Faversham Associates Ltd

Using MySQL

A brief introduction

Version: 1.0.0

Gavin Bungay

gavin.bungay@favershamassociates.com

December 12, 2017

This document is a brief outline of some of the most used features and functionality of MySQL.

The following assumes that MySQL has already been successfully installed.

For more information on installing it, take a look at our document:

'LAMP, Using Apache, MySQL and Python on Linux',

For information about using using MySQL with Python, try:

'Python and MySQL, Connecting to a MySQL database using mysqlclient',

There are links to the above documents below: Section 2; *See Also*; page 12

Contents

1 Using MySQL

1.1 Starting and Exiting the MySQL Client

MySQL comes with a client program, or monitor, that allows users to connect to and modify databases.

- Starting the MySQL client.

  At the shell prompt type:

  ```
  $ mysql -user=root -password=⟨yourpassword⟩
  ```

  The prompt will change to the MySQL one:

  ```
  mysql>
  ```

  Alternatively one can use:

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

  You will then be prompted for the password

- Exiting the MySQL client.

  At the MySQL prompt type:

  ```
  mysql> exit
  ```

  or

  ```
  mysql> \q
  ```

  This will return the user to the command shell.

  In the above example, we logged on as the `root` user. This is not a good idea for several reasons, not least, security.

1.2 Adding and Deleting Users

In the above example, we logged on as the `root` user. This is not a good idea for several reasons, not least, security.

One of the first tasks should be to create users, other than `root` and assign appropriate rights to use to connect to and manage your database.

- Adding a new user.

  The documentation says the command to create a new user is:

  ```
  mysql> CREATE USER '⟨username⟩'@'localhost'
         ➥ IDENTIFIED BY '⟨password⟩';
  ```

  So I tried to create a user, `gavin`:

  ```
  mysql> CREATE USER 'gavin'@'localhost'
         ➥ IDENTIFIED BY '⟨mypassword⟩';
  ```

  No matter what I tried, I still go the error, so I used:

  ```
  mysql> CREATE USER gavin
  ```

  instead, and that seemed to work. When I typed:

  ```
  mysql> SELECT User from mysql.user;
  ```

  I got the following response that confirmed the user had been added:

  ```
  +------------------+
  | User             |
  +------------------+
  | gavin            |
  | debian-sys-maint |
  | gavin            |
  | mysql.session    |
  | mysql.sys        |
  | root             |
  +------------------+
  6 rows in set (0.00 sec)
  ```

- Granting Rights (and Setting a Password)

  The user `gavin`, created above, exists, but has no rights to anything yet.

  The general command to assign rights is:

  ```
  mysql> GRANT ⟨rights⟩ ON ⟨database⟩ TO '⟨user⟩'@'localhost';
  ```

  Where ⟨`rights`⟩ can be one of the following:

- ALL PRIVILEGES - grants all privileges to the MySQL user
- CREATE - allows the user to create databases and tables
- DROP - allows the user to drop databases and tables
- DELETE - allows the user to delete rows from specific MySQL table
- INSERT - allows the user to insert rows into specific MySQL table
- SELECT - allows the user to read the database
- UPDATE - allows the user to update table rows

and ⟨`database`⟩ is either:

- ⟨`databasename`⟩.* - a specific database

or

- *.* - all databases.

I used a variant that also allows me to set the password:

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'gavin'@'localhost'
       ➥ IDENTIFIED BY '⟨password⟩';
```

I then logged out and logged in as this new user:

```
$ mysql -u gavin -p
```

For the rest of what follows, I was logged in as `gavin` rather than root.

1.3 Using the Client

The client allows the user to execute commands, below are a few simple, everyday examples.

- Listing all the databases

  At the MySQL prompt type:

  ```
  mysql> show databases;
  ```

  This will return a list of databases, on our system, immediately after installation this returned the following:

  ```
  +--------------------+
  | Database           |
  +--------------------+
  | information_schema |
  | mysql              |
  ```

```
| phpmyadmin         |
+--------------------+
3 rows in set (0.00 sec)
```

- To show current database

  At the MySQL prompt type:

  ```
  mysql> select DATABASE();
  ```

  This will return the current database, for example, if running the above command just after loading MySQL the user will get the following response:

  ```
  +------------+
  | DATABASE() |
  +------------+
  | NULL       |
  +------------+
  1 row in set (0.00 sec)
  ```

  This indicates that no database is currently being used. To select a database use the `use` command below.

- To use a particular database

  At the MySQL prompt type:

  `mysql> use ⟨database name⟩`

  (note: the semicolon (`;`) is not required.)

  This will display a message informing the user that the database has changed:

  ```
  Database changed
  ```

  For example if the user types:

  ```
  mysql> use mysql
  ```

  And then

  ```
  mysql> select DATABASE();
  ```

  This will show that the current database is no longer `NULL`

```
+------------+
| DATABASE() |
+------------+
| mysql      |
+------------+
1 row in set (0.00 sec)
```

- To create a database

  At the MySQL prompt type:

  ```
  mysql> create database ⟨database name⟩;
  ```

  For example, to create a database called `websites`:

  ```
  mysql> create database websites;
  ```

  If successful a confirmation message will be returned: g

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

  If we now list all the databases, we should see the new database:

  ```
  mysql> show databases;
  ```

  ```
  +--------------------+
  | Database           |
  +--------------------+
  | information_schema |
  | mysql              |
  | phpmyadmin         |
  | websites           |
  +--------------------+
  4 rows in set (0.00 sec)
  ```

  So far only an skeleton database has been create, if we list its tables we don't see anything interesting.

- Deleting a database.

  In database parlance, deleting an object such as a table is referred to as 'dropping' it.

  To delete a database, at the command prompt type:

  ```
  mysql> drop ⟨database name⟩;
  ```

  For example, to drop the `websites` table created above, use:

  ```
  mysql> drop website;
  ```

  If successful a confirmation message will be returned:


```
Query OK, 0 rows affected (0.09 sec)
```

If one attempts to drop a database when it doesn't exist,they will get an error like the one below:

```
ERROR 1008 (HY000): Can't drop database 'websites'; database doesn't exist
```

To stop this type of error, one can use `if exists` construct to test whether a database exists when attempting top drop it:

```
mysql> drop database if exists ⟨database name⟩;
```

This time, if the database doesn't exist one will get a non fatal warning rather than an error:

```
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

- Listing a database's tables.

  Change to that database, for example:

  ```
  mysql> use websites
  ```

  List its tables:

  ```
  mysql> show tables;
  ```

  If we do this with the `websites` database we will see something like:

  ```
  Empty set (0.00 sec)
  ```

  as the database currently contains no tables.

- Creating a table.

  The `create table;` command is used to create tables. The complete syntax for this command is, if not complex, is at least long and involved - to cover all the options. Here we are going to use a much simplified form. A table must have at least one column, so we're going to create a table called `sitedef` and add one column, `name`, which will hold up to twenty characters.

  To achieve this, at the MySQL prompt type:

  ```
  mysql> create table sitedef(
      ➥ name varchar(20)
      ➥ );
  ```

  This will return something similar to:

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

Note that the command was split over several lines, it's the semi-colon and not the new line that ends a statement.

Now if we list the tables in the `websites` database we should see the one we've just created.

```
mysql> show tables;
```

This now returns:

```
+--------------------+
| Tables_in_websites |
+--------------------+
| sitedef            |
+--------------------+
1 row in set (0.00 sec)
```

- Deleting a table.

  To delete (or drop) a table from the current database, at the MySQL prompt type:

  ```
  mysql> drop table ⟨table name⟩;
  ```

  For example:

  ```
  mysql> drop table sitedef;
  ```

  This returns a confirmation message, something like:

  ```
  Query OK, 0 rows affected (0.00 sec)
  ```

  Obviously, if a table is dropped, all the data it contained will also be deleted.

  If one tries to drop a table that doesn't exist they will get an error. To prevent this, use the `if exists` contruct to test whether a table exists before trying to delete it:

  ```
  mysql> drop table if exists ⟨table name⟩;
  ```

  This time, if the table doesn't exist one will get a non fatal warning rather than an error.

- Showing the columns in a table.

  The general command is:

  ```
  mysql> show columns from ⟨table name⟩;
  ```

  For example, to list the columns in the table `sitedef` (assuming it hasn't been dropped):

  ```
  mysql> show columns from sitedef;
  ```

  This returns the columns currently in the table:

  ```
  +-------+-------------+------+-----+---------+-------+
  | Field | Type        | Null | Key | Default | Extra |
  +-------+-------------+------+-----+---------+-------+
  | name  | varchar(20) | YES  |     | NULL    |       |
  +-------+-------------+------+-----+---------+-------+
  1 row in set (0.00 sec)
  ```

- Inserting data into a table.

  The general syntax is of the form:

  ```
  mysql> insert into ⟨table name⟩
         ➥ (⟨field1⟩, ⟨field2⟩, ⟨field3⟩, ..., ⟨fieldn⟩)
         ➥ values (⟨value1⟩, ⟨value2⟩, ⟨value3⟩, ..., ⟨valuen⟩);
  ```

  For example to add a website called 'gsb' to the sitedef table:

  ```
  mysql> insert into sitedef (name) values ('gsb');
  ```

  This return a message:

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

- Listing all the data in a table.

  A very simple version of the `select` statement that returns all the data from a table is:

  ```
  mysql> select * from ⟨table name⟩;
  ```

  For example to list all the data in the `sitedef` table:

  ```
  mysql> select * from sitedef;
  ```

  This returns:


```
+------+
| name |
+------+
| gsb  |
+------+
1 row in set (0.00 sec)
```

- Altering a table.

  To change the definition of a table use the: `alter table` command.

  This complete syntax is as long as the `create table`, in the following example, we are going to use a simple version to add a column to our `sitedef` table. The general version of the command to add a column is:

  `mysql> alter table ⟨table name⟩ add column ⟨column name⟩ ⟨column spec⟩;`

  We are going to add a column called 'title':

  ```
  mysql> alter table sitedef add column title varchar (50);
  ```

  This returned:

  ```
  Query OK, 1 row affected (0.10 sec)
  Records: 1  Duplicates: 0  Warnings: 0
  ```

  Now let's show the columns of the table to confirm the new one has been successfully added:

  ```
  mysql> show columns from sitedef;
  ```

  retuns:

  ```
  +-------+-------------+------+-----+---------+-------+
  | Field | Type        | Null | Key | Default | Extra |
  +-------+-------------+------+-----+---------+-------+
  | name  | varchar(20) | YES  |     | NULL    |       |
  | title | varchar(50) | YES  |     | NULL    |       |
  +-------+-------------+------+-----+---------+-------+
  2 rows in set (0.00 sec)
  ```

1.4 Using SQL Scripts

Typing commands in at the command line is fine for very simple commands, but it is inconvenient to have to type in long, complex ones. Luckily commands can be put in a file and then executed using the syntax:

`mysql> source ⟨script name⟩`

where

⟨`script name`⟩ includes its path.

For example, I created a file `selsites.sql` with the single line:

```
select * from sitedef;
```

Listing 1: Selecting all the data in a table

Now running:

`mysql> source  /Documents/Dev/mysql/selsites.sql`

returns:

```
+------+-------+
| name | title |
+------+-------+
| gsb  | NULL  |
+------+-------+
1 row in set (0.00 sec)
```

I have a slight confession, when I first attempted to run the above script, I got an error:

```
ERROR 1046 (3D000): No database selected
```

This was because I had restarted the MySQL client and not select the correct database with the `use websites` command before running it. Once I had done this, everything worked as expected. I am going to make a slight change to the `selsites.sql` script and place the `use` command at the top of the script to ensure the correct database is always selected automatically:

```
use websites;
select * from sitedef;
```

Listing 2: Selecting all the data in a table, selecting the correct database first

It is good practice to select the correct database at the top of all MySQL scripts.

2 See Also

'http://www.favershamassociates.com/Documents/lamp.pdf - LAMP, Using Apache, MySQL and Python on Linux'

'http://www.favershamassociates.com/Documents/py-mysql.pdf - Python and MySQL, Connecting to a MySQL database using mysqlclient'

'http://www.favershamassociates.com/Documents/Django.pdf - Django, Basic Installation using Python 3'

3 Copyright