Using a MySQL Database

Professor Don Colton, BYU Hawaii

March 19, 2003

In this course, we will learn to use an SQL database system called MySQL. The programming interface we will learn also applies to other database engines, such as Oracle. We are using MySQL because it runs fast and is essentially free, so you can run it on your own machine at home if you desire.

1 Working by Hand

There are two main ways to work with a database. First, we will show how to work with it by hand. Second, we will show how to work with it from another program. In the long run, you will do most of your database work using another program that you will write, such as a CGI program. When you are debugging, or when you are doing one-time kinds of things like creating tables, working by hand can be a good solution.

1.1 Connect to MySQL

Use this command (from stu.cs.byuh.edu) to connect to MySQL.

```
  mysql -p -h HHH -u UUU
```

Replace `HHH` with the host name. Replace `UUU` with your own username. When you are prompted for a password, type it in. Your instructor should tell you your hostname, username, and password. You will change your password later.

The `-p` tells mysql to prompt you for a password.

The `-h` tells mysql to what hostname to connect to.

The `-u` tells mysql what username to use.

1.2 A Few Alerts

Pay attention to capital and lower-case letters. On some platforms big and little letters are interchangable, so "DonC" and "donc" are treated the same. This is called "case insensitive." On other platforms the big and little letters are distinguished, so "DonC" and "donc" are different. This is called "case sensitive." Be aware of the difference.

The up-arrow is your friend. Press it several times. Press the down-arrow several times. Try the other arrow keys (left and right). When you have a line looking the way you want it, you can press ENTER from anyplace in the line. You do not need to be at the end of the line. Using these keys will speed up your work.

1.3 Change Your Password

The following command allows you to change your password to something you like better. The quotes, equal sign, parentheses, and semi-colon are all required as shown.

```
  set password = password("whatever");
```

Change your password. Type `exit` to quit. Then log back in using the new password.

Avoid using any of your existing passwords. That is because this password is going to end up in your programs, right in plain text. People may see it, so you may want to make it hard to remember. I recommend you invent something totally random, like "vlksH36E." Nobody will guess that. Nobody will remember that. And you will hardly ever need to type it in.

Certain characters can cause difficulty in a password (or a table name, for that matter). Characters to avoid include space, backslash (\), at (`@`), dollar (`$`), and quote (`"`). There may be others. These characters are "metacharacters" or "escape characters" which means that they have a special meaning. They escape from the normal meaning. For instance, when we print "`\n`" we do not expect to get a backslash and an n. We expect to get a newline. The backslash is special. Now that you have been warned, I will admit that you can use these special characters, but it requires extra care. For most people it is easier to avoid them.

1.4 What Databases Exist?

See what databases exist on this "host." Use the following command:

```
show databases;
```

It should give you a list of the databases that currently exist. Your instructor will tell you which one is assigned to you.

1.5 Creating a Database

You will probably not do this, but later you may want to set up your own server and databases. Here are the commands to create a database and to grant access rights to a user.

```
create database DDD;
grant all on DDD.* to UUU
  identified by "PPP";
```

In this example, `DDD` is the name of the database. `UUU` is the username. `PPP` is the password. Apparently a user can only have one password, so if you issue the command again with a different password, the first password is replaced.

1.6 Focus on Your Database

When you arrive in mysql, the instructor will have already granted you all rights within your own database. Tell mysql to focus on your database by typing the following command:

```
use DDD
```

where "`DDD`" is replaced by the name of the database assigned to you. `mysql` should then respond with "`Database changed.`" Notice that in most commands, a semi-colon ("`;`") is required after the command. On this command the semi-colon seems to be optional.

1.7 Databases Contain Tables

You will be working with tables within your database. You can see a list of the existing tables by using this command:

```
show tables;
```

Initially there should be nothing there. You should see a message saying something like "`Empty set.`"

1.8 Create a Table

Create a table by doing something like this:

```
create table scores (
  student varchar(50),
  score int(6)
  );
```

In this example, `scores` is the name of the table. It has two columns. One is called `student` and can hold a string of up to 50 characters. The second is called `score` and can hold a number up to nine digits, with a default printing width of six digits.

You can put that command all on one line if you like, or spread it out over multiple lines like shown above. mysql will continue prompting you for the remainder of your command until you put in the semi-colon "`;`" to tell it that you are done.

Now show tables again. You should see your new table.

1.9 Enter Data into Your Table

Insert something into your table by using commands like these.

```
insert into scores values ( "Fred", 100 );
insert scores values ( "Bob", 70 );
insert into scores ( score, student )
  values ( 95, "Anne" );
insert scores set student=Don, score=75;
```

1.10 Display the Data in Your Table

See what you have in your table by using a command like this:

```
select * from scores;
```

1.11 Think Beyond the Example

Invent your own table with three or more columns. Insert into it three or more rows. Be creative. When you have your table built and populated, do a "select *" on the table and show your instructor. (Table names and column names do not allow spaces.)

2 Working by Program

The previous section told how to work with a database by hand. In this section, we show how to work with it from another program. We will illustrate using the Perl DBI (data base interface). We will not show you everything that is possible. Instead, we focus on a few simple commands that will allow you to do some interesting things.

2.1 Connect to MySQL

In your Perl program, you can use the following lines to connect to and disconnect from a MySQL database. You are telling the computer the same things you had to specify by hand (in the section above) when you connected to MySQL. The difference is that this protocol is easier for programming, and the by-hand protocol is easier for humans.

```
use DBI; # to include needed definitions
$db = "DDD";
$host = "HHH";
$username = "UUU";
$password = "PPP";
$x = DBI->connect( "DBI:mysql:$db:$host",
  $username, $password, {RaiseError=>1} );
```

Of course, you should replace `DDD`, `HHH`, `UUU`, and `PPP` with the correct information. `DDD` might be `201DonC`. `HHH` might be `cgi.cs.byuh.edu`. If your host is the "localhost" you can leave that part off from the connect statement. RaiseError is an example of information you can send as part of the connect process. If the connect fails, `$x` will be false. You can say `if($x)` to test whether the connection worked.

After connecting, you can access any of the tables and data in the database. When you are done, you should disconnect as follows.

```
$x->disconnect(); # when you are done
```

2.2 Issue a Query

When you are connected to the database, you can issue queries (inquiries, requests, commands). Data retrieval is probably the most common activity. There is a four-step process for retrieving data from a table. The first two steps are prepare and execute.

```
$query = "select * from tablename";
$y = $x->prepare($query); # introduce task
if ( !$y ) { die "query failed\n" }
$y->execute(); # carry out the task
```

Replace `tablename` with the actual name of your table. It is `scores` in the examples above.

2.3 Viewing Results

If your query creates results (like `select` does), then after you have executed the query, the results are ready for you to process. You can fetch the results one row at a time. If there are many rows of output, you need to do this in a loop so you can fetch each of them. Here are two ways to get the data from each row. In the first way, we retrieve all the columns for one row into an array.

```
while ( @z = $y->fetchrow_array() ) {
  print $z[1]; # print the second column
  ( $xxx, $yyy, $zzz, $frog ) = @z;
  print $frog; # print the first column
}
```

Of course, the names `$xxx`, etc. can be whatever variable names you wish to use. In the second way, we retrieve all the columns into a hash. Columns are identified by their formal name within the table.

```
while ( $z = $y->fetchrow_hashref() ) {
  print $z->{price}; # print price column
}
```

After processing the rows we want, we can end at any time.

```
$y->finish(); # when done with this query
```

3 Display Your Data

Write a program that displays the rows from the table you created earlier. Connect. Query. Display results. Disconnect. Quit. At first, do not worry if your data do not line up neatly. But do separate them by at least a few spaces so they don't run together.

To make your data line up in neat columns, you can use the `\t` tab character (a quick and dirty solution), or you can format the data to a specific width, using `printf`. Here is an example.

```
$format = "student: %-20s score: %5.0f\n";
printf ( $format, $student, $score );
```

4 Advanced Queries

Here is some additional information you may find useful.

4.1 Column Data Types Allowed

In our example above, we created a table with two columns: student and score. Student was followed by the note "varchar(50)" and score was followed by the note "int(6)". Varchar and int are called data types or column types. Here is a more complete sample of the column types allowed in mySQL. For a complete list, consult the mySQL book.

tinyint	-128 .. 127 (one byte)
smallint	-32768 .. 32767 (two bytes)
mediumint	-8388608 .. 8388607 (three bytes)
int	9 digits (four bytes)
bigint	20 digits (eight bytes)
float	like C, four bytes
double	like C, eight bytes
decimal(m,d)	string, m+2 bytes
char(m)	string, m bytes
varchar(m)	string, 1 to m+1 bytes
tinytext	up to 256 bytes
text	up to 65536 bytes
date	YYYY-MM-DD, three bytes
time	hh:mm:ss, three bytes
datetime	eight bytes
timestamp	four bytes (auto updating)
year	one byte

4.2 Updating a Row

You will need an update query. In this example, `mystuff` is a table name, `desc` is the column to be changed, `yadda` is a new value, and `ID` is the column to be matched.

```
update inven set desc="yadda" where ID=37;
update inven set desc="yadda"
  where ID=37 and price=33.91;
update inven set desc="yadda", price=99.99
  where ID=19;
```

4.3 Deleting a Row

You will need a delete query. In this example, `inventory` is a table name, `ID` is the column to be matched, and `99` is a value. If you leave off the "where" part, all rows in your table will be deleted.

```
delete from inventory where ID=99;
```

4.4 How to Add a Column

To modify an existing table, use the `alter table` query. Here are some samples of things you can do:

```
alter table foo add price int after cost;
alter table bar change price float;
alter table bletch drop cost;
```

4.5 How to Delete a Table

To delete a table, use the `drop table` query.

```
drop table bletch;
```

4.6 Not Case Sensitive

MySQL is not case sensitive, so name your tables and rows with that in mind. However, perl **is** case sensitive, so within any programs you write be consistent in how you capitalize things.

5 Doing More

If you wish to go beyond this short introduction to database you may want to buy these books. (They are in the bookstore under "IS 431.")

- **Recommended:** *MySQL*, by: Paul DuBois. New Riders press. ISBN 0-7357-0921-1. SRP $49.99.
- **Alternate:** *Programming the Perl DBI*, by: Alligator Descartes and Tim Bunce. O'Reilly press. ISBN 1-56592-699-4. SRP $34.95.

"MySQL" is more expensive, but really covers both SQL and the Perl DBI well. I recommend it highly.

DB Lab Sign-off Sheet

Student Name: ______________________________

This sheet is to be completed by a CS tutor. The tutor should sign (or initial) and date each item when the student has demonstrated its completion.

dbselect Lab

In this assignment, you will build one or more CGI programs to interact with the tables in your database. The CGI must list all the tables in your database and allow the user to select one of them. After selecting a table, the CGI must, at a minimum, identify which table was selected. For full credit, the CGI must also list all rows and columns in that table. You are not required to identify the column names.

For this assignment, you are not permitted to `use CGI` in your program. You are also not permitted to encode any table-specific information into your program. (This is also known as "hard coding" your program for specific database tables.)

Key queries you may need include "`show tables`" and "`select *`".

1. When the CGI program is run from the web, it should show a list of all the tables the student has. Each table should be selectable in some way.

Tutor: _______________ Date: _______________

2. When a table is selected, a new screen should be drawn with the name of that table shown in a special way, and all rows of the selected table should be displayed neatly and clearly.

Tutor: _______________ Date: _______________

3. Have the student create a new table (by hand) and insert new rows into it while you watch. Tell the student what to put into at least one of the new rows. Then have the student run their CGI again. It should display the new table and its rows. There should be no need for the student to modify the CGI to make this work.

Tutor: _______________ Date: _______________

For credit, the student should show this sign-off sheet to their instructor.

dbupdate Lab

In this assignment, you will build a table of items for sale. For each item in your inventory, the table must include the following information: an id number, a description, a quantity on hand, the cost to you, and the price to your customers. Put several items into your table. This is the minimum. You are allowed to have additional information in your table at your discretion.

You must also build one or more CGI programs to interact with your table. The CGI must, at a minimum, (1) display all rows of your table and give the total inventory value (cost $\times$ quantity) for each row, and give a grand total inventory value across all rows. For each row, there must be (2) a way to modify the description, quantity, cost, and price. There must also be (3) a way to delete an item from the table. Having a quantity of zero should not automatically cause deletion. There must be (4) a way to add new rows to your table, either one at a time or several at once.

1. Have the student operate their CGI to produce a complete inventory list on the screen showing id number, item description, quantity on hand, cost, and price. For each row, the total value of that row (quantity times cost) must be shown. For the whole table, the grand total value for all rows must be shown.

Tutor: _______________ Date: _______________

2. Have the student demonstrate the changing of values in several rows. The student should let the tutor decide what the new values will be. Verify the totals are still correct.

Tutor: _______________ Date: _______________

3. Have the student demonstrate the creation of several new rows. Verify the totals are still correct.

Tutor: _______________ Date: _______________

4. Have the student demonstrate the deletion of several old rows. Verify the totals are still correct.

Tutor: _______________ Date: _______________

For credit, the student should show this sign-off sheet to their instructor.