Introduction to NMS SQL Database

NMS Computing Support provides every Informatics student with **5** databases for them to create and read / write to. NMS computing provides MariaDB (`https://mariadb.org/about/`) which is an open-source, drop-in replacement for MySQL, and you will see the names MySQL and MariaDB used interchangably. You should test your SQL on these databases in your Lab Practicals and as part of the Coursework.
Here is the current version that NMS provides:

mysql Ver 5.5.50-MariaDB

You can also download it and install it in your computer.

1 Setting up a Database

The NMS MySql student database management web application is at:

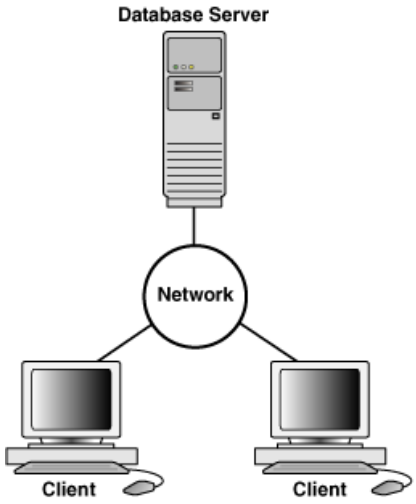
`https://nms.kcl.ac.uk/mysql`

1. Login with your KCL account (k-number and password)
2. Take note of your database username (this should be the same as your k-number).
3. You will see a MySQL password which you will use to connect to your database. You can change the password if you like. If you forget your database password, you can always retrieve it using this web application.
4. Click on **New** to create a database. A new database will appear in the list, and make sure to give database a name **without** any spaces or punctuation. Set the name to your user name and lab e.g. `k123456_lab`. This is the database we will use for the lab sessions. You can create up to 5 databases to work in the labs, assignments, etc.
5. Copy the line containing the commands for accessing your database (e.g., mysql -u k123456 -p -h mysql2.nms.kcl.ac.uk -P 33306 k123456_lab) and keep it safe.

In this session you will learn how to create and restore database backups. Note it is your responsibility to make backups of your database before and after each lab session.

2 Introduction to MySQL

The NMS database system uses a client-server architecture as follows:



- The **server** is the program that actually manipulates databases.
- To tell the server what to do, use a **client** program that communicates your intent by means of statements written in Structured Query Language (SQL).

2.1 Getting Started with NMS Server

Your command line access should be similar to

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName
```

This line is a connection command that uses different parameters:

- `-u yourUserName` specifies the MySQL user name to use when connecting to the server.
- `-p` indicates that a password is needed when connecting to the server.
- `-P` indicates the port used when connecting to the server.
- `-h mysql2.nms.kcl.ac.uk` indicates the name of the host containing the MySQL server.

2.2 Using *mysql* Client Program

Client programs are written for diverse purposes, but each interacts with the server by connecting to it, sending SQL statements to have database operations performed, and receiving the results.
Clients are installed locally on the machine from which you want to access MySQL, but the server can be installed anywhere (in this case a machine maintained by NMS). The *mysql* program is one of the clients included in MySQL distributions. You can use the *mysql* program from Windows or Linux. For example, in Linux can use the console to run the *mysql* client program by following the next steps:

1. Launch the terminal.
2. Paste the connection command line into the command prompt and press `Enter`. You will be asked to enter your password
3. Type your password and press `Enter`.
4. You will be shown a brief introduction message and then be placed at the `MariaDB [k12345_db]>` prompt.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 87973
Server version: 5.5.50-MariaDB MariaDB Server

Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [k12345_db]>
```

At the `MariaDB [k12345_db]>` prompt, you can enter SQL commands.

5. The `quit` command terminates your mysql session. Don't type it now, as you will have to start the session again, just remember this command to close your session once you have finished.

3 Creating a Sample Table

1. Create a simple table by typing the following SQL statement and press `Enter`:

```
MariaDB [k12345_db]> CREATE TABLE limbs (thing VARCHAR(20), legs INT, arms INT);
```

A **table** is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect.

2. And populate it with a few rows:

```
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('human',2,2);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('insect',6,0);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('squid',0,10);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('fish',0,0);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('centipede',100,0);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('table',4,0);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('armchair',4,2);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('phonograph',0,1);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('tripod',3,0);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('Peg Leg Pete',1,2);
MariaDB [k12345_db]> INSERT INTO limbs (thing,legs,arms) VALUES('space alien',NULL,NULL);
```

Here's a tip for entering the `INSERT` statements more easily: after entering the first one, press the up arrow to recall it, press Backspace (or Delete) a few times to erase characters back to the last open parenthesis, then type the data values for the next statement.

The table you just created is named limbs and contains three columns to record the number of legs and arms possessed by various life forms and objects. The physiology of the alien in the last row is such that the proper values for the arms and legs columns cannot be determined; NULL indicates "unknown value".

3. Verify that the rows were added to the limbs table by executing a `SELECT` statement:

```
MariaDB [k12345_db]> SELECT * FROM limbs;
```

You should get the following output:

```
+--------------+------+------+
| thing        | legs | arms |
+--------------+------+------+
| human        |    2 |    2 |
| insect       |    6 |    0 |
| squid        |    0 |   10 |
| fish         |    0 |    0 |
| centipede    |  100 |    0 |
| table        |    4 |    0 |
| armchair     |    4 |    2 |
| phonograph   |    0 |    1 |
| tripod       |    3 |    0 |
| Peg Leg Pete |    1 |    2 |
| space alien  | NULL | NULL |
+--------------+------+------+
```

The `SELECT` statement is used to retrieve information from a database. The most basic SELECT statement has only 2 parts:

`SELECT` *columnName* (1)
`FROM` *tableName* (2)

where: (1) indicates what columns you want to return and (2) indicates what table(s) those columns come from. If we want to retrieve all of the information about all of the columns (as in the previous example) in a table, we could use the asterisk (*) as a shortcut for all of the columns.

4 Getting Information about your Database

Let's start by taking a look on the database.

1. To know which tables are in your database type the command `SHOW TABLES;` and press `Enter`.

```
MariaDB [k12345_db]> SHOW TABLES;
```

You should get the following output:

```
+----------------------+
| Tables_in_k1512789db |
+----------------------+
| limbs                |
+----------------------+
1 row in set (0.00 sec)
```

The `SHOW TABLES` command lists all the tables in a database. In this case, you only have the table named `limbs` that you have just created.

2. To see information about the type of information stored in each table (that is the different columns on the table) use the command `DESCRIBE` as follows:

```
MariaDB [k12345_db]> DESCRIBE limbs;
```

This command gives you information about the columns in the specified table. In this case, we get the information about the columns on the weather table:

```
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| thing | varchar(20) | YES  |     | NULL    |       |
| legs  | int(11)     | YES  |     | NULL    |       |
| arms  | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

`Field` indicates the column name, `Type` is the data type for the column, `NULL` indicates whether the column can contain `NULL` values (which are empty or unknown values), `Key` indicates whether the column is indexed, and `Default` specifies the column's default value. `Extra` displays special information about columns.

5 Adding More Databases

1. Create another database (set the name to your user name and test, e.g., k12345_test) using the web application.
2. To get information about your databases use the command `SHOW databases;`

```
MariaDB [k12345_db]> SHOW databases;
```

   You should get the following information:

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| k12345_db          |
| k12345_test        |
+--------------------+
```

   `information_schema` is a database containing metadata, whereas the other databases are the databases you have just created.

3. The commands you run will affect the database you are currently using. To change the database you are using to the one you have just created execute the following command:

```
MariaDB [k12345_db]> USE k12345_test;
```

   Now you can see in the prompt that the database has been changed (`MariaDB [k12345_test]>`)

4. Create a simple table in the database k12345_db by typing the following SQL statement and press `Enter`:

```
MariaDB [k12345_test]> CREATE TABLE test (testText VARCHAR(20));
```

5. Retrieve the information about the tables in k12345_db. Only the table test should appear.
6. Since the table limbs is not in the database you are currently using, if you want to retrieve information from it you need to write the name of the database before the table name as follows:

```
MariaDB [k12345_test]>  Select * FROM k12345_db.limbs;
```

7. To remove the test database you can use the web application or the following command:

```
MariaDB [k12345_test]>  DROP SCHEMA k12345_test;
```

   Now you can see in the prompt that no database is selected (`MariaDB [(none)]>`)

8. Quit the mysql connection:

```
MariaDB [(none)]> quit;
```

6 Executing SQL Statements Read from a File

You may want to read statements stored in a file so that you need not enter them manually. Follow the steps below to do it:

1. To create a simple file to select the rows in the limbs table, execute the following command on the Unix terminal:

```
echo 'SELECT * FROM limbs' > select.sql
```

This creates a new file named `select.sql` in your working directory. In the terminal, if the notation `> fileName` is appended to any command (e.g., `echo` for printing) that normally writes its output to the terminal, the output of that command (e.g., the String `'SELECT * FROM limbs'`) will be written to the file named `fileName` (e.g., `select.sql`) instead of your terminal.

2. List the contents of the file your have just created:

```
less select.sql
```

As you can see it contains the select query. You can edit that file with a text editor and add more statements.

3. Then invoke mysql and redirect its input to read from that file:

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName<select.sql
```

As the greater-than character > is used for output redirection, the less-than character < is used to redirect the input of a command. Statements read from an input file (e.g., `select.sql`) substitute for what you'd normally enter interactively by hand. This is convenient for executing a set of statements on repeated occasions without entering them manually each time. SQL scripts also are useful for distributing statements to other people (for your first coursework you will submit a set of script files).

You will see the following output:

```
thing           legs    arms
human           2       2
insect          6       0
squid           0       10
fish            0       0
centipede       100     0
table           4       0
armchair        4       2
phonograph      0       1
tripod          3       0
Peg Leg Pete    1       2
space alien     NULL    NULL
```

7 Database Backup

The `mysqldump` client utility performs logical backups, producing a set of SQL statements that can be executed to reproduce the original database object definitions and table data. It dumps one or more MySQL databases for backup or transfer to another SQL server. The `mysqldump` command can also generate output in CSV, other delimited text, or XML format.

1. Create a backup of your database as follows:

```
mysqldump -u yourUserName -p -h mysql2.nms.kcl.ac.uk  -P 33306 yourDatabaseName >
    backup.sql
```

Note that `mysqldump` also needs connection parameters. This commands creates a backup that contains the contents of your database; i.e., the limbs table and its rows. In particular, the output of the mysqldump command is redirected to a file named `backup.sql`.

Now we are going to remove the rows of the limbs table and restore them using the backup file.

1. Connect to MySQL using your connection command line:

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk  -P 33306 yourDatabaseName
```

2. Delete all rows in the limbs table:

```
MariaDB [k12345_db]>  DELETE FROM limbs;
```

3. Verify that the rows were removed from the limbs table by executing a SELECT statement:

```
MariaDB [k12345_db]> SELECT * FROM limbs;
```

   You should get an empty set.

4. Quit the mysql connection:

```
MariaDB [k12345_db]> quit;
```

To restore the contents of limbs we can redirect `mysql`'s input to the backup file. By default, the `mysql` program reads input interactively from the terminal, but you can feed it statements using other input sources such as a file or program.

1. To invoke mysql and redirect its input to read from the backup file execute the following command:

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName < backup
    .sql
```

   In this case, we are redirecting the input of the `mysql` command to the file `backup.sql`. In this example, the command `mysql` will read the contents of file `backup.sql` and use them as input.

2. Connect to MySQL using your connection command line:

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName
```

3. Verify that the rows have been restored by executing a SELECT statement:

```
MariaDB [k12345_db]> SELECT * FROM limbs;
```

   You should get the following output:

```
+--------------+------+------+
| thing        | legs | arms |
+--------------+------+------+
| human        |    2 |    2 |
| insect       |    6 |    0 |
| squid        |    0 |   10 |
| fish         |    0 |    0 |
| centipede    |  100 |    0 |
| table        |    4 |    0 |
| armchair     |    4 |    2 |
| phonograph   |    0 |    1 |
| tripod       |    3 |    0 |
| Peg Leg Pete |    1 |    2 |
| space alien  | NULL | NULL |
+--------------+------+------+
```

   That is the rows that we have deleted have been restored from the backup file.

4. Quit the mysql connection:

```
MariaDB [k12345_db]> quit;
```

Note that in the backup file has been given as a relative path name, and the name is interpreted relative to the directory in which the `mysql` program was started.

8 Creating a Directory for Backups

1. Create a directory named db:

```
mkdir db
```

2. Create a new backup of your database and save it into the db:

```
mysqldump -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName > ./
    db/backup.sql
```

3. List the contents of the db directory to check that the backup has been created:

```
ls ./db
```

4. See the contents of the backup.sql file.

5. Restore the contents of the database using the file you have just created:

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName < ./db/
    backup.sql
```

Don't remove the limbs table as we will use it in the lab next week.

9 Accessing the NMS Server using Alternative Networks

If you plan on using your own laptop or you plan to access the NMS database from home, note that wifi or other networks will not allow direct access to the NMS MySQL server.
To connect to the MySQL Database from other networks you will first have to connect to the databases server using your kings credentials, not your database credentials. From the command prompt the SSH command to connect to the database server is (where k123456 is your username):

```
ssh yourUserName@bastion.nms.kcl.ac.uk
```

Once on `bastion.nms.kcl.ac.uk`, connect to the MySQL server with the usual command:

```
mysql -u yourUserName -p -h mysql2.nms.kcl.ac.uk -P 33306 yourDatabaseName
```

10 Personal Study

If you finish these activities early then you can start this personal study in the lab, otherwise you should be working on these tasks in your own time.

1. For the next lab you need to be familiarised with simple SELECT statements. Please have a look at:

   `https://dev.mysql.com/doc/refman/5.7/en/programs.html`

   and

   `http://www.mysqltutorial.org/mysql-select-statement-query-data.aspx`