The Bioinformatics Lab: Databases and SQL

Tatyana Goldberg

June 24, 2010

The programming assignment for the seventh week of the "Bioinformatics Lab" course was to set up a MySQL database server, to create a new database and to provide only special users access to this database from the localhost. The database was then filled with data using programming language Perl. Furthermore, phpMyAdmin was installed to handle the administration and management of MySQL databases through a web-based graphic user interface.

1 Introduction

Many web based applications can be simplified through the use of a standardized database. A *database* is purposed to efficiently store, manage and retrieve information. Large amounts of data are usually organized in *relational databases* [1]. Relational databases are made up of *relations*, more commonly called *tables*. Each column in the table has a unique name and contains different data. The columns are called *fields* or *attributes*. Each row has the same attributes and represents one unique record. The tables stored in a relational database are referenced to each other based on identifying columns, also called *keys*.

One of the the most commonly used relational database management systems is MySQL [2]. It is very fast, robust, reliable, easy to use and suitable for applications of any size. MySQL uses Structural Query Language (SQL) [3], the standard query language worldwide. It is an open source software and falls under the GNU General Public License [4].

A wide range of application programming interfaces (APIs) is provided by MySQL. A MySQL API specifies a collection of recipes for connecting to the system and executing MySQL commands in any programming language or environment, including Java, ODBC, Perl, PHP, Python, Ruby, Tcl, C and C++.

Several graphical administration applications, also called *frontends*, are available to communicate with MySQL and work with MySQL databases visually. PhpMyAdmin [5] is one of the most popular web applications for MySQL database management. It is an open source software written in PHP. Through this software a user can not only execute any SQL statement but can also export data to various formats, load files into tables, track changes on databases and tables and much more.

2 MySQL

This section presents the steps required for the installation and connection to the MySQL database server. A simple example to create a database, store it with data and give permissions to its users introduces some of the basic tasks you can perform in MySQL.

Please note that MySQL commands end with a semicolon. This tells MySQL to execute the command. Another point to note is that SQL statements are not case sensitive, however this can be the case for tables and column names.

2.1 Installation

The first step is to install the MySQL server package by typing:

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

The installer will start to download the necessary packages. It will also ask you to set up the MySQL root password. Without a root password your system is insecure.

The MySQL server package consists of a MySQL server, several client programs to access the server and a client library for writing your own client programs. When using MySQL there are actually two programs running because MySQL uses a server/client architecture. The database server is a program to manage the databases. It listens for client requests and accesses database contents to give the information required in those requests. The client programs are the programs that connect to the database server and issue SQL statements. Thus, you can use client programs on your system to access data on a MySQL server running on another computer. The client programs included in the `mysql-server` package are `mysql`, `mysqlaccess`, `mysqladmin`, `mysqlbug`, `mysqlcheck`, `mysqldump`, `mysqlimport`, `mysqlshow`, `mysqlslap` and others. The client programs used in our course are:

`mysql`, a general-purpose client for issuing queries and retrieving their results

`mysqladmin`, an administrative client that helps you to manage the server

`mysqldump`, a program for dumping the contents of databases and making database backups

MySQL Navigator

MySQL Navigator [6] is a GUI based MySQL database server client program. It is very simple to use and can be utilized for database creation, server administration and even editing the script files. In order to use the MySQL Navigator application you must install the `mysqlnavigator` package, of course.

2.2 Server Connection

To login to the MySQL server type the following:

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

The `mysql` command is a command line client that connects you to the MySQL server. The `-u` switch is used to specify the user name you want to connect as. The `-p` switch tells the server you want to connect using a password.

After entering the previous command, you should get the following response:

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

Now, you should see some introductory information. The mysql prompt tells you that `mysql` is ready for you to enter commands.:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version:  5.0.51a-24+lenny4 (Debian)
Type 'help;' or '\h' for help.  Type '\c' to clear the buffer.
mysql>
```

You can logout by typing `quit` (or \q):

```
mysql> quit
Bye
```

2.3 Administration

A MySQL system can have many users. The root user should generally be used for administrative purposes only because of the security reasons.

Database Privileges

One of the features of MySQL is that it supports a sophisticated privilege system. A privilege is a right to perform a particular action on a particular object, which is associated with a particular user. In MySQL only root has the highest level of privilege.

If you connect to the server via a root account you can view the tables contained in the `mysql` database by typing:

```
mysql> use mysql;
mysql> show tables;
```

At this point the most important tables included in the result of our query are the `user`, `db` and `host` tables. Each of these tables contain information about privileges to specify what a specific user can or can not do within a system. The tables consist of the following two types of fields:

- scope fields to identify host, user and a part of a database and
- privilege fields to identify which actions can be performed by a user from a particular scope field

The `user` table determines information whether a user is allowed to connect the MySQL server and whether he has any administrative privileges. The `db` table is used to decide which users can access which databases from which hosts. The `host` table supplements the `db` table. If a user wants to connect the database from multiple hosts, then no host will be listed for that user in the `db` table. Instead, there will be a set of entries for this user listed in the `host` table, one to specify the privileges for each user-host combination.

A Real-Life Example

In this example we will create a database called 'marios_db' for a new user 'mario' who wants to use the password 'mammamia'. Only the user 'mario' should have access to the database from localhost.

To access the system tables containing privileges, type:

```
mysql> use mysql;
```

Then edit the `host` table to give localhost permission to access all databases. Use '%' as a wildcard:

```
mysql> INSERT INTO
-> host(host, db, Select_priv, Insert_priv, Update_priv,
-> Delete_priv, Create_priv, Drop_priv)
-> VALUES('localhost','%','Y','Y','Y','Y','Y','Y');
```

Create a new user 'mario' who can access the MySQL server from localhost. Call the function `password()` to encrypt the password:

```
mysql> INSERT INTO
-> user(host, user, password)
-> VALUES('localhost','mario',password('mammamia'));
```

In order to give the user 'mario' permission to access the database 'marios_db' from localhost insert a record into the `db` table as follows:

```
mysql> INSERT INTO
-> db (host, db, user, Select_priv, Insert_priv, Update_priv,
-> Delete_priv, Create_priv, Drop_priv)
-> VALUES ('localhost','marios_db','mario','Y','Y','Y','Y','Y','Y');
```

Exit from the MySQL server and create the necessary database 'marios_db' using the `mysqladmin` command:

```
mysqladmin -u root -p create marios_db
```

After prompting for the password, the database is created. Now, MySQL must be reloaded to enable all the modifications you made:

```
mysqladmin -u root -p reload
```

2.4 Backup

It is always a good idea to backup your data and files by regularly making backup copies. The `mysqldump` client can be used to backup all MySQL databases, several databases, one database or just certain tables within a given database. One of the possible ways to use the `mysqldump` command to dump entire databases is:

```
mysqldump -all-databases > all.sql
```

3 Perl DBI

As described above there exist different ways to access the MySQL server and to manage the data stored in the databases. This can also be done using any of the programming languages of your choice. Here, a simple script template provides an example on how to communicate with MySQL from Perl and introduces important methods supported by the DBI interface [7].

The DBI is a database access module for the Perl programming language. It defines a set of methods and variables to provide a consistent database interface, independent of the database being used. This module does not come with the standard Perl distribution, therefore you have to install the `libdbi-perl` Debian package. After the installation you can proceed with your actual Perl code to work with the database.

```
#!/usr/bin/perl
use strict;
use warnings;
```

At the beginning of the code you must load the DBI module and establish the data source name (dsn). Dsn tells the DBI what kind of database it is connecting to. Then, set the user name and password. Finally, connect to the database and print an error if the connection was unsuccessful. From this command you get a database handle, which you can use to run your SQL queries.

```
#load module
use DBI;
# set the data source name
my $dsn='dbi:mysql:marios_db';

# set the user and password
my $user='mario';
my $pass='mammamia';

# now connect and get a database handle
my $dbh = DBI->connect($dsn,$user,$pass) or die "Can't connect to the
database:  $DBI::errstr\n";
```

The execution of a SQL query takes two steps. First, you must prepare the query and then execute it. After the query is executed the `finish` call allows it to reinitialize the handle so that you can execute it again for the next query.

```
# CREATE example
my $sth1=$dbh->prepare('CREATE TABLE pizza(id VARCHAR (12), price INT)');
$sth1->execute();
$sth1->finish();

# INSERT example
my $sth2=$dbh->prepare('INSERT INTO pizza(id,price) values ("tonno",5)');
$sth2->execute();
$sth2->finish();
```

```
# SELECT example
my $sth3=$dbh->prepare('SELECT * FROM pizza');
$sth3->execute();
while(my @row=$sth3->fetchrow_array()){
print "$row[0] - $row[1]\n"; }
$sth3->finish();
```

At the end close the connection to the database.

```
$dbh->disconnect();
```

4 phpMyAdmin

Assuming we have Apache2 web server, the scripting language PHP and the MySQL database server installed and running on our machines, we can set up phpMyAdmin, a tool to handle the administration of MySQL over the Internet.

To install the `phpMyAdmin` package, type this:

```
sudo apt-get install phpmyadmin
```

Choose Apache2 web server to be configured for phpMyAdmin. The configuration files are then located in `/etc/phpmyadmin`. The configuration file `/etc/phpmyadmin/apache.conf` sets the alias and the directory permissions for phpMyAdmin, directives for loading PHP, etc.

To set up phpMyAdmin under Apache2 you need to include the following line at the end of the file `/etc/apache2/apache2.conf`:

```
Include /etc/phpmyadmin/apache.conf
```

Now you need to reload the Apache2 web server in order to enable the new configuration:

```
sudo /etc/init.d/apache2 reload
```

Point your browser to the `https://localhost/phpmyadmin` to access phpMyAdmin. Now you should be able to log in using any user name you have set up in MySQL. The secure HTTPS (HTTP over SSL) URL is used to protect the login and password so that access is made by authorized users only.

5 Summary

MySQL is a powerful, fast and easy to use SQL database program. This program is widely used to construct database-based web sites that are sophisticated, portable and professional appearing. In this tutorial I provided information on how to install and configure MySQL. I also explained how to grant users with privileges to access the database to make it secure. MySQL can easily be integrated into Perl programs using DBI. The free graphical administration application phpMyAdmin makes the usage of MySQL more intuitive.

References

[1] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13 (6):377-387,1970.

[2] http://www.mysql.com/.

[3] Structured query language (sql). *International Business Machines*, 2006.

[4] http://www.gnu.org/licenses/gpl.html.

[5] http://www.phpmyadmin.net/.

[6] http://sourceforge.net/projects/mysqlnavigator/.

[7] http://dbi.perl.org/.