DB Class Notes

Followup for CSC330 Database Theory & Applications Spring 2022

"Be the shell!": in-class assignment

See also screencase video on YouTube (12 min).

1. Start SQLite with header on and column mode switched on from the command line (to find out, look at `sqlite3 --help`).

   Solution: On the command line[1], enter

   ```
   $ sqlite3 -header -column~
   ```

2. Check that you don't have a persistent database with `.database`.

   Solution:

   ```
   sqlite>  .database
   ```

3. Open your (existing) database `sqlite.db` with `.open`

   Solution:

   ```
   sqlite>  .open sqlite.db
   ```

   This database does not need to exist yet - if it does not, the binary file is created as an empty file in the directory where you were when you opened SQLite.

4. Check that you're now writing to `sqlite.db`

   Solution:

   ```
   sqlite>  .database
   ```

   You could also enter `.show` - the last line of its output shows the currently active database. You can change databases and have as many open as you like.

5. Check that in fact header is `ON` and the mode is `column`

   - with `.show` to show all output values
   - with `SELECT`

   Solution:

   ```
   sqlite> .show
   sqlite> SELECT * FROM customer;
   ```

This presumes that you have created at least one table named `customer`, and that it is in your database `sqlite.db`, and that the table has at least one row - otherwise nothing will be displayed.

6. Switch the output to a file with `.output sqlite.sql`

   Solution:

   ```
   sqlite> .output feb17.sql
   ```

   `.output` is one of the settings you saw with `.show`. It shows where the output is directed. The default is `stdout`, or the screen. You have now reset this to pipe all output into a file `sqlite.sql`.

7. Dump the content of your database with `.dump`

   Solution:

   ```
   sqlite> .dump
   ```

   This writes a version of the binary `.db` file to a text file `.sql`. It contains all the information necessary to recreate the database. This `.sql` can be used to port the database. On another computer, SQLite will recreate `sqlite.db` from `sqlite.sql` by importing it with the `.read` command.

8. Switch the output back to `stdout`

   Solution:

   ```
   sqlite> .output stdout
   sqlite> .show
   ```

   Until we redirect the output to `stdout`, every output we generate, including error messages, will be redirected to the `.sql` file.

9. Dump the content of your database again.

   Solution:

   ```
   sqlite> .dump
   ```

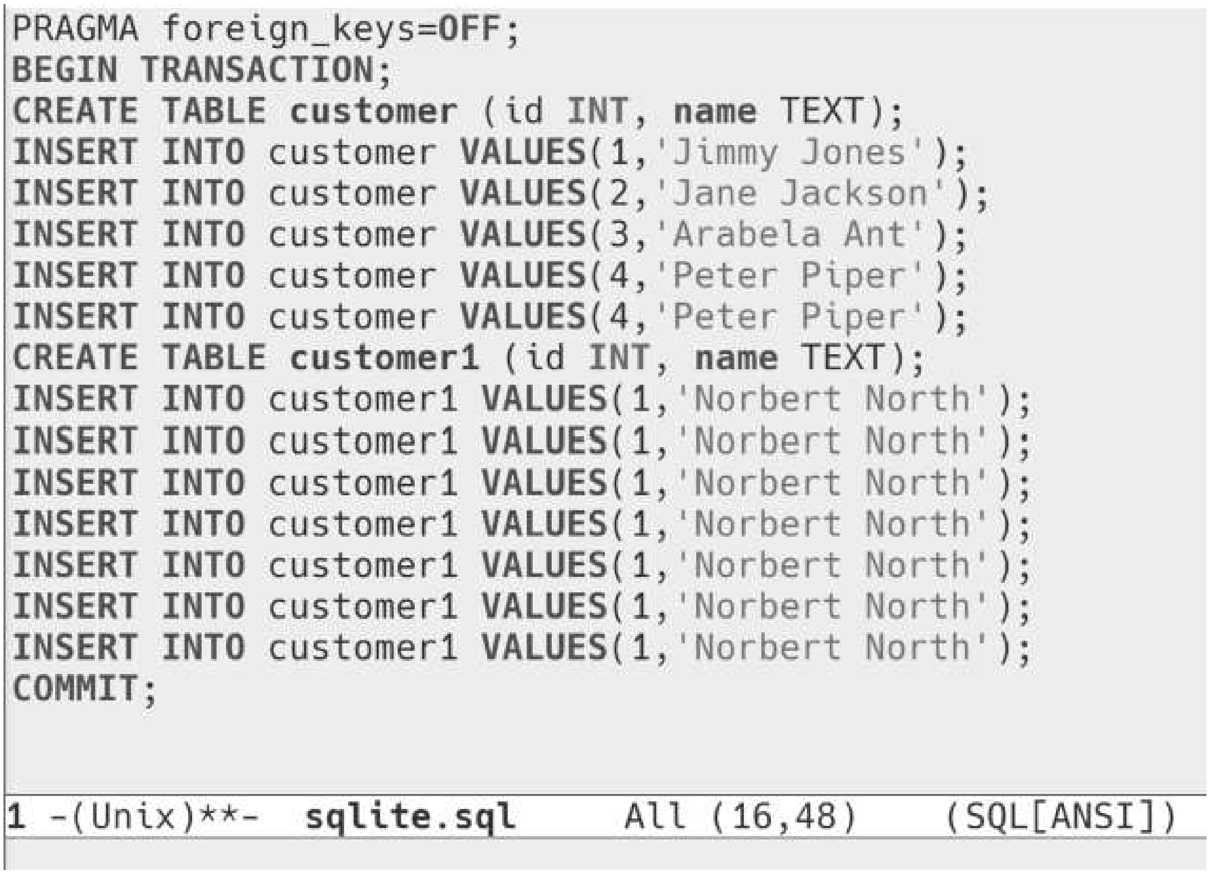
   Instead of dumping to an SQL file, we are now dumping the database to the screen.

10. Leave the SQLite shell and look at `sqlite.sql`

    Solution:

    ```
    sqlite> .q
    ```

    Depending on your PC (it doesn't seem to work under MacOS), the `.shell` command can be used to peek beyond the SQLite shell, and you could use this command to look at the directory listing - under Windows: `.shell DIR`.



```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE customer (id INT, name TEXT);
INSERT INTO customer VALUES(1,'Jimmy Jones');
INSERT INTO customer VALUES(2,'Jane Jackson');
INSERT INTO customer VALUES(3,'Arabela Ant');
INSERT INTO customer VALUES(4,'Peter Piper');
INSERT INTO customer VALUES(4,'Peter Piper');
CREATE TABLE customer1 (id INT, name TEXT);
INSERT INTO customer1 VALUES(1,'Norbert North');
INSERT INTO customer1 VALUES(1,'Norbert North');
INSERT INTO customer1 VALUES(1,'Norbert North');
INSERT INTO customer1 VALUES(1,'Norbert North');
INSERT INTO customer1 VALUES(1,'Norbert North');
INSERT INTO customer1 VALUES(1,'Norbert North');
INSERT INTO customer1 VALUES(1,'Norbert North');
COMMIT;



1 -(Unix)**-  sqlite.sql     All (16,48)    (SQL[ANSI])
```

Figure 1: sqlite.sql with the sqlite.db database dump

Footnotes:

[1] Org-mode has 11 different header arguments (see here) - you can initialize SQLite with the header shown below. A separate session buffer is not supported[2] - SQLite is run outside of Emacs and the results are displayed here. For a session demonstrating SQLite commands (which begin with a `.`), Org-mode is not well suited. For SQL commands, it is well suited.

```
#+begin_src sqlite :db ./sqlite/sqlite.db :header :column :results output
  ...
#+end_src
```

[2] Such a session buffer opens e.g. when you start R with `M-x R`.

Author: Marcus Birkenkrahe
Created: 2022-02-17 Thu 20:43
Validate