Introduction To MySQL
Installation Of MySQL 5.6.17

Instructor: Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790

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 5.6.17 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.
MySQL Community Server
5.6.17 GA
5.5.37 GA
5.7.4 m14

MySQL Enterprise Edition
Learn More »

MySQL Cluster CGE
Learn More »

New Releases
MySQL Workbench 6.1
(6.1.6 GA)

Developer Articles

Importing Raster Based Spatial Data into MySQL 5.7
By Matt Lord | April 25, 2014
GDAL 1.10+ (by OS Geo) supports converting spatial data between various raster and vector (points, lines, polygons, etc.) based formats, along with supporting MySQL. MySQL and ESRI Shapefiles are already being supported in this version.

New! MySQL Enterprise Monitor 3.0
Performance & Health Monitoring

Click here to go to download page
This should be the next page you see. Scroll down this page until you find the MySQL Community Edition and click that link.
The MySQL Community Server page.

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.
Use this selection window to select the proper platform/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.
If you are not using the all-inclusive loader, go back to the main download page and also download MySQL Workbench which contains the Administrator and MySQL Query Browser GUI tools.
Once again, go back to the main download page and select Connectors.
Download the Connector/J for use with Java applications.
Installing MySQL 5.6.17

- 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 8 over the next few pages, just to give you an idea of what you should be seeing.

- Note: Starting on page 108 are a sequence of screen shots that you would see if you are updating an older version of MySQL using the MSI Installer.
Installing MySQL 5.6.17 (cont.)

License Agreement

To install MySQL, you must accept the Oracle Software License Terms.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1231 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom
to share and change it. By contrast, the GNU General Public License
is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it. (SOME OTHER FREE SOFTWARE FOUNDATION SOFTWARE IS COVERED BY
the GNU Library General Public License Instead.) You can apply it to
your programs, too.

When we speak of free software, we are referring to freedom, not price.
Our General Public Licenses are designed to make sure that you have
the freedom to distribute copies of free software (and charge for this
service if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs; and that you know you can do these things.
Installing MySQL 5.6.17 (cont.)

Before the installation is performed, the Installer will check if there are newer versions of the products you are about to install / already installed are available.

- Connect to the Internet
- Fetch product update information

Option to skip the check for updates (not recommended).
Installing MySQL 5.6.17 (cont.)

Before the installation is performed, the Installer will check if there are newer versions of the products you are about to install / already installed are available.

- Connect to the Internet
- Fetch product update information

The operation is complete. Please click 'Next >' to continue.

[Options: Back, Next, Cancel]
Your choice here. For this course, a developer default, full, or custom set-up will work fine. Do Not Select Server Only or Client Only. I'm illustrating a custom set-up.
On a custom install, you'll go through each of the choices on the left menu list. First up will be setting the features. Selecting which connectors you want loaded. I've selected only the Connector/J for this server.
A requirements check looks for all of the supporting tools that MySQL needs. Any missing requirements will initiate a prompt and you’ll need to handle the issues separately. On most current Windows-based machines there should not be any problems.
You're now at the point to download and install the server configuration you've selected. Just click Execute (and stand a safe distance from your system 😊).
If all went well, you’ll see this screen evolve as each product is installed…when you get all green checkmarks, click Next.
You're now at the point to begin configuring the server. This is where you customize how the server is to behave. Click Next.
Select Config Type: Development Machine. Check TCP/IP enable. Port will default to 3306, which is fine. I have several MySQL servers running on different ports. We don’t need any advanced options at this point. Click Next.
Providing an easily remembered root user password. Doesn't matter if it is considered weak.

We don’t need to define any additional user accounts at this point. You can do this later if you wish.
MySQL is designed to run as a Windows service. Unless you want to start and stop it manually, allow this to happen.

Select standard system account.
Installing MySQL 5.6.17 (cont.)

Click Next to begin the MySQL Server configuration. It will start automatically as a service when this completes (see next slide).
You’re done. The MySQL Server is now running and the MySQL Workbench will start when you click Finish.
This is the MySQL WorkBench. It is basically waiting for you to select a server to connect to at this point. You will have only the one you just created, so your screen will look like this one. Click on the instance and login as the root user.
This is the MySQL WorkBench main screen. From here you can create, use, and manage databases as well as user accounts and the server itself.
This is the MySQL WorkBench main screen shown with the script file available on the course website for you to play around with loaded. This script will create and populate a small database. Under File, Select Open SQL Script. Navigate to where you placed the script file and open it into this editing window.
The script executed, which created and populated the database, then executed the query at the bottom of the script. Shown in the results window is the execution results of that query. Shown in the output window is the MySQL Server output.
A different query executed against the same database instance.
Running MySQL 5.6.17

• 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 5.6 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 5.6.17 (cont.)

Hopefully, you see this output from MySQL. The MySQL server is now awaiting a command from this client.
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.
List all databases managed by this MySQL server which are accessible to this client.

Terminate client connection.

```
mysql> show databases;
+-------------------------+
| Database                |
+-------------------------+
| information_schema      |
| bikesdb                 |
| colorsurvey             |
| mailinglist             |
| mysql                   |
| performance_schema      |
| project3                |
| test                    |
+-------------------------+
8 rows in set (0.00 sec)
mysql> exit;
```
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 acknowledges selection of `bikedb` database.
In the Workbench, you select a database by clicking on it.
Viewing the Schema of a Relation

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

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> use bikedb;
Database changed
mysql> describe bikes;

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>bikename</td>
<td>varchar(30)</td>
<td>NO</td>
<td>PRI</td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>size</td>
<td>int(2)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td>varchar(15)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>cost</td>
<td>int(6)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>purchased</td>
<td>date</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>mileage</td>
<td>int(6)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

6 rows in set (0.00 sec)
```

Specify which table’s schema to describe. All information regarding the schema visible to the user is displayed.
To see the details of a table’s schema, use the icons to select more or less detail. Details shown in information area below.
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.

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

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 straightforward. The example below illustrates a simple selection query on the `bikes` table of the `bikedb` database.

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

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>`
Dropping a Database in MySQL

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

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.

```sql
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 varchar(255) default null,  
    -> );
Query OK, 0 rows affected (0.04 sec)
mysql>
```
Manipulating Tables in MySQL (cont.)

Screen shot that describes the newly created table.

```sql
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)
```
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

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.

```
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
```

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

```
mysql> show tables;
+----------------+
| Tables_in_bikedb |
+----------------+
  | bikes         |
  | bluebikes     |
+----------------+
```

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

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:

```sql
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>
<thead>
<tr>
<th>Table Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISAM</td>
<td>MySQL’s original table handler</td>
</tr>
<tr>
<td>HEAP</td>
<td>The data for this table is only stored in memory</td>
</tr>
<tr>
<td>MyISAM</td>
<td>A binary portable table handler that has replaced ISAM</td>
</tr>
<tr>
<td>MERGE</td>
<td>A collection of MyISAM tables used as one table</td>
</tr>
<tr>
<td>BDB</td>
<td>Transaction-safe tables with page locking</td>
</tr>
<tr>
<td>InnoDB</td>
<td>Transaction-safe tables with row locking</td>
</tr>
</tbody>
</table>

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:

  ```sql
  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.)

<table>
<thead>
<tr>
<th>Action Syntax</th>
<th>Action Performed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>add [column] column_declaration</code> [first</td>
<td>after column_name]</td>
</tr>
<tr>
<td><code>alter [column] column_name</code> {set default literal</td>
<td>drop default}</td>
</tr>
<tr>
<td><code>change [column] column_name column_declaration</code></td>
<td>Modify column declaration with renaming of column</td>
</tr>
<tr>
<td><code>modify [column] column_declaration</code></td>
<td>Modify column declaration without renaming column</td>
</tr>
<tr>
<td><code>drop [column] column_name</code></td>
<td>Drop a column and all data contained within it.</td>
</tr>
<tr>
<td><code>rename [as] new_table_name</code></td>
<td>Rename a table</td>
</tr>
<tr>
<td><code>table_options</code></td>
<td>Change the table options</td>
</tr>
</tbody>
</table>

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.

```
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.
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.

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.

```sql
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   | NULL  |       |       |          |
| size           | int(2) | YES    | NULL   | NULL  |       |       |          |
| color          | varchar<15> | YES  | NULL   | NULL  |       |       |          |
| cost           | int(8) | YES    | NULL   | NULL  |       |       |          |
| purchased      | date   | YES    | NULL   | NULL  |       |       |          |
| mileage        | int(6) | YES    | NULL   | NULL  |       |       |          |
| races_ridden   | int(3) | YES    | NULL   | NULL  |       |       |          |
+----------------+-------+---------+---------+-------+-------+-------+----------+
8 rows in set (0.00 sec)

mysql>
```

More complicated alter table command.

Bikes table after the alteration.
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.
Examples: Inserting Data Into A Table

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

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;
+--------------------+------------+----------+--------+----------+---------+
<table>
<thead>
<tr>
<th>bikename</th>
<th>size</th>
<th>color</th>
<th>cost</th>
<th>purchased</th>
<th>mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battaglin Carrera</td>
<td>60</td>
<td>red/white</td>
<td>4000</td>
<td>2001-03-10</td>
<td>11200</td>
</tr>
<tr>
<td>Bianchi Corse Evo 4</td>
<td>58</td>
<td>celeste</td>
<td>5700</td>
<td>2004-12-02</td>
<td>300</td>
</tr>
<tr>
<td>Bianchi Evolution 3</td>
<td>58</td>
<td>celeste</td>
<td>4800</td>
<td>2003-11-12</td>
<td>2000</td>
</tr>
<tr>
<td>Bianchi Infinito</td>
<td>58</td>
<td>celeste</td>
<td>8900</td>
<td>2011-07-14</td>
<td>0</td>
</tr>
<tr>
<td>BMC SLCD - Swiss</td>
<td>58</td>
<td>red/white</td>
<td>8000</td>
<td>2010-06-23</td>
<td>0</td>
</tr>
<tr>
<td>Colnago Dream Rabobank</td>
<td>60</td>
<td>blue/orange</td>
<td>5500</td>
<td>2002-07-07</td>
<td>4300</td>
</tr>
<tr>
<td>Colnago Superissimo</td>
<td>59</td>
<td>red</td>
<td>3800</td>
<td>1996-03-01</td>
<td>13000</td>
</tr>
<tr>
<td>Eddy Merckx Domo</td>
<td>58</td>
<td>blue</td>
<td>5300</td>
<td>2004-02-02</td>
<td>0</td>
</tr>
<tr>
<td>Eddy Merckx Molteni</td>
<td>58</td>
<td>orange</td>
<td>5100</td>
<td>2004-08-12</td>
<td>0</td>
</tr>
<tr>
<td>Gianni Motta Personal</td>
<td>59</td>
<td>red/green</td>
<td>4400</td>
<td>2000-05-01</td>
<td>8700</td>
</tr>
<tr>
<td>Gios Torino Super</td>
<td>58</td>
<td>blue</td>
<td>2000</td>
<td>1998-11-08</td>
<td>9000</td>
</tr>
<tr>
<td>Ridley Damocles</td>
<td>58</td>
<td>blue</td>
<td>7500</td>
<td>2008-06-27</td>
<td>0</td>
</tr>
<tr>
<td>Ridley X-Fire</td>
<td>58</td>
<td>red</td>
<td>7500</td>
<td>2011-09-01</td>
<td>0</td>
</tr>
<tr>
<td>Schwinn Paramount P14</td>
<td>60</td>
<td>blue</td>
<td>1800</td>
<td>1992-03-01</td>
<td>200</td>
</tr>
</tbody>
</table>
+--------------------+------------+----------+--------+----------+---------+
15 rows in set (0.00 sec)
```

Using Form 1 for insertion – attribute order is not important.
Using Form 2 for insertion – attribute order is important.
Examples: Inserting Data Into A Table

```sql
mysql>
mysql>
-- Creates an initially empty table just like the bikes table
mysql> create table celestebikes like bikes;
Query OK, 0 rows affected (0.11 sec)

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

-- Table creation did not place any data into the table
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;
+-----------------+--------+--------+--------+-----------------+--------+--------+
| bike name       | size   | color  | cost   | purchased       | mileage|
+-----------------+--------+--------+--------+-----------------+--------+--------+
| Bianchi Corse    | 58     | celeste| 5700   | 2004-12-02      | 300    |
| Evolution        | 58     | celeste| 4800   | 2003-11-12      | 2000   |
| Bianchi          | 58     | celeste| 8900   | 2011-07-14      | 0      |
+-----------------+--------+--------+--------+-----------------+--------+--------+
3 rows in set (0.00 sec)
```

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

Using Form 3 for insertion
Examples: Inserting Data Into A Table

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.)

- Create a new database.
- Switch to the new database.
- Define schema for the new table.
- Insert some tuples:
  - `insert into states values ('Florida', 'FL', 'Tallahassee', 18328240, 54153);`
  - `insert into states values ('New York', 'NY', 'Albany', 194909297, 54556);`
  - `insert into states values ('Indiana', 'IN', 'Indianapolis', 6376792, 35789);`
  - `insert into states values ('Maryland', 'MD', 'Annapolis', 5633597, 9975);`
- Run a simple selection query on the new table:
  - `select * from states;`
Using Scripts with MySQL (cont.)

Specify which script to execute

Results of select query at end of script.

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>
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.

![File Example](image)

- 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.

```
C:\Program Files\MySQL\MySQL Server 5.1\bin>mysqlimport -u root -vr testdb c:\states.txt
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
```

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

Table updated
Importing Data Using the `mysqlimport` Utility

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

<table>
<thead>
<tr>
<th>name</th>
<th>abbrev</th>
<th>capital</th>
<th>population</th>
<th>square_miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Florida</td>
<td>FL</td>
<td>Tallahassee</td>
<td>18328240</td>
<td>54153</td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
<td>Albany</td>
<td>194909297</td>
<td>54556</td>
</tr>
<tr>
<td>Indiana</td>
<td>IN</td>
<td>Indianapolis</td>
<td>6376792</td>
<td>35789</td>
</tr>
<tr>
<td>Maryland</td>
<td>MD</td>
<td>Annapolis</td>
<td>5633597</td>
<td>9975</td>
</tr>
</tbody>
</table>

4 rows in set <0.00 sec>

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

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

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

8 rows in set <0.00 sec>

```sql
mysql> 
```
### mysqlimport Utility Options

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>--fields-optionally-enclosed-by='char'</td>
<td>Same as –fields-enclosed-by, but delimiter is used only to enclosed string-type columns, default is none.</td>
</tr>
<tr>
<td>--fields-escaped-by='char'</td>
<td>Specify the escape character placed before special characters; default is .</td>
</tr>
<tr>
<td>--lines-terminated-by='char'</td>
<td>Specify the separator used to terminate each row of data, default is \n (newline).</td>
</tr>
<tr>
<td>-u or --user</td>
<td>Specify your username</td>
</tr>
<tr>
<td>-p or --password</td>
<td>Specify your password</td>
</tr>
<tr>
<td>-h or --host</td>
<td>Import into MySQL on the named host; default is localhost.</td>
</tr>
<tr>
<td>-s or --silent</td>
<td>Silent mode, output appears only when errors occur.</td>
</tr>
<tr>
<td>-v or --verbose</td>
<td>Verbose mode, print more commentary on action.</td>
</tr>
<tr>
<td>-? or --help</td>
<td>Print help message and exit</td>
</tr>
</tbody>
</table>
Importing Data From A File With SQL Statement \texttt{Load Data Infile}

- Using the utility \texttt{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 \texttt{Load Data Infile}.

- The \texttt{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, ...)]
```

- **LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'filename'**
  - Either allow concurrent update or block until no other clients are reading from the specified table. See page 75.

- **[REPLACE | IGNORE]**
  - Sets the characters that delimit and enclose the fields and lines in the data file. Similar to mysqlimport syntax.

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

- **[FIELDS**
  - Ignores lines at the start of the file (miss header info)

- **[TERMINATED BY 'char']**
  - Used to load only certain columns (not entire rows)
Load Data Infile Example

String fields may be enclosed by double quotes in this file. Numeric values are not enclosed in quotes.

Fields are delimited by commas and lines are terminated by newline characters (an invisible \n)

Text file containing the data to be loaded into the database table.
Load data infile statement indicating all of the parameters which describe the configuration of the input file.

States table before addition of data

States table after addition of data
Load Data Infile Example 2

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.
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).

<table>
<thead>
<tr>
<th>name</th>
<th>abbrev</th>
<th>capital</th>
<th>population</th>
<th>square_miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Florida</td>
<td>FL</td>
<td>Tallahassee</td>
<td>18328240</td>
<td>54153</td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
<td>Albany</td>
<td>194909297</td>
<td>54556</td>
</tr>
<tr>
<td>Indiana</td>
<td>IN</td>
<td>Indianapolis</td>
<td>6376792</td>
<td>35799</td>
</tr>
<tr>
<td>Maryland</td>
<td>MD</td>
<td>Annapolis</td>
<td>5635997</td>
<td>9975</td>
</tr>
<tr>
<td>California</td>
<td>CA</td>
<td>Sacramento</td>
<td>36756666</td>
<td>155973</td>
</tr>
<tr>
<td>Texas</td>
<td>TX</td>
<td>Austin</td>
<td>22118509</td>
<td>261914</td>
</tr>
<tr>
<td>South Carolina</td>
<td>SC</td>
<td>Columbia</td>
<td>4147152</td>
<td>30111</td>
</tr>
<tr>
<td>Georgia</td>
<td>GA</td>
<td>Atlanta</td>
<td>9685754</td>
<td>47224</td>
</tr>
<tr>
<td>Illinois</td>
<td>IL</td>
<td>Springfield</td>
<td>12653544</td>
<td>55593</td>
</tr>
<tr>
<td>Maine</td>
<td>ME</td>
<td>Augusta</td>
<td>1305728</td>
<td>30865</td>
</tr>
<tr>
<td>Michigan</td>
<td>MI</td>
<td>Lansing</td>
<td>10079985</td>
<td>56809</td>
</tr>
<tr>
<td>Oregon</td>
<td>OR</td>
<td>Salem</td>
<td>3559596</td>
<td>96003</td>
</tr>
<tr>
<td>Arizona</td>
<td>AZ</td>
<td>Phoenix</td>
<td>5580811</td>
<td>113642</td>
</tr>
</tbody>
</table>

13 rows in set (0.00 sec)

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).

<table>
<thead>
<tr>
<th>name</th>
<th>abbrev</th>
<th>capital</th>
<th>population</th>
<th>square_miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Florida</td>
<td>FL</td>
<td>Tallahassee</td>
<td>18328240</td>
<td>54153</td>
</tr>
<tr>
<td>New York</td>
<td>NY</td>
<td>Albany</td>
<td>194909297</td>
<td>54556</td>
</tr>
<tr>
<td>Indiana</td>
<td>IN</td>
<td>Indianapolis</td>
<td>6376792</td>
<td>35799</td>
</tr>
<tr>
<td>Maryland</td>
<td>MD</td>
<td>Annapolis</td>
<td>5635997</td>
<td>9975</td>
</tr>
<tr>
<td>California</td>
<td>CA</td>
<td>Los Angeles</td>
<td>36756666</td>
<td>155973</td>
</tr>
<tr>
<td>Texas</td>
<td>TX</td>
<td>Austin</td>
<td>22118509</td>
<td>261914</td>
</tr>
<tr>
<td>South Carolina</td>
<td>SC</td>
<td>Columbia</td>
<td>4147152</td>
<td>30111</td>
</tr>
<tr>
<td>Georgia</td>
<td>GA</td>
<td>Atlanta</td>
<td>9685754</td>
<td>47224</td>
</tr>
<tr>
<td>Illinois</td>
<td>IL</td>
<td>Springfield</td>
<td>12653544</td>
<td>55593</td>
</tr>
<tr>
<td>Maine</td>
<td>ME</td>
<td>Augusta</td>
<td>1305728</td>
<td>30865</td>
</tr>
<tr>
<td>Michigan</td>
<td>MI</td>
<td>Lansing</td>
<td>10079985</td>
<td>56809</td>
</tr>
<tr>
<td>Oregon</td>
<td>OR</td>
<td>Salem</td>
<td>3559596</td>
<td>96003</td>
</tr>
<tr>
<td>Arizona</td>
<td>AZ</td>
<td>Phoenix</td>
<td>5580811</td>
<td>113642</td>
</tr>
</tbody>
</table>

13 rows in set (0.00 sec)
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**
  ```sql
  replace [low priority | delayed] [ignore] [into] table_name
          [set] column_name1 = expression1,
          column_name2 = expression2, ...
  ```

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

  **Form 3**
  ```sql
  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.

```sql
>>> use bidedb;
>>> select * from bluebikes;
<table>
<thead>
<tr>
<th>bikename</th>
<th>color</th>
<th>price</th>
<th>total_miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gios Torino Super</td>
<td>blue</td>
<td>2000</td>
<td>9000</td>
</tr>
<tr>
<td>Schwinn Paramount</td>
<td>blue</td>
<td>1800</td>
<td>200</td>
</tr>
</tbody>
</table>
2 rows in set (0.00 sec)

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

>>> select * from bluebikes;
<table>
<thead>
<tr>
<th>bikename</th>
<th>color</th>
<th>price</th>
<th>total_miles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gios Torino Super</td>
<td>blue</td>
<td>4200</td>
<td>11000</td>
</tr>
<tr>
<td>Schwinn Paramount</td>
<td>blue</td>
<td>1800</td>
<td>200</td>
</tr>
</tbody>
</table>
2 rows in set (0.00 sec)
```
Using `Replace` (cont.)

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.)

Global update within the relation. All tuples have their price field increased by 5%.
Using `update` (cont.)

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][SQ_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.
Table 4.1: Details about the mysql Database

The mysql database contains user information about the various users. See page 92.

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

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

Details on user privileges. See page 91.

The output of `show databases` command shows the databases available in the mysql database.
### Contents of the `user` Table

```sql
mysql> use mysql;
Database changed
mysql> describe user;

+-----------------+-------------------+-------+-------+---------+-------+
| Field           | Type              | Null | Key   | Default | Extra |
|-----------------+-------------------+-------+-------+---------+-------+
| Host            | varchar(60)       | P     | PRI   | N       | N     |
| User            | varchar(16)       | P     | PRI   | N       | N     |
| Password        | varchar(41)       | P     | PRI   | N       | N     |
| Select_priv     | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Insert_priv     | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Update_priv     | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Delete_priv     | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Create_priv     | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Drop_priv       | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Reload_priv     | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Shutdown_priv   | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Process_priv    | enum( 'N', 'Y')   | N     | N     | N       | N     |
| File_priv       | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Grant_priv      | enum( 'N', 'Y')   | N     | N     | N       | N     |
| References_priv | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Index_priv      | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Alter_priv      | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Show_db_priv    | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Super_priv      | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Create_tmp_table_priv | enum( 'N', 'Y') | N   | N     | N       | N     |
| Lock_tables_priv| enum( 'N', 'Y')   | N     | N     | N       | N     |
| Execute_priv    | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Repl_slave_priv | enum( 'N', 'Y')   | N     | N     | N       | N     |
| Repl_client_priv| enum( 'N', 'Y')   | N     | N     | N       | N     |
| ssl_type        | enum( '', 'ANY', 'X509', 'SPECIFIED') | N | N | N | N |
| ssl_cipher      | blob              | N     | N     | N       | N     |
| x509_issuer     | blob              | N     | N     | N       | N     |
| x509_subject    | blob              | N     | N     | N       | N     |
| max_questions   | int(11) unsigned  | N     | N     | N       | N     |
| max_updates     | int(11) unsigned  | N     | N     | N       | N     |
| max_connections | int(11) unsigned  | N     | N     | N       | N     |
+-----------------+-------------------+-------+-------+---------+-------+
31 rows in set (0.00 sec)
```
Contents of the user_info Table

```sql
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

```sql
mysql> \
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') |
| Column_priv| set('Select','Insert','Update','References') |
+----------+----------------+
8 rows in set (0.00 sec)
```

```sql
mysql> \
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>YES</td>
<td>PRI</td>
<td>PRI PRI PRI PRI MUL</td>
<td>CURRENT_TIMESTAMP</td>
</tr>
<tr>
<td>Db</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table_name</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grantor</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table_priv</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column_priv</td>
<td></td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Contents of the db Table

```sql
mysql> describe db;
+----------------+------------+------+-----+----------------+-------+
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>char(60)</td>
<td>NO</td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Db</td>
<td>char(64)</td>
<td>NO</td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>char(16)</td>
<td>NO</td>
<td>PRI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Insert_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Update_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>DeletePriv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Create_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Drop_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Grant_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>References_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Index_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Alter_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Create_tmp_table_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Lock_tables_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Create_view_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Show_view_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Create_routine_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Alter_routine_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Execute_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Event_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
<tr>
<td>Trigger_priv</td>
<td>enum('N','Y')</td>
<td>NO</td>
<td></td>
<td></td>
<td>N</td>
</tr>
</tbody>
</table>
+----------------+------------+------|-----+----------------+-------+
```

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 user 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

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Operations Permitted</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL or ALL PRIVILEGES</td>
<td>All privileges except for GRANT</td>
</tr>
<tr>
<td>ALTER</td>
<td>Change a table definition using ALTER TABLE excluding the creation and dropping of indices.</td>
</tr>
<tr>
<td>CREATE</td>
<td>Create database or tables within a database.</td>
</tr>
<tr>
<td>CREATE TEMPORARY TABLES</td>
<td>Create temporary tables.</td>
</tr>
<tr>
<td>DELETE</td>
<td>Ability to perform deletions from tables. (Delete DML statements).</td>
</tr>
<tr>
<td>DROP</td>
<td>Ability to drop databases or tables.</td>
</tr>
<tr>
<td>INSERT</td>
<td>Ability to insert data into tables.</td>
</tr>
<tr>
<td>SHUTDOWN</td>
<td>Ability to shutdown the MySQL server.</td>
</tr>
</tbody>
</table>
Displaying Privileges with \textbf{SHOW}

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

  \texttt{SHOW GRANTS FOR username@hostname}

- An example is shown below:

```sql
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 \texttt{REVOKE}

- Revocation of privileges in MySQL is accomplished with the \texttt{revoke} command.

- The format of the \texttt{revoke} command is:

  \begin{verbatim}
  REVOKE privileges [(column_list)]
  ON database_name.table_name
  FROM username@hostname
  \end{verbatim}

- An example is shown on the next page.
Example - Revoking User Privileges with `REVOKE`

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.
More Details On The MySQL Workbench

• The Workbench contains a fairly extensive set of administrator tools for maintaining your MySQL Server instances.

• The following slides illustrate some of these features. I’d encourage you to play around with the Workbench and get familiar with using it.
Starting and Stopping the Server

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

The database server instance is running. Stop Server

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

Startup Message Log

- 2014-02-11 08:39:48 - Workbench will use cmd shell commands to start/stop this instance
- 2014-02-11 08:39:48 - Status check of service 'MySQL56' returned running
Server log files for details on server status.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Thread</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014-02-11 08:33:05</td>
<td>9236</td>
<td>Note</td>
<td>InnoDB: 5.6.16 started; log sequence number 21621696</td>
</tr>
<tr>
<td>2014-02-11 08:33:05</td>
<td>9236</td>
<td>Note</td>
<td>Server hostname (bind-address): &quot;&quot;: port: 3310</td>
</tr>
<tr>
<td>2014-02-11 08:33:05</td>
<td>9236</td>
<td>Note</td>
<td>IPv6 is available.</td>
</tr>
<tr>
<td>2014-02-11 08:33:05</td>
<td>9236</td>
<td>Note</td>
<td>- &quot;:&quot; resolves to &quot;:&quot;;</td>
</tr>
<tr>
<td>2014-02-11 08:33:05</td>
<td>9236</td>
<td>Note</td>
<td>Server socket created on IP: &quot;:&quot;.</td>
</tr>
<tr>
<td>2014-02-11 08:33:05</td>
<td>9236</td>
<td>Note</td>
<td>C:\Program Files\MySQL\MySQL Server 5.6\bin\mysqld.exe: ready for connection Version: '5.6.16' socket: &quot;&quot;: port: 3310 MySQL Community Server (GPL) plugin 'FEDERATED' is disabled.</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>option 'innodb-autotest-increment': unsigned value 67108864 adjusted to 10</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Using atoms to ref count buffer pool pages</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: The InnoDB memory heap is disabled</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Mutexes and rw_locks use Windows interlocked functions</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Compressed tables use zlib 1.2.3</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Initializing buffer pool, size = 740.0M</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Completed initialization of buffer pool</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Highest supported file format is Barracuda</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Initializing buffer pool, size = 740.0M</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Completed initialization of buffer pool</td>
</tr>
<tr>
<td>2014-02-11 08:33:12</td>
<td>4904</td>
<td>Note</td>
<td>InnoDB: Highest supported file format is Barracuda</td>
</tr>
</tbody>
</table>

Log File Location: C:\ProgramData\MySQL\MySQL Server 5.6\data\HEC-236-G4TB8Y1.err
Log File Size: 54.4 kB
Showing: 703 records starting at byte offset 0
Client connections to the server.

<table>
<thead>
<tr>
<th>Id</th>
<th>User</th>
<th>Host</th>
<th>DB</th>
<th>Command</th>
<th>Time</th>
<th>State</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>root</td>
<td>localhost:64695</td>
<td>bikedb</td>
<td>Sleep</td>
<td>294</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>3</td>
<td>root</td>
<td>localhost:64696</td>
<td>bikedb</td>
<td>Sleep</td>
<td>294</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>4</td>
<td>root</td>
<td>localhost:64704</td>
<td>None</td>
<td>Sleep</td>
<td>246</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>6</td>
<td>root</td>
<td>localhost:64716</td>
<td>bikedb</td>
<td>Sleep</td>
<td>115</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>7</td>
<td>root</td>
<td>localhost:64717</td>
<td>bikedb</td>
<td>Sleep</td>
<td>116</td>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td>9</td>
<td>root</td>
<td>localhost:64719</td>
<td>None</td>
<td>Query</td>
<td>0 init</td>
<td>NULL</td>
<td>SHOW FULL PROCESLIST</td>
</tr>
<tr>
<td>10</td>
<td>root</td>
<td>localhost:64720</td>
<td>None</td>
<td>Sleep</td>
<td>3</td>
<td></td>
<td>NULL</td>
</tr>
</tbody>
</table>
User privileges
Options file for fine tuning server behavior.
An EER diagram reverse engineered from a MySQL database. To start this process, select Database, then Reverse Engineer.
Installing MySQL 5.6.17 (cont.)

Update Products

The MySQL Installer guides you through the installation and configuration of your MySQL products. Run it from the Start Menu to perform maintenance tasks later.

Select one of the actions below:

- **Install Updates**
  Select this action item to update the installed MySQL products on this machine to the latest version.

- **Add / Modify Products and Features**
  Add additional MySQL products or make changes to the features of already installed products.

- **Remove MySQL Products**
  Remove individual MySQL products from your machine. Also select this item to remove all MySQL products.
Installing MySQL 5.6.17 (cont.)

Before continuing, the Installer will determine if updates are available for the products you are about to install or upgrade.

- Connect to the Internet
- Fetch product update information

Product upgrades
- Upgrade MySQL Server 5.6.16 to MySQL Server 5.6.17
- Upgrade MySQL For Excel 1.1.1 to MySQL For Excel 1.2.0

Skip the check for updates (not recommended)
Installing MySQL 5.6.17 (cont.)

Before continuing, the Installer will determine if updates are available for the products you are about to install or upgrade.

- Connect to the Internet
- Fetch product update information

Product upgrades:
- Upgrade MySQL Server 5.6.16 to MySQL Server 5.6.17
- Upgrade MySQL Workbench CE 6.0.9 to MySQL Workbench CE 6.1.4
- Upgrade MySQL For Excel 1.1.1 to MySQL For Excel 1.2.1
- Upgrade Connector/J 5.1.29 to Connector/J 5.1.30

Skip the check for updates (not recommended)
Installing MySQL 5.6.17 (cont.)

The following products will be installed or updated:

<table>
<thead>
<tr>
<th>Product</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL Server 5.6.17</td>
<td>To be upgraded</td>
</tr>
<tr>
<td>MySQL Workbench CE 6.1.4</td>
<td>To be upgraded</td>
</tr>
<tr>
<td>MySQL For Excel 1.2.1</td>
<td>To be upgraded</td>
</tr>
<tr>
<td>Connector/J 5.1.30</td>
<td>To be upgraded</td>
</tr>
</tbody>
</table>

Click [Execute] to install or update the following packages.
Installing MySQL 5.6.17 (cont.)

The following products will be installed or updated:

- MySQL Server 5.6.17: Install success
- MySQL Workbench CE 6.1.4: Install success
- MySQL For Excel 1.2.1: Install error
- Connector/J 5.1.30: Install success
Installing MySQL 5.6.17 (cont.)

The following products will now be configured:

<table>
<thead>
<tr>
<th>Product</th>
<th>Action to be performed</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL Server 5.6.17</td>
<td>Initial Configuration.</td>
<td></td>
</tr>
</tbody>
</table>

Show Details >
Installing MySQL 5.6.17 (cont.)

In order to maintain data integrity following a server upgrade, it's important that we check your database and upgrade it, if necessary.

The check and upgrade process needs to be performed using the MySQL root user account which was created when the MySQL server was initially installed. Please enter:

User: root@localhost
Password: [blank]

Skip database upgrade check and process. (Not recommended)
Installing MySQL 5.6.17 (cont.)

The installation procedure has been completed.

- [ ] Copy Log to Clipboard
- [x] Start MySQL Workbench after Setup

Finish Cancel
Installing MySQL 5.6.17 (cont.)