Net-centric introduction to computing

Databases I

m Jenkin + h Roumani

1. Introduction

Suppose that you need to represent a collection of LP's. LP's (or records) were how music was distributed or sold prior to the development of CD's and on-line music. Usually 12" in diameter, LP's are vinyl structures with music information embedded on both sides. The vinyl was sold inside a cardboard sleeve. These sleeves were organized into collections — often contained in milk crates that were re-purposed from the dairy industry.

Many people amassed rather large collection of these LP's. Such collections were not easily accessed. It takes some effort to search through a collection for a given album, or to find (for example) all albums pressed in a given year, or by the same artist. Albums were often stored in a particular order to make finding a specific album easier. For example, if the albums are sorted by the name of the artist, then finding albums by artist is easier. But there would be other orderings of albums that are useful, but changing the order could be a time-consuming process. This was a running joke in the movie Serendipity. One way of making it easier to represent one's record collection is to build a representation of the collection that can be represented in a computer. Such a representation could be used to search for specific albums, or albums with specific properties. From a computer science point of view, the basic problem here is the development of an appropriate software infrastructure to support the representation and manipulation of data that represents this information. Normally in computer science this type of representation is known as a database.

This lab deals with the nuts and bolts of databases. In particular in this lab you will learn how to manipulate a very simple database implemented in sqlite. This lab will not make you a database wizard, but it will introduce you to some of the details of what databases are and how they are used in web-based applications.

sqlite is a simple database tool found in most modern operating systems, and indeed exists as a standard way of storing data in Android devices. sqlite is, as it name suggests, a lightweight version of sql. It is not designed to support large numbers of users accessing it nor particularly large databases. That being said, its access language is the same as that used by much larger databases and sqlite itself can be found as the core representation scheme in almost every Android application.

A database is nothing more or less than an organized collection of data. Databases allow users to enter, access, manipulate and analyze data stored within them. Information within a database can be taught of as information organized in a collection of tables. Each table consists of data organized in terms of rows and columns. Each row represents a single record, while each column represents the data associated with that record. One way to think about a database table is to

consider a table in a spreadsheet program such as Excel. The data is organized into rows. Each column has the same format, but the data in a given row "belongs" to the record that the row represents.

Designing an appropriate collection of tables to represent a specific collection of information is a complex task. In this lab you will manipulate a rather simple database consisting of a single table that represents information related to a collection of records.

Although there are many commercial database systems available, many of them "speak" SQL. SQL stands for structured query language and is the standard language for communicating with a relational database system.

Each database system has a different user interface and uses different ways of storing the information contained within the database. Large database systems are designed to be extremely efficient, have sophisticated mechanisms for dealing with maintaining versions of the system as changes are made to the data, and include the ability to roll back changes if that is required. In this lab you will use a relatively simple database tool to manipulate a database that will be key to the database used in the next lab.

The database that you are going to build and manipulate in this lab and the next represents a record collection. In building this representation we need to decide what it is we are going to to represent in the software. For an album you could imagine wanting to represent

- The title of the album (call this album)
- The artist of the album (call this artist)
- The year of pressing (call this pressing)
- The artwork of the album (call this cover)

We will also want to assign to each album a unique id (called id) that will allow us to refer to an album in a unique way.

Given this structure you can imagine that there are number of operations that we might be interested in performing on this representation.

Inserting a new record into the database. If you purchase a new LP, then a new record should be inserted into the database.

Deleting a record from the database. If you loose an LP, then the corresponding record should be deleted from the database.

Selecting records from the database. Find all LP's that have a specific property, such as being released in a given year or being played by a given artist.

In the next lab you will develop the front end software — the user interface — to deal with the database. Here you will deal with the nuts and bolts that makes the database work.

The virtual machine provided with this course has a database system installed within it called sqlite3. Although you can build different databases with this tool, the default location for the database used in the next lab can be found in the directory db under the user's home directory. The database file that is used is called data.db. If at any time during this lab (or the next) you wish to return the database to its original state use the terminal program to go to that directory and execute the following commands

```
rm data.db

sqlite3 data.db <data.sql
```

2. sqlite

Sqlite is an embedded database system. That is, unlike most commercial database systems that require a dedicated server to actually store the data, sqlite is intended to be compact and to use ordinary disk files to represent the information stored in the database. A complete database with many tables and other elements is actually stored in a single file. This makes sqlite a good choice for small databases that are associated with lightweight processes such as those found in hand-held devices to maintain things list contact lists and the like.

sqlite comes 'standard' in a wide number of different operating systems. In particular it comes installed in the virtual machine used in this course. sqlite is accessed through the terminal application and is invoked using the command sqlite3.

Full documentation on sqlite and sqlite3 can be found on the sqlite home page www.sqlite.org. Examples of key sqlite3 commands are given below. See here for a short list of some of the commands that are available.

2.1 Some sqlite3 commands

Full and complete descriptions of all of the sqlite3 commands can be found in the readings given at the end of this document and also from various online sources including TutorialsPoint. The following give examples of how to do various things in sqlite3 in the context of representing a collection of music in a sql table.

Creating/using a database file.

```
sqlite3 my_database.db
```

This will start the process of creating a database file (here my_database.db) or make changes/provide access to the file .

Creating a table inside the database.

```
create table collection (
```

```
        id int primary key not null,
        album text not null,
        artist text not null,
        year int not null,
        cover text
);
```

This command creates a table of a given name (here collection). A table definition defines the columns in the table. Each column has an identifier. In the example above the columns are called id, album, artist, year and cover. Each column has a particular type. In the sample table created here, the column id and year are int's (integers) while all other columns are text (character strings). In some tables it is possible to have null entries for columns. In this table only the cover is allowed to be null. All other columns are marked as 'not null'. Finally, the id is the 'primary key'. That is, each value must be unique.

Inserting a record into a table.

```
insert into collection (101, 'Led Zeppelin IV', 'Led Zeppelin', 1971, 'foo.jpg');
```

This adds a new record into the table where the order of fields in the insert command matches the order of the declaration of the table itself. Types must match, and the inserted value for the primary key must be unique. There can be no record with id 101 in the table already.

Upon execution of this command the table will contain a new record for 'Zeppelin IV'.

Retrieving information from a table.

```
select album from collection with year = 1971;
```

This will retrieve all album values with year = 1971. If you want to retrieve all values use "select * from collection;".

Deleting records from a table.

```
delete from collection where year = 1971;
```

This will delete all records with year = 1971.

In the exercises below you will use these, and other sqlite3 commands to manipulate the database that you will use in the next lab.

3. The shell

The terminal application found under the applications menu in the virtual box provides access to a terminal emulator. This is an application that emulates what a terminal used to look like when accessing a computer. It provides a command line interface (CLI) to the computer. Within a command line interface commands are executed by typing them, and then terminating the command — causing it to execute — using the return key.

There are many different CLI's available for unix machines. By default, the one you are provided with in the terminal application runs the bash shell. Full details on the bash shell can be found here. Some useful commands include
ls - list files
rm filename - remove the file filename
cd directory - change directory to the directory given
man man - provide the manual page for man
sqlite3 file.db - run sqlite3 on the file file.db

4. Under the hood

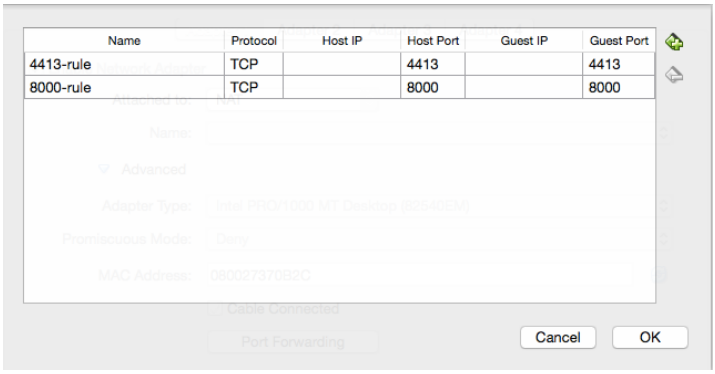
From the very first lab in this course you have used a service that was provided 'under the hood' that allowed you to access resources stored inside the virtual machine running on your laptop. If you think of the URL that you entered inside the html2apk application running on the Android, you entered a string like

http://10.0.1.1:8000/zip/lab0

to get lab0 from the virtual machine running on your laptop onto the Android device. How could this possibly have worked?

So you should be able to put the pieces together at this point in the course. The URL entered in the Android application specified a resource on 10.0.1.1 that was available on port 8000. The virtual box 'forwards' port 8000 requests to a service running inside the virtual box. This service looks for a resource called zip/lab0 which (as you can imagine) creates a zip archive of the contents of the folder server/lab0 and transmits it back to the Android device. (So no magic here, just some simple software.)

But how does this work? First, lets verify that port 8000 works as described above. If you look at the settings tab in the VBOX application, and then from there to Networking, you will see a 'port forwarding' tab. Clicking on this reveals that



TCP requests to port 8000 on your laptop are transmitted to port 8000 in the virtual box. Within the virtual box a service is started when the virtual machine boots an application (written in JavaScript) that can be found in /eecs/fac/package/nodeServer is executed. This listens to requests on port 8000 and

responds to them. Looking at the code shows that the code actually supports a number of commands. For fun try (from somewhere outside of the virtual box), using a browser, the following (where 10.0.1.2 is the ip address of your machine)

http://10.0.1.2:8000/ip

This should return a page that displays your IP address. And of course there is the zip resource you have been using from the first lab. But there is also a command that exposes a database stored on the virtual machine to the outside world. Execute

http://10.0.1.2:8000/sql?query=select * from collection

This will return a json formatted string of the result of the query. Note that not all browsers can grok json strings, although firefox certainly can. The database that is served is stored in the file db/data.db in your account on the virtual machine. This file was created by executing the command

sqlite3 data.db <default.sql

using the files in that directory. You can modify the contents of data.db by stopping the server, changing the file data.db, and then restarting the server. To stop the server use the command

node_stop

and to start the server again use the command

node_start

5. Exercises

As with other labs in this course, lab exercises are broken down into three sections, A, B and C. Exercises in section A are Pre-Lab exercises. All Pre-lab exercises **must** be completed prior to attending your lab. You will not be allowed to participate in the lab if you have not completed these exercised prior to attending the lab. A backup lab is scheduled each week for students who miss/were unprepared for their normal lab time. You will also get much more out of each laboratory if you spend some time going through the B and C exercises for each laboratory before attending your laboratory session.

5A. Pre-lab

1. Use the terminal application to run the man (manual) command on the ls, mkdir, echo and cd commands.
2. Complete the pre-lab quiz on the course moodle page.

5B. In-lab

1. Use the terminal application to change your directory to /home/user/db and reset the database to the original version using the following commands. Before running these commands. Before executing these commands, look them up using either the internet or the man command to figure out what they will do.

```
node_stop
rm data.db
sqlite3 data.db <data.sql
node_start
```

2. The file data.sql is a text file containing commands that were used in the question above to re-populate the database that you deleted. Use the more command to look at the contents of data.sql. What is the name of the table that is created by this process and what fields does it define?

   more data.sql

3. The data.sql file you used above also inserts a number of records into the collection table. Based on the information in that file, what Beatles albums are inserted into the database?

4. Run the sqlite3 program to access the file data.db and execute the following commands. What do these commands do?

   .tables

   .schema collection

   select * from collection;

   select * from collection where year=1971;

   select artist,album from collection where year=1971;

   select album from collection where year=1971 and artist="Who, The";

   .quit

5. Run the sqlite3 program to access the file data.db and execute the following commands. What do these commands do? What would happen if you replace the 115 value with 110? Why?

   insert into collection (115, "Houses of the Holy", "Led Zeppelin", 1973, "junk");

   select * from collection where artist="Led Zeppelin";

   .quit

6. Run the sqlite3 program to access the file data.db and execute the following commands do?

   6.1. select * from collection;

   6.2. delete from collection where artist="Led Zeppelin";

6.3. select * from collection;

7. After doing the operations above, reset the database to its original format.
8. You can access the database from the web using the url

   http://10.0.1.2:8000/sql?query=XXX

   Replace the query XXX with 'select * from collection' (no quotes). Convince yourself that you can select, delete and add records to the database using the web-based interface. Document this in your eReport. Are changes in the database reflected in the database when accessed through the command line?
9. Copy the file default.sql to mine.sql. Edit mine.sql so that it adds an additional 10 records to the database. Note: the cover art must be a url to a jpeg image of size at least 100x100 pixels. Once you have done so, delete the file data.db and re-create the file data.db using 'mine.sql'. (Note: If you ever mess things up, simply recreate the database using default.sql, which you carefully made a backup of at the beginning of this step.)
10. Access the database through the web interface to convince yourself that it is working properly through this.

5C. Advanced

1. You may not find the current collection of albums align with your artistic interests. Delete the 'oldies' from mine.sql and replace them with albums more closely aligned with your interests.
2. Add the field condition to the collection table. Field should be a non-null text, and should have one of the values E, VG, G, U or P (standing for the condition of the album with E be excellent and P being poor. U here is unknown). Populate the database so that the album quality has a valid value from the set above.
3. A database typically has more than one table. Add a second table to the database called artists. Artists have a unique id and a non-null text name. Populate the artists table with the artists from your albums. Verify that you can

access both the collection and artists tables from both the command line and the web.

4. Further Reading

Aditya, S K and Karn, V K. (2014) *Android SQLite Essentials*, Packt Publishing.

Das, S. (2014). *SQLite for Mobile Apps Simplified*. Amazon Publishing.

Haldar, S. (2015). *SQLite Database System and Implementation*. Google Books.

Kreibich, J A. (2010). *Using SQLite*, O'Reilly Media.

Newman, C. (2004). *SQLite,* Sams Publishing.

5. Credits

The image on the front cover is from the Ngorongoro crater in Tanzania.