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.

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.
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: 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=3475 -p`
   - You will be prompted for your MySQL password.
   - Enter the nine 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.

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.
   - 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.

Next, let’s create our first table. Click this link: http://www.cs.uoregon.edu/Classes/13W/cis110/cis110_13w/Nations.sql
and copy the Create Table command from Nations.sql and paste it into MySQL command line.

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

```
mysql> show tables;
+-------------------+
| Tables_in_susanQ  |
+-------------------+
| Nations           |
+-------------------+
```
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.

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.

4. Retrieving Information: Selecting All Data.
The simplest form of SELECT retrieves everything from a table:

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

5. Retrieving Information: Selecting Particular Rows.
```
SELECT * FROM Nations WHERE Interest = 'Beach';
```
Note that strings are usually quoted with single quotes (').
```
SELECT * FROM Nations WHERE (N_S = 'S' AND Interest = 'Beach');
```

```
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.

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

8. Quitting MySQL.
```

mysql> exit;