

Lab 1

Contents

Overview	3
SQLite and your environment.....	3
Installing sqlite3	3
Running sqlite3	3
Dot commands in sqlite3 v SQL (and a note on SQL).....	4
 Figure 1: How thing should look in gui, related to commands.....	4
Figure 2: Creating person in cmd.....	6

Overview

All the labs are important, but in its own way this lab is *the* most important. If you follow it carefully and concentrate on what you are doing, then you will be well set up to use sqlite3, the environment you are working with for the rest of the course **and the coursework**. This lab introduces you to:

- SQLite and how it can be launched (a reminder of what you have already been shown elsewhere).
- How to create a database tables using SQL in the sqlite3 shell.
- The difference between an SQL environment, such as sqlite3, and SQL itself
- Some ‘.dot’ commands within sqlite3

Once you complete the lab, it is a good plan to go over what you did in your Terminal/CMD, maybe copy some of the commands you entered and make notes on what they did. You could even repeat the lab again, just to get some practice. The reason is, it’s not enough to merely *understand* what you are doing... you should *practice* what you do again and again. In the end, you won’t just understand something, you be very good at it too.

By the way...if you notice any mistakes in any of the labs, please contact me so that I can correct them. Much appreciated!

SQLite and your environment

Installing sqlite3

There are some instructions on GCU Learn (and some videos) regarding sqlite3. See under the Software section, and sqlite3. Before you go any further, ensure that you are able (on whatever computer you are using) to enter and exit the sqlite3 inner shell.

Running sqlite3

Based on the instructions and the meaning of LAB_HOME (explained in the sqlite link on GCU Learn), ensure that you are inside the lab1 folder by:

```
<cd LAB_HOME\1>
```

Following the method with which you are familiar with (again, see GCU Learn Software/sqlite3), launch sqlite3 with the argument (lab1.db). Then when you are successfully inside the sqlite3 shell type:

```
<.databases>
```

See Figure 1 you should see my inputs to the command line, which achieved the following:

- Changed directory to folder 1, which is in LABS_HOME.
- Entered into the sqlite3 shell
- Use an sqlite3 ‘.dot’ command <.databases> to list the databases

And you should also see from the last command that lab1.db was written to disk in folder '1'. Consequently, lab1.db can now be seen in the Windows 'File Explorer' (or 'Finder' on Mac) if you open the folder '1' from there. Its worth doing this, just to check that you have created your database before you do the work for the lab; you should do this at the beginning of each lab and always double check you created the lab*.db file in the appropriate '*' folder, where * is in the set {1, 2, 3, 4, 5, 6}.

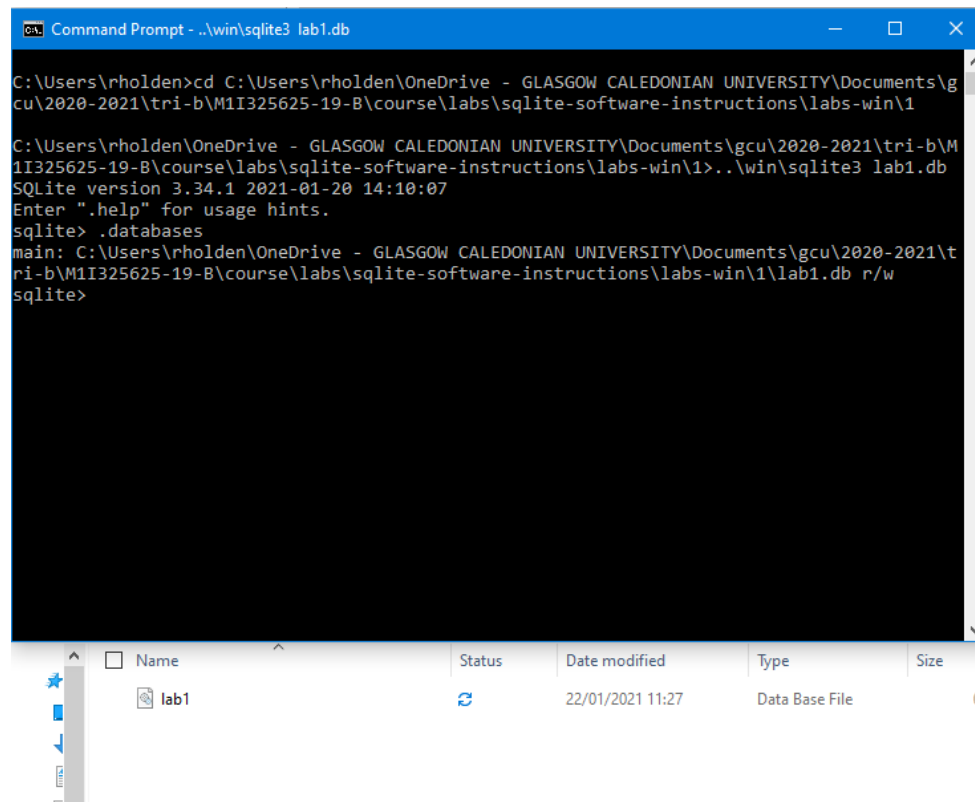


Figure 1: How thing should look in gui, related to commands

Dot commands in sqlite3 v SQL (and a note on SQL)

When you are in the sqlite3 shell, there are a set of sqlite3 commands. Don't misunderstand what these are, conceptually; for example, these are nothing to do with SQL (Structured Query Language)...

SQL is best thought of as standard language for the actions of creating, reading, updating, and deleting (**CRUD**) data and databases. There are many implementations of this language and not all vendors implement all the SQL standard, which is why different versions of SQL will be found between different vendors and database environments (sqlite3, MySQL, Oracle). So, you can think of the particular version of SQL as part of the particular environment you are using (e.g., sqlite3), but that particular SQL might not work if you use it to query inside a different environment (e.g., MySQL) because of the potential differences of SQL implementation; I say might not, of course, because each vendor implements 'part of' the standard and logic therefore tells us that the different implementations (in sqlite3 and MySQL, for example) might overlap. Don't take the risk though... if you write your own SQL the in MySQL environment (which we are not using this on this course), instead of

sqlite3, then also your coursework, you take big risks... for the coursework you will be using sqlite3 and I will run your code in the sqlite3 shell.

...so, back to the 'dot' commands. Each different environment (MySQL, sqlite3) will have its own special 'tools'. In sqlite3, these commands prefixed with a dot. You just used one such command to list the databases currently 'attached' to the sqlite3 shell:

<.databases> = List names and files of attached databases

And if you use another:

.help

You will see a list of the tools available:

.archive ...	Manage SQL archives
.auth ON OFF	Show authorizer callbacks
.backup ?DB? FILE	Backup DB (default "main") to FILE
.bail on off	Stop after hitting an error. Default OFF
.binary on off	Turn binary output on or off. Default OFF
.cd DIRECTORY	Change the working directory to DIRECTORY
*	
*	
.system CMD ARGS...	Run CMD ARGS... in a system shell
.tables ?TABLE?	List names of tables matching LIKE pattern TABLE
*	
*	
etc	

In the remainder of this lab, start to take a look at the documentation for sqlite3, available here:

<https://www.sqlite.org/cli.html>

... and explore some of the SQL commands suggested in the documentation.

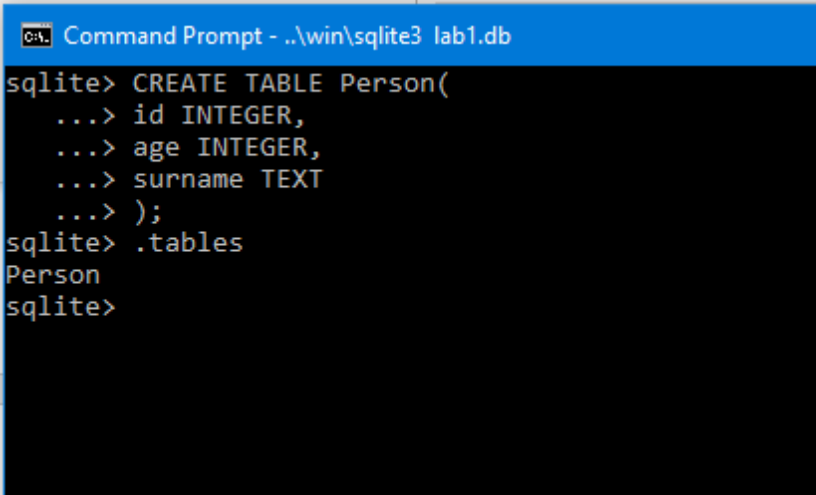
For example, see if you can create some tables of your own that follow the general form on the left, below, for example like the one on the right:

General form of table	Specific example
CREATE TABLE table_name(attribute1 TYPE, attribute1 TYPE, attribute1 TYPE);	CREATE TABLE Person(id INTEGER, age INTEGER, surname TEXT);

... see if you can create some tables. Once you have created some tables (try and create 5 different ones) in your database, then you can list them by typing:

<.tables>

An example of how this would look in windows CMD is shown in Figure 2.

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt - ..\win\sqlite3 lab1.db". The prompt is "sqlite>". The user has entered the command "CREATE TABLE Person(" followed by three indented lines: "id INTEGER,", "age INTEGER,", and "surname TEXT". The prompt is now "sqlite>". The user has entered ".tables" and the output "Person" is displayed. The prompt is now "sqlite>".

```
C:\> Command Prompt - ..\win\sqlite3 lab1.db
sqlite> CREATE TABLE Person(
...> id INTEGER,
...> age INTEGER,
...> surname TEXT
...> );
sqlite> .tables
Person
sqlite>
```

Figure 2: Creating person in cmd.

N.B. Each separate line here, in CMD/Terminal, was 'entered' by pressing the return key. Please get used to doing this. Don't be lazy in arranging your code, such as

CREATE TABLE Person(id INTEGER, age INTEGER, surname TEXT);

That code above might work ok, but if you do this, your code will be much harder to read for you to correct the mistakes. So, order your code nicely, take your time, and concentrate on each character you type.

When you finished the lab, you can exit sqlite3 by typing **<.exit>** to return control to CMD/Terminal.

~~~~~