A Basic MySQL Tutorial

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: http://faq.asphosthelpdesk.com/article.php?id=182

About MySQL

MySQL is an open source database management software that helps users store, organize, and retrieve data. It is a very powerful program with a lot of flexibility—this tutorial will provide the simplest introduction to MySQL

How to Install MySQL on Ubuntu and CentOS

If you don't have MySQL installed on your server instance, you can quickly download it.

Ubuntu:

```
sudo apt-get install mysql-server
```

Centos:

```
sudo yum install mysql-server
/etc/init.d/mysqld start
```

How to Access the MySQL shell

Once you have MySQL installed on your server instance, you can access the MySQL shell by typing the following command into terminal:

```
mysql -u root -p
```

After entering the root MySQL password into the prompt (not to be confused with the root control panel password), you will be able to start building your MySQL database.

Two points to keep in mind:

- All MySQL commands end with a semicolon; if the phrase does not end with a semicolon, the command will not execute.
- Also, although it is not required, MySQL commands are usually written in uppercase and databases, tables, usernames, or text are in lowercase to make them easier to distinguish. However, the MySQL command line is not case sensitive.

How to Create and Delete a MySQL Database

MySQL organizes its information into databases; each one can hold tables with specific data.

You can quickly check what databases are available by typing:

```
SHOW DATABASES;
```

Your screen should look something like this:

```
 mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.01 sec)
```

Creating a database is very easy:

```
 CREATE DATABASE database name;
```

In this case, for example, we will call our database "events."

```
 mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| events             |
| mysql              |
| performance_schema |
| test               |
+--------------------+
5 rows in set (0.00 sec)
```

In MySQL, the phrase most often used to delete objects is Drop. You would delete a MySQL database with this command:

```
 DROP DATABASE  database name;
```

How to Access a MySQL Database

Once we have a new database, we can begin to fill it with information.

The first step is to create a new table within the larger database.

Let"s open up the database we want to use:

```
USE events;
```

In the same way that you could check the available databases, you can also see an overview of the tables

that the database contains.

```
SHOW tables;
```

Since this is a new database, MySQL has nothing to show, and you will get a message that says, "Empty setâ€•

How to Create a MySQL Table

Let"s imagine that we are planning a get together of friends. We can use MySQL to track the details of the event.

Let"s create a new MySQL table:

```
CREATE TABLE potluck (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(20),
food VARCHAR(30),
confirmed CHAR(1),
signup_date DATE);
```

This command accomplishes a number of things:

1. It has created a table called potluck within the directory, events.
2. We have set up 5 columns in the table—id, name, food, confirmed, and signup date.
3. The "idâ€• column has a command (INT NOT NULL PRIMARY KEY AUTO_INCREMENT) that automatically numbers each row.
4. The "nameâ€• column has been limited by the VARCHAR command to be under 20 characters long.
5. The "foodâ€• column designates the food each person will bring. The VARCHAR limits text to be under 30 characters.
6. The "confirmedâ€• column records whether the person has RSVP"d with one letter, Y or N.
7. The "dateâ€• column will show when they signed up for the event. MySQL requires that dates be written as yyyy-mm-dd

Let"s take a look at how the table appears within the database using the "SHOW TABLES;" command:

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_events |
+------------------+
| potluck          |
+------------------+
1 row in set (0.01 sec)
```

We can remind ourselves about the table"s organization with this command:

```
 DESCRIBE potluck;
```

Keep in mind throughout that, although the MySQL command line does not pay attention to cases, the table and database names are case sensitive: potluck is not the same as POTLUCK or Potluck.

```
 mysql>DESCRIBE potluck;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| name        | varchar(20) | YES  |     | NULL    |                |
| food        | varchar(30) | YES  |     | NULL    |                |
| confirmed   | char(1)     | YES  |     | NULL    |                |
| signup_date | date        | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

How to Add Information to a MySQL Table

We have a working table for our party. Now it"s time to start filling in the details.

Use this format to insert information into each row:

```
INSERT INTO `potluck` (`id`,`name`,`food`,`confirmed`,`signup_date`) VALUES
(NULL, "John", "Casserole","Y", '2012-04-11');
```

Once you input that in, you will see the words:

```
 Query OK, 1 row affected (0.00 sec)
```

Let"s add a couple more people to our group:

```
INSERT INTO `potluck` (`id`,`name`,`food`,`confirmed`,`signup_date`) VALUES
(NULL, "Sandy", "Key Lime Tarts","N", '2012-04-14');
INSERT INTO `potluck` (`id`,`name`,`food`,`confirmed`,`signup_date`) VALUES
(NULL, "Tom", "BBQ","Y", '2012-04-18');
INSERT INTO `potluck` (`id`,`name`,`food`,`confirmed`,`signup_date`) VALUES
(NULL, "Tina", "Salad","Y", '2012-04-10');
```

We can take a look at our table:

```
mysql> SELECT * FROM potluck;
+----+-------+---------------+-----------+-------------+
| id | name  | food          | confirmed | signup_date |
+----+-------+---------------+-----------+-------------+
|  1 | John  | Casserole     | Y         | 2012-04-11  |
|  2 | Sandy | Key Lime Tarts | N         | 2012-04-14  |
|  3 | Tom   | BBQ           | Y         | 2012-04-18  |
|  4 | Tina  | Salad         | Y         | 2012-04-10  |
+----+-------+---------------+-----------+-------------+
```

```
4 rows in set (0.00 sec)
```

How to Update Information in the Table

Now that we have started our potluck list, we can address any possible changes.

For example: Sandy has confirmed that she is attending, so we are going to update that in the table.

```
UPDATE `potluck`
SET
`confirmed` = 'Y'
WHERE `potluck`.`name` ='Sandy';
```

You can also use this command to add information into specific cells, even if they are empty.

How to Add and Delete a Column

We are creating a handy chart, but it is missing some important information: our attendees" emails.

We can easily add this:

```
ALTER TABLE potluck ADD email VARCHAR(40);
```

This command puts the new column called "email" at the end of the table by default, and the VARCHAR command limits it to 40 characters.

However, if you need to place that column in a specific spot in the table, we can add one more phrase to the command.

```
 ALTER TABLE potluck ADD email VARCHAR(40) AFTER name;
```

Now the new "emailâ€• column goes after the column "nameâ€•.

Just as you can add a column, you can delete one as well:

```
ALTER TABLE potluck DROP email;
```

I guess we will never know how to reach the picnickers.

How to Delete a Row

If needed, you can also delete rows from the table with the following command:

```
DELETE from [table name] where [column name]=[field text];
```

For example, if Sandy suddenly realized that she will not be able to participate in the potluck after all, we

could quickly eliminate her details.

```
mysql> DELETE from potluck  where name='Sandy';
Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM potluck;
+----+------+-----------+-----------+-------------+
| id | name | food      | confirmed | signup_date |
+----+------+-----------+-----------+-------------+
|  1 | John | Casserole | Y         | 2012-04-11  |
|  3 | Tom  | BBQ       | Y         | 2012-04-18  |
|  4 | Tina | Salad     | Y         | 2012-04-10  |
+----+------+-----------+-----------+-------------+
3 rows in set (0.00 sec)
```

Notice that the id numbers associated with each person remain the same.