Beginning MySQL Tutorial

By W.J. Gilmore

Table of Contents

The Database

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention (http://www...) would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.

One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server, developed by T.c.X. DataKonsultAB. MySQL, available for download at http://www.mysql.com, offers the database programmer with an array of options and capabilities rarely seen in other database servers. What's more, MySQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section, as there is a charge for licensing the product.

These capabilities range across a number of topics, including the following:

- Ability to handle an unlimited number of simultaneous users.
- Capacity to handle 50,000,000+ records.
- Very fast command execution, perhaps the fastest to be found on the market.
- Easy and efficient user privilege system.

However, perhaps the most interesting characteristic of all is the fact that it's **free**. That's right, T.c.X offers MySQL as a free product to the general public.

So Who's Using MySQL?

This database server has gained enormous popularity within corporate circles, including the following organizations:

- Silicon Graphics (http://www.sgi.com)
- Siemens (http://www.siemens.com)

If that isn't convincing enough to read on, you can obtain a much larger list of MySQL users at the MySQL User's List.

This article will aid the reader in learning the basic operations of the MySQL server, including how to make a proper connection, set up the server for consequential manipulation and execute basic commands. If you already have a basic understanding of MySQL, feel free to skip ahead to the more advanced sections. Remember, however, that the commands covered within this article are among the most important commands within the language, as they form the basis for the more advanced commands.

Important note: Before reading on, it is of importance to note that the MySQL server is an extra service that, while offered by many Internet Service Providers, usually must be requested before it can be used. Be sure to ask your Provider to establish and configure a MySQL account for you before beginning the exercises in the following pages.

Part 1: At First Glance

MySQL is most commonly entered through **telnet**. (A nice Telnet program, Easyterm, can be found at http://www.arachnoid.com) Once the telnet connection to the web server has been accomplished, a second command provides access to the MySQL server. The procedure to make these connection is as follows:

1. Connect to telnet. This involves the insertion of the given ISP username and password.

```
login: devshed Password: ******** Last login: Wed Aug 12 09:49:14 from 195.103.124.222
Copyright 1992, 1993, 1994, 1995, 1996 Berkeley Software Design, Inc. Copyright (c) 1980,
1983, 1986, 1988, 1990, 1991, 1993, 1994 The Regents of the University of California. All
rights reserved. BSDI BSD/OS 2.1 Kernel #12: Mon Feb 23 13:46:27 EST 1998 You have
new mail. www24:mywww/devshed#
```

2. Connect to MySQL. This involves the insertion of the username and password given specifically for MySQL use. This information has probably been provided to you at your request to the ISP provider.

```
www24:mywww/devshed# mysql −u devshed −p
```

Syntax: mysql −h hostname −u username −p[password]
Or
mysql −h hostname −u username −−password=password

The user will then be prompted for a password, as prompted by −p.

```
Enter password: *******
```

Assuming MySQL has been correctly installed and configured, the user will see output similiar to the following:

```
Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is
49 to server version: 3.21.23−beta−log Type 'help' for help. mysql>
```

(**Note:** If an error message pertaining to "Access denied" is the result of connection attempts, you should consult the MySQL documentation included with the software, the MySQL mailing list found at http://www.mysql.com, as well as your ISP provider. These resources will aid greatly in resolving these problems.)

Once connected to the database, we are free to execute the various commands of the MySQL language. However before we are able to modify the database, we must first connect to it, via the command:

```
mysql> use devshed;
```

Result:

```
Database changed Mysql>
```

You now are connected to the database. Note that the command was followed by a semi−colon (;). Almost all commands in MySQL are followed by a semi−colon.

At the disposition are a number of administrative commands. These commands can be viewed simply by typing help, \h or ? at the command line:

```
mysql> help help (\h) Display this text ? (\h) Synonym for `help' clear (\c) Clear command
connect (\r) Reconnect to the server. Optional arguments are db and host edit (\e) Edit
command with $EDITOR exit (\) Exit mysql. Same as quit go (\g) Send command to mysql
server print (\p) print current command quit (\q) Quit mysql rehash (\#) Rebuild completion
hash status (\s) Get status information from the server use (\u) Use another database. Takes
database name as argument Connection id: 49 (Can be used with mysqladmin kill) mysql>
```

Perhaps not all functions will not be immediately useful, however it would be wise to test each one to see exactly what it entails. However, functions such as: status, use, print, connect, clear, and quit will probably prove to be very useful from the start, so be sure to become familiar with them.

You should now have a basic understanding of how to connect to the server, select the database, and perform basic commands. The next section will cover the concepts and techniques needed to properly setup up the database for manipulation.

Part 2: Datatypes and Tables

A database is really nothing more than a hierarchy of increasingly complex data structures. In MySQL, the acknowledged structure for holding blocks (or **records**) of information is called the **table**.

These records, in turn, are made up of the smallest object that can be manipulated by the user, known as the **datatype**. Together, one or more of these datatypes form a record. A table holds the collection of records that make up part of the database. We can consider the hierarchy of a database to be that of the following:

Database < Table < Record < Datatype

Datatypes come in several forms and sizes, allowing the programmer to create tables suited for the scope of the project. The decisions made in choosing proper datatypes greatly influence the performance of a database, so it is wise to have a detailed understanding of these concepts.

MySQL Datatypes

MySQL is capable of many of the datatypes that even the novice programmer has probably already been exposed to. Some of the more commonly used include:

CHAR (M)

CHAR's are used to represent fixed length strings. A CHAR string can range from 1−255 characters. In later table creation, an example CHAR datatype would be declared as follows:

ex.

```
car_model CHAR(10);
```

VARCHAR (M)

VARCHAR is a more flexible form of the CHAR data type. It also represents data of type String, yet stores this data in variable length format. Again, VARCHAR can hold 1−255 characters. VARCHAR is usually a wiser choice than CHAR, due to it's variable length format characteristic. Although, keep in mind that CHAR is much faster than VARCHAR, sometimes up to 50%.
(A CHAR stores the whole length of the declared variable, regardless of the size of the data contained within, whereas a VARCHAR only stores the length of the data, thus reducing size of the database file.)

ex.

```
car_model VARCHAR(10);
```

INT (M) [Unsigned]

The INT datatype stores integers ranging from −2147483648 to 2147483647. An optional "unsigned" can be denoted with the declaration, modifying the range to be 0 to 4294967295

ex.

```
 light_years INT;
```

Valid integer: '−24567'. Invalid integer: '3000000000'.

ex.

```
light_years INT unsigned;
```

Valid integer: '3000000000'. Invalid integer: '−24567'.

FLOAT [(M,D)]

A FLOAT represents small decimal numbers, used when a somewhat more precise representation of a number is required.

ex.

```
rainfall FLOAT (4,2);
```

This could be used to represent rainfall average in centimeters per year, which could be a decimal value. More specifically, FLOAT (4,2) states the fact that rainfall can hold up to four characters and two decimal places. Thus,

42.35 is valid, accurately represented.
324.45 is invalid, rounded to 324.5.
2.2 is valid, accurately represented.
34.542 is invalid, rounded to 34.54.

*Note: Due to the fact that FLOAT is rounded, those wishing to represent money values would find it wise to use **DECIMAL**, a datatype found within MySQL that does not round values. Consult the documentation for a complete explanation.*

DATE

Stores date related information. The default format is 'YYYY−MM−DD', and ranges from '0000−00−00' to '9999−12−31'. MySQL provides a powerful set of date formatting and manipulation commands, too numerous to be covered within this article. However, one can find these functions covered in detail within the MySQL documentation.

```
the_date DATE;
```

TEXT / BLOB

The text and blob datatypes are used when a string of 255 − 65535 characters is required to be stored. This is useful when one would need to store an article such as the one you are reading. However, there is no end space truncation as with VARCHAR AND CHAR. The only difference between BLOB and TEXT is that TEXT is compared case insensitively, while BLOB is compared case sensitively.

SET

A datatype of type string that allows one to choose from a designated set of values, be it one value or several values. One can designate up to 64 values.

ex.

```
transport SET ("truck", "wagon") NOT NULL;
```

From the above declaration, the following values can be held by transport:

```
""
"truck"
"wagon"
"truck,wagon"
```

ENUM

A datatype of type string that has the same characteristics as the SET datatype, but only one set of allowed values may be chosen. Usually only takes up one byte of space, thus saving time and space within a table.

ex.

```
transport ENUM ("truck", "wagon") NOT NULL;
```

From the above declaration, the following values can be held by transport:

```
""
"truck"
"wagon"
```

Records

Together, a group of declared datatypes form what is known as a record. A record can be as small as one data variable, or as many as deemed needed. One or more records form the structure of a table.

The Bigger Picture: Tables

Before we can execute commands on the database, we must first create a table in which data can be stored. This is accomplished in the following manner:

```
mysql> CREATE TABLE test (
> name VARCHAR (15),
> email VARCHAR (25),
> phone_number INT,
> ID INT NOT NULL AUTO_INCREMENT,
> PRIMARY KEY (ID));
```

Ensuing output:

```
Query OK, 0 rows affected (0.10 sec)
mysql>
```

The first table in your database has now been created. *Note: no two tables can have the same name. Note(2): Each dataspace is more often referred to as a* ***column****.*

Column Characteristics:

- A name may not be made up of strictly numbers.
- A name may start with a number.
- A name may be up to 64 characters.

Other table options:

The following options can be placed after any datatype, adding other characteristics and capabilities to them.

- Primary Key. Used to differentiate one record from another. No two records can have the same primary key. This is obviously useful when it is imperative that no two records are mistaken to be the

other.

- Auto_Increment. A column with this function is automatically incremented one value (previous + 1) when an insertion is made into the record. The datatype is automatically incremented when 'NULL' is inserted into the column.
- NOT NULL. Signifies that the column can never be assigned a NULL value.

ex.

```
soc_sec_number INT PRIMARY KEY;
```

No two soc_sec_number records can hold the same value.

```
ID_NUMBER INT AUTO_INCREMENT;
```

Automatically increments in value, starting at '1', with every subsequent insertion.

Table−Relevant Commands

We can execute a number of useful commands pertaining to the tables, such as the following:

Show Tables

```
mysql> show tables;
```

Result:
This will list all tables currently existing within the database.

Show Columns

```
mysql> show columns from test;
```

Result:
This will return the columns and column information pertaining to the designated table.

Take a minute to execute each one of the above commands after you have created the test table. They will prove very helpful as your database increases in size and complexity.

You should now have a basic understanding of the creation of tables, one of the most important concepts of the MySQL server. You now know that tables are constructed using datatypes, which when grouped together form a record. In the next section, we will begin learning how to manipulate the database.

Part 3: Manipulating the Database

A database can be manipulated in four possible ways: addition, deletion, modification, and search. These topics will all be briefly covered in the following two sections. However, before we begin, I would like to highlight the fact that SQL, like many computer languages, is somewhat particular about command syntax. The slightest error in placement of a parentheses, comma, or semicolon will almost surely end in error. As a result, take care to be attentive of command syntax.

Insertion of records

Note: The originally created table, test, created in the last section will be used to illustrate the examples in this section. Here it is again, for quick reference:

```
mysql> CREATE TABLE test ( > name VARCHAR (15), > email VARCHAR (25), >
phone_number INT, > ID INT NOT NULL AUTO_INCREMENT, > PRIMARY KEY (ID));
```

Insertion of data into the table is accomplished, logically enough, using the INSERT command.

```
mysql> INSERT INTO test VALUES mysql> ('Bugs Bunny', 'carrots@devshed.com', mysql>
5554321, NULL);
```

Result, assuming the command was correctly entered:

```
Query OK, 1 row affected (0.02 sec) mysql>
```

So what happened?

- Single quotations were placed around the datatypes VARCHAR. All datatypes of type STRING (i.e. char, varchar, text, blob, etc.) must be surrounded in single quotes, or an error will occur.
- There were no single quotes surrounding the phone number. Datatypes of type INT do not require single quotes.
- NULL ? A NULL allows any datatype with the characteristic AUTO_INCREMENT to be automatically assigned a value. If it is the first record inserted into the database, it is assigned the value '1'. Otherwise, it is assigned the previously inserted value + 1 (i.e. if the previously inserted value was '2', then the next would be '3'). In addition, the insertion of NULL into a variable of type TIMESTAMP causes that variable to be given the value of the current date.

Note: It is of importance to remember that the same number of values must be inserted as datatypes are contained within a record. In the above example, if one attempted to insert only three values instead of four, the insertion would fail. The same result applies if one attempted to insert five values.

Example:

```
mysql> insert into test values('doggy'); ERROR 1058: Column count doesn't match value
count mysql>
```

Note (2): One of the advantageous aspects of MySQL is it's ability to convert without trouble between datatypes. MySQL automatically converts between integers, strings, and dates without problems.

Selection

A database would not be much use if one was not able to search and extract data from it. In MySql terms, this is accomplished through the SELECT statement.

```
mysql> SELECT * FROM test mysql> WHERE (name = "Bugs Bunny");
```

Result:

name	email	phone	ID
Bugs Bunny	carrots@devshed.com	5554321	1

Let's assume we have inserted four differing records, all bearing the same name of "Bugs Bunny", yet having different email addresses and phone numbers. The table test, would look somewhat like the following:

name	email	phone	ID
Bugs Bunny	carrots@devshed.com	5554321	1
Bugs Bunny	peppers@devshed.com	5554331	2
Bugs Bunny	lettuce@devshed.com	5554341	3
Bugs Bunny	celery@devshed.com	5554351	4

Deletion

One can also delete records inserted into the table. This is accomplished through the DELETE command.

```
mysql> DELETE FROM test mysql> WHERE (name = "Bugs Bunny");
```

Result:
This would result in the deletion of all records within the table test containing name "Bugs Bunny".

Another example:

```
mysql> DELETE FROM test mysql> WHERE (phone_number = 5554321);
```

Result: (Using the previously illustrated example)

name	email	phone	ID
Bugs Bunny	peppers@devshed.com	5554331	2
Bugs Bunny	lettuce@devshed.com	5554341	3
Bugs Bunny	celery@devshed.com	5554351	4

Modification

MySQL also has the capability of modifying data already entered into the table. This is accomplished through the UPDATE command.

```
mysql> UPDATE test SET name = 'Daffy Duck' mysql> WHERE name = "Bugs Bunny";
```

name	email	phone	ID
Daffy Duck	peppers@devshed.com	5554331	2
Daffy Duck	lettuce@devshed.com	5554341	3
Daffy Duck	celery@devshed.com	5554351	4

This section, we covered the core MySQL database manipulation functions, basic insertion, deletion, modification, and search. The next section will elaborate on these capabilities, providing extended functioning and flexibility when manipulating the database.

Part 4: Advanced MySQL Commands

What we have covered so far is but a small part of what MySQL is capable of. Let's delve a little deeper into the language, exploring some of the more advanced commands of the language.

Logical Operations

MySQL includes full support of all basic logical operations.

AND (

```
mysql> SELECT * FROM test WHERE mysql> (name = "Bugs Bunny") AND mysql>
(phone_number = 5554321);
```

Result:
All records containing the name "Bugs Bunny" AND the phone number '5554321' will be displayed to the screen.

OR (||)

```
mysql> SELECT * FROM test WHERE mysql> (name = "Bugs Bunny") OR mysql>
(phone_number = 5554321);
```

Result:
All records containing the name "Bugs Bunny" OR the phone number '5554321' will be displayed to the screen.

NOT (!)

```
mysql> SELECT * FROM test WHERE mysql> (name != "Bugs Bunny");
```

Result:
All records NOT containing the name "Bugs Bunny" will be displayed to the screen.

Order By

```
mysql> SELECT * FROM test WHERE mysql> (name = "Bugs Bunny") ORDER BY
mysql> phone_number;
```

Result:
All records containing the name "Bugs Bunny" will be displayed to the screen, ordered in respect to the phone_number.

Search functions

MySQL offers the user the ability to perform both general and specific searches on data.

```
mysql> SELECT * FROM test WHERE mysql> (name LIKE "%gs Bunny");
```

Result:
All records containing the partial string "gs Bunny" will be displayed to the screen. This would include such names as: "Bugs Bunny", "ags Bunny", "gs Bunny", and "234rtgs Bunny".

Notice that "LIKE" has been used instead of the equals sign (=). "LIKE" signifies that one is searching for an estimate of the data requested, and not necessarily an exact copy.

The '%' sign could be placed anywhere within the string. The method in which the server searches for a string is dependent upon where one places the '%' sign.

```
mysql> SELECT * FROM test WHERE mysql> (name LIKE "Bugs Bunny%");
```

Result:
All records containing the partial string "Bugs Bunny" will be displayed to the screen. This would include such names as: "Bugs Bunnys", "Bugs Bunnyyyy453", "Bugs Bunnytrtrtrtrtr", but not "gs Bunny".

Focused Search Results

One can also perform searches and display only certain columns.

```
mysql> SELECT name FROM test WHERE mysql> (name = "Bugs Bunny");
```

Result:

name

Alter table

Another very important function of MySQL is the ability to modify previously created tables. This is accomplished via the ALTER statement. This function allows one to add, modify, and delete columns, as well as rename the table, among other functions.

Example: Rename the table

```
mysql> ALTER table test RENAME mytest;
```

Example: Add a column

```
mysql> ALTER table mytest ADD birthday DATE;
```

Example: Modify a column

```
mysql> ALTER table mytest CHANGE mysql> name newname VARCHAR (25);
```

Example: Delete a column

```
mysql> ALTER table mytest DROP newname;
```

Executing the above four functions would modify test, creating the following table:

```
mysql> TABLE mytest ( > email VARCHAR (25), > phone_number INT, > ID INT
AUTO_INCREMENT, > birthday DATE );
```

The topics covered within this article are but a short introduction of the capabilities of MySQL. However, these functions form the basis of almost all advanced commands to be found in the language. Above all, the most important lesson that one can remember is to practice, study the documentation, and become an active member of the mailing list archives (And obviously read DevShed frequently!). Only by taking an enthusiastic, even "aggressive" approach to the language can one successfully master it.