Table of Contents

For this class you should be working with a local installation of MySQL I am assuming a default installation of MySQL 5.5 or higher with InnoDB tables and strict mode. I assume that you are not trying to do your assignments and experiments on a production server; the things that I ask you to do in this class should not cause problems but student do occasionally make mistakes that could cause problems and you do not want to do that on your job.

You will also need clients for working with the database. The client provides ways to let the user interact with the database engine. A client will let you create a database, create tables, insert and modify data and run queries.

Generally you will have both a command line interface and a GUI interface. You should be comfortable with both.

One of the advantages to having a local installation of the dbms and having root access is that you can create databases for different purposes. I will ask you to create several databases over the semester. You may want to also create the tennis database that is used in the van Lans book.

Many of the demos assume that you have set up the databases mentioned and have created the zoo table.

You also need to have a text editor and know how to open a command window on your computer.

1. User accounts and databases

When you install MySQL you will have a root account and password-which you created during installation. You would use this account for administrative purposes only. This class does not deal with administrative tasks. For the things you do in this class, you should create a regular user account that you will use routinely. It is not a good idea to use an administrative account for this class.

You could use the admin account to create the databases for this class and then create a user with privileges on those databases. Create the database a_testbed and the database a_vets. I am going to use the lower case version of the database name since case of the database name and the table names can be an issue with some operating systems. If you want to make all the databases for class now, make the following databases. The names of these databases all start with the characters 'a_' as an organization convention (databases for my 155A class start with 'a_' and databases for my 155P class start with 'p_'.

- `a_testbed` we will use this for miscellaneous tables for experimenting
- `a_vets` we use this for the first few demos and assignments
- `a_emp` this and the next two databases are used for most of the demos
- `a_oe`
- `a_prd`
- `a_bkinfo` this and the following are used for later assignments
- `a_bkorders`
- `a_plants`
- `a_xml` this is used at the end of the semester for the xml unit

Demo 01: The command to create a database is `CREATE DATABASE` followed by the database name. You use a semicolon to tell the dbms to execute the command.

```
Create database a_testbed;
```

A dba would need to take more things into considerations- but this version is sufficient for our databases.

Demo 02: This is the command to create a user named cls_demo with a password of demo_pss.

```
Create user 'cls_demo'@'localhost' identified by 'demo_pss';
```

You can also create a user who does not need a password by omitting the "identified by" clause. A user without a password would not be a good idea on a production system, but it can be OK for the databases we use in class. The @'localhost' phrase assumes that your installation of MySQL is on your local computer. Pay attention to the quotation marks in these commands.

Demo 03: The command to let that user work with the a_testbed database.

```
Grant all privileges on a_testbed.* to 'cls_demo'@'localhost' ;
```

Compare this to the grant statement the van Lans uses. First, he grants all privileges to the user BOOKSQL on all of the databases and all of their objects (On *.*); my command limits this user to one specific database. Second, the author includes the clause "With Grant Option" which lets this user grant privileges to other users. When you create a general purpose user for the class, use the author's grant syntax so that your general purpose user has full privileges. I am going to be a more cautious administrator and give the user the rights to work with a specific database. The root account can add additional database privileges to a user later.

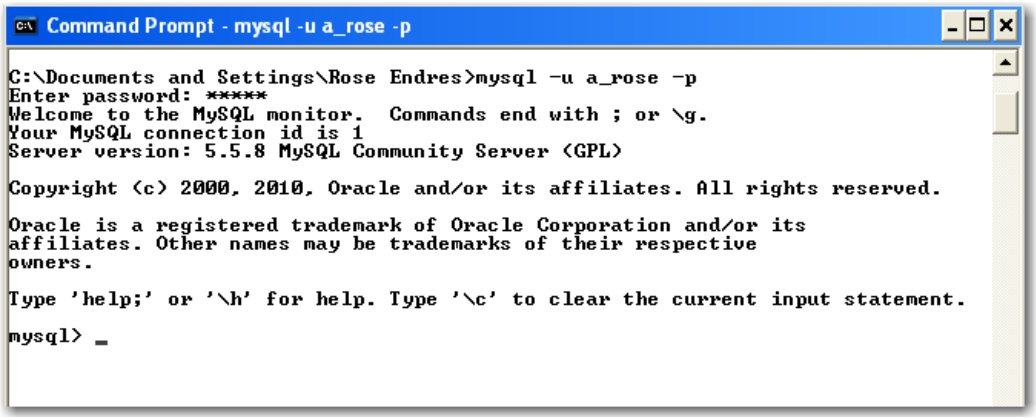
You can create all the databases first and then grant privileges on those databases to the user you created. I will supply sql for the various tables in these databases as the semester progresses.

2. Using the mysql command line client

This is a command line client. On a windows machine, where MySQL is properly installed, you can start the client from a C:> prompt using the Command Prompt program. (on a windows machine this is probably in All Programs→ Accessories. You can then pin this to your start menu to make it easier to find. If you right click the title bar for this window you can find options for changing the color scheme etc.)

Start up the client with the command mysql and the command option –u followed by your user name and -p to be prompted for the password for your account. My account user name is a_rose.

```
mysql –u a_rose –p
```



You can give some commands without being in any database.

The user a_rose can see several databases. In addition to the output of the command itself- which is a table display, you also get the row count and the time it took to run the query.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| a_bkinfo           |
| a_bkorders         |
| a_emp              |
| a_oe               |
| a_prd              |
| a_testbed          |
| a_vets             |
| a_xml              |
+--------------------+
9 rows in set (0.00 sec)
```

If I log in as another user and give the show databases command, that user can see only the databases he was given privileges to and information_schema.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| books              |
| vets               |
+--------------------+
3 rows in set (0.00 sec)
```

The rest of this document uses my regular ' a_rose' account.

```
mysql> select version(), current_date;
+-----------+--------------+
| version() | current_date |
+-----------+--------------+
| 5.5.8     | 2012-07-04   |
+-----------+--------------+
1 row in set (0.00 sec)
```

In addition to the output of the command itself- which is a table display, you also get the row count and the time it took to run the query.

2.1. Command terminators

So far we have been using the semicolon to terminate a command and tell the client to run that command. You can also use \G to end the command and get the data displayed vertically. This is useful for displays which have very few rows but the column values might be long. It is not particular helpful for most table displays. (This is an upper case G.)

```
mysql> select version(), current_date\G
*************************** 1. row ***************************
   version(): 5.5.8
current_date: 2012-07-04
1 row in set (0.00 sec)
```

You can also use a lower case \g as a command terminator

```
mysql> select version(), current_date\g
+-----------+--------------+
| version() | current_date |
+-----------+--------------+
| 5.5.8     | 2012-07-04   |
+-----------+--------------+
1 row in set (0.00 sec)
```

2.2. Entering queries

You can select a literal or expressions.

```
mysql> select 'Hello World', 60*60*24;
+-------------+----------+
| Hello World | 60*60*24 |
+-------------+----------+
| Hello World |    86400 |
+-------------+----------+
1 row in set (0.00 sec)
```

You can put more than one query on a line and they execute one after the other. This also shows column aliases;

```
mysql> select 'Hello' as greeting; select now() as "clock time";
+----------+
| greeting |
+----------+
| Hello    |
+----------+
1 row in set (0.00 sec)

+---------------------+
| clock time          |
+---------------------+
| 2012-07-04 21:55:47 |
+---------------------+
1 row in set (0.00 sec)
```

You can enter a single command on multiple physical lines. You get more secondary prompts until you give the semicolon delimiter followed by the Enter key.

```
mysql> select 'Hello' as Greeting1,
    -> 'Good morning' as Greeting2
    -> ,
    -> now()
    -> ;
+-----------+--------------+---------------------+
| Greeting1 | Greeting2    | now()               |
+-----------+--------------+---------------------+
| Hello     | Good morning | 2012-07-04 21:56:49 |
+-----------+--------------+---------------------+
1 row in set (0.00 sec)
```

2.3. Quitting a command

Sometimes you are entering an SQL command and decide that you do not want to run it. Perhaps you realize that the syntax is wrong or that you selected the wrong data to work with. You can use \c to quit a command if you decide not to run it. If you have an open quote delimiter, you will need to close the delimiter before using the \c command.

For example: Suppose you are trying to run one of the previous demos and accidentally enter the wrong quotes. Here I am opening with a single quote but trying to end the literal with a double quote.

```
mysql> select 'Hello World", 60 * 60 *24;
    '>
```

Note that the prompt is now a quote followed by >. This is the client's way of trying to tell you that you have not closed the delimiter. If you try to quit this by entering \c, you get another prompt.

```
mysql> select 'Hello World", 60 * 60 *24;
    '> \c
    '>
```

What you need to do in this case is close the literal with a single quotes and then use the \c command.

```
mysql> select 'Hello World", 60 * 60 *24;
    '> \c
    '> '\c
mysql>
```

2.4. Secondary prompts

The command line interface has a series of secondary prompts with the following meanings.

mysql> This is the prompt to enter a new command

-> This is the prompt for the next line in a multi-line command

The following will probably happen to you. You type in the leading ' for a string and do not type the ending quote. Note that the prompt changed somewhat in an attempt to let you know that it is waiting for the ending quote. The carriage return you entered is in the string literal. You can enter a closing quote and the terminator (and get error messages on the sql) or terminate the command.

If you open a delimiter and do not close it you will get a following prompt to indicate the delimiter needed to close the literal.

`'>` waiting for the completion of a string with a single quote

`">` waiting for the completion of a string with a double quote

`'>` waiting for the completion of a string with a back tick quote

`/*>` waiting for the completion of an extended comment

You can enter a literal that includes a new line character. This is not terribly useful but it is interesting.

```
mysql> select 'Hello
    '> World', 24 *7;
+-------------+-------+
| Hello
World | 24 *7 |
+-------------+-------+
| Hello
World |   168 |
+-------------+-------+
```

2.5. Warnings

Sometimes you can enter a query that runs but MySQL has a warning about the query. The following query tries to multiply the number 5 by a string value. MySQL will run this but if you look at the output, the product is 0 and the feedback line says there is a warning.

```
mysql> Select 5 * 'cat';
+-----------+
| 5 * 'cat' |
+-----------+
|         0 |
```

```
+-----------+
1 row in set, 1 warning (0.00 sec)
```

You can give the command `show warnings` to see the warning message.

```
mysql> show warnings;
+---------+------+----------------------------------------+
| Level   | Code | Message                                |
+---------+------+----------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'cat' |
+---------+------+----------------------------------------+
1 row in set (0.00 sec)
```

You could also use the \W command to signal that you want all of the warning messages displayed without having to ask for them. To turn this option off use \w with a lower case w.

```
mysql> \W
Show warnings enabled.
mysql> Select 5 * 'cat';
+-----------+
| 5 * 'cat' |
+-----------+
|         0 |
+-----------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Truncated incorrect DOUBLE value: 'cat'
```

3. Some useful commands to get started.

3.1. What databases do we have access to

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| a_bkinfo           |
| a_bkorders         |
| a_emp              |
| a_oe               |
| a_prd              |
| a_testbed          |
| a_vets             |
| a_xml              |
+--------------------+
```

If you see the database mysql, don't work with it directly. You may have a test database that was installed when you installed mysql. The information_schema database is a system db. The other databases are user created databases.

3.2. What is the current database?

database() is a function and we are asking to see the return value of that function. If you have just logged in and have not selected any database, then the return value is NULL. Be careful that you do *not* have a space between the function name and the opening parentheses.

```
mysql> select database();
+------------+
| database() |
+------------+
| NULL       |
+------------+
```

3.3. Select a database to use

Generally you want to do your work in the context of a specific database. You can set the database with the Use command.

```
mysql> use a_testbed;
Database changed
mysql> select database();
+--------------+
| database()   |
+--------------+
| a_testbed    |
+--------------+
```