

CNT 4714: Enterprise Computing Summer 2012

Introduction To MySQL Installation Of MySQL 5.5.25

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
<http://www.cs.ucf.edu/courses/cnt4714/sum2012>

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida



MySQL RDBMS

- MySQL is a **database server** (although it does come with a set of simple client programs). The current stable version is 5.5.25 and can be downloaded from www.mysql.com.
- It is typically used in **thin client** environments. In other words, it is used in client-server systems where the bulk of the processing and storage takes place on the server, and the client is little more than a dumb terminal.
- MySQL performs multithreaded processing, which means that multiple clients are allowed to connect to it and run queries simultaneously. This makes MySQL extremely fast and well suited to client-server environments such as Web sites and other environments that process numerous transactions for multiple users.





The world's most popular open source database

Contact a MySQL Representative Search Login Register

MySQL.com

Downloads (GA) Customer Login



Home Products Services Partners Customers Why MySQL? News & Events How to Buy

GET STARTED

- Try Now
- MySQL Enterprise Edition
- Free Web Seminars
- White Papers
- ISVs and OEMs
- MySQL Training

- 1
- 2
- 3
- 4
- 5
- 6
- 7



OTN Developer Day: MySQL October 18 in London, UK

Click here to go to download page

Register Now »



http://www.mysql.com/downloads/

File Edit View Favorites Tools Help

Google Search Sign In

MySQL :: MySQL Downloads (Generally Available)

MySQL Downloads (Generally Available)

- MySQL Community Server
- MySQL Enterprise Edition
- MySQL Cluster
- MySQL Cluster CGE
- MySQL Workbench (GUI Tool)
- MySQL Connectors



MySQL Installer for Windows

All MySQL Products. For All Windows Platforms. In One Package.

[Download](#)

- New Releases (GA)**
- MySQL Installer 5.5 (5.5.25 GA)
 - MySQL Community Server 5.5 (5.5.25 GA)
 - MySQL Cluster 7.1 (7.1.12 GA)
 - MySQL Cluster 7.2 (7.2.6 GA)
 - Connector/Net 6.4 (6.4.5 GA)

MySQL Community Server

(Current Generally Available Release: 5.5.25)

MySQL Community Server is a freely downloadable version of the world's most

Contact Sales

USA/Canada - Toll Free:



MySQL :: Download MySQL Community Server - Windows Internet Explorer

http://www.mysql.com/downloads/mysql/

File Edit View Favorites Tools Help

MySQL.com Downloads (GA)

Current Downloads (Generally Available)

Download MySQL Community Server

MySQL Community Edition is a freely downloadable version of the world's most popular open source database that is supported by an active community of open source developers and enthusiasts.

MySQL Cluster Community Edition is available as a separate download.

The reason for this change is so that MySQL Cluster can provide more frequent updates and support using the latest sources of MySQL Cluster Carrier Grade Edition.

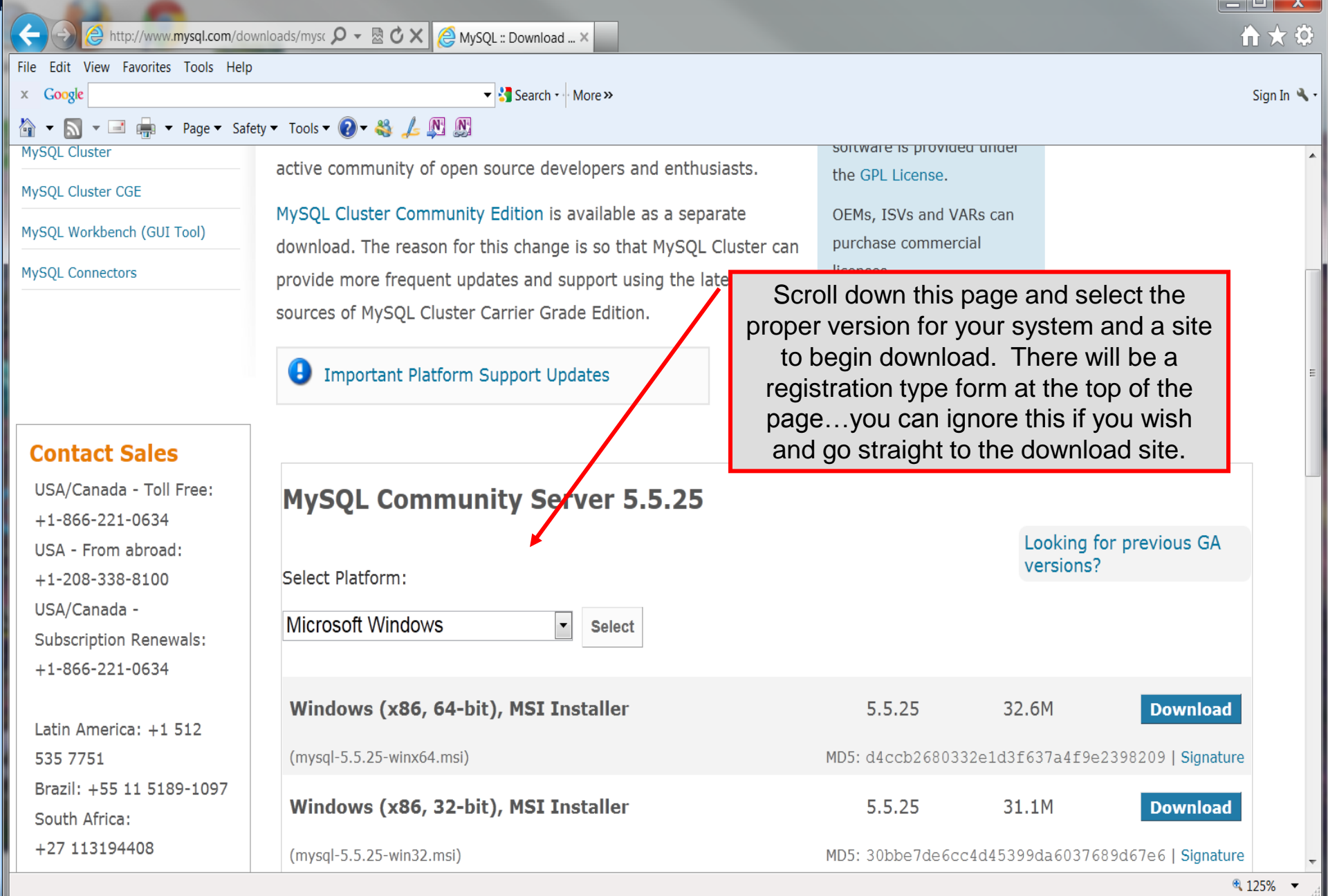
MySQL open source software is provided under the [GPL License](#).

OEMs, ISVs and VARs can purchase commercial licenses.



The MySQL Community Server page.





Scroll down this page and select the proper version for your system and a site to begin download. There will be a registration type form at the top of the page...you can ignore this if you wish and go straight to the download site.

MySQL Community Server 5.5.25

Select Platform:

Microsoft Windows

Windows (x86, 64-bit), MSI Installer

5.5.25

32.6M

[Download](#)

(mysql-5.5.25-winx64.msi)

MD5: d4ccb2680332e1d3f637a4f9e2398209 | [Signature](#)

Windows (x86, 32-bit), MSI Installer

5.5.25

31.1M

[Download](#)

(mysql-5.5.25-win32.msi)

MD5: 30bbe7de6cc4d45399da6037689d67e6 | [Signature](#)

Contact Sales

USA/Canada - Toll Free:

+1-866-221-0634

USA - From abroad:

+1-208-338-8100

USA/Canada -

Subscription Renewals:

+1-866-221-0634

Latin America: +1 512

535 7751

Brazil: +55 11 5189-1097

South Africa:

+27 113194408



Go back to the main download page and also download MySQL Workbench which contains the Administrator and MySQL Query Browser GUI tools.

The Query Browser and Administrator come in a bundle with some other tools. Scroll down and select the correct option for your machine.

Contact Sales

USA/Canada - Toll Free:
+1-866-221-0634
USA - From abroad:
+1-208-338-8100
USA/Canada -
Subscription Renewals:
+1-866-221-0634

Latin America: +1 512
535 7751
Brazil: +55 11 5189-1097
South Africa:
+27 113194408
UK: +44 207 553 8447
Ireland: +353 1 8031050
Germany:
+49 89 143 01280

The Community (OSS) Edition is available from this page under the [GPL](#).

Download source packages of LGPL libraries: [\[+\]](#)

[MySQL Workbench Prerequisites >](#)

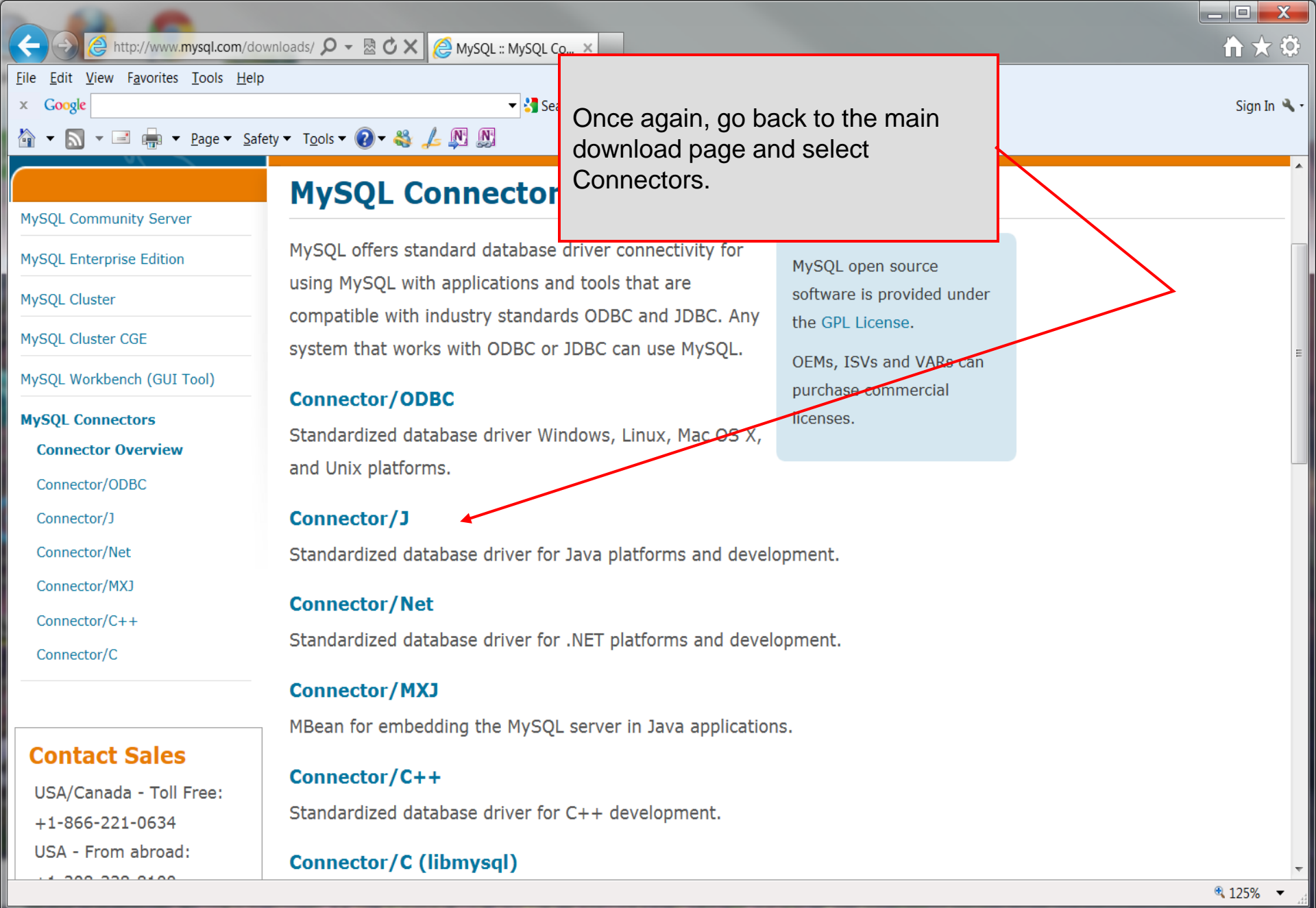
MySQL Workbench 5.2.40

Select Platform:

Microsoft Windows

| | | | |
|---|--------|-------|--------------------------|
| Windows (x86, 32-bit), MSI Installer (mysql-workbench-gpl-5.2.40-win32.msi) | 5.2.40 | 25.6M | Download |
| Windows (x86, 32-bit), ZIP Archive (mysql-workbench-gpl-5.2.40-src.zip) | 5.2.40 | 20.9M | Download |
| Windows (x86, 32-bit), ZIP Archive | 5.2.40 | 30.5M | Download |





Once again, go back to the main download page and select Connectors.

MySQL open source software is provided under the GPL License. OEMs, ISVs and VARs can purchase commercial licenses.

MySQL Connector

MySQL offers standard database driver connectivity for using MySQL with applications and tools that are compatible with industry standards ODBC and JDBC. Any system that works with ODBC or JDBC can use MySQL.

Connector/ODBC

Standardized database driver Windows, Linux, Mac OS X, and Unix platforms.

Connector/J

Standardized database driver for Java platforms and development.

Connector/Net

Standardized database driver for .NET platforms and development.

Connector/MXJ

MBean for embedding the MySQL server in Java applications.

Connector/C++

Standardized database driver for C++ development.

Connector/C (libmysql)

- MySQL Community Server
- MySQL Enterprise Edition
- MySQL Cluster
- MySQL Cluster CGE
- MySQL Workbench (GUI Tool)
- MySQL Connectors**
 - Connector Overview**
 - Connector/ODBC
 - Connector/J
 - Connector/Net
 - Connector/MXJ
 - Connector/C++
 - Connector/C

Contact Sales
USA/Canada - Toll Free:
+1-866-221-0634
USA - From abroad:
+1-300-330-8488



MySQL.com Downloads (GA)

Current Downloads (Generally Available)

Download Connector/J

MySQL Connector/J is the official JDBC driver for MySQL.

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

Connector/J 5.1.20

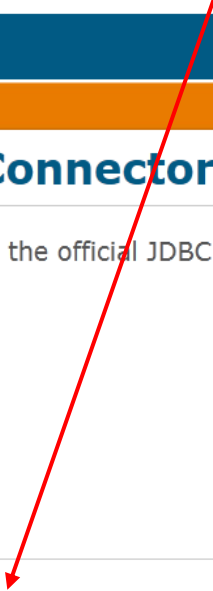
Select Platform:

Platform Independent

Looking for previous GA versions?

| | | | |
|---|--------|------|--------------------------|
| Platform Independent (Architecture Independent), | 5.1.20 | 3.7M | Download |
|---|--------|------|--------------------------|

Download the Connector/J for use with Java applications.



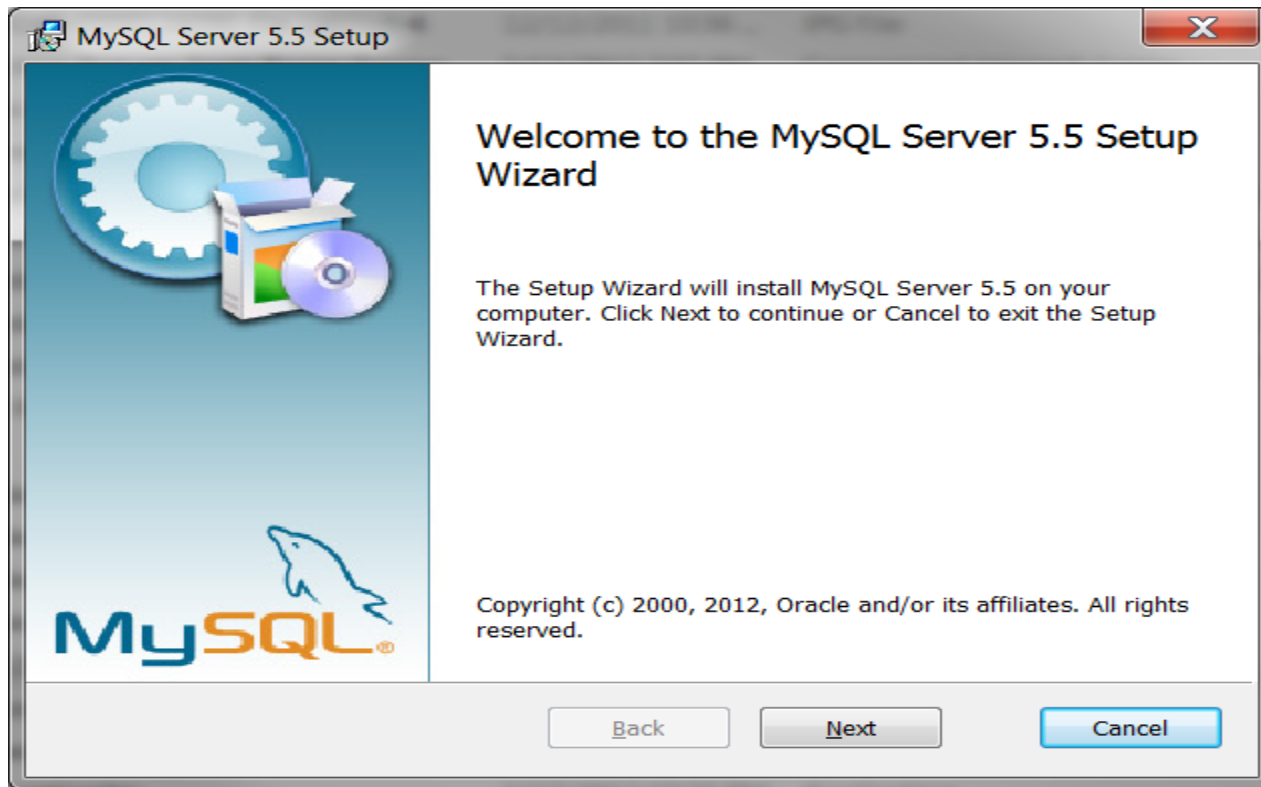
Installing MySQL 5.5.25

- Once you've got MySQL downloaded, go through the installation process. It may vary somewhat depending on platform.
- I've illustrated the basic install on Windows 7 over the next few pages, just to give you an idea of what you should be seeing.

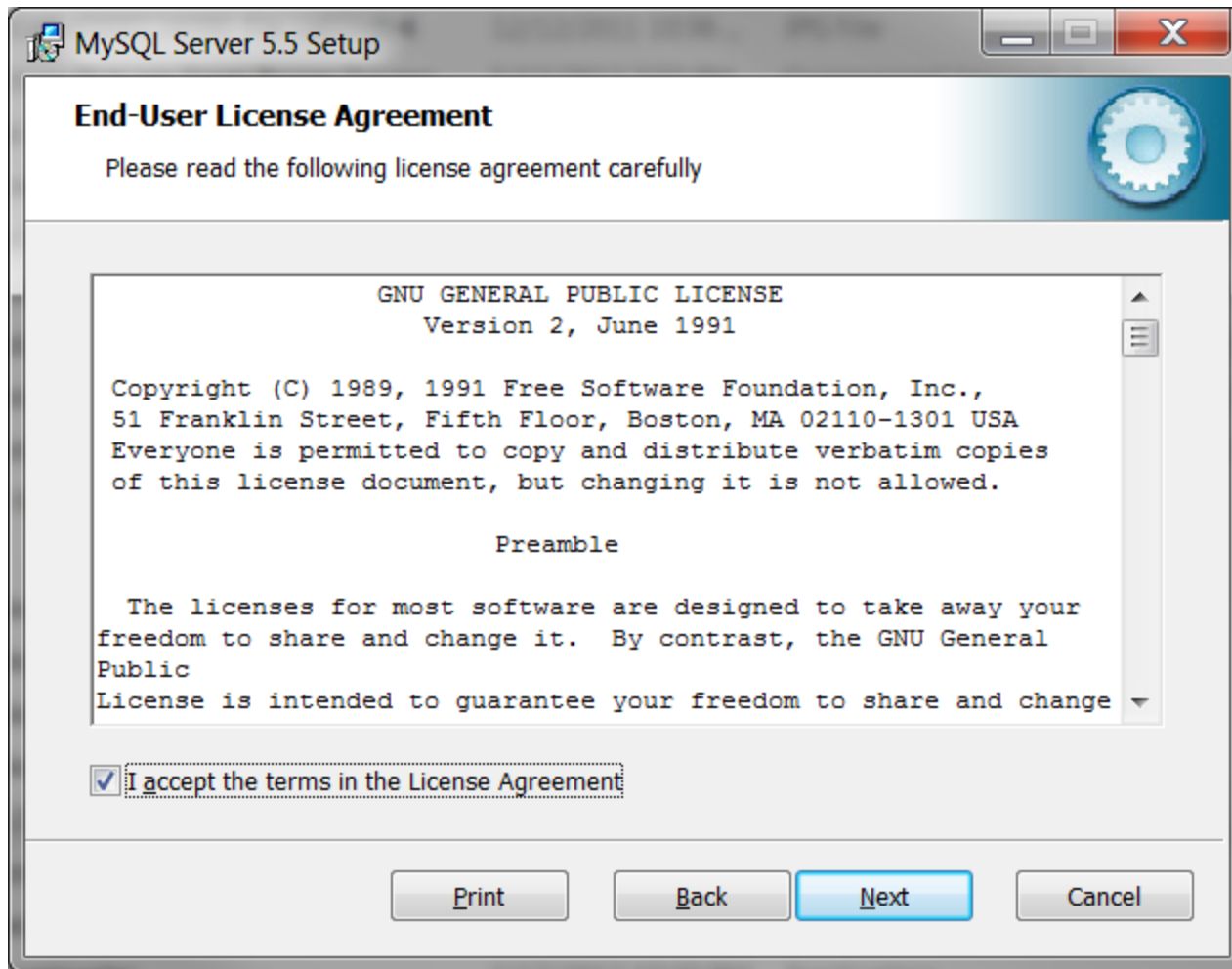


Installing MySQL 5.5.25

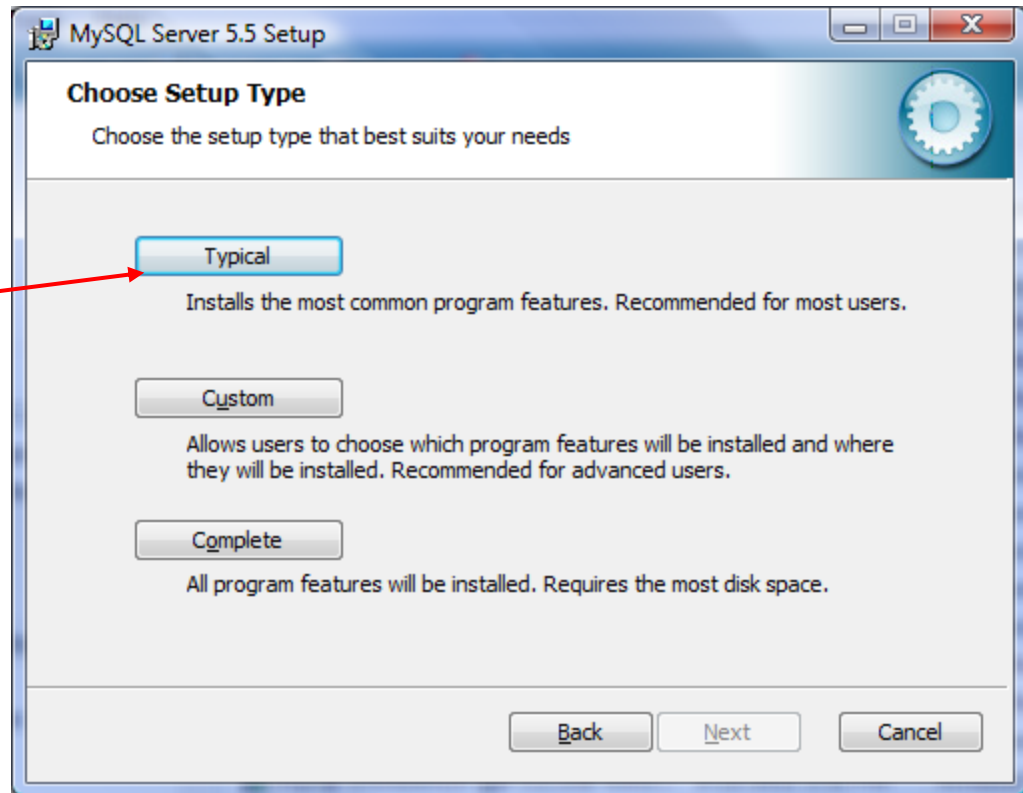
- Once the Windows installer is running you should see the following window appear:
- Click Next and accept the terms on the next window.



Installing MySQL 5.5.25 (cont.)



Installing MySQL 5.5.25 (cont.)

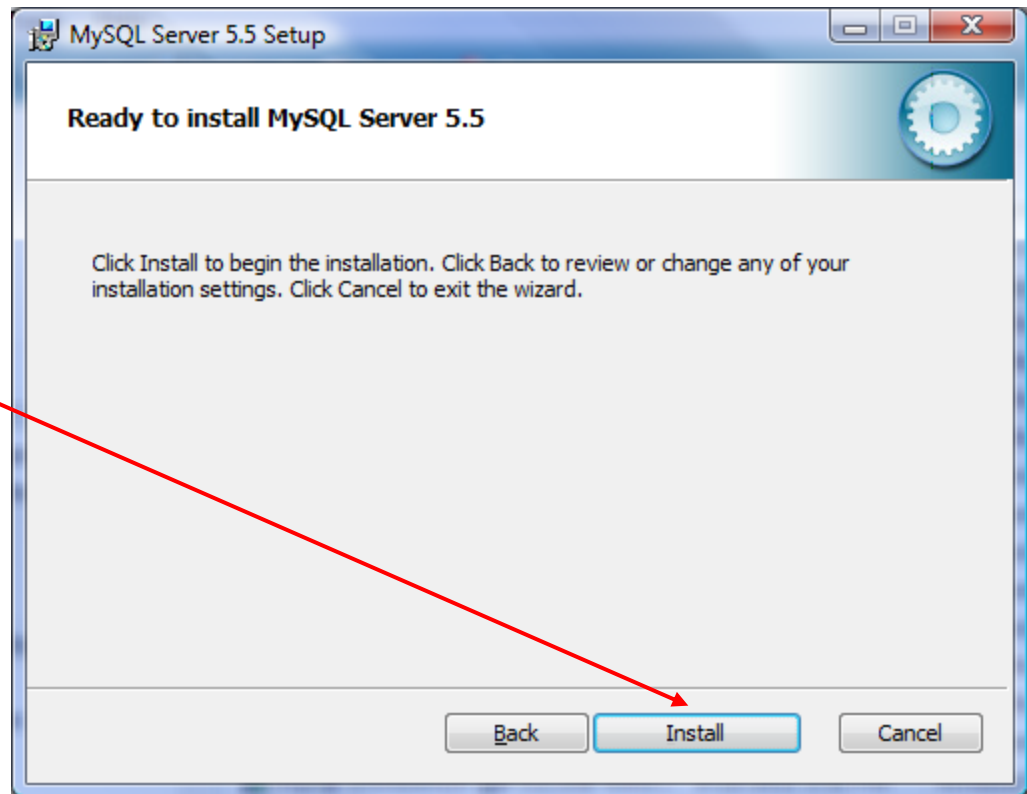


Your choice here.
For this course, a
typical set-up will be
fine.

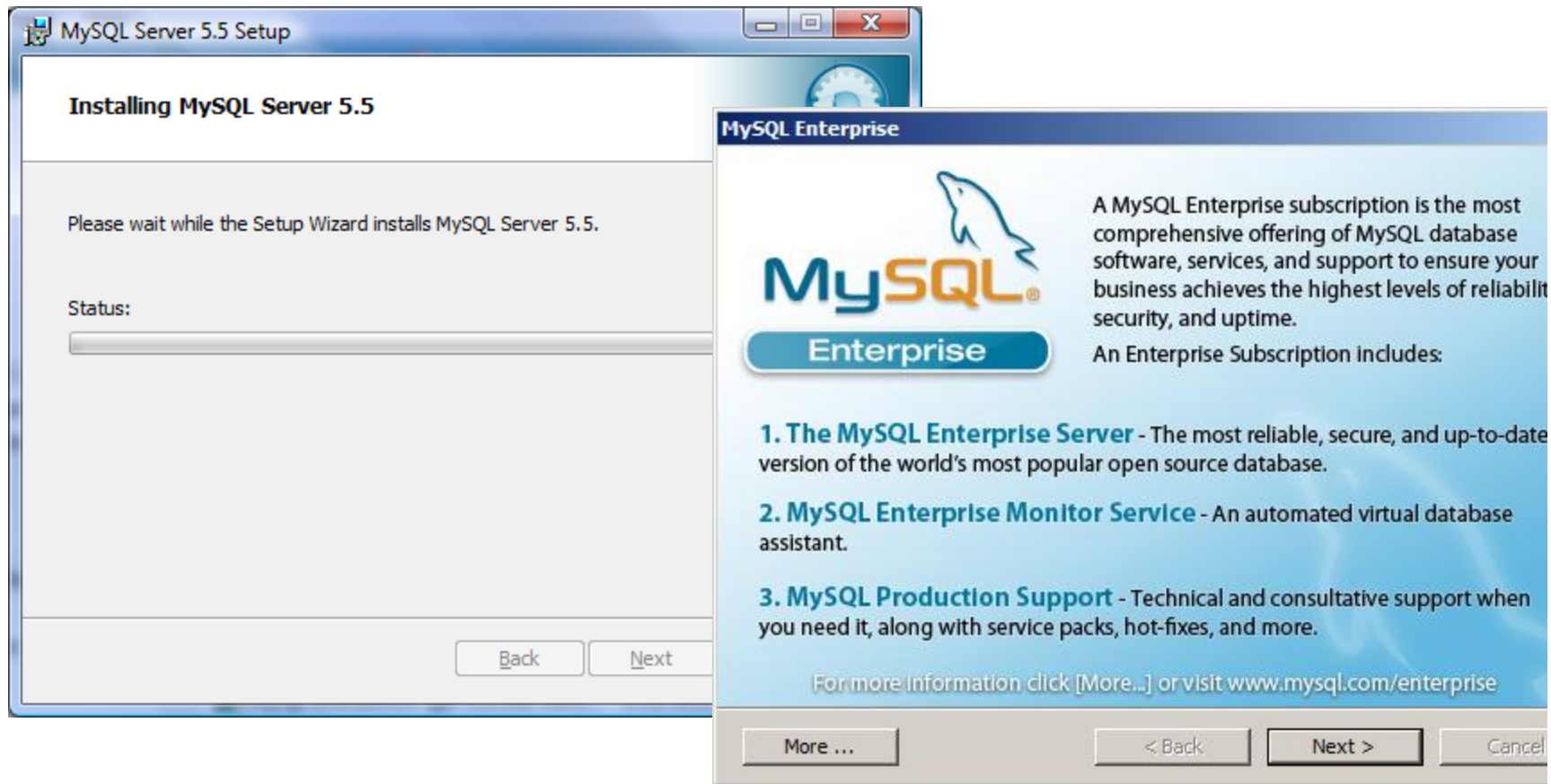


Installing MySQL 5.5.25 (cont.)

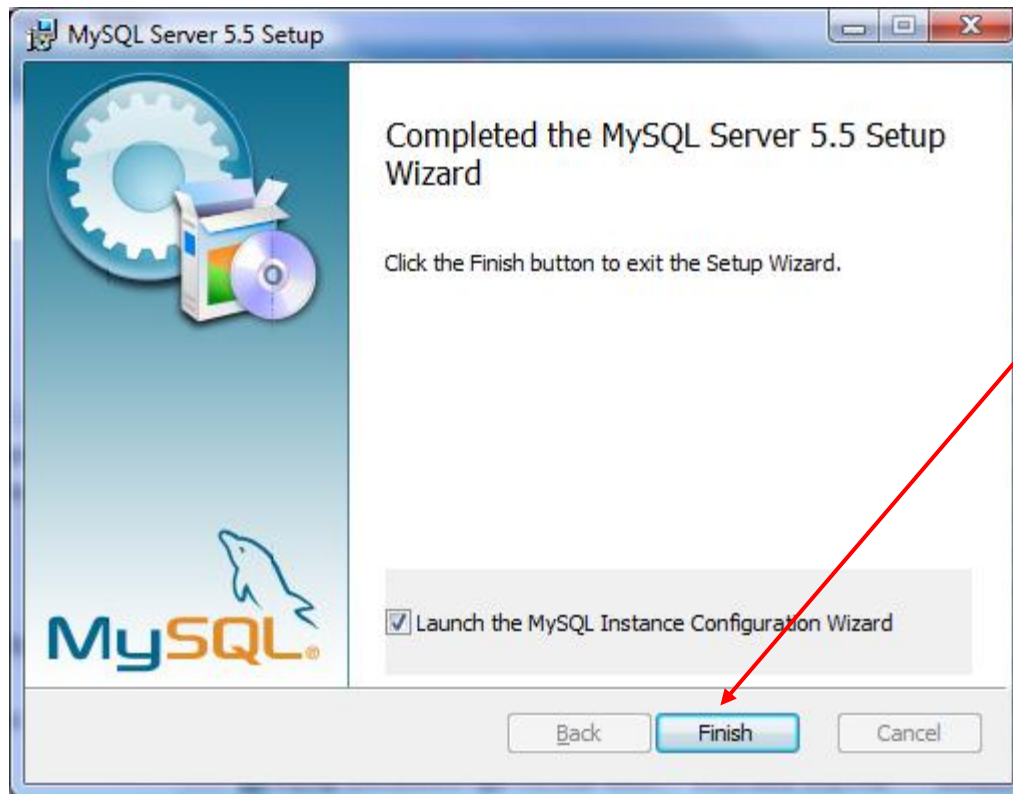
Click Install



Installing MySQL 5.5.25 (cont.)



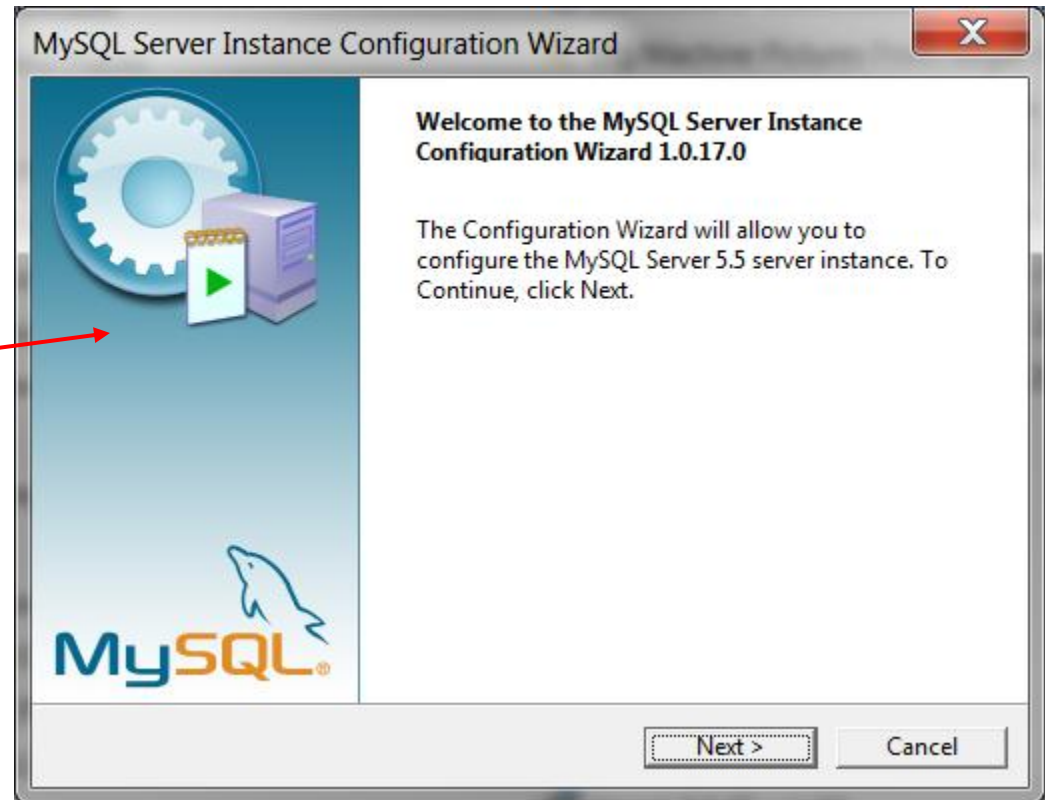
Installing MySQL 5.5.25 (cont.)



Click Finish



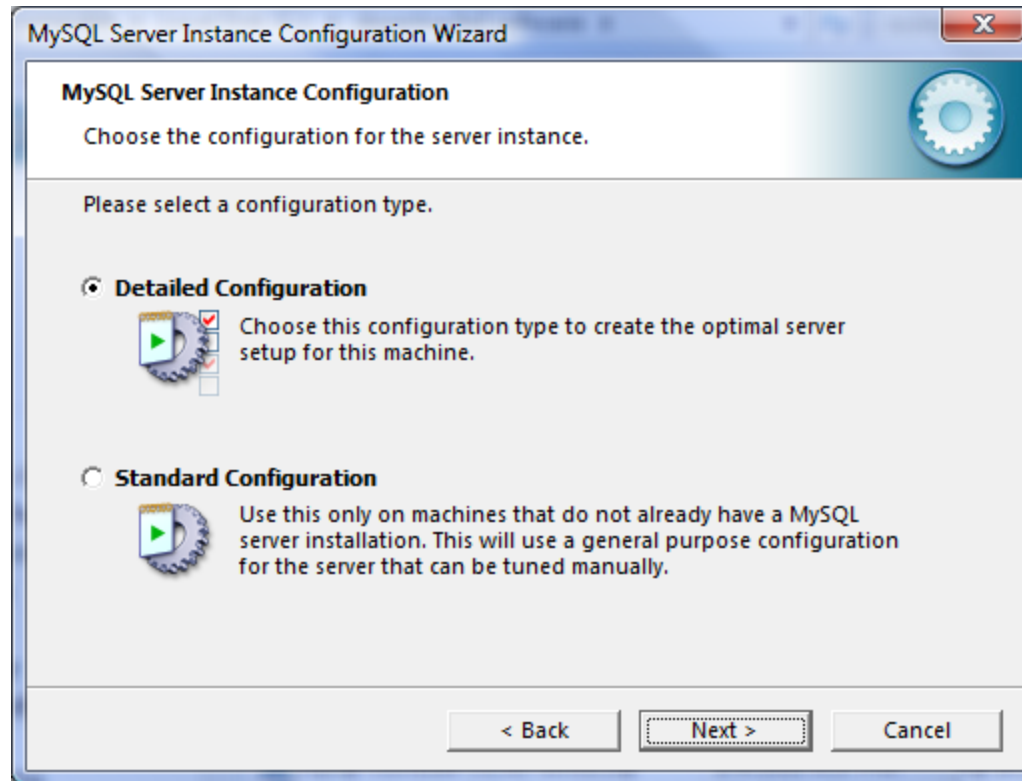
Installing MySQL 5.5.25 (cont.)



Initial server
configuration window



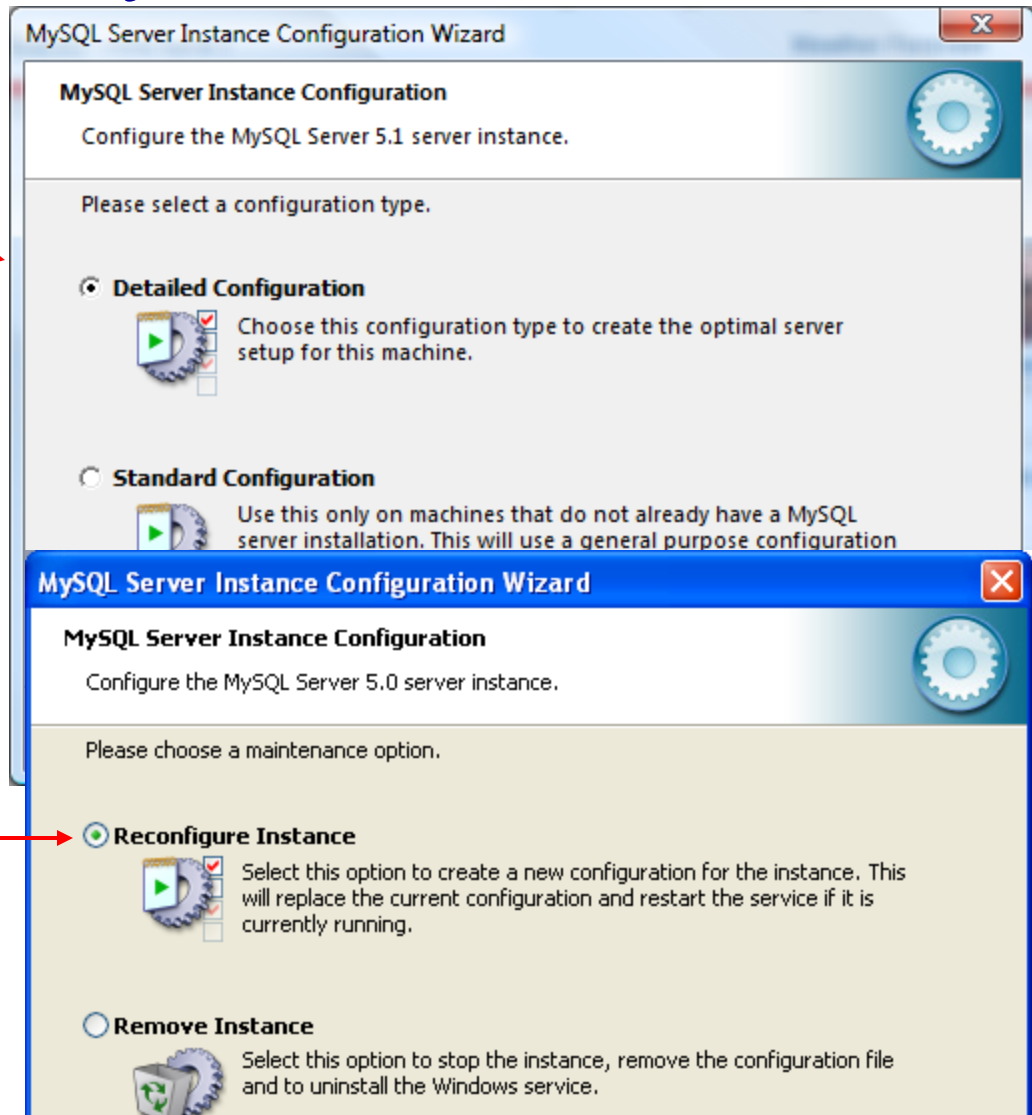
Installing MySQL 5.5.25 (cont.)



Installing MySQL 5.5.25 (cont.)

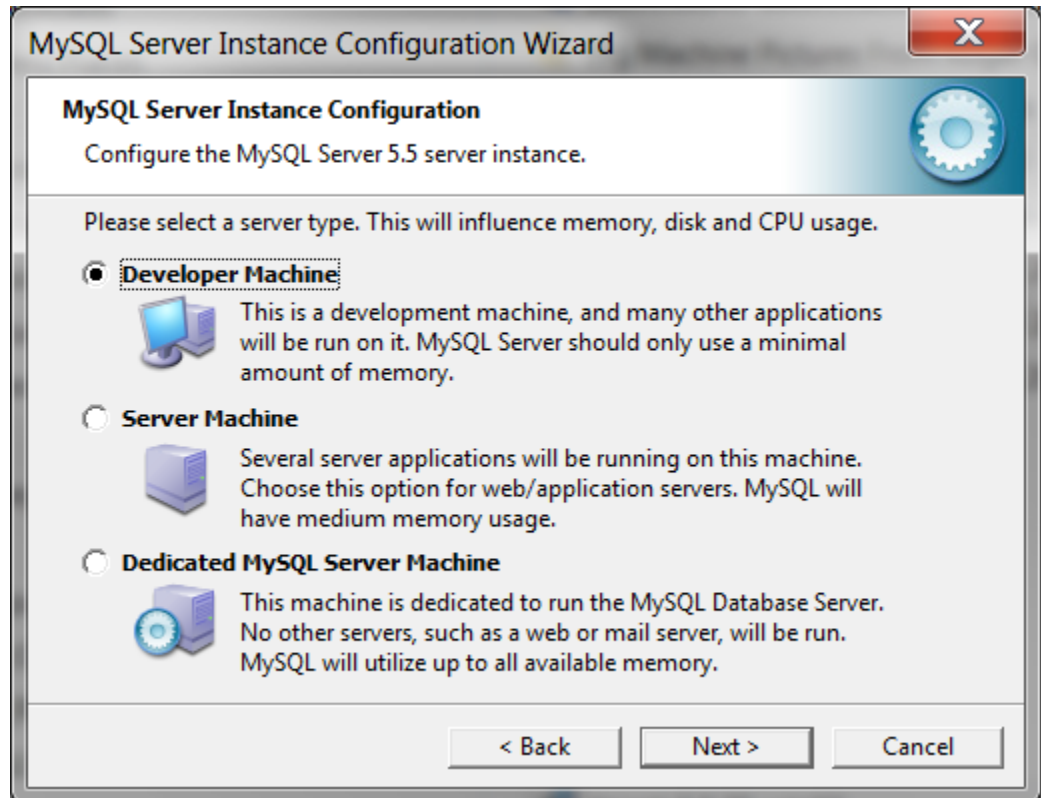
Your choice here. If you are not sure if there is already a MySQL server on your machine, choose the detailed configuration setting.

If you already have an instance of a MySQL server on your machine, you'll see this screen first, followed by the one above. Select reconfigure instance.



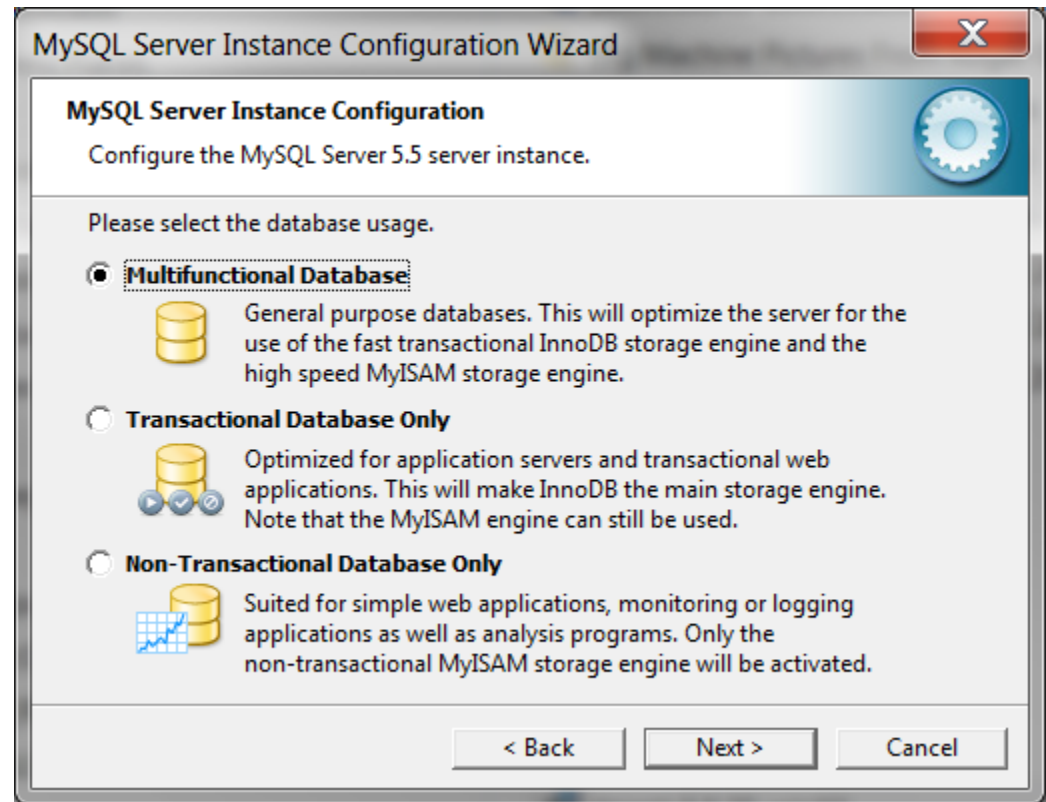
Installing MySQL 5.5.25 (cont.)

Choose the developer machine option



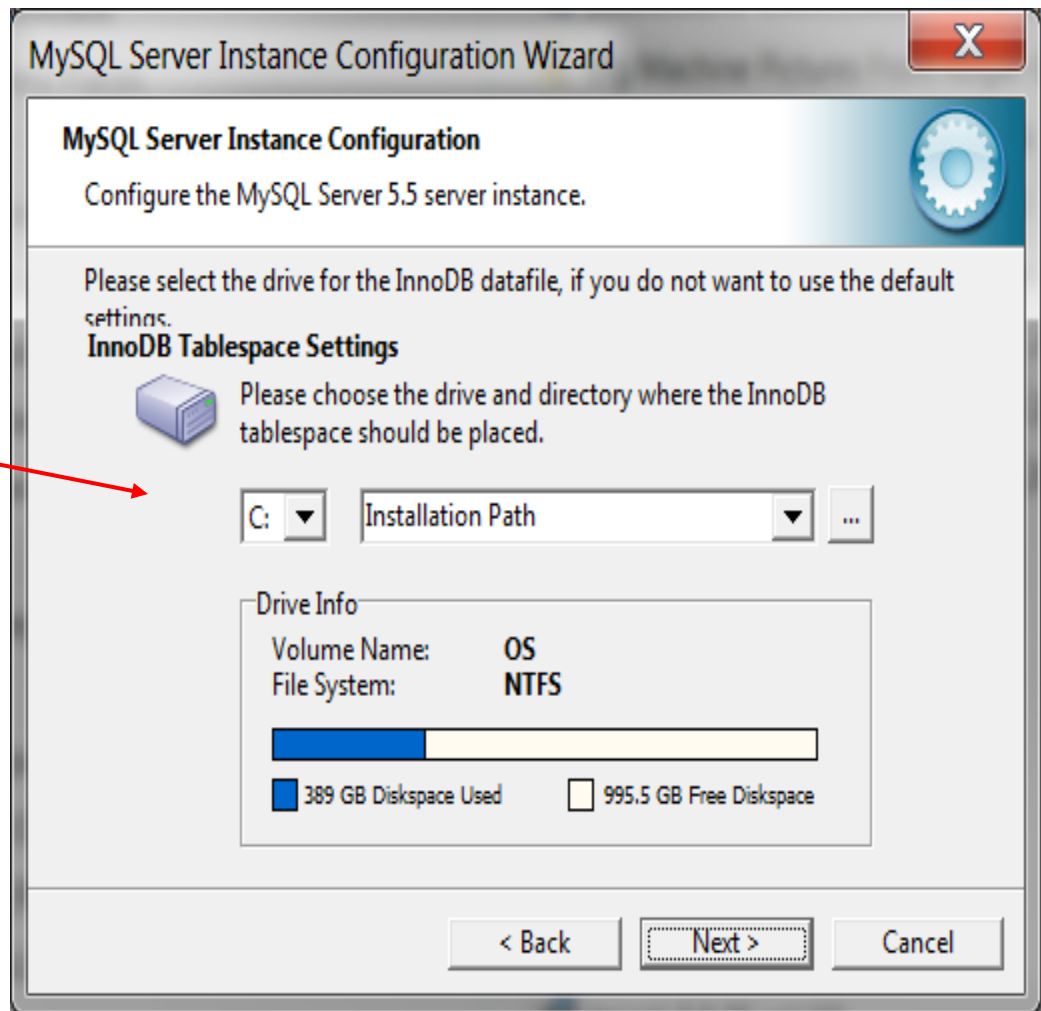
Installing MySQL 5.5.25 (cont.)

Choose the
multifunctional
database option

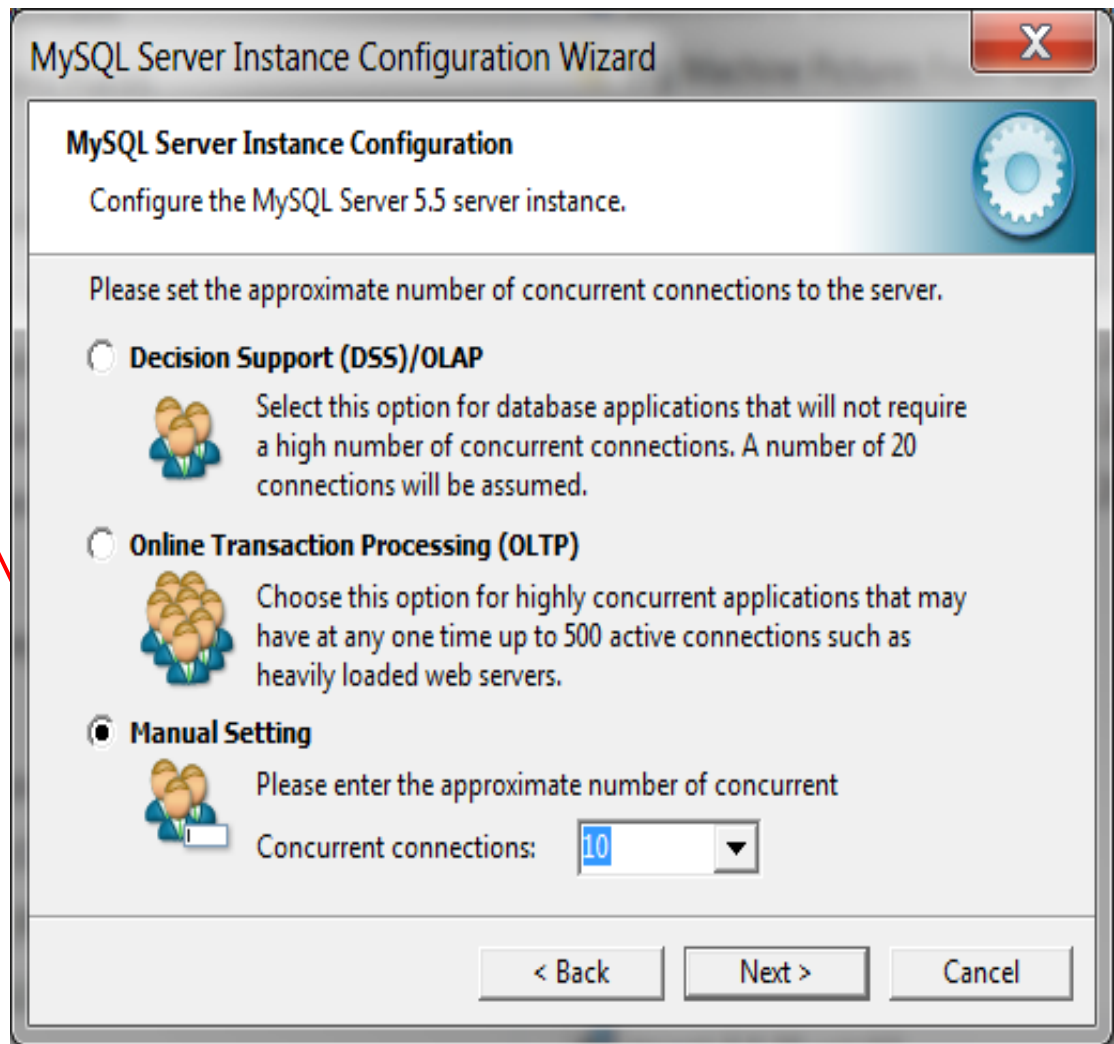


Installing MySQL 5.5.25 (cont.)

Choose the installation path to keep InnoDB tables in same area as other MySQL files



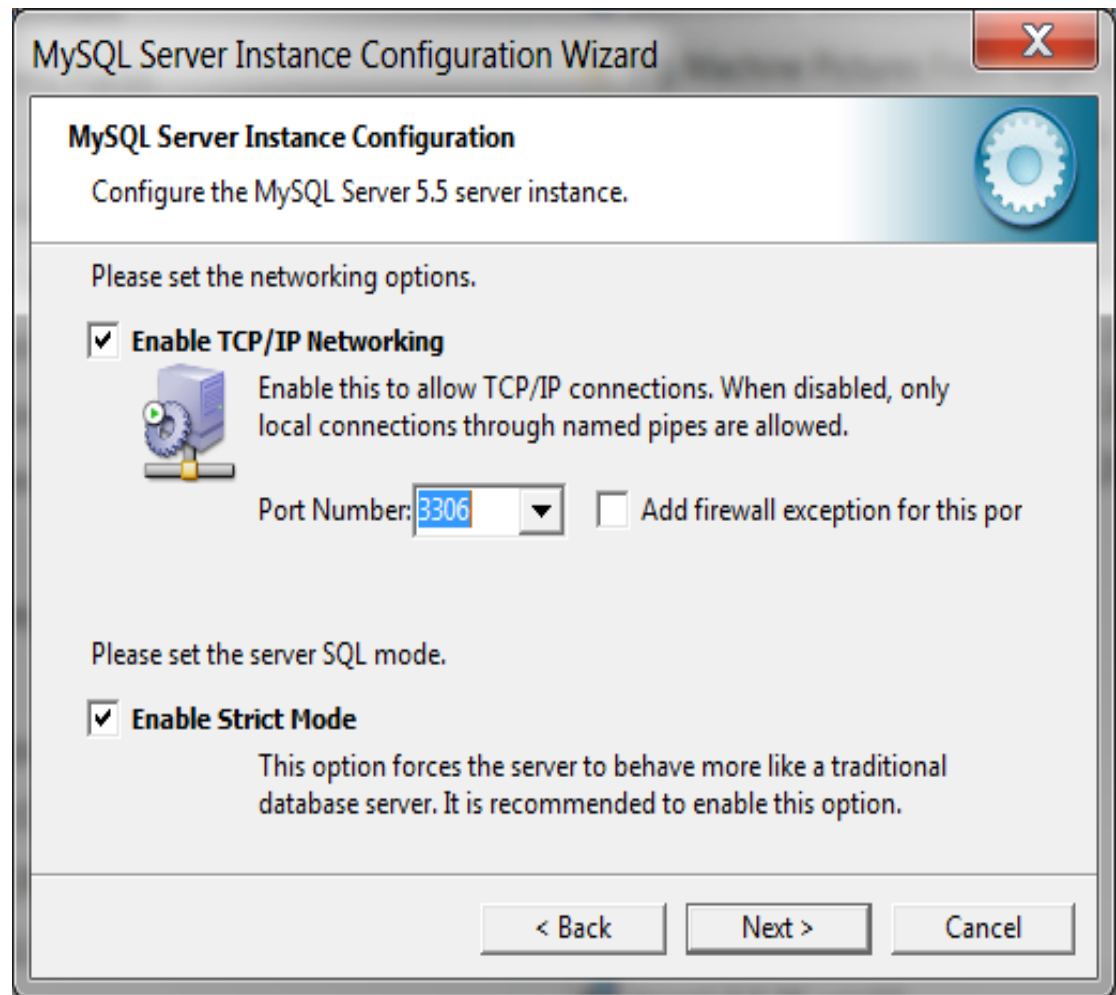
Installing MySQL 5.5.25 (cont.)



Select manual setting for this option. The default is 15, I set mine to 10, but you can use any number you would like, but pick something greater than 3 or 4.



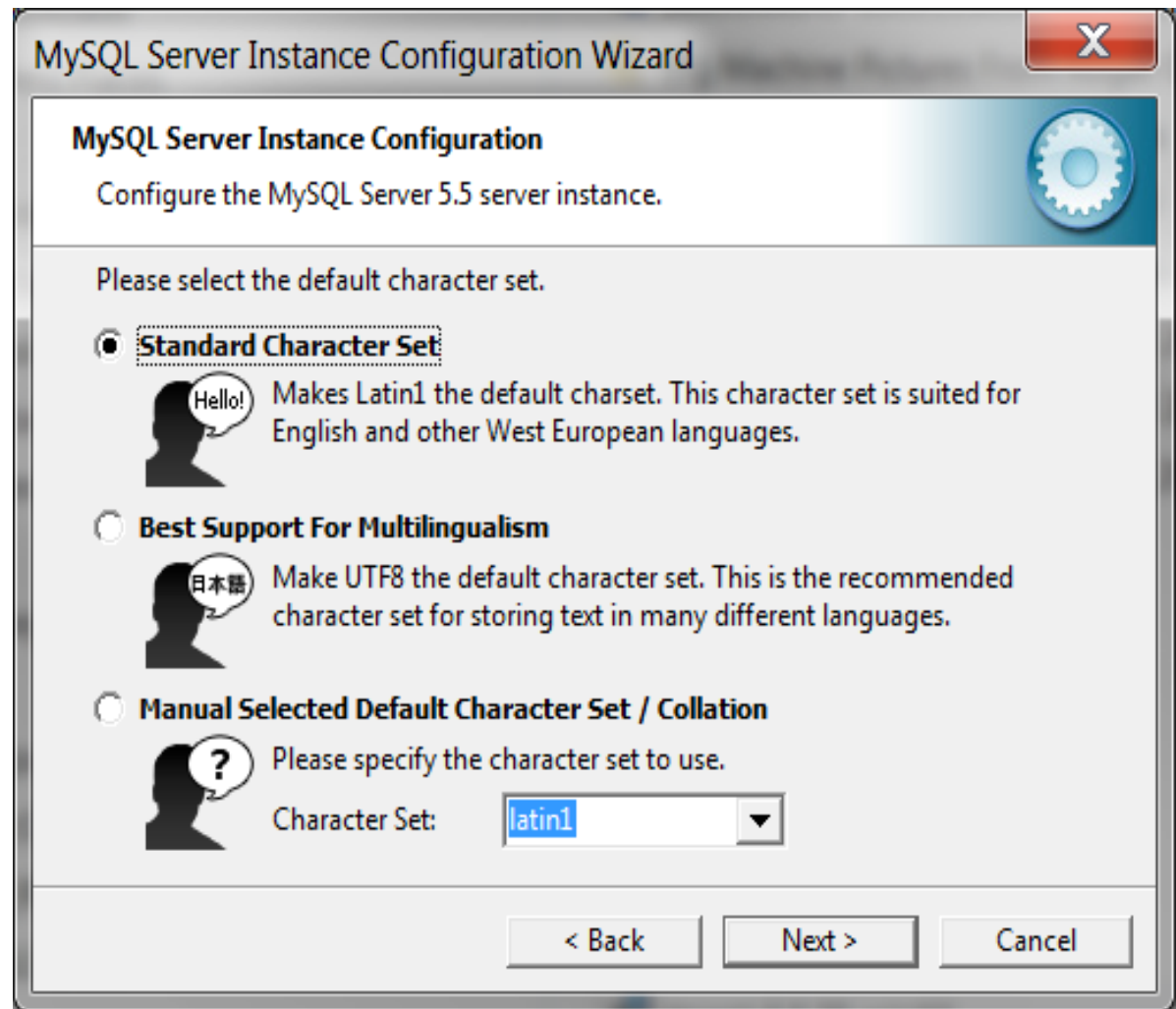
Installing MySQL 5.5.25 (cont.)



Accept all defaults in this window



Installing MySQL 5.5.25 (cont.)



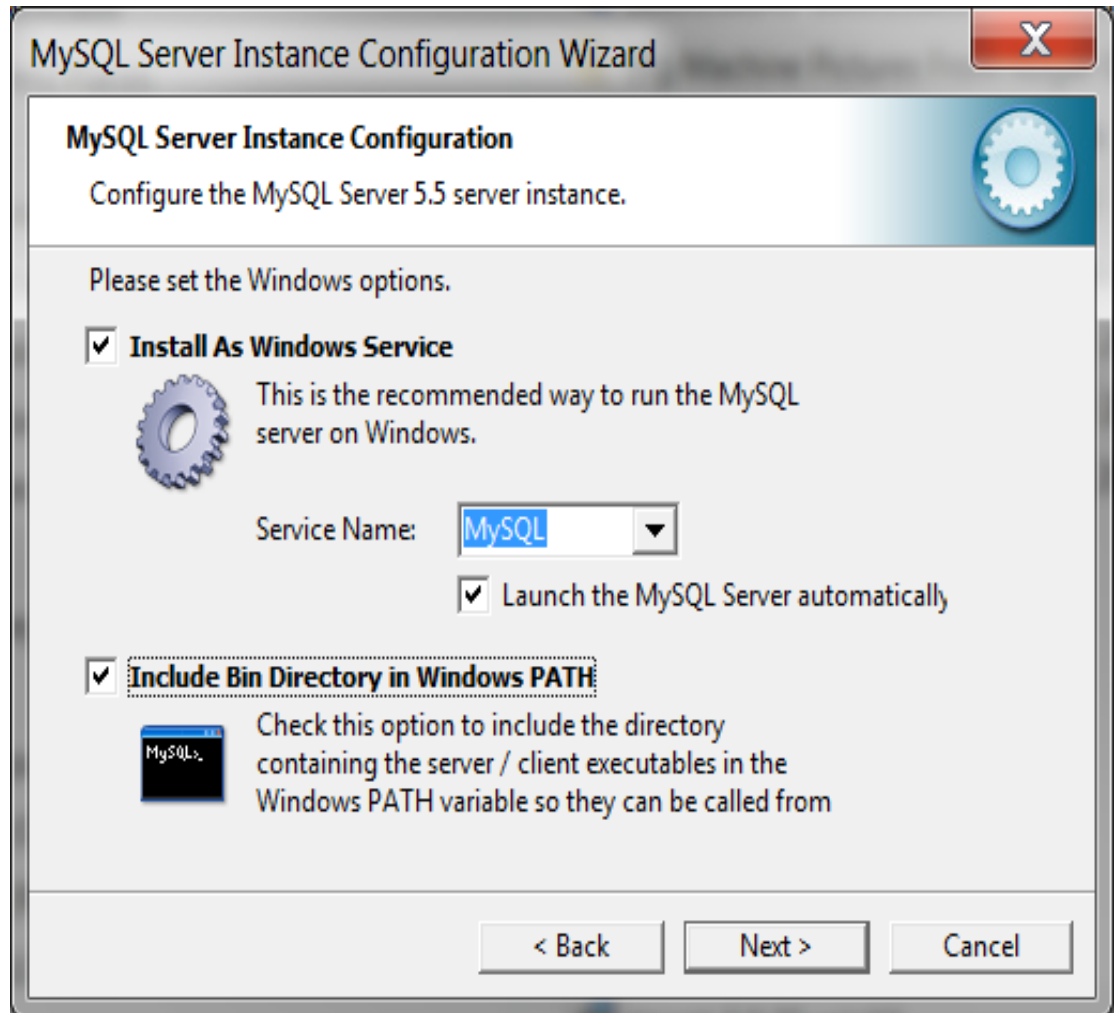
Your choice again



Installing MySQL 5.5.25 (cont.)

Accept default options

This option is not marked by default, but you can mark and accept it if you want to include MySQL file locations in your PATH statement.



The screenshot shows the 'MySQL Server Instance Configuration Wizard' window. The title bar reads 'MySQL Server Instance Configuration Wizard'. The main content area is titled 'MySQL Server Instance Configuration' and contains the text 'Configure the MySQL Server 5.5 server instance.' Below this, it says 'Please set the Windows options.' There are three main options, each with a checkbox and a gear icon:

- Install As Windows Service**
This is the recommended way to run the MySQL server on Windows.
Service Name:
- Launch the MySQL Server automatically**
- Include Bin Directory in Windows PATH**
Check this option to include the directory containing the server / client executables in the Windows PATH variable so they can be called from

At the bottom of the window, there are three buttons: '< Back', 'Next >', and 'Cancel'.



Installing MySQL 5.5.25 (cont.)

Accept default setting and enter a password for the root (superuser with all privileges by default). Enabling root access from remote machines is only necessary if you will be accessing the DB as the root user from a remote machine – we will not be doing this in this course.


Do not enable this option

MySQL Server Instance Configuration Wizard

MySQL Server Instance Configuration
Configure the MySQL Server 5.5 server instance.

Please set the security options.


Modify Security Settings

 New root password: Enter the root password.

Confirm: Retype the password.

Enable root access from remote machines

Create An Anonymous Account

 This option will create an anonymous account on this server. Please note that this can lead to an insecure system.

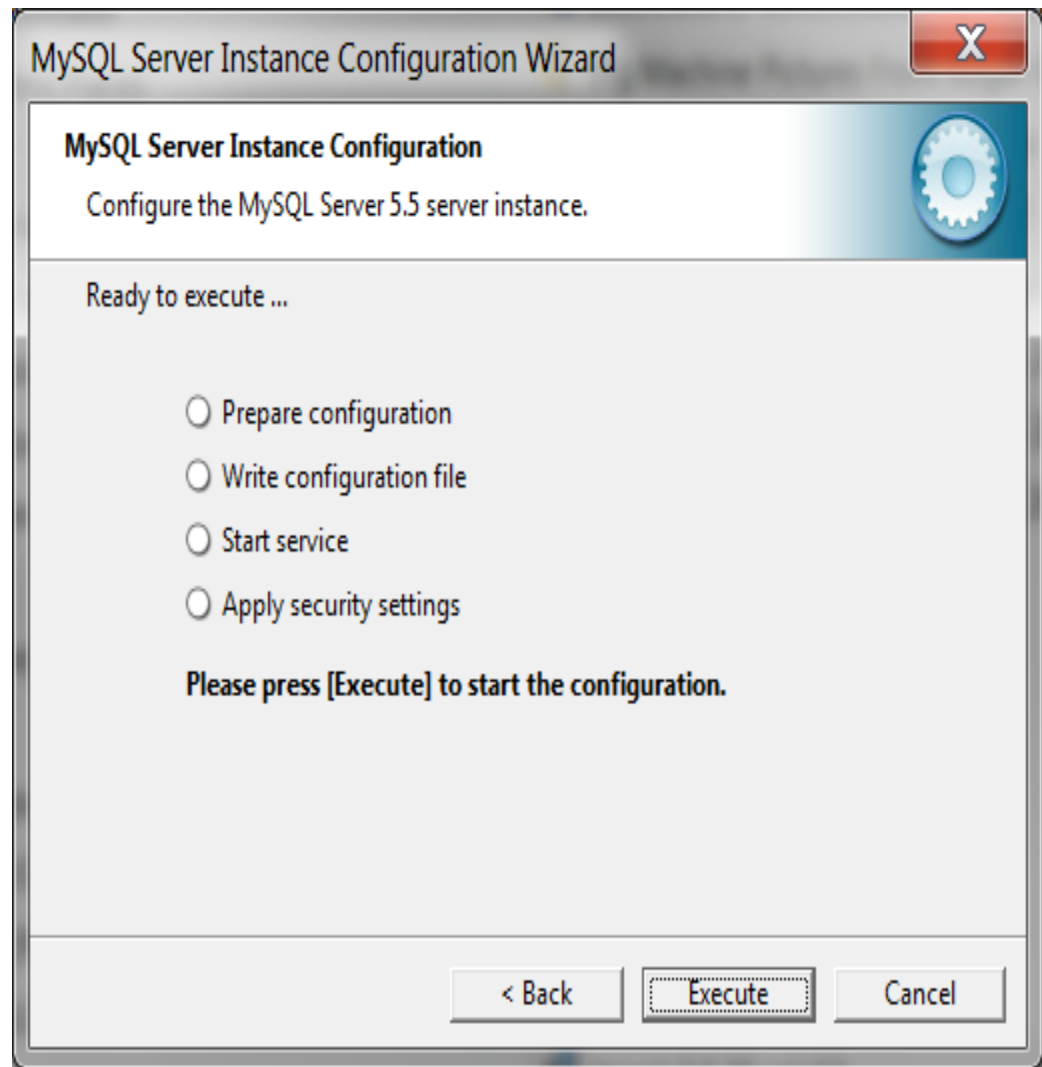
< Back Next > Cancel



Installing MySQL 5.5.25 (cont.)

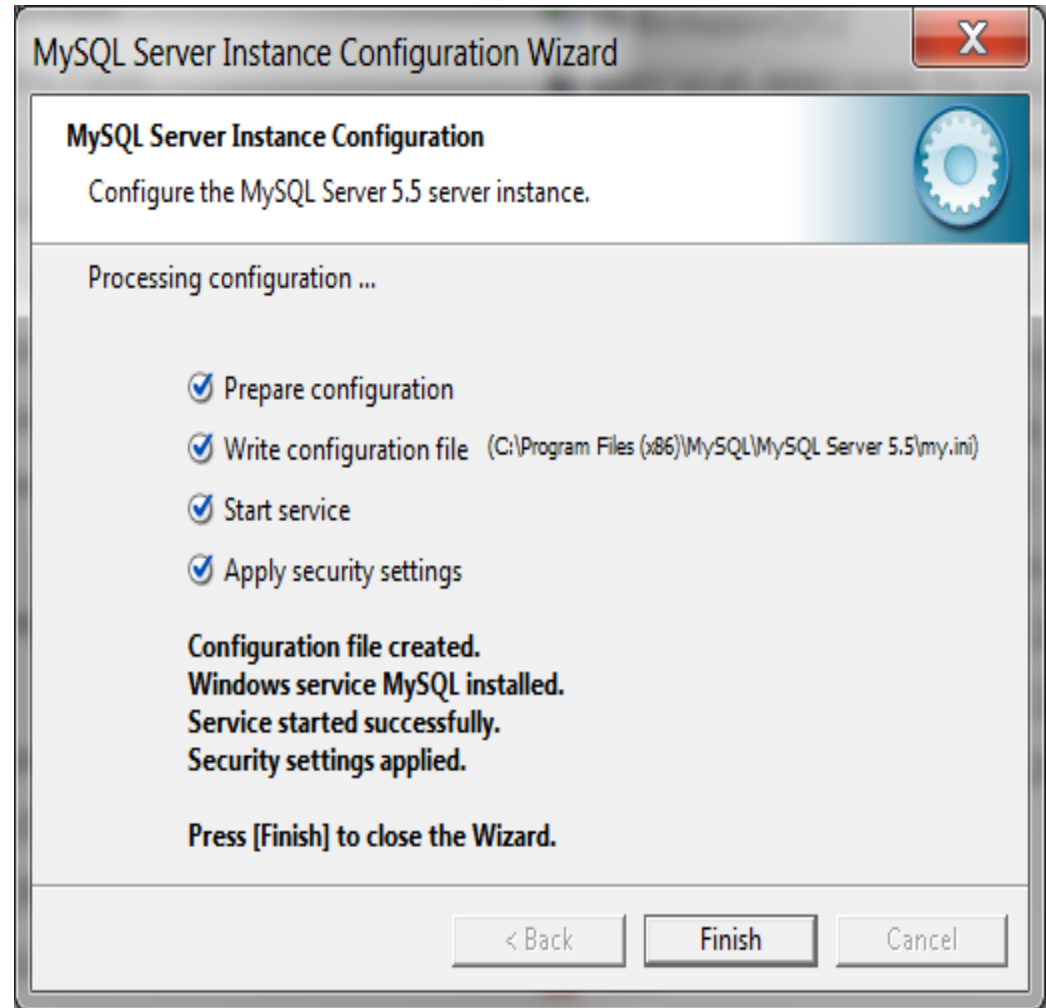
Configuration is about to begin. Now cross your fingers, toes, and anything else you have, take a deep breath, click the Execute button and close your eyes for a few seconds.

When they all have green check marks in them – you're good to go!



Installing MySQL 5.5.25 (cont.)

You've successfully installed MySQL!!



Running MySQL 5.5.25

- If you've successfully installed MySQL, it should now be running as a service on your machine. It will start automatically when your machine boots.
- Go into your listing of programs (from the start menu at the bottom: All Programs) and you should see MySQL appear. Since you will be running MySQL clients a lot, it will be easier if you pin the MySQL 5.5 Command Line Client to the start menu.
- To verify that MySQL is running properly as a service you can either check the process window or run a MySQL client.



Running MySQL 5.5.25 (cont.)

```
MySQL 5.5 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.5.25 MySQL Community Server (GPL)
Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> status
-----
C:\Program Files (x86)\MySQL\MySQL Server 5.5\bin\mysql.exe  Uer 14.14 Distrib 5
.5.25, for Win32 (x86)

Connection id:          2
Current database:
Current user:           root@localhost
SSL:                    Not in use
Using delimiter:        ;
Server version:         5.5.25 MySQL Community Server (GPL)
Protocol version:       10
Connection:             localhost via TCP/IP
Server characterset:    latin1
Db      characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
TCP port:               3309
Uptime:                 3 min 1 sec

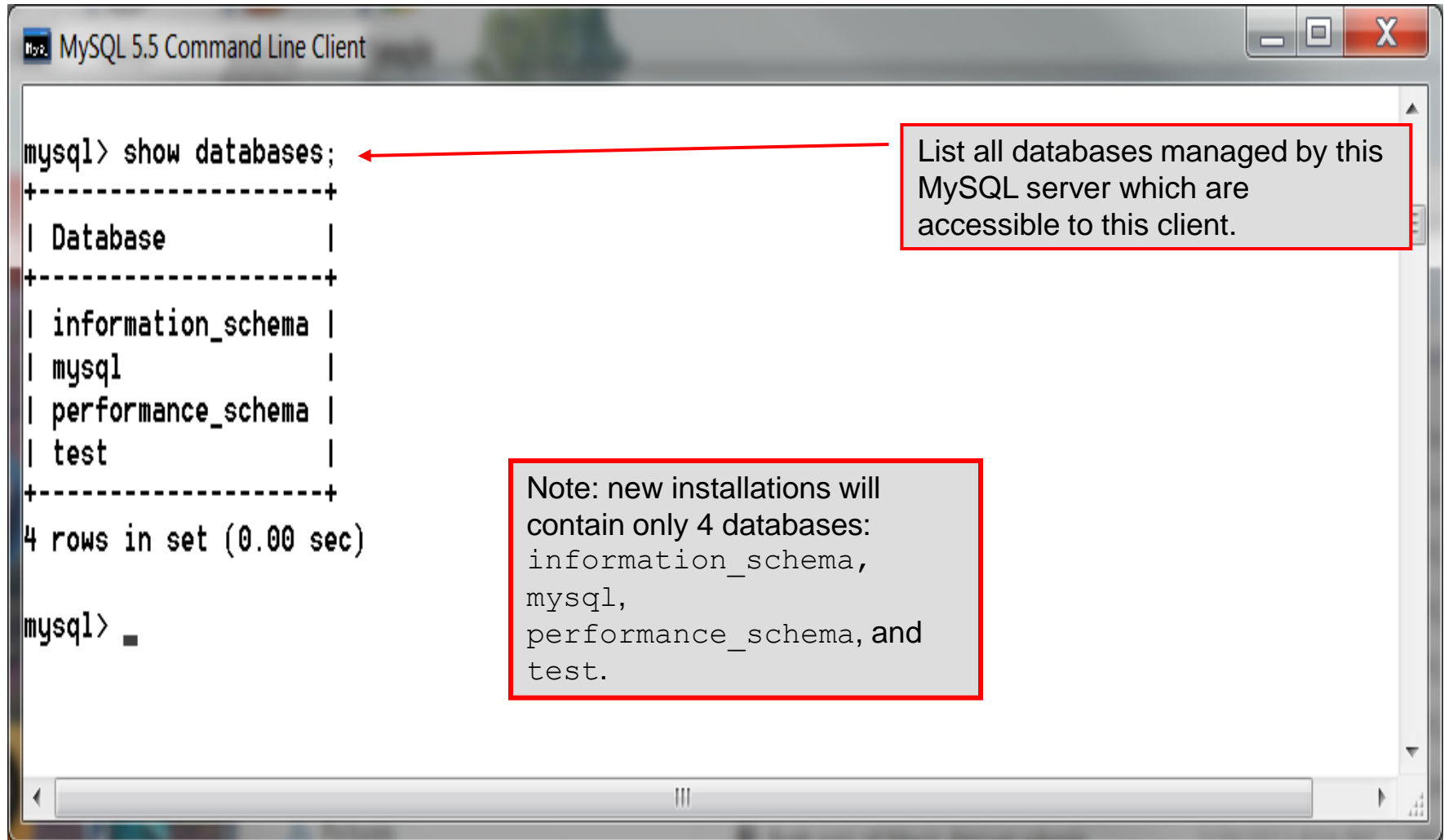
Threads: 1  Questions: 13  Slow queries: 0  Opens: 33  Flush tables: 1  Open tab
les: 26  Queries per second avg: 0.071
-----
mysql> _
```

Server version

Hopefully, you see this output from MySQL. The MySQL server is now awaiting a command from this client.



Running MySQL 5.5.25 (cont.)



```
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
| performance_schema     |
| test                    |
+-----+
4 rows in set (0.00 sec)

mysql> 
```

List all databases managed by this MySQL server which are accessible to this client.

Note: new installations will contain only 4 databases: information_schema, mysql, performance_schema, and test.



Running MySQL 5.5.25 (cont.)

```
MySQL 5.5 Command Line Client

Connection id:          2
Current database:
Current user:          root@localhost
SSL:                   Not in use
Using delimiter:      ;
Server version:        5.5.20 MySQL Community Server (GPL)
Protocol version:     10
Connection:            localhost via TCP/IP
Server characteraset:  latin1
Db characteraset:     latin1
Client characteraset: latin1
Conn. characteraset:  latin1
TCP port:              3309
Uptime:                1 min 31 sec

Threads: 1  Questions: 5  Slow queries: 0  Opens: 33  Flush tables: 1  Open tabl
es: 26  Queries per second avg: 0.054

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb        |
| colorsurvey  |
| guestbook    |
| mailinglist  |
| mysql        |
| performance_schema |
| prog3        |
| project4     |
| test        |
+-----+

10 rows in set (0.05 sec)

mysql> exit;
```

List all databases managed by this MySQL server which are accessible to this client.

Terminate client connection.



Specifying A Database Within MySQL

- Unless, it is specifically stated, in the following slides we'll assume that the user has root-level privileges.
- To select a database for use in MySQL the `use` command must be issued. In the example below, we'll select the `bikedb` database.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb      |
| colorsurvey  |
| guestbook   |
| mailinglist  |
| mysql       |
| performance_schema |
| prog3       |
| test        |
+-----+
9 rows in set (0.00 sec)

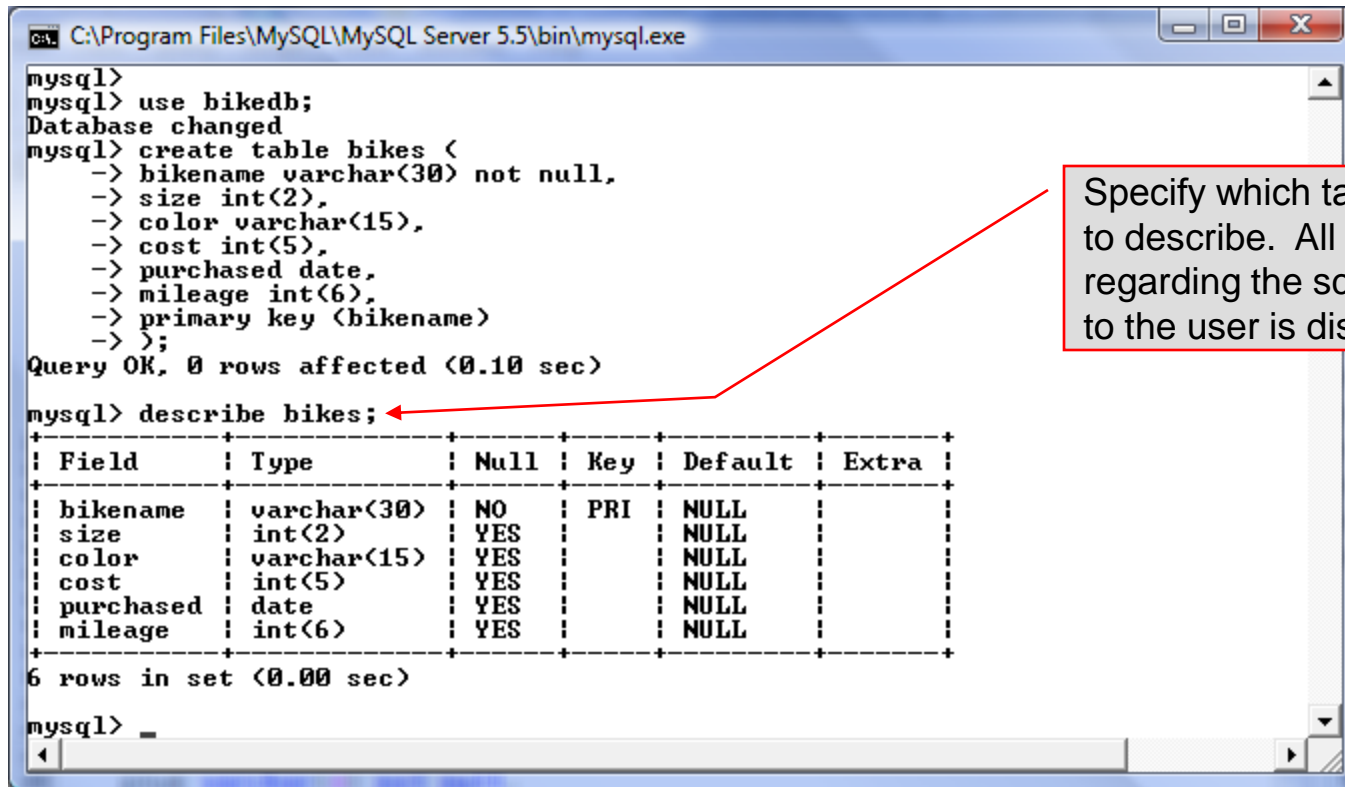
mysql> use bikedb;
Database changed
mysql> _
```

MySQL acknowledges selection of bikedb database.



Viewing the Schema of a Relation

- To see the schema of a relation within a database, use the `describe <tablename>` command as illustrated below.



```
C:\Program Files\MySQL\MySQL Server 5.5\bin>mysql.exe
mysql>
mysql> use bikedb;
Database changed
mysql> create table bikes (
  -> bikename varchar(30) not null,
  -> size int(2),
  -> color varchar(15),
  -> cost int(5),
  -> purchased date,
  -> mileage int(6),
  -> primary key (bikename)
  -> );
Query OK, 0 rows affected (0.10 sec)

mysql> describe bikes;
```

| Field | Type | Null | Key | Default | Extra |
|-----------|-------------|------|-----|---------|-------|
| bikename | varchar(30) | NO | PRI | NULL | |
| size | int(2) | YES | | NULL | |
| color | varchar(15) | YES | | NULL | |
| cost | int(5) | YES | | NULL | |
| purchased | date | YES | | NULL | |
| mileage | int(6) | YES | | NULL | |

```
6 rows in set (0.00 sec)

mysql> _
```

Specify which table's schema to describe. All information regarding the schema visible to the user is displayed.



Viewing the Relations of a Database

- Once a database has been selected you can see the relations (tables) within that database with the show tables command as illustrated below.

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe
-> mileage int(6),
-> primary key (bikename)
-> );
Query OK, 0 rows affected (0.10 sec)

mysql> describe bikes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| bikename   | varchar(30)   | NO   | PRI | NULL     |       |
| size       | int(2)        | YES  |     | NULL     |       |
| color      | varchar(15)   | YES  |     | NULL     |       |
| cost       | int(5)        | YES  |     | NULL     |       |
| purchased  | date          | YES  |     | NULL     |       |
| mileage    | int(6)        | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_bikedb |
+-----+
| bikes             |
| bluebikes         |
+-----+
2 rows in set (0.00 sec)

mysql> _
```

Show tables command lists all the relations within a database visible to the user. There are two tables in this database.



Running a Simple Select Query in MySQL

- Within the MySQL monitor, running an SQL query is straight forward. The example below illustrates a simple selection query on the `bikes` table of the `bikedb` database.

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe
mysql>
mysql>
mysql> select * from bikes;
```

| bikename | size | color | cost | purchased | mileage |
|------------------------|------|-----------------|------|------------|---------|
| Battaglin Carrera | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300 |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000 |
| Bianchi Infinito | 58 | celeste | 8900 | 2011-07-14 | 0 |
| BMC SLC01 - Swiss | 58 | red/black/white | 8000 | 2010-06-23 | 0 |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300 |
| Colnago Superissimo | 59 | red | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo | 58 | blue/black | 5300 | 2004-02-02 | 0 |
| Eddy Merckx Molteni | 58 | orange | 5100 | 2004-08-12 | 0 |
| Gianni Motta Personal | 59 | red/green | 4400 | 2000-05-01 | 8700 |
| Gios Torino Super | 60 | blue | 2000 | 1998-11-08 | 9000 |
| Ridley Damocles | 58 | blue/black | 7500 | 2008-06-27 | 0 |
| Ridley X-Fire | 58 | red/white | 7500 | 2011-09-01 | 0 |
| Schwinn Paramount P14 | 60 | blue | 1800 | 1992-03-01 | 200 |

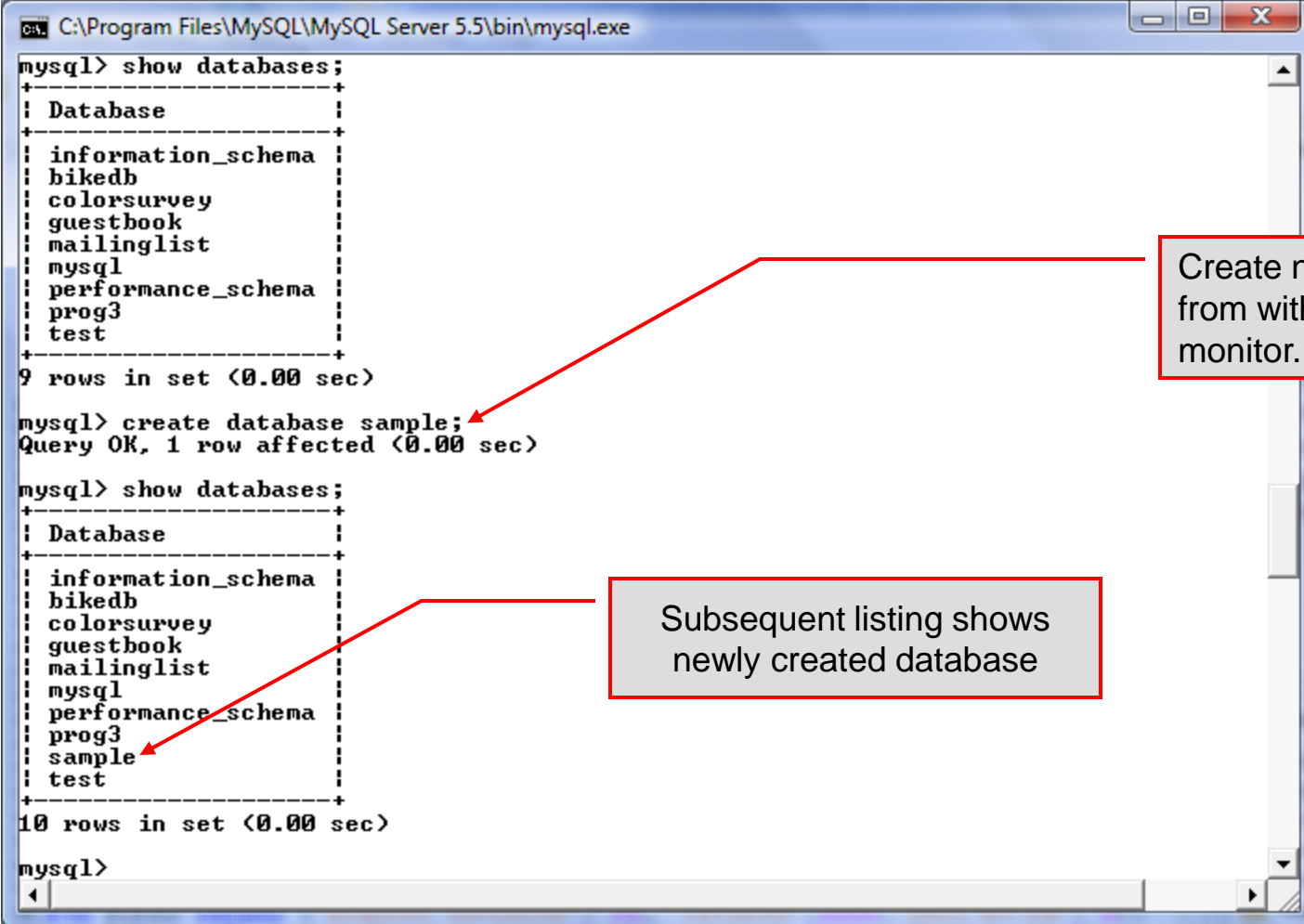
```
14 rows in set (0.00 sec)
mysql>
```

The tuples within the `bikes` table are displayed as the result of the query.



Creating a Database in MySQL

- From the MySQL monitor enter create database <db name>



```
C:\Program Files\MySQL\MySQL Server 5.5\bin>mysql.exe
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb       |
| colorsurvey  |
| guestbook    |
| mailinglist  |
| mysql        |
| performance_schema |
| prog3        |
| test         |
+-----+
9 rows in set (0.00 sec)

mysql> create database sample;
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb       |
| colorsurvey  |
| guestbook    |
| mailinglist  |
| mysql        |
| performance_schema |
| prog3        |
| sample       |
| test         |
+-----+
10 rows in set (0.00 sec)

mysql>
```

Create new database from within MySQL monitor.

Subsequent listing shows newly created database



Dropping a Database in MySQL

- From the MySQL monitor execute the `drop database <db name>` command.

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb          |
| coloursurvey    |
| guestbook       |
| mailinglist     |
| mysql           |
| performance_schema |
| prog3           |
| sample          |
| test            |
+-----+
10 rows in set (0.00 sec)

mysql> drop database sample;
Query OK, 0 rows affected (0.00 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb          |
| coloursurvey    |
| guestbook       |
| mailinglist     |
| mysql           |
| performance_schema |
| prog3           |
| test            |
+-----+
9 rows in set (0.00 sec)

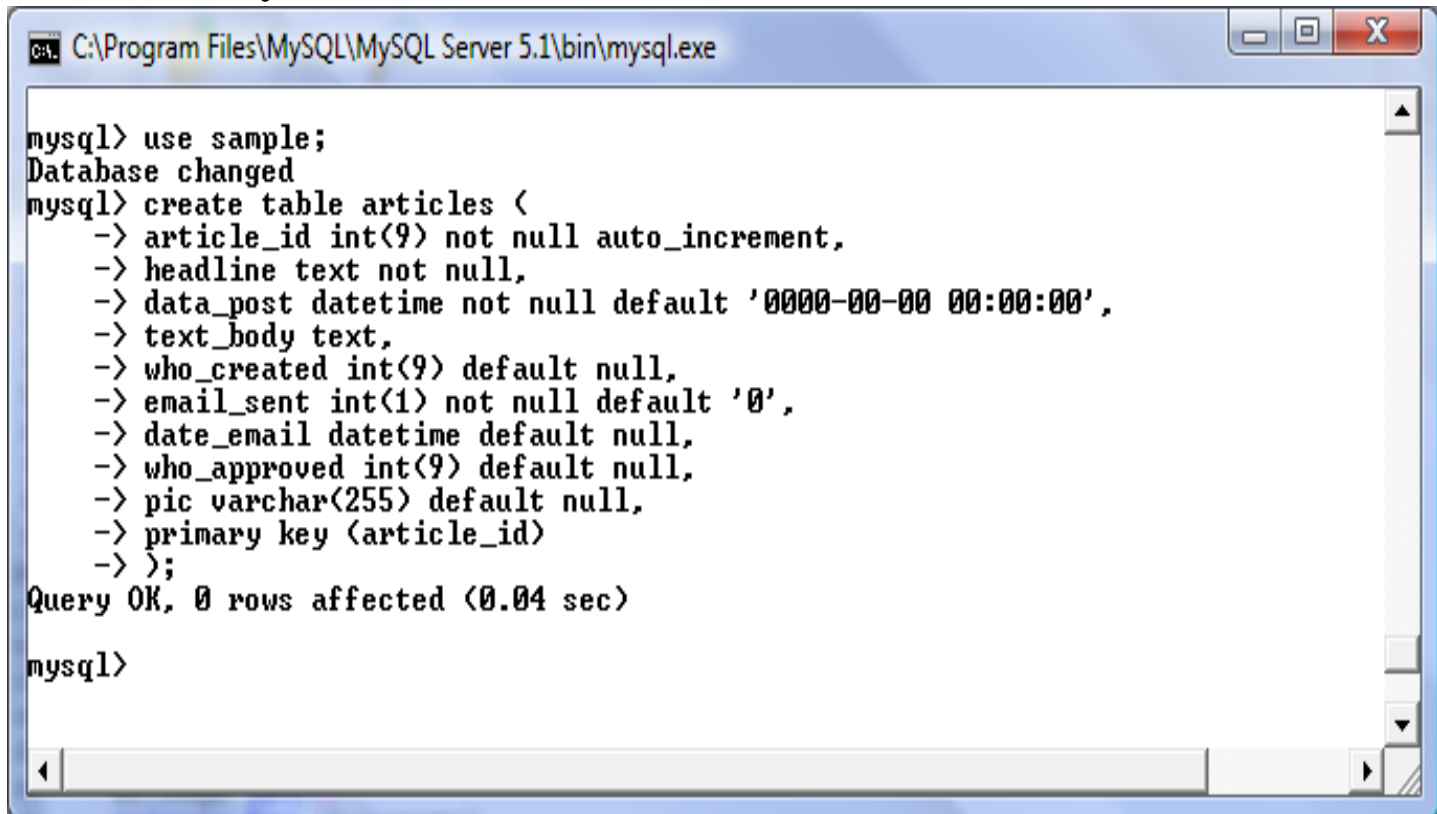
mysql> _
```

From within the MySQL monitor, no warning is given when dropping a database. Be very sure that this is what you want to do before you do it.



Manipulating Tables in MySQL

- The creation of a database does not place any relations into the database. Relations must be separately created.
- To create a table within a database, first select the database (or create one if you haven't already done so), then execute the create table command.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

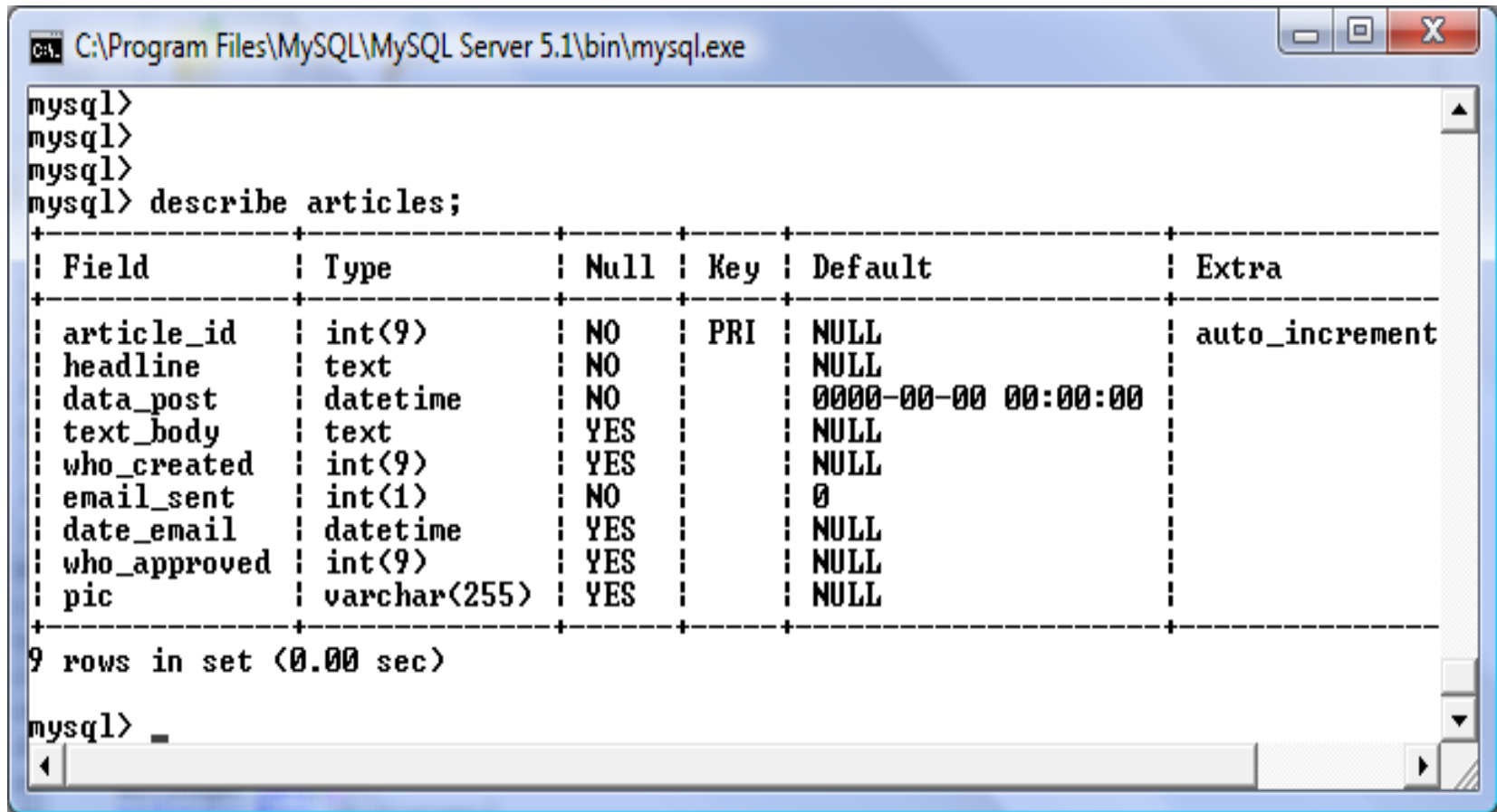
mysql> use sample;
Database changed
mysql> create table articles (
  -> article_id int(9) not null auto_increment,
  -> headline text not null,
  -> data_post datetime not null default '0000-00-00 00:00:00',
  -> text_body text,
  -> who_created int(9) default null,
  -> email_sent int(1) not null default '0',
  -> date_email datetime default null,
  -> who_approved int(9) default null,
  -> pic varchar(255) default null,
  -> primary key (article_id)
  -> );
Query OK, 0 rows affected (0.04 sec)

mysql>
```



Manipulating Tables in MySQL (cont.)

Screen shot that describes the newly created table.



```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql>
mysql>
mysql>
mysql> describe articles;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default                | Extra          |
+-----+-----+-----+-----+-----+-----+
| article_id | int(9)    | NO   | PRI | NULL                    | auto_increment |
| headline   | text      | NO   |     | NULL                    |                |
| data_post  | datetime  | NO   |     | 0000-00-00 00:00:00    |                |
| text_body  | text      | YES  |     | NULL                    |                |
| who_created | int(9)    | YES  |     | NULL                    |                |
| email_sent | int(1)    | NO   |     | 0                        |                |
| date_email | datetime  | YES  |     | NULL                    |                |
| who_approved | int(9)    | YES  |     | NULL                    |                |
| pic        | varchar(255) | YES  |     | NULL                    |                |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql> _
```



Manipulating Tables in MySQL (cont.)

- The `create table` command has the following general format:

```
create [temporary] table
[if not exists] tablename
[(create_definition, ...)]
[table_options] [select_statement];
```

- If the `[if not exists]` clause is present, MySQL will produce an error message if a table with the specified name already exists in the database, otherwise the table is created.



Manipulating Tables in MySQL (cont.)

- A temporary table exists only for the life of the current database connection. It is automatically destroyed when the connection is closed or dies.
- Two different connections can use the same name for a temporary table without conflicting with one another.
- Temporary tables are most useful when queries get complex and intermediate results become useful. Also, versions of MySQL earlier than version 4.1 do not have subselect capability and temporary tables are a convenient way to simulate subselect query results.

Note: Non-root users require special permission to be able to create temporary tables. These users must have the `Create_tmp_tables` privilege set in the user grant table. We'll see more on this later.



Creating A Temporary Table From A Select Query

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe
+-----+-----+-----+-----+-----+-----+
| Battaglin Carrera      | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4   | 58 | celeste   | 5700 | 2004-12-02 | 300   |
| Bianchi Evolution 3   | 58 | celeste   | 4800 | 2003-11-12 | 2000  |
| Bianchi Infinito      | 58 | celeste   | 8900 | 2011-07-14 |       |
| BMC SLC01 - Swiss     | 58 | red/black/white | 8000 | 2010-06-23 |       |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300  |
| Colnago Superissimo   | 59 | red       | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo      | 58 | blue/black | 5300 | 2004-02-02 |       |
| Eddy Merckx Molteni   | 58 | orange    | 5100 | 2004-08-12 |       |
| Gianni Motta Personal | 59 | red/green  | 4400 | 2000-05-01 | 8700  |
| Gios Torino Super     | 60 | blue      | 2000 | 1998-11-08 | 9000  |
| Ridley Damocles       | 58 | blue/black | 7500 | 2008-06-27 |       |
| Ridley X-Fire         | 58 | red/white  | 7500 | 2011-09-01 |       |
| Schwinn Paramount P14 | 60 | blue      | 1800 | 1992-03-01 | 2000  |
+-----+-----+-----+-----+-----+
14 rows in set (0.00 sec)

mysql> create temporary table celestebikes
-> select *
-> from bikes
-> where color = "celeste";
Query OK, 3 rows affected (0.11 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> show tables;
+-----+
| Tables_in_bikedb |
+-----+
| bikes             |
| bluebikes         |
+-----+
2 rows in set (0.00 sec)

mysql> select * from celestebikes;
+-----+-----+-----+-----+-----+-----+
| bikename      | size | color   | cost | purchased | mileage |
+-----+-----+-----+-----+-----+-----+
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300   |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000  |
| Bianchi Infinito   | 58 | celeste | 8900 | 2011-07-14 | 0     |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

A SELECT query produces a result set which has been extracted from one or more tables. A table can be created with the results of this data using the create table command.

Notice that temporary tables do not appear in a table listing.



Manipulating Tables in MySQL (cont.)

- Recall that the `create table` command has the following general format:

```
create [temporary] table
[if not exists] tablename
[(create_definition, ...)]
[table_options]
[select_statement];
```

- The table options allow you to specify the MySQL table type. The table type can be anyone of the six types listed in the table on the next slide.



Manipulating Tables in MySQL (cont.)

| Table Type | Description |
|------------|--|
| ISAM | MySQL's original table handler |
| HEAP | The data for this table is only stored in memory |
| MyISAM | A binary portable table handler that has replaced ISAM |
| MERGE | A collection of MyISAM tables used as one table |
| BDB | Transaction-safe tables with page locking |
| InnoDB | Transaction-safe tables with row locking |

MySQL Table Types

ISAM, HEAP, and MyISAM are available for MySQL versions 3.23.6 or later.

MERGE, BDB, and InnoDB are available for MySQL versions 4.0 and later.

Default table type is InnoDB for MySQL versions 5.5.20.x.



Altering A Table

- After a table has been created, it is possible to change the specifications of its schema. This is done through the `alter table` command:

```
alter table table_name action_list
```

- Note: Changing the schema of a table in a database is not something that is done very often once the database has been created. The time for altering the schema is during the design phase. Altering the schema of an operational database is a very dangerous thing.
- Multiple changes to the table can be made at the same time by separating actions with commas in the `action_list`.
- The possible attribute (column) actions that can be used are shown in the table on the following slide.



Altering A Table (cont.)

| Action Syntax | Action Performed |
|--|--|
| <code>add [column] <i>column_declaration</i> [first after <i>column_name</i>]</code> | Add a column to the table |
| <code>alter [column] <i>column_name</i> {set default <i>literal</i> drop default}</code> | Specify new default value for a column or remove old default |
| <code>change [column] <i>column_name</i> <i>column_declaration</i></code> | Modify column declaration with renaming of column |
| <code>modify [column] <i>column_declaration</i></code> | Modify column declaration without renaming column |
| <code>drop [column] <i>column_name</i></code> | Drop a column and all data contained within it. |
| <code>rename [as] <i>new_table_name</i></code> | Rename a table |
| <code><i>table_options</i></code> | Change the table options |

Actions performed by `alter table` (column related) command

column_name represents the current name of the column, *column_declaration* represents the new declaration, in the same format as if it were in a `create` command.



Altering A Table (cont.)

- The screen shot below shows an example of altering a table.

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> describe bikes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| bikename   | varchar(30)   | NO   | PRI | NULL    |      |
| size       | int(2)        | YES  |     | NULL    |      |
| color      | varchar(15)   | YES  |     | NULL    |      |
| cost       | int(6)        | YES  |     | NULL    |      |
| purchased  | date          | YES  |     | NULL    |      |
| mileage    | int(6)        | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> alter table bikes
-> add column races_won int(3) default 0;
Query OK, 10 rows affected (0.05 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> describe bikes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| bikename   | varchar(30)   | NO   | PRI | NULL    |      |
| size       | int(2)        | YES  |     | NULL    |      |
| color      | varchar(15)   | YES  |     | NULL    |      |
| cost       | int(6)        | YES  |     | NULL    |      |
| purchased  | date          | YES  |     | NULL    |      |
| mileage    | int(6)        | YES  |     | NULL    |      |
| races_won  | int(3)        | YES  |     | 0       |      |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

mysql>
```

Schema of bikes before alteration

There are 10 rows affected because this table currently contains 10 tuples (rows) and the new attribute has been added to all rows.

Bikes table after the addition of a new column named races_won



Altering A Table (cont.)

- The screen shot below shows the tuples currently in the bikes table after the addition of the new attribute illustrating that all of the tuples have assumed the default value on the new attribute.

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> select * from bikes;
+-----+-----+-----+-----+-----+-----+-----+
| bikename          | size | color      | cost | purchased | mileage | races_won |
+-----+-----+-----+-----+-----+-----+-----+
| Colnago Dream Rabobank | 60   | blue/orange | 5500 | 2002-07-07 | 4300    | 0          |
| Bianchi Evolution 3   | 58   | celeste    | 4800 | 2003-11-12 | 2000    | 0          |
| Eddy Merckx Molteni   | 58   | orange     | 5100 | 2004-08-12 | 0       | 0          |
| Eddy Merckx Domo      | 58   | blue/black  | 5300 | 2004-02-02 | 0       | 0          |
| Battaglin Carrera    | 60   | red/white   | 4000 | 2001-03-10 | 11200   | 0          |
| Gianni Motta Personal | 59   | red/green   | 4400 | 2000-05-01 | 8700    | 0          |
| Gios Torino Super     | 60   | blue       | 2000 | 1998-11-08 | 9000    | 0          |
| Schwinn Paramount P14 | 60   | blue       | 1800 | 1992-03-01 | 200     | 0          |
| Bianchi Corse Evo 4   | 58   | celeste    | 5700 | 2004-12-02 | 300     | 0          |
| Colnago Superissimo   | 59   | red        | 3800 | 1996-03-01 | 13000   | 0          |
+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql>
```

Every tuple in the table has the default value for the new attribute.



Altering A Table (cont.)

- The screen shot below illustrates dropping a column from a table.
- Note that in general, this type of operation may not always be allowed due to constraint violations.

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> alter table bikes
  -> drop column races_won;
Query OK, 10 rows affected (0.03 sec)
Records: 10 Duplicates: 0 Warnings: 0

mysql> describe bikes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| bikename   | varchar(30)   | NO   | PRI | NULL    |       |
| size       | int(2)        | YES  |     | NULL    |       |
| color      | varchar(15)   | YES  |     | NULL    |       |
| cost       | int(6)        | YES  |     | NULL    |       |
| purchased  | date          | YES  |     | NULL    |       |
| mileage    | int(6)        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

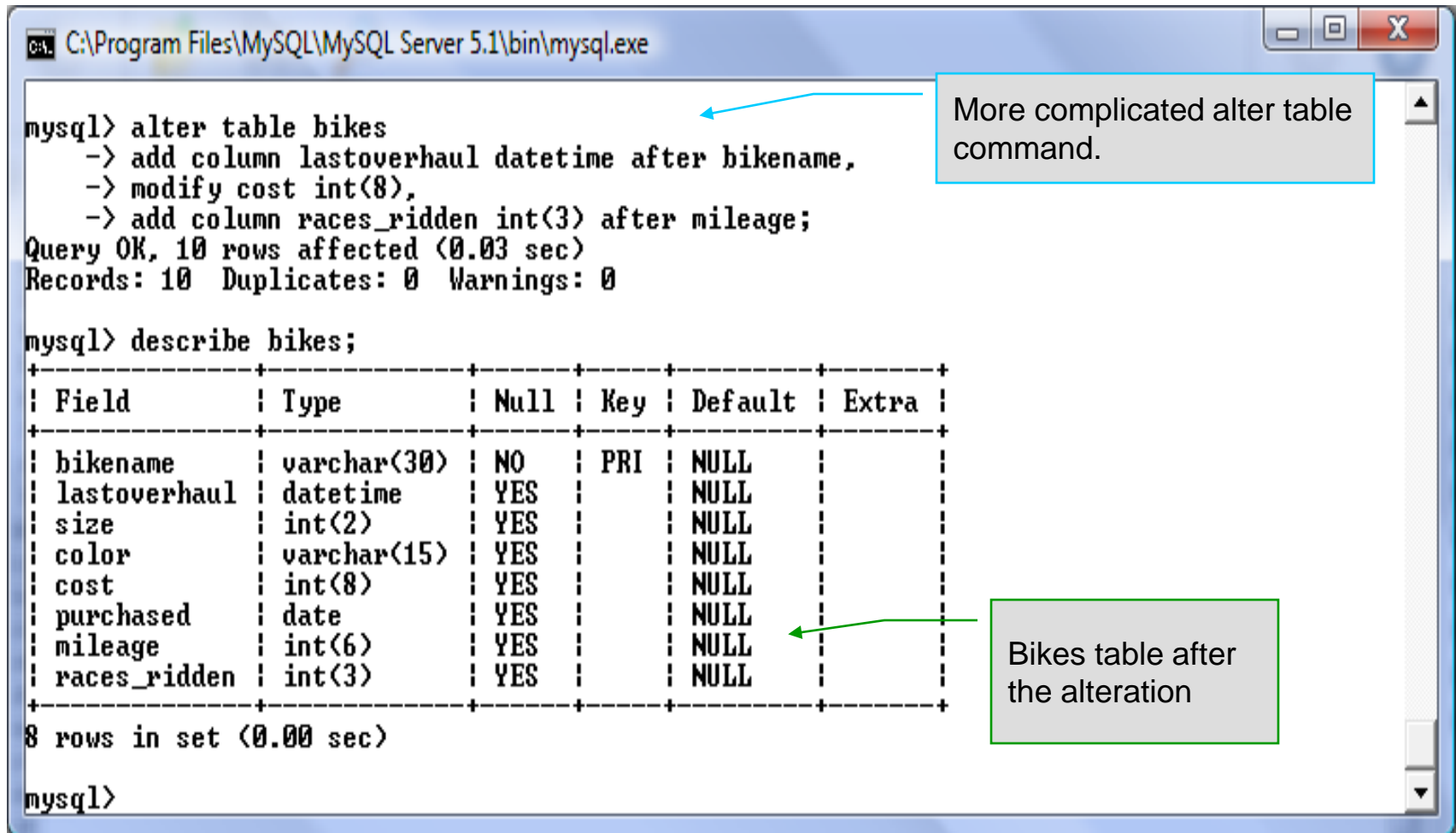
mysql>
```

The attribute races_won has been eliminated from the table.



Altering A Table (cont.)

- The screen shot below shows a more complicated example of altering a table.



The screenshot shows a MySQL command prompt window with the following text:

```
mysql> alter table bikes
  -> add column lastoverhaul datetime after bikename,
  -> modify cost int(8),
  -> add column races_ridden int(3) after mileage;
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> describe bikes;
```

| Field | Type | Null | Key | Default | Extra |
|--------------|-------------|------|-----|---------|-------|
| bikename | varchar(30) | NO | PRI | NULL | |
| lastoverhaul | datetime | YES | | NULL | |
| size | int(2) | YES | | NULL | |
| color | varchar(15) | YES | | NULL | |
| cost | int(8) | YES | | NULL | |
| purchased | date | YES | | NULL | |
| mileage | int(6) | YES | | NULL | |
| races_ridden | int(3) | YES | | NULL | |

8 rows in set (0.00 sec)

```
mysql>
```

Two callout boxes are present: a blue one pointing to the ALTER TABLE command and a green one pointing to the DESCRIBE output.



Inserting Data Into A Table

- Data can be entered into a MySQL table using either the `insert` or `replace` commands.
- The `insert` statement is the primary way of getting data into the database and has the following form:

Form 1 `insert [low priority | delayed] [ignore] [into]table_name`
`[set] column_name1 = expression1,`
`column_name2 = expression2, ...`

Form 2 `insert [low priority | delayed] [ignore] [into]table_name`
`[(column_name,...)]values (expression,...), (...)`

Form 3 `insert [low priority | delayed] [ignore] [into]table_name`
`[(column_name,...)] select...`



Inserting Data Into A Table (cont.)

- Form 1 of the insert statement is the most verbose, but also the most common. The `set` clause explicitly names each column and states what value (evaluated from each `expression`) should be put into the table.
- Form 2 (insert values) requires just a comma separated list of the data. For each row inserted, each data value must correspond with a column. In other words, the number of values listed must match the number of columns and the order of the value list must be the same as the columns. (In form 1, the order is not critical since each column is named.)
- Form 3 is used to insert data into a table which is the result set of a `select` statement. This is similar to the temporary table example seen earlier in the notes.
- The following couple of pages give some examples of the different forms of the `insert` command.



| | | | | | |
|------------------------|----|-----------------|------|------------|-------|
| Battaglin Carrera | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300 |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000 |
| Bianchi Infinito | 58 | celeste | 8900 | 2011-07-14 | 0 |
| BMC SLC01 - Swiss | 58 | red/black/white | 8000 | 2010-06-23 | 0 |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300 |
| Colnago Superissimo | 59 | red | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo | 58 | blue/black | 5300 | 2004-02-02 | 0 |
| Eddy Merckx Molteni | 58 | orange | 5100 | 2004-08-12 | 0 |
| Gianni Motta Personal | 59 | red/green | 4400 | 2000-05-01 | 8700 |
| Gios Torino Super | 60 | blue | 2000 | 1998-11-08 | 9000 |
| Ridley Damocles | 58 | blue/black | 7500 | 2008-06-27 | 0 |
| Ridley X-Fire | 58 | red/white | 7500 | 2011-09-01 | 0 |
| Schwinn Paramount P14 | 60 | blue | 1800 | 1992-03-01 | 200 |

14 rows in set (0.00 sec)

```
mysql> insert into bikes
-> set bikename = "Eddy Merckx EM7",
-> cost=9500,
-> mileage=100,
-> purchased="2011-01-01",
-> color="red/white/blue",
-> size=58;
```

Query OK, 1 row affected (0.03 sec)

```
mysql> select * from bikes;
```

| bikename | size | color | cost | purchased | mileage |
|------------------------|------|-----------------|------|------------|---------|
| Battaglin Carrera | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300 |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000 |
| Bianchi Infinito | 58 | celeste | 8900 | 2011-07-14 | 0 |
| BMC SLC01 - Swiss | 58 | red/black/white | 8000 | 2010-06-23 | 0 |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300 |
| Colnago Superissimo | 59 | red | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo | 58 | blue/black | 5300 | 2004-02-02 | 0 |
| Eddy Merckx EM7 | 58 | red/white/blue | 9500 | 2011-01-01 | 100 |
| Eddy Merckx Molteni | 58 | orange | 5100 | 2004-08-12 | 0 |
| Gianni Motta Personal | 59 | red/green | 4400 | 2000-05-01 | 8700 |
| Gios Torino Super | 60 | blue | 2000 | 1998-11-08 | 9000 |
| Ridley Damocles | 58 | blue/black | 7500 | 2008-06-27 | 0 |
| Ridley X-Fire | 58 | red/white | 7500 | 2011-09-01 | 0 |
| Schwinn Paramount P14 | 60 | blue | 1800 | 1992-03-01 | 200 |

15 rows in set (0.00 sec)

```
mysql> _
```

Examples: Inserting Data Into A Table

Using Form 1 for insertion – attribute order is not important.



mysql> select * from bikes;

| bikename | size | color | cost | purchased | mileage |
|------------------------|------|-----------------|------|------------|---------|
| Battaglin Carrera | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300 |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000 |
| Bianchi Infinito | 58 | celeste | 8900 | 2011-07-14 | 0 |
| BMC SLC01 - Swiss | 58 | red/black/white | 8000 | 2010-06-23 | 0 |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300 |
| Colnago Superissimo | 59 | red | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo | 58 | blue/black | 5300 | 2004-02-02 | 0 |
| Eddy Merckx EM7 | 58 | red/white/blue | 9500 | 2011-01-01 | 100 |
| Eddy Merckx Molteni | 58 | orange | 5100 | 2004-08-12 | 0 |
| Gianni Motta Personal | 59 | red/green | 4400 | 2000-05-01 | 8700 |
| Gios Torino Super | 60 | blue | 2000 | 1998-11-08 | 9000 |
| Ridley Damocles | 58 | blue/black | 7500 | 2008-06-27 | 0 |
| Ridley X-Fire | 58 | red/white | 7500 | 2011-09-01 | 0 |
| Schwinn Paramount P14 | 60 | blue | 1800 | 1992-03-01 | 200 |

15 rows in set (0.00 sec)

mysql> insert into bikes
-> values("Ridley Crosswind",58,"black",6500,"2010-04-05",2000);

Query OK, 1 row affected (0.05 sec)

mysql> select * from bikes;

| bikename | size | color | cost | purchased | mileage |
|------------------------|------|-----------------|------|------------|---------|
| Battaglin Carrera | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300 |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000 |
| Bianchi Infinito | 58 | celeste | 8900 | 2011-07-14 | 0 |
| BMC SLC01 - Swiss | 58 | red/black/white | 8000 | 2010-06-23 | 0 |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300 |
| Colnago Superissimo | 59 | red | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo | 58 | blue/black | 5300 | 2004-02-02 | 0 |
| Eddy Merckx EM7 | 58 | red/white/blue | 9500 | 2011-01-01 | 100 |
| Eddy Merckx Molteni | 58 | orange | 5100 | 2004-08-12 | 0 |
| Gianni Motta Personal | 59 | red/green | 4400 | 2000-05-01 | 8700 |
| Gios Torino Super | 60 | blue | 2000 | 1998-11-08 | 9000 |
| Ridley Crosswind | 58 | black | 6500 | 2010-04-05 | 2000 |
| Ridley Damocles | 58 | blue/black | 7500 | 2008-06-27 | 0 |
| Ridley X-Fire | 58 | red/white | 7500 | 2011-09-01 | 0 |
| Schwinn Paramount P14 | 60 | blue | 1800 | 1992-03-01 | 200 |

16 rows in set (0.00 sec)

mysql> _

Using Form 2 for insertion – attribute order is important.



Examples: Inserting Data Into A Table

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe
mysql>
mysql>
mysql> create table celestebikes like bikes;
Query OK, 0 rows affected (0.11 sec)

mysql> select * from celestebikes;
Empty set (0.00 sec)

mysql> insert into celestebikes
-> select *
-> from bikes
-> where color = "celeste";
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select * from celestebikes;
+-----+-----+-----+-----+-----+-----+
| bikename          | size | color  | cost  | purchased | mileage |
+-----+-----+-----+-----+-----+-----+
| Bianchi Corse Evo 4 | 58   | celeste | 5700  | 2004-12-02 | 300    |
| Bianchi Evolution 3 | 58   | celeste | 4800  | 2003-11-12 | 2000   |
| Bianchi Infinito   | 58   | celeste | 8900  | 2011-07-14 | 0      |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Creates an initially empty table just like the bikes table

Table creation did not place any data into the table

Using Form 3 for insertion

This table contains the name and cost of those bikes whose color was celeste from the source table.



Examples: Inserting Data Into A Table

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe
3 rows in set (0.00 sec)
mysql> drop table celestebikes;
Query OK, 0 rows affected (0.05 sec)

mysql> create table celestebikes (
  -> name varchar(30),
  -> paint varchar(15),
  -> price int(6),
  -> miles_ridden int(6),
  -> primary key (name)
  -> );
Query OK, 0 rows affected (0.10 sec)

mysql> insert into celestebikes
  -> select bikename, color, cost, mileage
  -> from bikes
  -> where color = "celeste";
Query OK, 3 rows affected (0.05 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> select * from celestebikes;
+-----+-----+-----+-----+
| name          | paint  | price | miles_ridden |
+-----+-----+-----+-----+
| Bianchi Corse Evo 4 | celeste | 5700  | 300          |
| Bianchi Evolution 3 | celeste | 4800  | 2000         |
| Bianchi Infinito   | celeste | 8900  | 0            |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

Create an initially empty table with a schema different from the base table.

Using Form 3 for insertion

This table contains the those bike tuples whose color was celeste from the source table.



Using Scripts with MySQL

- Entering data to create sample databases using conventional SQL commands is tedious and prone to errors. A much simpler technique is to use scripts. The following illustrates two techniques for invoking scripts in MySQL. The third and more preferable option is to use the MySQL Workbench tool (see page 98 and on.)
- Create your script file using the text editor of your choice.
- Comments in the SQL script files begin with a # symbol.
- In the script file example shown on the next slide, I drop the database in the first SQL command. Without the if exists clause, this will generate an error if the database does not exist. The first time the script executes (or subsequent executions if the database is dropped independently) the error will be generated...simply ignore the error.



Using Scripts with MySQL (cont.)

```
*C:\state script.sql - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
template.html commentform.html fourthCSS.css state script.sql
1 #SQL commands in a script file
2 drop database if exists testdb;
3
4 create database testdb;
5
6 use testdb;
7
8 create table states (
9     name varchar(15) not null,
10    abbrev char(2),
11    capital varchar(25),
12    population integer,
13    square_miles integer,
14    primary key (name)
15 );
16
17 insert into states values ('Florida', 'FL', 'Tallahassee', 18328240, 54153);
18 insert into states values ('New York', 'NY', 'Albany', 194909297, 54556);
19 insert into states values ('Indiana', 'IN', 'Indianapolis', 6376792, 35789);
20 insert into states values ('Maryland', 'MD', 'Annapolis', 5633597, 9975);
21
22 select * from states;
```

Drop the database if it already exists.

Create a new database.

Switch to the new database.

Define schema for the new table.

Insert some tuples

Run a simple selection query on the new table.

Structured Query Language file nb char: 616 nb line: 22



Using Scripts with MySQL (cont.)

```
C:\Program Files\MySQL\MySQL Server 5.5\bin\mysql.exe

mysql> source c:\state script.sql
Query OK, 0 rows affected, 1 warning (0.00 sec)

Query OK, 1 row affected (0.00 sec)

Database changed
Query OK, 0 rows affected (0.10 sec)

Query OK, 1 row affected (0.05 sec)
Query OK, 1 row affected (0.05 sec)
Query OK, 1 row affected (0.06 sec)
Query OK, 1 row affected (0.06 sec)

+-----+-----+-----+-----+-----+
| name   | abbrev | capital   | population | square_miles |
+-----+-----+-----+-----+-----+
| Florida | FL     | Tallahassee | 18328240  | 54153         |
| Indiana | IN     | Indianapolis | 6376792   | 35789         |
| Maryland | MD    | Annapolis  | 5633597   | 9975          |
| New York | NY    | Albany     | 194909297 | 54556         |
+-----+-----+-----+-----+-----+

4 rows in set (0.00 sec)

mysql> _
```

Specify which script to execute

Results of select query at end of script.



Importing Data Using the `mysqlimport` Utility

- As with many things in MySQL there are several ways to accomplish a specific task. For getting data into tables, the `mysqlimport` utility is also useful.
- The `mysqlimport` utility reads a range of data formats, including comma- and tab- delimited, and inserts the data into a specified database table. The syntax for `mysqlimport` is:

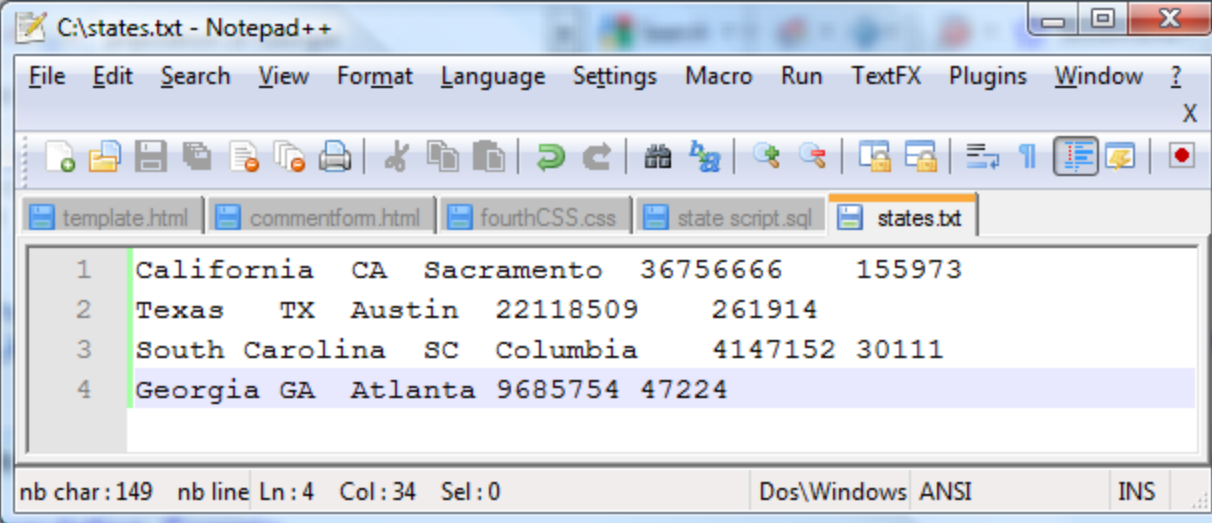
```
mysqlimport [options] database_name file1 file2 ...
```

- This utility is designed to be invoked from the command line.
- The name of the file (excluding the extension) must match the name of the database table into which the data import will occur. Failure to match names will result in an error.



Importing Data Using the `mysqlimport` Utility (cont.)

- The file shown below was created to import additional data into the `states` table within the `testdb` database used in the previous example.



```
C:\states.txt - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ?
template.html commentform.html fourthCSS.css state script.sql states.txt
1 California CA Sacramento 36756666 155973
2 Texas TX Austin 22118509 261914
3 South Carolina SC Columbia 4147152 30111
4 Georgia GA Atlanta 9685754 47224
nb char:149 nb line Ln:4 Col:34 Sel:0 Dos\Windows ANSI INS
```

- In this case, the default field delimiter (tab), default field enclosure (nothing), and the default line delimiter (`\n`) were used. Many options are available and are illustrated in the table on pages 65-66.



Importing Data Using the `mysqlimport` Utility

Importing a “data file” into a MySQL database table using the `mysqlimport` utility

```
Prompt
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysqlimport -u root -vr testdb c:\st
Connecting to localhost
Selecting database testdb
Loading data from SERVER file: c:/states.txt into states
testdb.states: Records: 4 Deleted: 0 Skipped: 0 Warnings: 3
Disconnecting from localhost

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

See tables on pages 23-24 for listing of options.

Table updated



Importing Data Using the mysqlimportUtility

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

+-----+-----+-----+-----+-----+
| name   | abbrev | capital   | population | square_miles |
+-----+-----+-----+-----+-----+
| Florida | FL     | Tallahassee | 18328240 | 54153 |
| New York | NY    | Albany     | 194909297 | 54556 |
| Indiana | IN    | Indianapolis | 6376792 | 35789 |
| Maryland | MD   | Annapolis  | 5633597 | 9975 |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from states;
+-----+-----+-----+-----+-----+
| name           | abbrev | capital       | population | square_miles |
+-----+-----+-----+-----+-----+
| Florida        | FL     | Tallahassee  | 18328240 | 54153 |
| New York       | NY    | Albany       | 194909297 | 54556 |
| Indiana        | IN    | Indianapolis  | 6376792 | 35789 |
| Maryland       | MD    | Annapolis    | 5633597 | 9975 |
| California     | CA    | Sacramento   | 36756666 | 155973 |
| Texas          | TX    | Austin       | 22118509 | 261914 |
| South Carolina | SC    | Columbia     | 4147152 | 30111 |
| Georgia        | GA    | Atlanta      | 9685754 | 47224 |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> _
```

Table **before** another client updated the table using the mysqlimport utility.

Table **after** another client updated the table using the mysqlimport utility.



mysqlimportUtility Options

| Option | Action |
|-------------------------------|--|
| -r or --replace | Causes imported rows to overwrite existing rows if they have the same unique key value. |
| -i or --ignore | Ignores rows that have the same unique key value as existing rows. |
| -f or --force | Forces mysqlimport to continue inserting data even if errors are encountered. |
| -l or --lock | Lock each table before importing (a good idea in general and especially on a busy server). |
| -d or --delete | Empty the table before inserting data. |
| --fields-terminated-by='char' | Specify the separator used between values of the same row, default \t (tab). |
| --fields-enclosed-by='char' | Specify the delimiter that encloses each field, default is none. |



mysqlimport Utility Options (cont.)

| Option | Action |
|--|--|
| --fields-optionally-enclosed-by='char' | Same as --fields-enclosed-by, but delimiter is used only to enclosed string-type columns, default is none. |
| --fields-escaped-by='char' | Specify the escape character placed before special characters; default is \. |
| --lines-terminated-by='char' | Specify the separator used to terminate each row of data, default is \n (newline). |
| -u or --user | Specify your username |
| -p or --password | Specify your password |
| -h or --host | Import into MySQL on the named host; default is localhost. |
| -s or --silent | Silent mode, output appears only when errors occur. |
| -v or --verbose | Verbose mode, print more commentary on action. |
| -? or --help | Print help message and exit |



Importing Data From A File With SQL

Statement `Load Data Infile`

- Using the utility `mysqlimport` to load data into a table from an external file works well if the user has access to a command window or command line.
- If you have access via a connection to only the MySQL database, or you are importing data from within an executing application, you will need to use the SQL statement `Load Data Infile`.
- The `Load Data Infile` statement also provides a bit more flexibility since the file name does not need to match the table name. Other than that the options are basically the same and the same results are accomplished.
- The example on page 70 illustrates this SQL command which is available in MySQL.



Importing Data From A File With SQL

Statement Load Data Infile (cont.)

- The basic form of the Load Data Infile statement is:

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'filename'  
[REPLACE | IGNORE]  
INTO TABLE tablename  
[FIELDS  
  [TERMINATED BY 'char' ]  
  [ [OPTIONALLY] ENCLOSED BY 'char' ]  
  [ESCAPED BY '\\char' ] ]  
[LINES  
  [STARTING BY 'char' ]  
  [TERMINATED BY 'char' ] ]  
[IGNORE number LINES]  
[(column_name, ... )]
```

Either allow concurrent update or block until no other clients are reading from the specified table. See page 75.

Same as `-r` and `-i` options in `mysqlimport` utility – either replace or ignore rows with duplicate keys.

Sets the characters that delimit and enclose the fields and lines in the data file. Similar to `mysqlimport` syntax.

Ignores lines at the start of the file (miss header info)

Used to load only certain columns (not entire rows)



Load Data Infile Example

The screenshot shows a Notepad++ window titled 'C:\states2.txt - Notepad++'. The menu bar includes File, Edit, Search, View, Format, Language, Settings, Macro, Run, TextFX, and Plugin. The toolbar contains various icons for file operations and editing. The taskbar shows several open files: fourthCSS.css, state script.sql, states.txt, states2.sql, states3.txt, and states2.txt. The main text area contains the following data:

```
1 Illinois,IL, Springfield,12653544, 55593
2 Maine,ME, Augusta,1305728, 30865
3 Michigan,MI, Lansing,10079985, 56809
4 Oregon,OR, Salem,3559596, 96003
5 Arizona,AZ, Phoenix,5580811, 113642
```

Two callout boxes provide additional information:

- The top callout box (pointing to the first line) states: "String fields may be enclosed by double quotes in this file. Numeric values are not enclosed in quotes."
- The bottom callout box (pointing to the commas and newlines) states: "Fields are delimited by commas and lines are terminated by newline characters (an invisible \n)".

The status bar at the bottom shows: nb char:172 nb line:5 Ln:1 Col:2 Sel:0 Dos\Windows ANSI INS

Text file containing the data to be loaded into the database table.



```
mysql>
mysql> select * from states;
```

| name | abbrev | capital | population | square_miles |
|----------------|--------|--------------|------------|--------------|
| Florida | FL | Tallahassee | 18328240 | 54153 |
| New York | NY | Albany | 194909297 | 54556 |
| Indiana | IN | Indianapolis | 6376792 | 35789 |
| Maryland | MD | Annapolis | 5633597 | 9975 |
| California | CA | Sacramento | 36756666 | 155973 |
| Texas | TX | Austin | 22118509 | 261914 |
| South Carolina | SC | Columbia | 4147152 | 30111 |
| Georgia | GA | Atlanta | 9685754 | 47224 |

States table before addition of data

8 rows in set (0.00 sec)

```
mysql> load data infile 'c:/states2.txt'
-> into table states
-> fields
-> terminated by ','
-> optionally enclosed by '"';
```

Load data infile statement indicating all of the parameters which describe the configuration of the input file.

Query OK, 5 rows affected (0.00 sec)
Records: 5 Deleted: 0 Skipped: 0 Warnings: 0

```
mysql> select * from states;
```

| name | abbrev | capital | population | square_miles |
|----------------|--------|--------------|------------|--------------|
| Florida | FL | Tallahassee | 18328240 | 54153 |
| New York | NY | Albany | 194909297 | 54556 |
| Indiana | IN | Indianapolis | 6376792 | 35789 |
| Maryland | MD | Annapolis | 5633597 | 9975 |
| California | CA | Sacramento | 36756666 | 155973 |
| Texas | TX | Austin | 22118509 | 261914 |
| South Carolina | SC | Columbia | 4147152 | 30111 |
| Georgia | GA | Atlanta | 9685754 | 47224 |
| Illinois | IL | Springfield | 12653544 | 55593 |
| Maine | ME | Augusta | 1305728 | 30865 |
| Michigan | MI | Lansing | 10079985 | 56809 |
| Oregon | OR | Salem | 3559596 | 96003 |
| Arizona | AZ | Phoenix | 5580811 | 113642 |

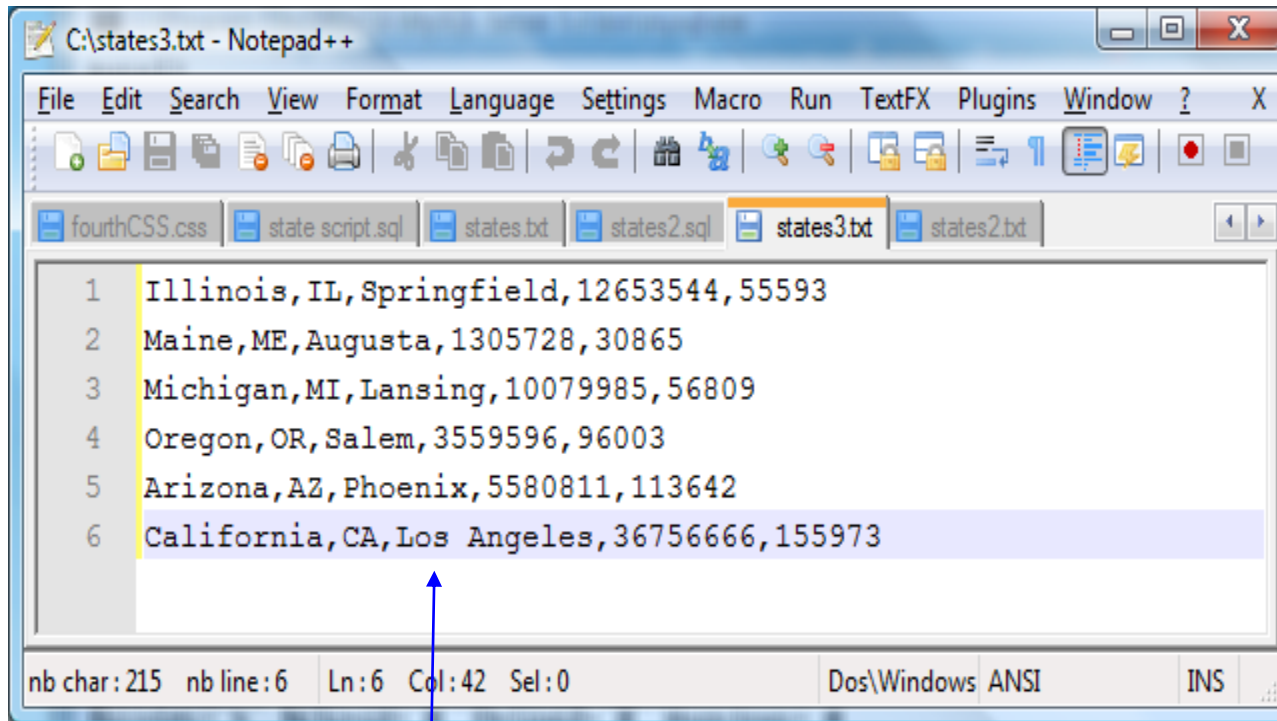
States table after addition of data

13 rows in set (0.00 sec)

```
mysql> _
```



Load Data Infile Example 2



```
C:\states3.txt - Notepad++
File Edit Search View Format Language Settings Macro Run TextFX Plugins Window ? X
fourthCSS.css state script.sql states.txt states2.sql states3.txt states2.txt
1 Illinois, IL, Springfield, 12653544, 55593
2 Maine, ME, Augusta, 1305728, 30865
3 Michigan, MI, Lansing, 10079985, 56809
4 Oregon, OR, Salem, 3559596, 96003
5 Arizona, AZ, Phoenix, 5580811, 113642
6 California, CA, Los Angeles, 36756666, 155973
nb char: 215 nb line: 6 Ln: 6 Col: 42 Sel: 0 Dos\Windows ANSI INS
```

Text file containing the data to be loaded into the database table.

California already exists in the states table – this one will replace the value of the capital with a different value.



mysql> select * from states;

| name | abbrev | capital | population | square_miles |
|----------------|--------|--------------|------------|--------------|
| Florida | FL | Tallahassee | 18328240 | 54153 |
| New York | NY | Albany | 194909297 | 54556 |
| Indiana | IN | Indianapolis | 6376792 | 35789 |
| Maryland | MD | Annapolis | 5633597 | 9975 |
| California | CA | Sacramento | 36756666 | 155973 |
| Texas | TX | Austin | 22118509 | 261914 |
| South Carolina | SC | Columbia | 4147152 | 30111 |
| Georgia | GA | Atlanta | 9685754 | 47224 |
| Illinois | IL | Springfield | 12653544 | 55593 |
| Maine | ME | Augusta | 1305728 | 30865 |
| Michigan | MI | Lansing | 10079985 | 56809 |
| Oregon | OR | Salem | 3559596 | 96003 |
| Arizona | AZ | Phoenix | 5580811 | 113642 |

13 rows in set (0.00 sec)

```
mysql> load data infile 'c:/states3.txt'
-> replace into table states
-> fields
-> terminated by ','
-> optionally enclosed by '"';
```

Query OK, 12 rows affected (0.00 sec)
Records: 6 Deleted: 6 Skipped: 0 Warnings: 0

mysql> select * from states;

| name | abbrev | capital | population | square_miles |
|----------------|--------|--------------|------------|--------------|
| Florida | FL | Tallahassee | 18328240 | 54153 |
| New York | NY | Albany | 194909297 | 54556 |
| Indiana | IN | Indianapolis | 6376792 | 35789 |
| Maryland | MD | Annapolis | 5633597 | 9975 |
| California | CA | Los Angeles | 36756666 | 155973 |
| Texas | TX | Austin | 22118509 | 261914 |
| South Carolina | SC | Columbia | 4147152 | 30111 |
| Georgia | GA | Atlanta | 9685754 | 47224 |
| Illinois | IL | Springfield | 12653544 | 55593 |
| Maine | ME | Augusta | 1305728 | 30865 |
| Michigan | MI | Lansing | 10079985 | 56809 |
| Oregon | OR | Salem | 3559596 | 96003 |
| Arizona | AZ | Phoenix | 5580811 | 113642 |

13 rows in set (0.00 sec)

mysql> _

States table **before** addition of data

Same basic configuration as in previous example except that we have instructed MySQL to replace duplicate key value rows with new values (in this case replacing California's capital).

States table **after** addition of data. Note that California's capital has been changed!



The Ignore Clause of the Insert Command

- While the normal issues of data type compatibility are always of concern, there are other issues to deal with when inserting data into tables.
- There is the possibility that a duplicate of a key may be entered. If so, you will see an error like this:

```
ERROR 1062: Duplicate entry '2' for key 1
```

- It is possible to subdue errors by using the keyword `ignore` in the `insert` statement. By using `ignore` any duplicate rows will simply be ignored. They won't be imported, and the data at the related row of the target table will be left untouched.
 - In your application, you would be wise to check how many rows were affected (imported) whenever using `ignore` because ignoring a record may constitute a failure condition in your application that needs to be handled.



Low Priority and Delayed Inserts

- If you specify `insert low-priority`, the insert waits until all other clients have finished reading from the table before the insert is executed.
- If you specify `insert delayed`, the client performing the action gets an instant acknowledgement that the insert has been performed, although in fact the data will only be inserted when the table is not in use by another thread.
 - This may be useful if you have an application that needs to complete its process in minimum time, or simply where there is no need for it to wait for the effect of an insert to take place. For example, when you're adding data to a log or audit trail.
 - This feature applies only to ISAM or MyISAM type files.



Inserting/Replacing Data Using Replace

- Data can also be entered into a MySQL table using the `replace` command.
- The `replace` statement has forms similar to the `insert` statement:

Form 1 `replace` [`low priority` | `delayed`] [`ignore`] [`into`] *table_name*
[`set`] *column_name1* = *expression1*,
column_name2 = *expression2*, ...

Form 2 `replace` [`low priority` | `delayed`] [`ignore`] [`into`] *table_name*
[(*column_name*,...)] `values` (*expression*,...), (...)...

Form 3 `replace` [`low priority` | `delayed`] [`ignore`] [`into`] *table_name*
[(*column_name*,...)] `select`...



Using replace

- The `replace` statement works similar to `insert`. It always tries to insert the new data, but when it tries to insert a new row with the same primary or unique key as an existing row, it deletes the old row and replaces it with the new values.
- The following examples will illustrate how `replace` operates.

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

mysql> use bikedb;
Database changed
mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 2000  | 9000        |
| Schwinn Paramount P14 | blue  | 1800  | 200         |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> replace into bluebikes
-> values ('Gios Torino Super', 'blue', 4200, 11000);
Query OK, 2 rows affected (0.00 sec)

mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4200  | 11000       |
| Schwinn Paramount P14 | blue  | 1800  | 200         |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> _
```

Changing non-key values. Simplest form of data replacement.



Using Replace (cont.)

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4200  | 11000       |
| Schwinn Paramount P14 | blue  | 1800  | 200         |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> replace into bluebikes
  -> values ('Ridley Damocles','blue',8500,1000);
Query OK, 1 row affected (0.00 sec)

mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4200  | 11000       |
| Schwinn Paramount P14 | blue  | 1800  | 200         |
| Ridley Damocles   | blue  | 8500  | 1000        |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Specifying values for a non-existent key. Basically the same as an insert since the key value being replaced does not currently exist.



Performing Updates on Tables

- The `update` command allows you to modify the values of the existing data in a table. The basic format of the statement is:

```
update [low priority] [ignore] table_name
  set column_name1 = expression1,
      column_name2 = expression2, ...
  [where where_definition]
  [limit num];
```

- There are basically two parts to the statement: the `set` portion to declare which column to set to what value; and the `where` portion, which defines which rows are to be affected.
- `Limit` restricts the number of rows affected to `num`.



Using update (cont.)

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4200  | 11000       |
| Schwinn Paramount P14 | blue  | 1800  | 200         |
| Ridley Damocles    | blue  | 8500  | 1000        |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> update bluebikes
-> set price=price*1.05;
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4410  | 11000       |
| Schwinn Paramount P14 | blue  | 1890  | 200         |
| Ridley Damocles    | blue  | 8925  | 1000        |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Global update within the relation. All tuples have their price field increased by 5%



Using update (cont.)

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe
mysql>
mysql>
mysql>
mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4410  | 11000       |
| Schwinn Paramount P14 | blue  | 1890  | 200         |
| Ridley Damocles    | blue  | 8925  | 1000        |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> update bluebikes
  -> set price=price*1.05
  -> where price > 4500;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from bluebikes;
+-----+-----+-----+-----+
| bikename          | color | price | total_miles |
+-----+-----+-----+-----+
| Gios Torino Super | blue  | 4410  | 11000       |
| Schwinn Paramount P14 | blue  | 1890  | 200         |
| Ridley Damocles    | blue  | 9371  | 1000        |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

Specific update, only tuples satisfying the select condition (those with price greater than 4500) will have their price field increased by 5%.



Select Queries in MySQL

- The `select` command in MySQL is basically the same as in the standard SQL, however, it does have some additional features. The basic format of the statement is (not all options are shown – for complete details see the SQL Manual):

```
SELECT [ALL | DISTINCT | DISTINCTROW] [HIGH_PRIORITY]
      [STRAIGHT JOIN] [SQL_SMALL_RESULT] [SQL_BIG_RESULT]
      [SQL_BUFFER_RESULT] [SQL_CACHE | SQL_NO_CACHE]
      select_expression, ...
[INTO {OUTFILE | DUMPFILE} 'path/to/filename' export_options]
[FROM table_references
  WHERE where_definition]
  [GROUP BY {col_name | col_alias | col_pos | formula}
        [asc | desc], ...]
  [HAVING where_definition]
  [ORDER BY {col_name | col_alias | col_pos | formula}
        [asc | desc], ...]
  [LIMIT [offset, ] num_rows]
  [PROCEDURE procedure_name];
```



MySQL RDBMS (cont.)

- MySQL features a user permissions system, which allows control over user's access to the databases under MySQL control.
- There are very few competitors of MySQL (Oracle, Sybase, DB2, and SQL Server) that can match the level of sophistication provided by MySQL's permissions system in terms of granularity and level of security provided.

Note that I did not include Microsoft Access in the list above. There are a couple of reasons for this; Access concentrates on the client front-end, although available in shareable versions, it lacks the management system that is a key part of any RDBMS. Access provides virtually no user authentication capabilities nor does it have multithreading processing capabilities, in its normal form.



Authorization in MySQL

- `mysql` and the various utility programs such as `mysqladmin`, `mysqlshow`, and `mysqlimport` can only be invoked by a valid MySQL user.
- Permissions for various users are recorded in **grant tables** maintained by MySQL.
- As the root user, you have access to all the databases and tables maintained by the MySQL Server.
- One of these databases is named `mysql` and contains the various information on the users who have access to this installation of MySQL. Some of the tables which comprise this database are shown on the next few pages.



Tables in the mysql Database

The mysql database contains user information

Details on user privileges at the database level. See page 94.

Specific details on privileges at the table level. See page 93

Details on user privileges. See page 91.

Details about the various users. See page 92.

C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| bikedb      |
| mysql      |
| sample     |
| test       |
| testdb     |
+-----+
6 rows in set (0.00 sec)
```

```
mysql> show tables;
+-----+
| Tables_in_mysql |
+-----+
| columns_priv |
| db            |
| event        |
| func         |
| general_log  |
| help_category |
| help_keyword |
| help_relation |
| help_topic   |
| host        |
| ndb_binlog_index |
| plugin      |
| proc        |
| procs_priv  |
| servers     |
| slow_log    |
| tables_priv |
| time_zone   |
| time_zone_leap_second |
| time_zone_name |
| time_zone_transition |
| time_zone_transition_type |
| user        |
| user_info   |
+-----+
24 rows in set (0.00 sec)
```



Contents of the user Table

```
outt; - Notepad
File Edit Format View Help
mysql> use mysql;
Database changed
mysql> describe user;
```

| Field | Type | Null | Key | Default | Extra |
|-----------------------|--------------------------------------|------|-----|---------|-------|
| Host | varchar(60) | | PRI | | |
| User | varchar(16) | | PRI | | |
| Password | varchar(41) | | | | |
| Select_priv | enum('N','Y') | | | N | |
| Insert_priv | enum('N','Y') | | | N | |
| Update_priv | enum('N','Y') | | | N | |
| Delete_priv | enum('N','Y') | | | N | |
| Create_priv | enum('N','Y') | | | N | |
| Drop_priv | enum('N','Y') | | | N | |
| Reload_priv | enum('N','Y') | | | N | |
| Shutdown_priv | enum('N','Y') | | | N | |
| Process_priv | enum('N','Y') | | | N | |
| File_priv | enum('N','Y') | | | N | |
| Grant_priv | enum('N','Y') | | | N | |
| References_priv | enum('N','Y') | | | N | |
| Index_priv | enum('N','Y') | | | N | |
| Alter_priv | enum('N','Y') | | | N | |
| Show_db_priv | enum('N','Y') | | | N | |
| Super_priv | enum('N','Y') | | | N | |
| Create_tmp_table_priv | enum('N','Y') | | | N | |
| Lock_tables_priv | enum('N','Y') | | | N | |
| Execute_priv | enum('N','Y') | | | N | |
| Repl_slave_priv | enum('N','Y') | | | N | |
| Repl_client_priv | enum('N','Y') | | | N | |
| ssl_type | enum('', 'ANY', 'X509', 'SPECIFIED') | | | N | |
| ssl_cipher | blob | | | | |
| x509_issuer | blob | | | | |
| x509_subject | blob | | | | |
| max_questions | int(11) unsigned | | | 0 | |
| max_updates | int(11) unsigned | | | 0 | |
| max_connections | int(11) unsigned | | | 0 | |

```
31 rows in set (0.00 sec)
```



Contents of the `user_info` Table

```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

mysql> describe user_info;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key  | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| User           | varchar(16)   | NO   | PRI  | NULL    |       |
| Full_name      | varchar(60)   | YES  | MUL  | NULL    |       |
| Description    | varchar(255)  | YES  |      | NULL    |       |
| Email          | varchar(80)   | YES  |      | NULL    |       |
| Contact_information | text         | YES  |      | NULL    |       |
| Icon           | blob          | YES  |      | NULL    |       |
+-----+-----+-----+-----+-----+-----+

6 rows in set (0.02 sec)

mysql>
```



Contents of the tables_priv Table

```
mysql> \t;
mysql> describe tables_priv;
+-----+-----+
| Field      | Type                                     |
+-----+-----+
| Host      | char(60)                                |
| Db        | char(64)                                |
| User      | char(16)                                 |
| Table_name| char(64)                                 |
| Grantor   | char(77)                                 |
| Timestamp | timestamp                               |
| Table_priv| set('Select','Insert','Update','Delete','Create','Drop','Grant','References','Index','Alter') |
| Column_priv| set('Select','Insert','Update','References') |
+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql> show full columns in information_schema.tables_priv;
+-----+-----+-----+-----+-----+-----+
| Table      | Column      | Null | Key | Default      | Extra |
+-----+-----+-----+-----+-----+-----+
| tables_priv | Timestamp   | YES  | PRI | CURRENT_TIMESTAMP |      |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```



Contents of the db Table

```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe
mysql> describe db;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Host           | char(60)      | NO   | PRI |          |       |
| Db             | char(64)      | NO   | PRI |          |       |
| User          | char(16)      | NO   | PRI |          |       |
| Select_priv   | enum('N','Y') | NO   |     | N        |       |
| Insert_priv   | enum('N','Y') | NO   |     | N        |       |
| Update_priv   | enum('N','Y') | NO   |     | N        |       |
| Delete_priv   | enum('N','Y') | NO   |     | N        |       |
| Create_priv   | enum('N','Y') | NO   |     | N        |       |
| Drop_priv     | enum('N','Y') | NO   |     | N        |       |
| Grant_priv    | enum('N','Y') | NO   |     | N        |       |
| References_priv | enum('N','Y') | NO   |     | N        |       |
| Index_priv    | enum('N','Y') | NO   |     | N        |       |
| Alter_priv    | enum('N','Y') | NO   |     | N        |       |
| Create_tmp_table_priv | enum('N','Y') | NO   |     | N        |       |
| Lock_tables_priv | enum('N','Y') | NO   |     | N        |       |
| Create_view_priv | enum('N','Y') | NO   |     | N        |       |
| Show_view_priv | enum('N','Y') | NO   |     | N        |       |
| Create_routine_priv | enum('N','Y') | NO   |     | N        |       |
| Alter_routine_priv | enum('N','Y') | NO   |     | N        |       |
| Execute_priv  | enum('N','Y') | NO   |     | N        |       |
| Event_priv    | enum('N','Y') | NO   |     | N        |       |
| Trigger_priv  | enum('N','Y') | NO   |     | N        |       |
+-----+-----+-----+-----+-----+-----+
22 rows in set (0.00 sec)

mysql>
```



How The Grant Tables Work

- The various grant tables work together to define access capabilities for the various users of the databases in MySQL. The tables represent a hierarchy which begins at the database level and moves downward to finer and finer granularity in access capabilities.
- To understand how the grant tables work, it is necessary to understand the process that MySQL goes through when considering a request from a client.

Step 1: A user attempts to connect to the MySQL server. The `user` table is consulted, and on the basis of the username, password, and host from which the connection is occurring, the connection is either refused or accepted. (MySQL actually sorts the user table and looks for the first match.)



How The Grant Tables Work (cont.)

Step 2: If the connection is accepted, any privilege fields in the `user` table that are set to 'Y' will allow the user to perform that action on any database under the server's control. For administrative actions such as shutdown and reload, the entry in the `user` table is deemed absolute, and no further grant tables are consulted.

Step 3: Where the user makes a database-related request and the `user` table does not allow the user to perform that operations (the privilege is set to 'N'), MySQL consults the `db` table (see page 84).

Step 4: The `db` table is consulted to see if there is an entry for the user, database, and host. If there is a match, the `db` privilege fields determine whether the user can perform the request.



How The Grant Tables Work (cont.)

Step 5: If there is a match on the db table's Db and User files but Host is blank, the host table is consulted to see whether there is a match on all three fields. If there is, the privilege fields in the host table will determine whether the use can perform the requested operation. Corresponding entries in the db and host tables must both be 'Y' for the request to be granted. Thus, an 'N' in either table will block the request.

Step 6: If the user's request is not granted, MySQL checks the tables_priv (see page 83) and columns_priv tables. It looks for a match on the user, host, database, and table to which the request is made (and the column, if there is an entry in the columns_priv table). It adds any privileges it finds in these tables to the privileges already granted. The sum of these privileges determines if the request can be granted.



Managing User Privileges with GRANT and REVOKE

- The basic granting and revocation of privileges in MySQL are accomplished through the `grant` and `revoke` commands.
- The format of the `grant` command is:

```
GRANT privileges [(column_list)]  
ON database_name.table_name  
TO username@hostname [IDENTIFIED BY 'password']  
[REQUIRE [SSL | X509]  
    [CIPHER cipher [AND] ]  
    [ISSUER issuer [AND] ]  
    [SUBJECT subject ] ]  
[WITH GRANT OPTION |  
    MAX_QUERIES_PER_HOUR num |  
    MAX_UPDATES_PER_HOUR num |  
    MAX_CONNECTIONS_PER_HOUR num ]
```



Some of the Privileges Assigned with GRANT

| Privilege | Operations Permitted |
|-------------------------|---|
| ALL or ALL PRIVILEGES | All privileges except for GRANT |
| ALTER | Change a table definition using ALTER TABLE excluding the creation and dropping of indices. |
| CREATE | Create database or tables within a database. |
| CREATE TEMPORARY TABLES | Create temporary tables. |
| DELETE | Ability to perform deletions from tables. (Delete DML statements). |
| DROP | Ability to drop databases or tables. |
| INSERT | Ability to insert data into tables. |
| SHUTDOWN | Ability to shutdown the MySQL server. |

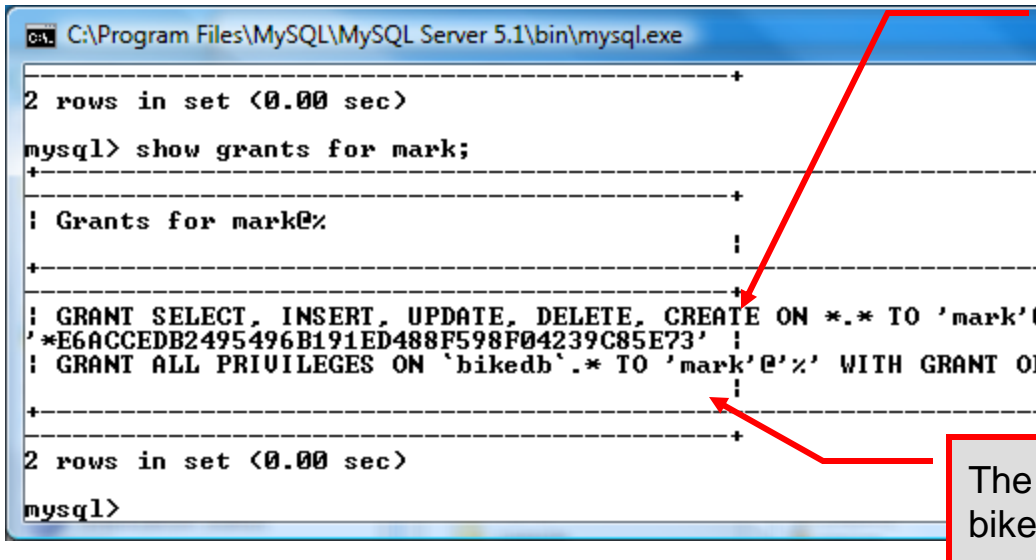


Displaying Privileges with SHOW

- The SQL command SHOW is used to display the grant privileges for a given user.
- The syntax for the SHOW command is:

SHOW GRANTS FOR *username@hostname*

- An example is shown below:



```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysql.exe

2 rows in set (0.00 sec)
mysql> show grants for mark;
+-----+
| Grants for mark@% |
+-----+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON *.* TO 'mark'@'%' IDENTIFIED BY PASSWORD '*E6ACCEDB2495496B191ED488F598F04239C85E73' |
| GRANT ALL PRIVILEGES ON `bikedb`.* TO 'mark'@'%' WITH GRANT OPTION |
+-----+

2 rows in set (0.00 sec)
mysql>
```

This user has only SELECT, INSERT, UPDATE, DELETE, and CREATE global privileges.

The user has all privileges on the bikedb database.



Revoking User Privileges with REVOKE

- Revocation of privileges in MySQL is accomplished with the `revoke` command.

- The format of the `revoke` command is:

```
REVOKE privileges [(column_list)]  
ON database_name.table_name  
FROM username@hostname
```

- An example is shown on the next page.



Example - Revoking User Privileges with REVOKE

```
C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

+-----+
| Grants for mark@% |
+-----+
+-----+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON *.* TO 'mark'@'%' IDENTIFIED BY PASSWORD |
| '*E6ACCEDB2495496B191ED488F598F04239C85E73' |
| GRANT SELECT ON `testdb`.* TO 'mark'@'%' |
+-----+
| GRANT ALL PRIVILEGES ON `hikedb`.* TO 'mark'@'%' WITH GRANT OPTION |
+-----+
| GRANT SELECT ON `testdb`.`states` TO 'mark'@'%' |
+-----+

4 rows in set (0.00 sec)

mysql> revoke select
-> on testdb.states
-> from mark;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for mark;
+-----+
| Grants for mark@% |
+-----+
+-----+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE ON *.* TO 'mark'@'%' IDENTIFIED BY PASSWORD |
| '*E6ACCEDB2495496B191ED488F598F04239C85E73' |
| GRANT SELECT ON `testdb`.* TO 'mark'@'%' |
+-----+
| GRANT ALL PRIVILEGES ON `hikedb`.* TO 'mark'@'%' WITH GRANT OPTION |
+-----+

3 rows in set (0.00 sec)

mysql> _
```

User has SELECT privilege on testdb.states table.

Revoking user's SELECT privilege on testdb.states.

User's grant listing shows that they no longer have SELECT privilege on testdb.states table.

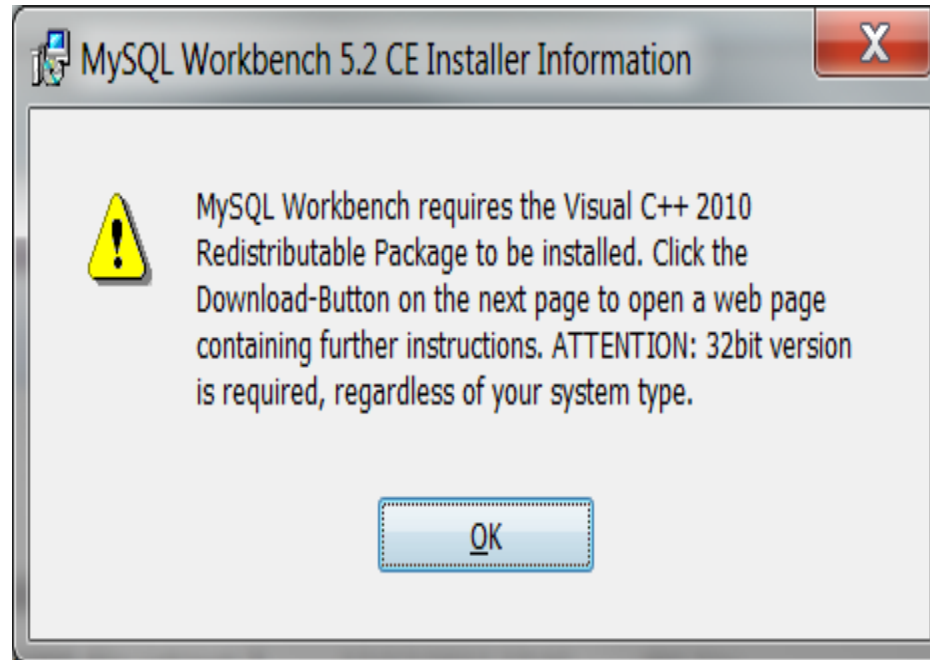


The MySQL Workbench

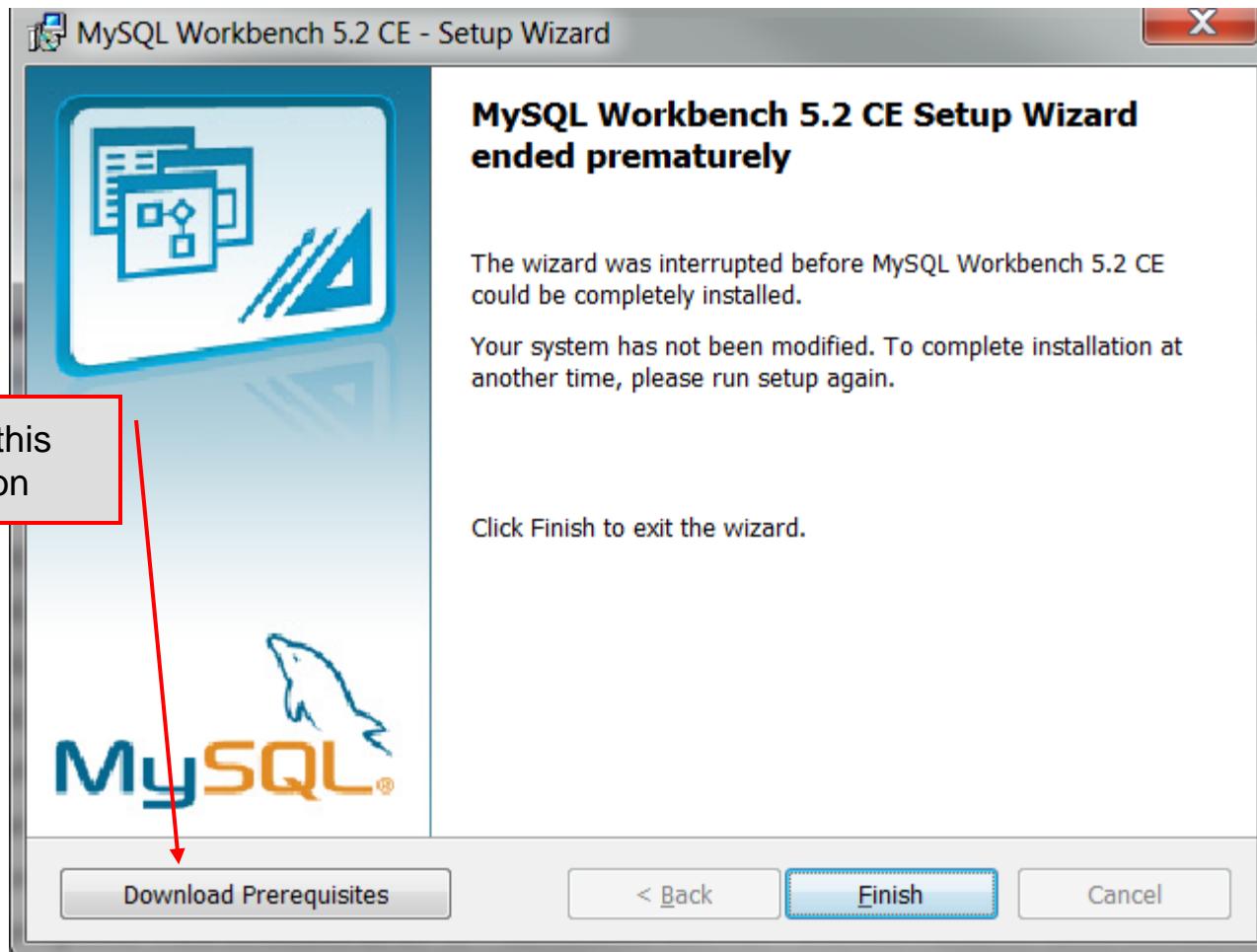
- From MySQL you can download a GUI-based tool that allows you to create, manipulate, and administer MySQL databases.
- The current version of this tool (5.2.40) does not implement full functionality of the GRANT command down to the attribute level.
- This tool also contains some system administrator functionality for monitoring system resources and utilization.
- You can download this tool at: <http://www.mysql.com/products/> (see page 7).
- The install/set-up for this tool as well as a few screen shots of this tool and its capabilities are shown in the next few slides.

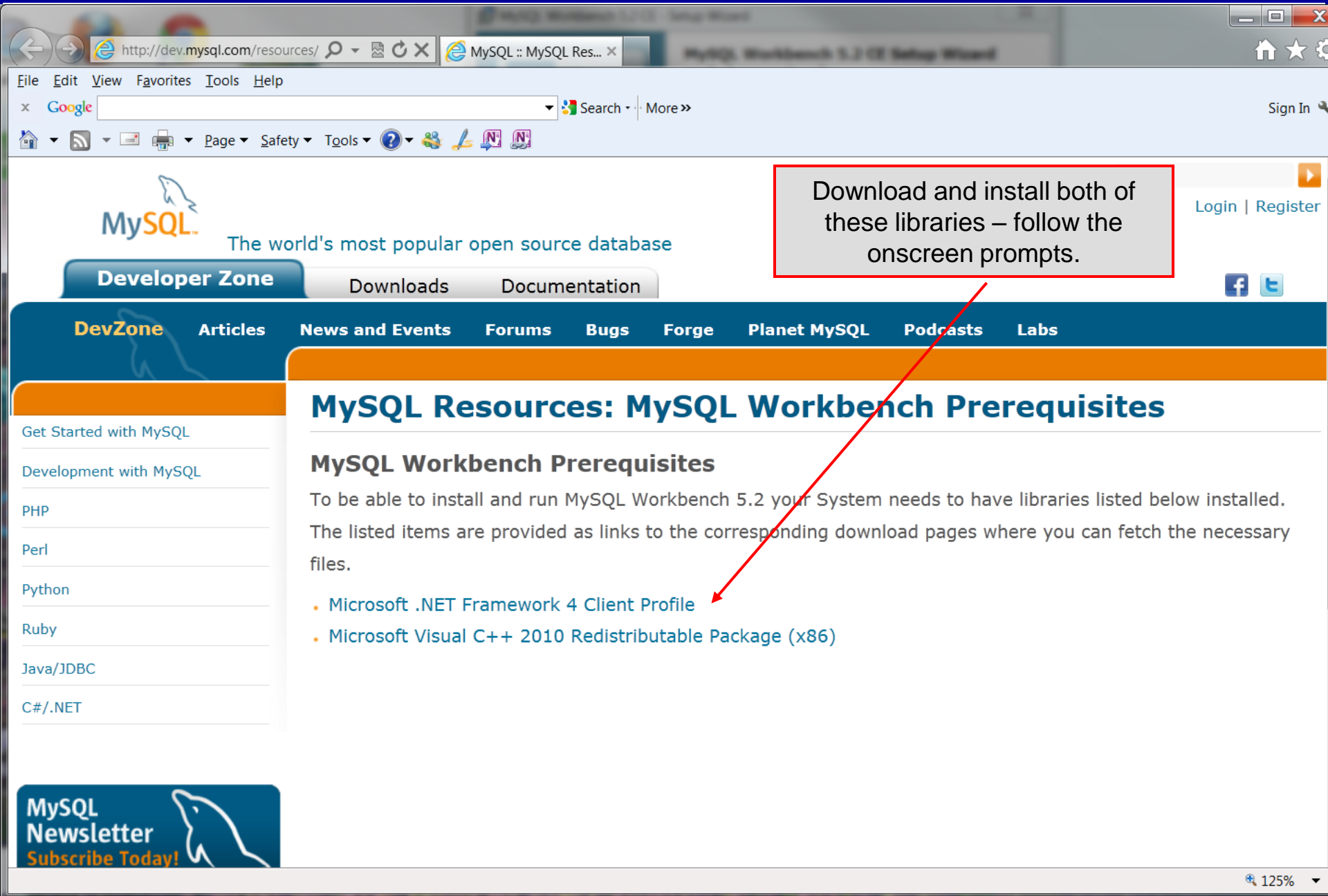


The MySQL Workbench



The MySQL Workbench





Download and install both of these libraries – follow the onscreen prompts.

MySQL Resources: MySQL Workbench Prerequisites

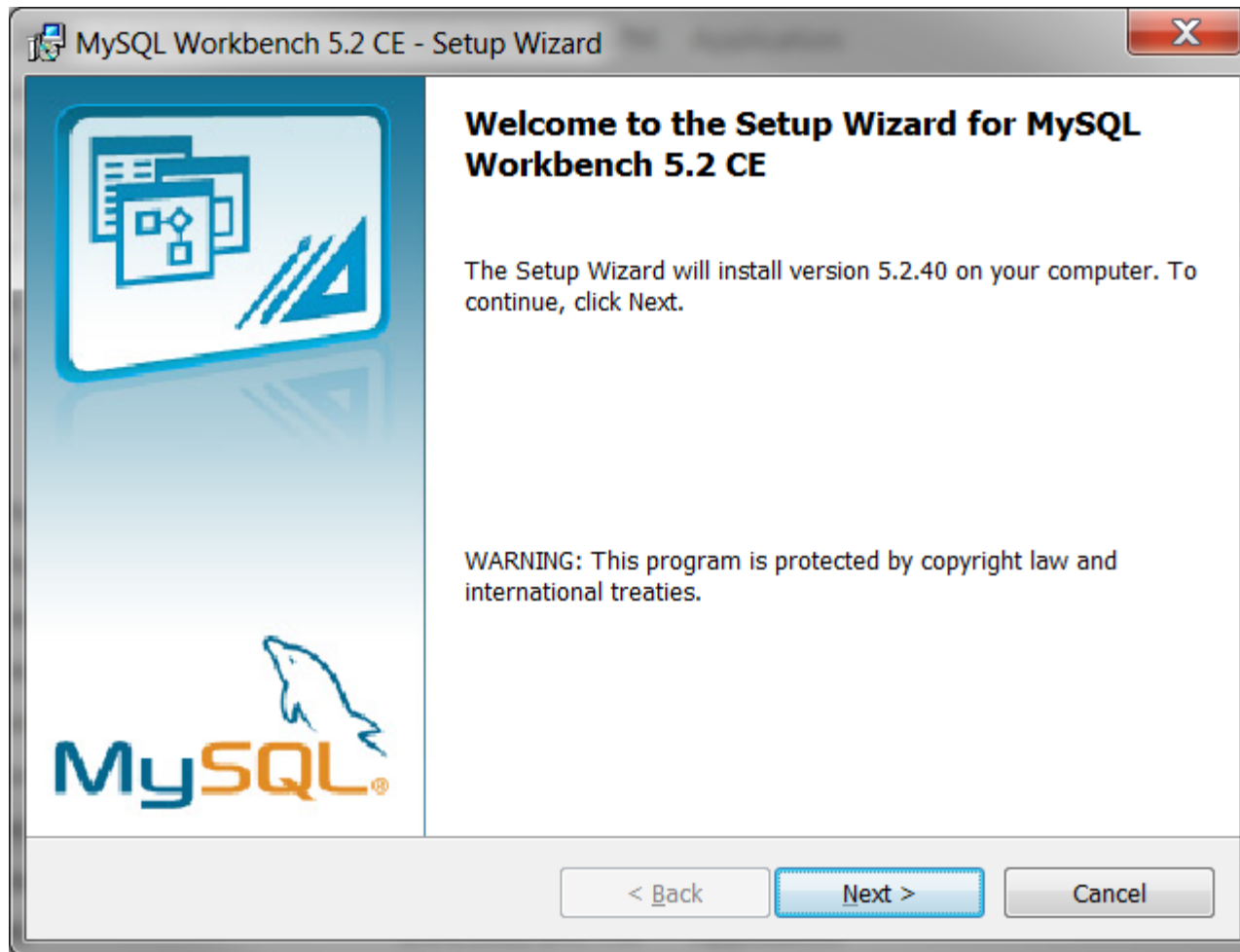
MySQL Workbench Prerequisites

To be able to install and run MySQL Workbench 5.2 your System needs to have libraries listed below installed. The listed items are provided as links to the corresponding download pages where you can fetch the necessary files.

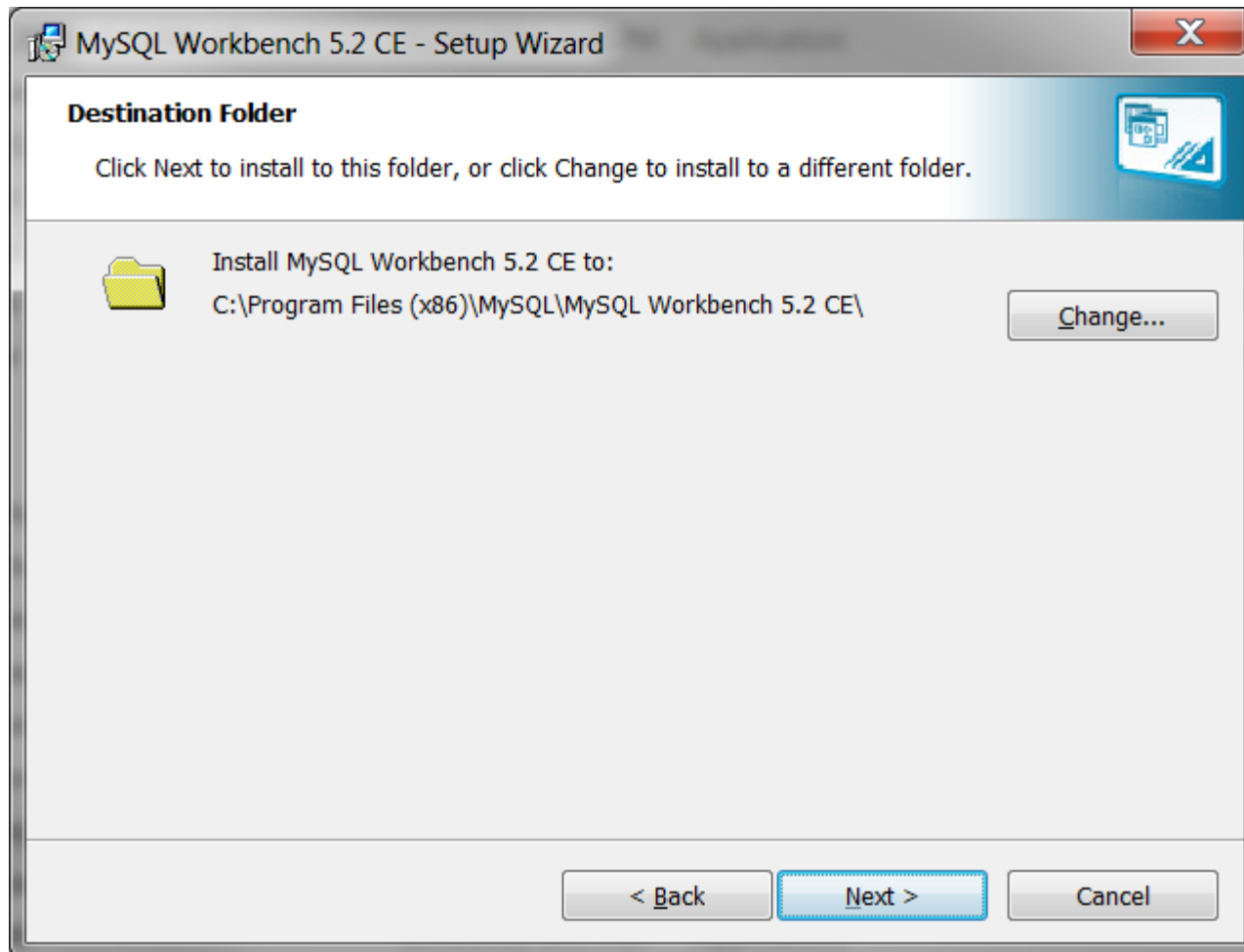
- [Microsoft .NET Framework 4 Client Profile](#)
- [Microsoft Visual C++ 2010 Redistributable Package \(x86\)](#)



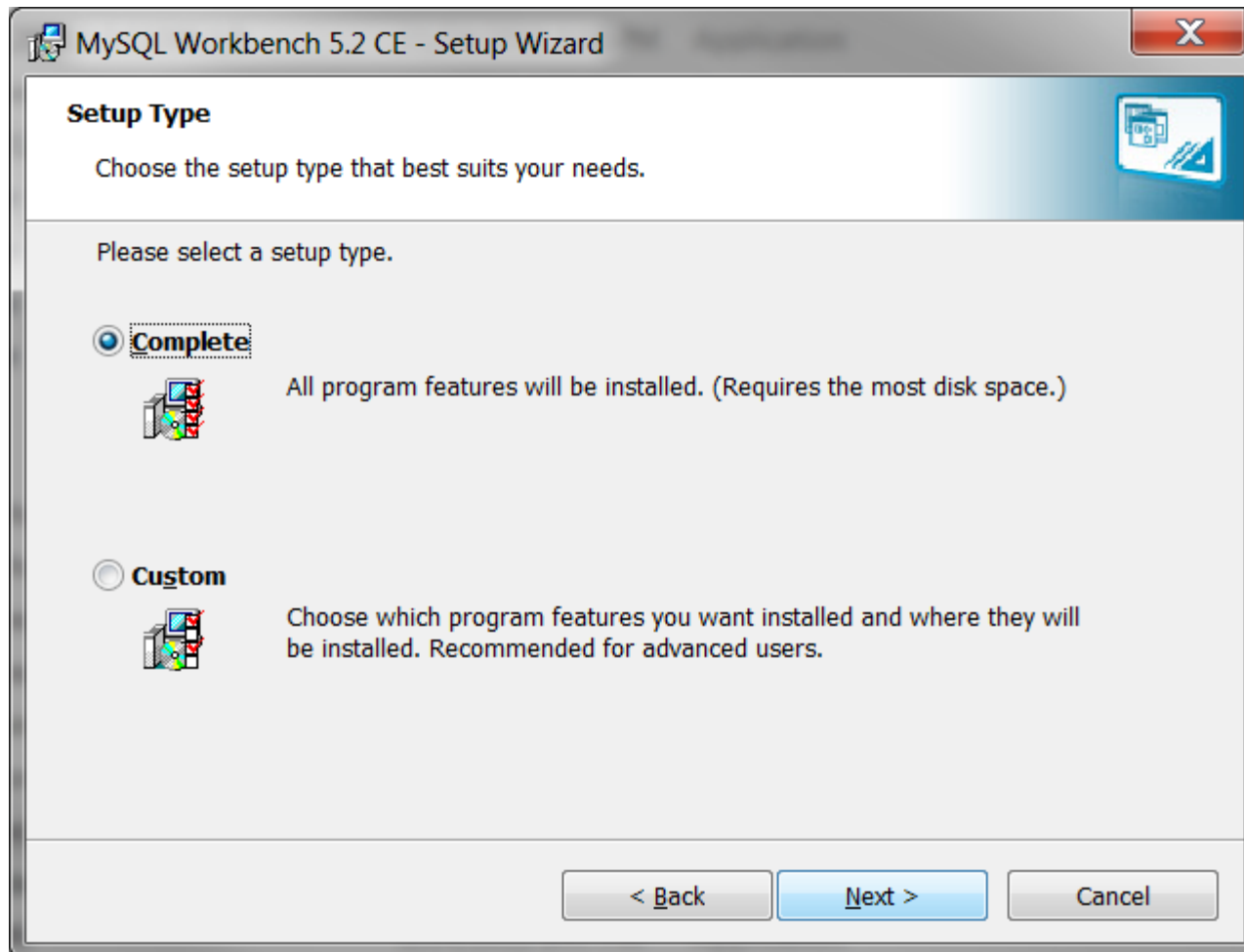
The MySQL Workbench



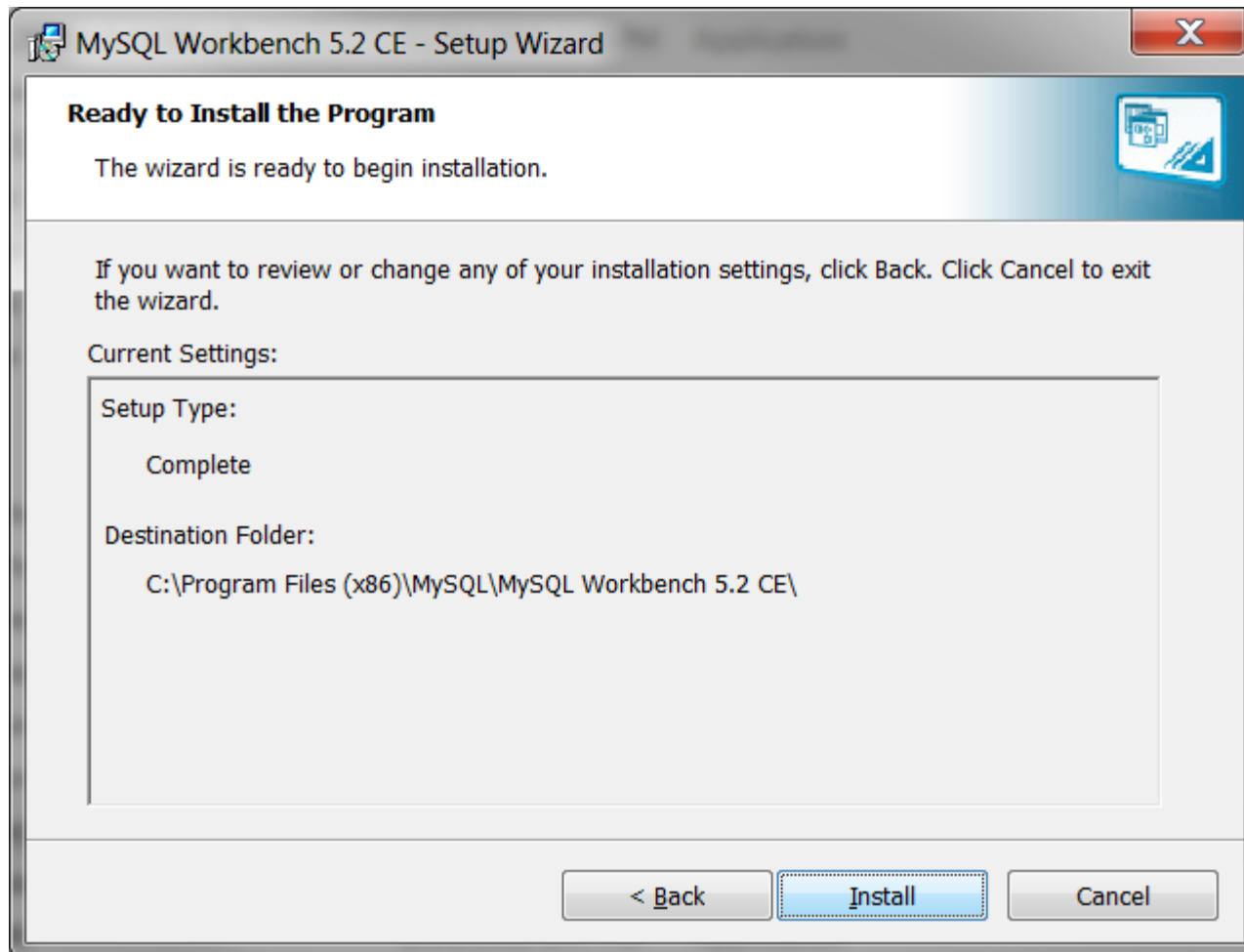
The MySQL Workbench



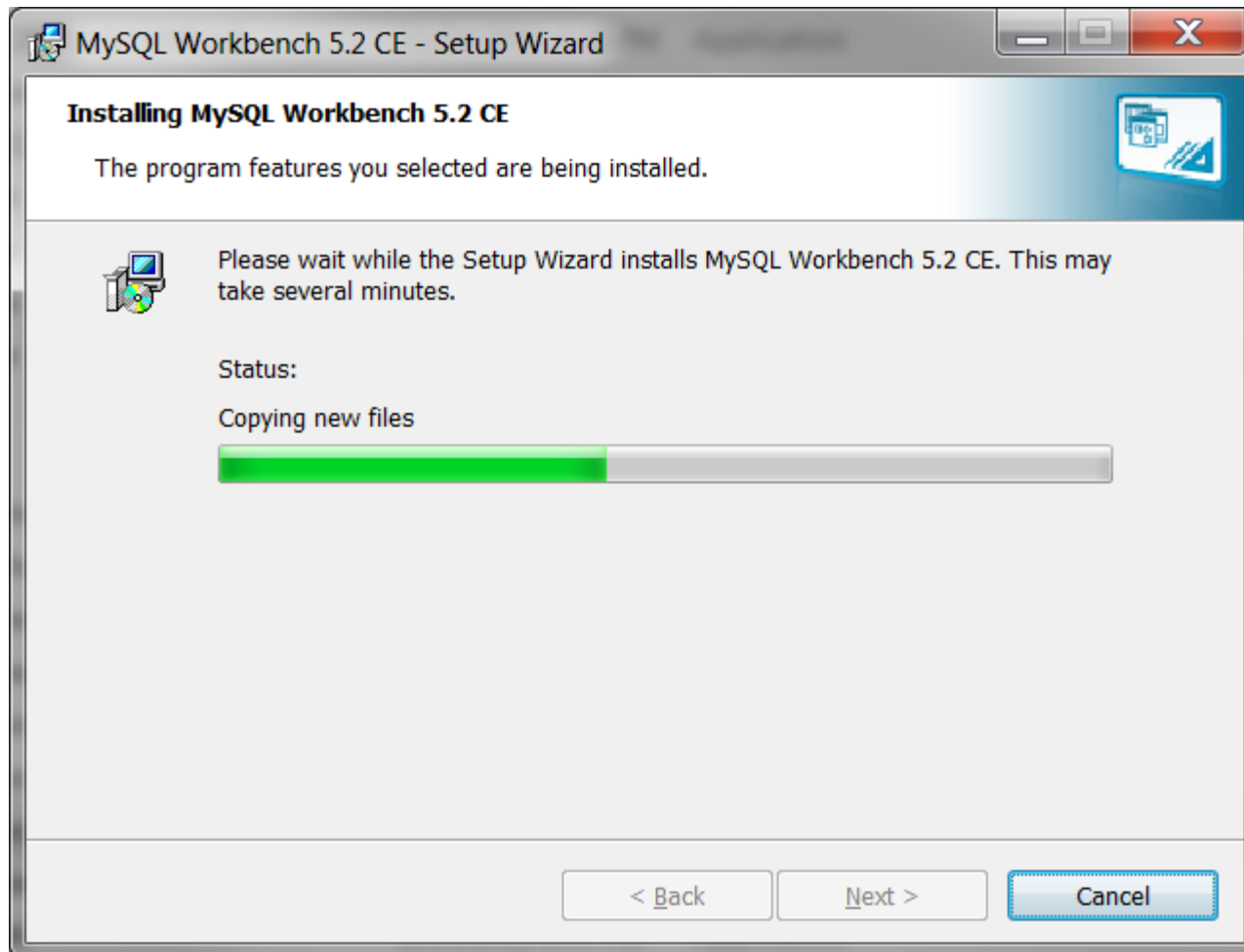
The MySQL Workbench



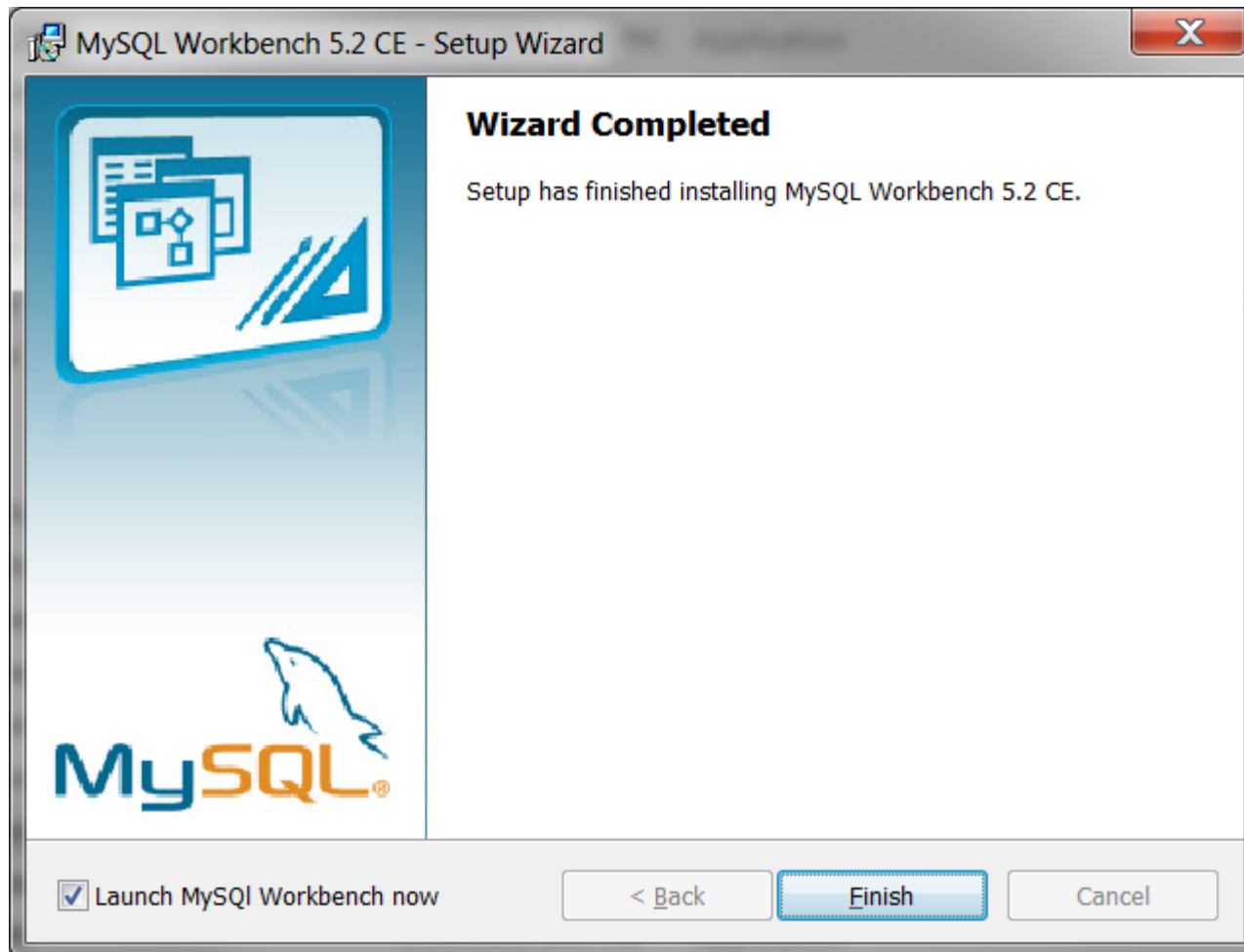
The MySQL Workbench



The MySQL Workbench



The MySQL Workbench



Workbench Central

Workspace



SQL Development

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



Open Connection to Start Querying

Or click a DB connection to open the SQL Editor.



Local instance MySQL

User: root Host: localhost:3309



New Connection

Add a new database connection for querying.



Edit Table Data

Select a connection and schema table to edit.



Edit SQL Script

Open an existing SQL Script file for editing.



Manage Connections

Modify connection settings or add connections.



Data Modeling

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



Open Existing EER Model

Or select a model to open or click here to browse.



Create New EER Model

Create a new EER Model from scratch.



Create EER Model From Existing Database

Create by connecting and reverse engineering.



Create EER Model From SQL Script

Import an existing SQL file.



Server Administration

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



Server Administration

Or click to manage a database server instance.



Local MySQL

Local Type: Windows



New Server Instance

Register a new server instance to manage.



Manage Import / Export

Create a dump file or restore data from a file.



Manage Security

Manage user accounts and assign privileges.

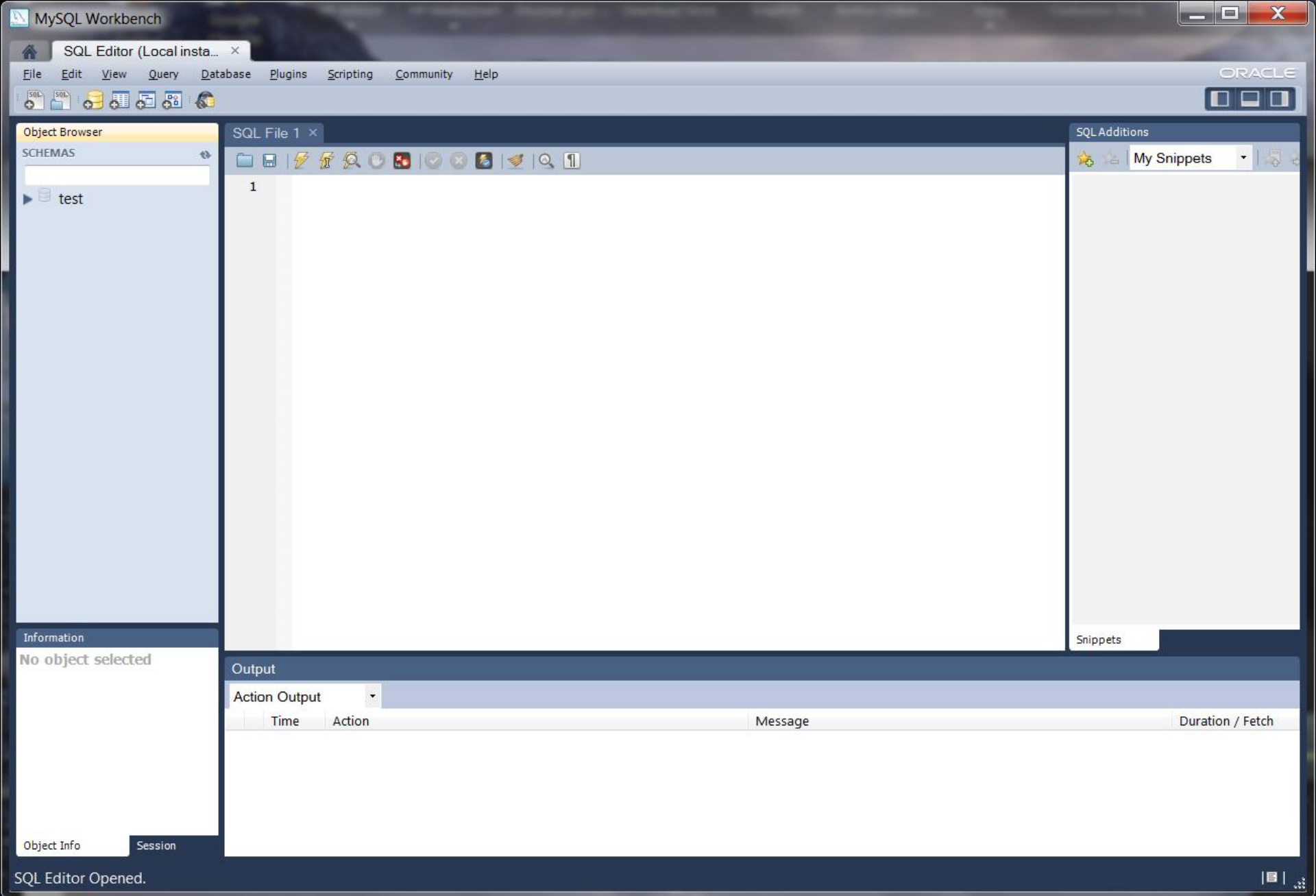


Manage Server Instances

Add, delete and update server instance settings.

Ready.





Workbench Scripting Shell

File Browser

- User Scripts
- User Modules
- User Libraries

Shell Snippets bikedbscriptsql

```
1  ?# Script file for creating the bikedb that is used in many of the SQL and MyS
2  # examples for the CNT 4714 notes
3
4  drop database if exists bikedb;
5
6  create database bikedb;
7
8  use bikedb;
9
10 create table bikes (
11     bikename varchar(30) not null,
12     size int(2),
13     color varchar(15),
14     cost int(6),
15     purchased date,
16     mileage int(6),
17     primary key (bikename)
18 );
```

Debugging

Files Globals Classes Modules Output Breakpoints Debug Info



MySQL Workbench

SQL Editor (Local insta... x)

File Edit View Query Database Plugins Scripting Community Help

ORACLE

Object Browser

SCHEMAS

- bikedb
 - Tables
 - bikes
 - bluebikes
 - Views
 - Routines
 - test

SQL File 1 bikedbscript SQL File 3* x

```
1 • select * from bikes
```

Filter: Edit: Export: Autosize:

| bikename | size | color | cost | purchased | mileage |
|------------------------|------|-----------------|------|------------|---------|
| Battaglin Carrera | 60 | red/white | 4000 | 2001-03-10 | 11200 |
| Bianchi Corse Evo 4 | 58 | celeste | 5700 | 2004-12-02 | 300 |
| Bianchi Evolution 3 | 58 | celeste | 4800 | 2003-11-12 | 2000 |
| Bianchi Infinito | 58 | celeste | 8900 | 2011-07-14 | 0 |
| BMC SLC01 - Swiss | 58 | red/black/white | 8000 | 2010-06-23 | 0 |
| Colnago Dream Rabobank | 60 | blue/orange | 5500 | 2002-07-07 | 4300 |
| Colnago Superissimo | 59 | red | 3800 | 1996-03-01 | 13000 |
| Eddy Merckx Domo | 58 | blue/black | 5300 | 2004-02-02 | 0 |
| Eddy Merckx Molteni | 58 | orange | 5100 | 2004-08-12 | 0 |
| Gianni Motta Personal | 59 | red/green | 4400 | 2000-05-01 | 8700 |
| Gios Torino Super | 60 | blue | 2000 | 1998-11-08 | 9000 |
| Ridley Damocles | 58 | blue/black | 7500 | 2008-06-27 | 0 |
| Ridley X-Fire | 58 | red/white | 7500 | 2011-09-01 | 0 |

bikes 1 Apply Cancel Snippets

Information

No object selected

Object Info Session

Output

Action Output

| | Time | Action | Message | Duration / Fetch |
|---|-------------|--|--|-----------------------|
| ✓ | 19 18:30:35 | select * from bikes LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |
| ✓ | 20 18:30:35 | create table bluebikes (bikename varchar(30), color ... | 0 row(s) affected | 0.062 sec |
| ✓ | 21 18:30:35 | # Use a form 3 insert statement to load the bluebikes table... | 2 row(s) affected Records: 2 Duplicates: 0 Warnings: 0 | 0.000 sec |
| ✓ | 22 18:30:35 | select * from bluebikes LIMIT 0, 1000 | 2 row(s) returned | 0.000 sec / 0.000 sec |
| ✓ | 23 18:32:45 | select * from bikes LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |

Query Completed



Task and Object Browser

MANAGEMENT

- Server Status
- Startup / Shutdown
- Status and System Variables
- Server Logs

CONFIGURATION

- Options File

SECURITY


- Users and Privileges

DATA EXPORT / RESTORE

- Data Export
- Data Import/Restore


Server Status

INFO




Name: **Local MySQL**
 Host: **localhost**
 Server: **5.5.25**
 Status: **Running**

SYSTEM



CPU: 36% Mem: 31%

SERVER HEALTH



Connection Usage: 4 Traffic: 7.97 KB/s Query Cache Hitrate: 4.76% Key Efficiency: 0.00%

CONNECTIONS

| Id | User | Host | DB | Command | Time | State | Info |
|----|------|-----------------|--------|---------|------|-------|------------------|
| 2 | root | localhost:50474 | | Sleep | | 56 | |
| 3 | root | localhost:50475 | bikedb | Sleep | | 91 | |
| 5 | root | localhost:50766 | | Query | | 0 | SHOW PROCESSLIST |
| 6 | root | localhost:50767 | | Sleep | | 2 | |

WB Admin Opened



Task and Object Browser

MANAGEMENT

- Server Status
- Startup / Shutdown
- Status and System Variables
- Server Logs

CONFIGURATION

- Options File

SECURITY

- Users and Privileges**

DATA EXPORT / RESTORE

- Data Export
- Data Import/Restore

Users and Privileges

Server Access Management Schema Privileges

User Accounts

Select an account to edit or click Add Account to create a new one

| User | From Host |
|------|-----------|
| root | localhost |

Login Administrative Roles Account Limits

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

Limit Connectivity to Hosts Matching: * % and _ wildcards may be used

Password: Type a password to reset it.

Confirm Password: Enter password again to confirm.

Add Account Remove Revoke All Privileges Revert Apply Refresh

WB Admin Opened

