



In this lab, you will again connect to the MySQL server, but this time, we'll use the PHPMyAdmin web interface to do so. Note that you will connect to the same database, so the username (NetId) and password will be the same, but this time you'll use your web browser instead of PuTTY and login.fos.auckland.ac.nz. You will also need to keep the lab sheets from the last two lectures handy.

You will learn how to use some of the features of PHPMyAdmin to:

- send SQL queries to the DB server (inserts, selects, updates, alter table statements)
- export the contents of your database to a text file as SQL statements (handy for backing up a database)
- delete records from a table
- drop tables
- import scripts with SQL statements (handy for moving databases around)

Note again that these lab sheets are not assessed, but they may be discussed in tutorials and their content is examinable!

Estimated time to complete this lab: 30 minutes.

### A) PHPMyAdmin

Point your browser at: <https://www.cs.auckland.ac.nz/courses/compsci280s1c/phpmyadmin/>

Log in with your NetId and the database password that was e-mailed to you. After you have logged in, select your database on the left and click on it. The top of the window should now show something like this:

 [studdb-mysql.cs](#) ▶  [stu\\_fred001\\_COMPSCI\\_280\\_C\\_S1\\_2011](#)

(Don't worry that it says cs rather than fos here – just some leftover confusion from some reorganisation within the Faculty of Science).

Below this heading, you should see the tabs:

 [Structure](#)  [SQL](#)  [Search](#)  [Query](#)  [Export](#)  [Import](#)  [Operations](#)

#### [Structure](#)

Click on Structure and you'll see a list of all tables. If you've done the previous exercises, this should show your **city** table. In databases with more than one table, you'll see these listed here, too.

Further down, there's a field where you can start the creation of a new table, bypassing the need to enter actual SQL statements. We won't use this now but you're welcome to try this out later by creating another table of your choice (as always: experiment, experiment, experiment!)

To the right of your **city** table, there are six symbols. The first lets you browse the table (so you can see and manipulate the records), the second shows you the structure – basically the result of the CREATE TABLE and ALTER TABLE statements you have used. The remaining four let you search in the table, insert a record into the table, empty the table of all records, and finally drop (delete) the table itself.

Click on the first symbol, and you'll see a list of all records in your table. Next to each record, there's a pen symbol to let you edit (update) that record, and a red X to delete the record. Edit one of your records to convince yourself that it works. Then click the browser's back button until you get back to the structure tab of the database.

Now click on the second symbol, which shows you the structure of the table. Note that you can use this to change table definitions without having to know the ALTER TABLE syntax. As the semester progresses, you are welcome to use this feature in your assignments and for your other projects.

For now, go back to the structure tab of the database and click on its neighbour, SQL.



The SQL tab is very simple: It lets you input one or more SQL statements, such as inserts, updates, selects etc. in a very similar way to the command line interface you used in the first two lab sheet revisions. That's all, really, nothing fancy. One useful feature is the option to have the query shown again. This allows you to edit it in the case of multiple inserts or updates, or when you have a complex select statement and want to construct it step-by-step, or of course when you've made a mistake with any sort of query and you want to try again because you got an error.

### **B) A tale of two cities**

Insert two more records into the city table, one via a SQL query in the SQL tab, and one via the insert button next to the **city** table entry in the Structure tab. Do an appropriate SELECT in the SQL tab to verify that the inserts were successful.

### **C) Exporting the database**

Click on the Export tab. Untick "Extended inserts" under "Data" and then click on "Go". Save the file on your drive and open it with a text editor (e.g., Notepad++, Eclipse).

You'll see a commented list of SQL statements that creates the tables in the database and then populates them with insert statements. Keep this file. Do not proceed beyond here if you cannot see the statements that create your table(s) and insert the data you inserted manually.

### **D) Deleting a record**

In this section, use the SQL tab only.

First task: Delete the city with the smallest population, using a WHERE clause specifying the **cityId**. Do an appropriate SELECT to verify that the delete was successful and hasn't affected any of the other records.

Next: Delete all cities from a particular country of your choice in the table. Again, use an appropriate SELECT to verify that the deletes were successful and haven't affected any of the remaining records.

Finally: Delete all records from the table so it ends up empty. Again check with a SELECT to ensure the dataset is empty.

### **E) Dropping the table**

Now, let's get rid of the table altogether with a DROP TABLE statement. Use:

```
show tables;
```

in the SQL tab and/or the Structure tab to verify that you've zapped the table. Note: A table doesn't have to be empty to be dropped. If you drop a table that isn't empty, all records in it are deleted. Caution: An RDBMS will not warn you about this!

### **F) Phoenix from the ashes**

If you've just gotten a bit annoyed with me for letting you delete all the stuff you painstakingly entered, there's help. Click on the Import tab. There, you'll find a file upload field. Click on "Browse" and find the text file with the SQL statements you saved above. Select it and click on "GO". After a few seconds, PHPMyAdmin should tell you that it has successfully executed a number of queries.

Now go to the Structure tab to verify that your table is back and browse it to verify the records have been restored, too.

### **G) Population growth**

Assuming for a moment that all data you have found about the cities is a few years out of date, their population will have grown, say by 5%. Using the SQL tab, update all records in the **city** table such that their population increases by this amount. Note that an UPDATE without a WHERE clause will update all records in the table, but you need to ensure that each record is updated by the proper amount. Write the query down here:

```
UPDATE _____;
```

Do an appropriate SELECT in the SQL tab to verify that the updates were successful. Plan B if this fails: update individual records to get back to the original state, or repeat E) and F).