Lab Week 8: Learning MySQL

In a Relational Database data are stored in tables (a.k.a. relations). A table stores information in rows (records) and columns (fields).

Relational databases include a Structured Query Language (SQL), which is an application programming interface (API).

Developed in the 1970s relational databases are based on an underlying mathematical model, have great expressive power and are a core computer science technology that has migrated to all fields.

The RDBMS market is dominated by Oracle, IBM DB2, and Microsoft SQL Server, which account for 85% of worldwide RDBMS revenue.

A more mature technology than XML, RDBMS' support data-intensive applications such as large websites (Facebook, Yahoo, WalMart, Sears, Amazon, . . .).
What is MySQL?

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

The MySQL Website (http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html) provides this information about MySQL:

MySQL is a relational database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. This adds speed and flexibility.

The SQL part of “MySQL” stands for “Structured Query Language.” SQL is the most common standardized language used to access databases.

MySQL software is Open Source.

Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs.

If you need to embed MySQL code into a commercial application, you can buy a commercially licensed version from Oracle.

MySQL is named after co-founder Michael Widenius' daughter, My.
Connecting to the MySQL Server

Creating MySQL databases and tables requires communicating with a MySQL server.

Our CIS 110 MySQL server is running on ix.cs.uoregon.edu.

Each 110 student has a MySQL account and a MySQL database on the server.

1. Your MySQL Account

   Your MySQL username:  
   *yourDuckID*

   Your MySQL Password:  
   the last six digits of your UO ID

   Your MySQL database name:  
   *yourDuckID*

2. Connecting to MySQL

   MySQL Terminal is a Unix command-line interface for interacting with the MySQL server

   You will run MySQL Terminal from the shell.uoregon.edu command line:

   3. Use SSH to connect to shell.uoregon.edu.

4. Login to the MySQL Server.

   At the Unix command line, enter this command:

   ```
   mysql -u DuckID -h ix.cs.uoregon.edu --port=3305  -p
   ```

   You will be prompted for your MySQL password.

   Enter the last six digits of your UO ID.
Learning MySQL

A database named yourDuckID has already been created for you.

You will create a Nations table (shown in ch. 15 of FIT5) in your database and load it with information shown in the textbook.

Then you can answer different sorts of questions about Nations by retrieving data from the tables. This section shows you how to perform the following operations:

• Use a database
• Create a table
• Load data into the table
• Retrieve data from the table in various ways

A command consists of an SQL statement followed by a semicolon (;).

Commands may be entered in any letter-case. The following queries are equivalent:

```
SELECT VERSION(), CURRENT_DATE;
select version(), current_date;
Select vErSiOn(), current_DATE;
```

Follow along with your lab instructor to learn how to use basic MySQL commands.

If you make a spelling mistake when typing a command, you will see a MySQL error message.

You can use the up-arrow on your keyboard to retrieve the command, and then edit the command by using the left- and right-arrow keys.

These exercises were adapted from the MySQL Reference Manual Tutorial.

1. Important First Step: Select your Database with the USE database command.

The USE database statement tells MySQL to use database as the default (current) database for subsequent statements.

database remains the default until the end of the session.

The name of your database is your DuckID. So, for example, the command that Magnus McDonald would use to select his database would be:

```
USE mcdonald7;
```
You will substitute your actual DuckID in the above USE command.

2. Create a Table.

At this point your database is empty, as SHOW TABLES tells you:

```
mysql> show tables;
Empty set (0.01 sec)
```

*If you forget the semicolon*, the prompt changes from `mysql>` to `->`.

This is how `mysql` indicates that it has not yet seen a complete statement and is waiting for the rest. Just type the semicolon and hit Enter.

Now, create a new folder (110/SQL/) on your computer and download the Nations.sql file to 110/SQL/.

Right-click this link and choose Save Link As: [http://bit.ly/Q7fVLi](http://bit.ly/Q7fVLi)

Then copy the Create Table command from Nations.sql and paste it into the MySQL command line.

Once you have created a table, SHOW TABLES should produce some output:

```
mysql> show tables;
+---------------------+
<table>
<thead>
<tr>
<th>Tables_in_susanQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nations</td>
</tr>
</tbody>
</table>
+---------------------+
```

To verify that your table was created the way you expected, use a DESCRIBE statement:

```
mysql> describe Nations;
```

3. Inserting Data into a Table.

After creating your table, you need to populate it, using the INSERT statement.

Copy the five INSERT INTO commands from Nations.sql and paste them into the MySQL command line.

Then create your own INSERT INTO command using the following data from Fig. 15.7 of FIT5:
4. **Retrieving Information**: Selecting All Data.

The simplest form of SELECT retrieves everything from a table:

```sql
mysql> SELECT * FROM Nations;
```

5. **Retrieving Information**: Selecting Particular Rows.

```sql
SELECT * FROM Nations WHERE Interest = 'Beach';
```

Note that strings are usually quoted with single quotes (`'`).

```sql
SELECT * FROM Nations
WHERE (N_S = 'S' AND Interest = 'Beach')
```


```sql
SELECT Name  FROM Nations;
SELECT Name, Domain FROM Nations
SELECT Name, Domain FROM Nations
WHERE Interest = 'Kabuki';
```

7. **Retrieving Information**: Sorting Rows

You may have noticed in the preceding examples that the result rows are displayed in no particular order. It is often easier to examine query output when the rows are sorted in some meaningful way. To sort a result, use an ORDER BY clause.

```sql
SELECT * FROM Nations ORDER BY Name;
SELECT Name, Interest FROM Nations
ORDER BY Interest DESC;
```

8. **Quitting MySQL**.

```sql
mysql> exit;
```

If time remains in your lab, start working on the Post Lab Exercises.
After your Lab: More MySQL

Complete the following tasks after your lab:

1. **Insert a new record into your Nations table.**

   Use the INSERT INTO command to insert the data shown in figure 15.7, p. 475, and figure 15.10, p. 478.

   Here’s how to do it:

   a) create a new folder, 110/SQL/

   b) In Aptana, right-click this new SQL folder and choose File > New > File. Name the new file, `post-lab.sql`.

   c) Use Apatana to create your INSERT INTO command in `post-lab.sql`.

   d) Login to mysql. See step 4, *Connecting to the MySQL Server*, above.

   Select your database with the USE command.

   e) Copy the INSERT INTO command from `post-lab.sql`, and paste it into MySQL.

2. **Pattern Matching.**

   MySQL pattern matching enables you to use “_” to match any single character and “%” to match an arbitrary number of characters (including zero characters).

   In MySQL, patterns are case-insensitive by default. Some examples are shown here. You do not use = or <> when you use SQL patterns; use the LIKE or NOT LIKE comparison operators instead.

   To find names beginning with “C”:
SELECT * FROM Nations WHERE Name LIKE 'C%';

To find names ending with “land”:

SELECT * FROM Nations WHERE Name LIKE '%land';

To find names containing a “w”:

SELECT * FROM Nations WHERE Name LIKE '%w%';

To find names containing exactly five characters, use five instances of the “_” pattern character:

SELECT * FROM pet WHERE name LIKE '_____';

3. Counting Rows.

Databases are often used to answer the question, “How often does a certain type of data occur in a table?” For example, you might want to know how many nations are in the table.

COUNT(*) counts the number of rows, so the query to count your Nations looks like this:

SELECT COUNT(*) FROM Nations;

You can use COUNT() if you want to find out how many countries have history as an interest:

SELECT COUNT(*) FROM Nations
WHERE Interest = 'History';


Write an SQL query to retrieve the following information: All nations with Beach as an interest.

Write an SQL query to retrieve the following information: All nations in the northern hemisphere with Beach as an interest.

5. Create an aliens_abduction Table in your Database.

Download aliens_abduction.sql to 110/SQL/, following these steps:

In Aptana, right-click your SQL folder and choose New > File. Name the file aliens_abduction.sql.


Login to mysql. See step 4, Connecting to the MySQL Server, above.

Copy the Create Table command from aliens_abduction.sql
and paste it into MySQL.

Copy and paste the INSERT INTO commands from aliens_abduction.sql and paste them into MySQL.

Run a few SELECT queries in MySQL to make sure the aliens_abduction table is correct.

(Credits: aliens_abduction.sql is from Head First PHP and MySQL, by O'Reilly Publishing.)

6. **Insert a new record into the aliens_abduction table.**

   In Aptana, open the file aliens_abduction.sql.

   Create a new INSERT INTO command with your own data, as follows:

   a) Copy one of the INSERT INTO commands, and paste it at the end of the file. Modify it to make a new INSERT INTO command with your data. Do not use your actual email address in the command.

   If you are not logged into MySQL, do so now. See step 4, Connecting to the MySQL Server, above.

   Copy the new INSERT INTO command, and paste it into MySQL.

   Run a SELECT query to verify that it succeeded.

7. **The UPDATE command** is used to update existing data in database tables. Use this command when you want to change the value of a record in a table.

   UPDATE Nations SET Interest = 'Reggae'
   WHERE Name = 'Jamaica';

8. **The DELETE command** is used to remove records from a table:

   DELETE FROM Nations
   WHERE Name = 'Jamaica';

9. **The DROP TABLE Command.**

   This command is documented here for completeness only. Do not use this command unless you totally mangle a table and want to start over,

   DROP TABLE removes one or more tables. All table data and the table definition are deleted from your database.

   DROP TABLE aliens_abduction;