail. ([Bug#7912](#))
- When using blobs and parameter replacement in a statement with `WHERE CURSOR OF`, the SQL is truncated. ([Bug#5853](#))
- Connector/ODBC would return too many foreign key results when accessing tables with similar names. ([Bug#4518](#))

## A.32. Changes in MySQL Connector/ODBC 3.51.14 (08 March 2007)

Functionality added or changed:

- Use of `SQL_ATTR_CONNECTION_TIMEOUT` on the server has now been disabled. If you attempt to set this attribute on your connection the `SQL_SUCCESS_WITH_INFO` will be returned, with an error number/string of `HYC00: Optional feature not supported`. ([Bug#19823](#))
- Added `auto is null` option to Connector/ODBC option parameters. ([Bug#10910](#))
- Added `auto-reconnect` option to Connector/ODBC option parameters.
- Added support for the `HENV` handlers in `SQLEndTran()`.

Bugs fixed:

- On 64-bit systems, some types would be incorrectly returned. ([Bug#26024](#))
- When retrieving `TIME` columns, C/ODBC would incorrectly interpret the type of the string and could interpret it as a `DATE` type instead. ([Bug#25846](#))
- Connector/ODBC may insert the wrong parameter values when using prepared statements under 64-bit Linux. ([Bug#22446](#))
- Using Connector/ODBC, with `SQLBindCol` and binding the length to the return value from `SQL_LEN_DATA_AT_EXEC` fails with a memory allocation error. ([Bug#20547](#))
- Using `DataAdapter`, Connector/ODBC may continually consume memory when reading the same records within a loop (Windows Server 2003 SP1/SP2 only). ([Bug#20459](#))
- When retrieving data from columns that have been compressed using `COMPRESS()`, the retrieved data would be truncated to 8KB. ([Bug#20208](#))
- The ODBC driver name and version number were incorrectly reported by the driver. ([Bug#19740](#))
- A string format exception would be raised when using iODBC, Connector/ODBC and the embedded MySQL server. ([Bug#16535](#))
- The `SQLDriverConnect()` ODBC method did not work with recent Connector/ODBC releases. ([Bug#12393](#))

## A.33. Changes in MySQL Connector/ODBC 3.51.13 (Never released)

Connector/ODBC 3.51.13 was an internal implementation and testing release.

## A.34. Changes in MySQL Connector/ODBC 3.51.12 (11 February 2005)

Functionality added or changed:

- N/A

Bugs fixed:

- Using stored procedures with ADO, where the `CommandType` has been set correctly to `adCmdStoredProc`, calls to stored procedures would fail. (Bug#15635)
- File DSNs could not be saved. (Bug#12019)
- `SQLColumns()` returned no information for tables that had a column named using a reserved word. (Bug#9539)

## A.35. Changes in MySQL Connector/ODBC 3.51.11 (28 January 2005)

Bugs fixed:

- `mysql_list_dbcolumns()` and `insert_fields()` were retrieving all rows from a table. Fixed the queries generated by these functions to return no rows. (Bug#8198)
- `SQLGetTypeInfo()` returned `tinyblob` for `SQL_VARBINARY` and nothing for `SQL_BINARY`. Fixed to return `varbinary` for `SQL_VARBINARY`, `binary` for `SQL_BINARY`, and `longblob` for `SQL_LONGVARBINARY`. (Bug#8138)