Homework 2

Intro to Relational Databases

This exercise introduces you to relational databases, databases that provide a conceptual model of "relations," which you can think of as "tables." Your goal is to explore the use of the SQLite command-line client to investigate the schema of a trivial relational database, and to practice the fundamental SQL queries of CRUD operations.

Step 1: Install SQLite 3

Using the text and online SQLite documentation, install a copy of SQLite on your development machine. You will always want to be able to execute *sqlite3* from anywhere on your filesystem, so be sure to add the location of the *sqlite3* executable to your PATH. You know this is right when you can start a shell, and execute *sqlite3* regardless of your current working directory.

Step 2: Open and Explore a Database

Using the *homework02a.sqlite.db* file, open the program with the SQLite command-line client. Issue the following commands, and observe the result.

- `.schema`
- `SELECT * FROM artists;`
- `.mode column`
- `.headers on`
- `SELECT * FROM artists;`
- `.exit`

You invoked four "dot-commands," specific to SQLite, and one SQL query (twice). Investigate the SQLite documentation to see more about these "dot-commands" and the SELECT statement.

Step 3: cRud

The "R" in CRUD refers to operations that allow you to read data that resides in a database. This manifests as the SELECT statement in SQL. Try all of these:

- `SELECT * FROM badname;`
- `SELECT * FROM artists;`
- `SELECT COUNT(*) FROM artists;`
- `SELECT name FROM artists;`
- `SELECT name FROM artists ORDER BY name;`
- `SELECT name FROM artists LIMIT 4;`
- `SELECT name, state FROM artists;`
- `SELECT name AS "artist name" FROM artists;`
- `SELECT id, name FROM artists ORDER BY name LIMIT 2;`
- `SELECT name FROM artists WHERE zip_code = '97701';`

The SELECT statement has many options, but, at minimum, notice how we typically must specify one or more column names, and use a FROM clause that specifies the name of the table.

Step 4: Conceptual and Physical Schemas

Now open the second database file, *homework02b.sqlite.db*. Issue the .schema command, and notice the new field in the schema. Don't see the difference? Open the previous database file and observe the schema, and look at the second database file again. You should notice that there is a new field in the table.

Now, issue one of the previous queries:

```
SELECT name, state FROM artists;
```

Notice how, despite the additional field in the table, the same query still works.

Now, imagine that one day the *physical* file format of a SQLite database changes. Should your query change? Why or why not?

Step 5: Crud

The "C" in CRUD refers to operations that add data to the database. For example, adding a new artist's details to the *artists* table. (Do not confuse this with the CREATE TABLE command, which is for modifying the database schema.) Try the following commands:

- `INSERT INTO artists(name) VALUES('foo');`
- `INSERT INTO artists(name, phone_number, street_address, city, state, zip_code, country) VALUES('Britney Spears', 'fee', 'fi', 'fo', 'fum', 'foo', 'bar');`
- `SELECT * FROM artists;`

Master the INSERT statement syntax. See our text and the SQLite documentation.

Step 6: crUd

The "U" in CRUD refers to operations that modify existing data in the database. For example, changing an artist's address to "1500 SW Chandler Ave.". Try the following commands:

- `UPDATE artists SET country = 'USA' WHERE name = 'Britney Spears';`
- `SELECT * FROM artists;`

Notice how the country value has changed from "bar" to "USA". Pay close attention to the UPDATE statement above; how did the WHERE clause affect the UPDATE operation?

Now try another command:

- `UPDATE artists SET country = 'Oh no!';`
- `SELECT * FROM artists;`

What do you see? What happened? Why is the WHERE clause important?

Try one more UPDATE command:

- `UPDATE artists SET phone = '123-123-1234', state = 'OH';`
- `SELECT * FROM artists;`

Notice how you can specify multiple field-value pairs in the SET clause of a single UPDATE statement.

Master the UPDATE statement syntax. See our text and the SQLite documentation.

Step 7: cruD

The "D" in CRUD refers to operations that delete data from the database. Careful with this one! It's easy to delete all the records in a table if you don't know how the DELETE command works. Try it now:

- `DELETE FROM artists WHERE name = 'chickenbutt';`
- `SELECT * FROM artists;`

Notice how the artists remain intact - there were no records with the name "chickenbutt" in the artists table. (Unless you previously added an artist with the name "chickenbutt," in which case you must like hardcore punk rock, because only a punk band would name themselves "chickenbutt.")

Let's really delete a record from the database:

- `DELETE FROM artists WHERE name = 'Bananarama';`
- `SELECT * FROM artists;`

Notice how the greatest artist from the 80s was deleted from the database.

Now try the following command:

- `DELETE FROM artists WHERE zip_code = '97703';`
- `SELECT * FROM artists;`

How many records were deleted? Why?

Now, let's make an intentional mistake, so you don't unintentionally make it in the future. You don't want to be the person who brings all of Netflix down, do you? You can make a copy of the database file if you want, but you can also just re-download the original later, if you like.

- `DELETE * FROM artists;`
- `SELECT * FROM artists;`

Great, the Internet just broke. (What happened? Why?)

Master the DELETE statement syntax. See our text and the SQLite documentation.

Pro Tips

SQL syntax isn't really case-sensitive, but you should treat it as such.

Observe the conventions of SQL syntax, eg. operations in ALL_CAPS.

SQL is based on an agreed-upon standard - you can "talk SQL" to any relational database management system. However, each vendor may have additions to the language, specific to their RDBMS, and not all systems support all of the features defined in the SQL language.

Although the words SELECT, UPDATE, DELETE, and INSERT are imperative verbs, SQL is a *declarative* language - we describe the operation we want, but the DBMS controls *how* it fulfills our request.

In practice, that previous statement isn't entirely true. (Wat?!)