Lab Exercise 06
Introduction to SQLite 3

Written and prepared by Owen Riddy

Lab Exercises

SQLite 3 is a popular database because in practice it requires no configuration and is designed with a cooperative spirit regarding how it is used. Indeed, it is not suitable for many uses as a database because it is too forgiving. It makes an excellent choice for an application that wants to store data somewhere for private use.

Exercise 1 - SQLite 3 Familiarisation

Goal

Prepare a SQLite 3 database and become familiar with the program context.

References

- https://www.sqlite.org/docs.html -> Tools -> Command-Line Shell (sqlite3.exe)
- The SQLite command reference (accessible with .help inside SQLite)

Instructions

0) Download required files.

lab6.sql

1) Start SQLITE

```
       $ sqlite3 --version
       3.34.1 2021-01-20 14:10:07 10e20c0b43500cfb9bbc0eaa061c57514f715d87238f4d835880cd846b9ealt1
       $ sqlite3
```

2) Familiarise yourself with the help command

```
   sqlite> .help
```

3) Open a database file

```
       sqlite> .shell whoami
       sqlite> .shell hostname
       sqlite> .shell pwd
       sqlite> .shell ls
       sqlite> .databases
       sqlite> .open lab6.db
       sqlite> .databases
```

Observe that many of these commands are shell commands (ie, they are being run in a shell and then being displayed in SQLite). They are to orient you so you can check you are operating on the right computer and directory. HINT: Use the Up arrow key to access previous commands. This will save your valuable time.

4) Load data into the database

```
       sqlite> .read lab6.sql
       sqlite> .tables
       Actors     AppearsIn  BelongsTo  Directors  Directs    Movies
       sqlite> SELECT * FROM Actors LIMIT 3;
       1|13|Nick|m
       2|Abagnale Jr.|Frank|m
       3|Abbott|Dalton|m
```

5) Observe the relation schema

```
       sqlite> .schema
       CREATE TABLE Movies (
         id          integer,
         title       varchar(256),
         year        integer check (year >= 1900),
         primary key (id)
       );
       CREATE TABLE BelongsTo (
       ...
```

6) Quit the database

```
       sqlite> .quit
       $
```

7) Check that the database can be opened without error.

```
       $ sqlite3 lab6.db
       SQLite version 3.34.1 2021-01-20 14:10:07
       Enter ".help" for usage hints.
       sqlite>
```

Outputs

You should now have a database file to work with for the remainder of the lab. It will now be assumed that you can open and close SQLite and access the database as appropriate.

Extensions

8) Window Functions were introduced in SQLite 3.25. Are they supported on a CSE lab? The CSE Labs use the Debian GNU/Linux 11 (bullseye) distribution - by researching on the internet, can you determine what year this operating system was released?

9) It is possible to run the bash shell inside SQLite! For example:

```
sqlite> .shell bash
$ pwd
/import/adams/1/z3251243/lab
$ exit
exit
sqlite>
```

Some interesting extension questions to test your understanding of shells: If you start a bash shell in SQLite and change to a different directory, will SQLite now use that directory by default after you exit bash? If you change directory in SQLite using `.shell cd` will it now use that directory by default? If you start a bash shell in SQLite, what directory does it start in?

Exercise 2 - Recovering from Errors

Goal

To debug simple errors in an SQL file.

References

- The SQLite command reference (accessible with .help inside SQLite 3)
- The SQLite SQL dialect:
- https://www.sqlite.org/docs.html -> Programming Interfaces -> SQL Syntax
- https://www.sqlite.org/docs.html -> Programming Interfaces -> Result and Error Codes

Instructions

0) Download required files.

ex02_bad.sql

1) Attempt to load the file into SQLite using:

```
sqlite> .read ex02_bad.sql
```

2) Observe the errors! Fix the script so that it runs properly.

Outputs

The output of the fixed script is expected to be:

1) A text file - ex02_movie_list.txt - in the Working Directory (ie. the directory reported by pwd). It contains about 65 movie titles.

2) The following output in the SQLite command line tool:

```
sqlite> .read ex02_solution.sql
1|13|Nick|m
2|Abagnale Jr.|Frank|m
3|Abbott|Dalton|m

1|Cameron|James
2|von Trier|Lars
3|Park|Chan-wook

1|The Abyss|1989
2|Aliens|1986
3|Avatar|2009

1|49|Archaeologist
2|28|French Policeman
3|5|Infant John Connor

25|Comedy
25|War
1|Action

1|1
1|2
1|3

sqlite>
```

Exercise 3 - Basic SQL Exercises

Goal

To write SQL in SQLite and convince yourself it is similar to PostgreSQL.

References

- The SQLite .tables and .schema command
- The SQLite SQL dialect:
- https://www.sqlite.org/docs.html -> Programming Interfaces -> SQL Syntax

Instructions

Create views (using CREATE VIEW) that report:

1) Name & gender of the 10 actors with internal database ID numbers >= 2990.

2) The 10 male actors with the highest database ID numbers.

3) The ID of the movie with the highest ID number each year since 2000.

4) BelongsTo (left) joined with Movies

5) Movie title and genre of movies that are either Dramas or War movies

6) Movie title and year of dramas released after 2005.

7) Director given & family name(s), and movies directed (with year) in from 1990-2000

8) Movies (with year) directed by Park Chan-wook

9) 2 numbers: how many actors do not have a recorded given name, and how many do not have a recorded family name

10) All attributes of the actors relation where the actor in question has an unknown family name

!! Outer Joins This is an opportunity to test the behaviour of outer joins in the presence of NULLS. Outer joins are an important operation but somewhat rare compared to the more popular INNER JOIN. Make a prediction about what will happen, then Left- and Right- outer join the q10 view with itself using familyName as the join key.

11) A list of actor pairs with the same last name, but each pair only appears once (so if actors A and B share a last name C, either A C|B C or B C|A C appears in the output but not both. Report the given and family names as a single attribute.

12) All actors with a given name Gary and also display any actors who share a last name with a given that Gary. [HINT: Outer join]

Outputs

The SQL statements are expected to return the following data:

1)

```
sqlite> SELECT * FROM q1;
Esti|f
Sean|f
Ho-jeong|f
Jin-seo|f
Su-kyeong|f
Lucyna|f
Grace|f
Hilary Rose|f
Dianne|f
Catherine|f
```

2)

```
sqlite> SELECT * FROM q2;
Nick|m
Frank|m
Dalton|m
Joe|m
Ian|m
Lewis|m
Stan|m
George|m
Paul|m
Seth|m
```

3)

```
sqlite> SELECT * FROM q3;
51
52
61
62
17
49
63
58
24
```

4)

```
        sqlite> SELECT * FROM q4;
        25|Comedy|25|1941|1979
        25|War|25|1941|1979
        1|Action|1|The Abyss|1989
        1|Adventure|1|The Abyss|1989
        1|Sci-Fi|1|The Abyss|1989
        1|Thriller|1|The Abyss|1989
        1|Drama|1|The Abyss|1989
        2|Action|2|Aliens|1986
        2|Horror|2|Aliens|1986

        ....

        54|Crime|54|Wild at Heart|1990
        54|Drama|54|Wild at Heart|1990
        54|Romance|54|Wild at Heart|1990
        54|Thriller|54|Wild at Heart|1990
        9|Short|9|Xenogenesis|1978
        9|Sci-Fi|9|Xenogenesis|1978
```

5)

```
        sqlite> SELECT * FROM q5;
        1941|War
        The Abyss|Drama
        Amistad|Drama
        Antichrist|Drama
        Artificial Intelligence: AI|Drama
        Barton Fink|Drama
        Blood Simple.|Drama
        Blue Velvet|Drama
        Boksuneun naui geot|Drama
        Catch Me If You Can|Drama
        Chinjeolhan geumjassi|Drama
        Close Encounters of the Third Kind|Drama
        The Color Purple|Drama
        Dancer in the Dark|Drama
        Dogville|Drama
        The Elephant Man|Drama
        Empire of the Sun|Drama
        Empire of the Sun|War
        Epidemic|Drama
        Eraserhead|Drama
        E.T.: The Extra-Terrestrial|Drama
        Europa|Drama
        Europa|War
        Fargo|Drama
        Gongdong gyeongbi guyeok JSA|Drama
        Gongdong gyeongbi guyeok JSA|War
        The Hudsucker Proxy|Drama
        Inland Empire|Drama
        Lost Highway|Drama
        Medea|Drama
        Mulholland Dr.|Drama
        No Country for Old Men|Drama
        Oldboy|Drama
        Rabbits|Drama
        Saibogujiman kwenchana|Drama
        Saving Private Ryan|Drama
        Saving Private Ryan|War
        Schindler's List|Drama
        Schindler's List|War
        Simpan|Drama
        The Terminal|Drama
        Thirst|Drama
        Titanic|Drama
        Twin Peaks: Fire Walk with Me|Drama
        Wild at Heart|Drama
```

6)

```
        sqlite> SELECT * FROM q6;
        Antichrist|2009
        Chinjeolhan geumjassi|2005
        Inland Empire|2006
        No Country for Old Men|2007
        Saibogujiman kwenchana|2006
        Thirst|2009
```

7)

```
        sqlite> SELECT * FROM q7;
        James|Cameron|Terminator 2: Judgment Day|1991
        James|Cameron|Titanic|1997
        James|Cameron|True Lies|1994
```

```
Lars|von Trier|Europa|1991
Chan-wook|Park|Moon Is the Sun's Dream|1992
Chan-wook|Park|Saminjo|1997
Chan-wook|Park|Simpan|1999
Steven|Spielberg|Amistad|1997
Steven|Spielberg|Hook|1991
Steven|Spielberg|Jurassic Park|1993
Steven|Spielberg|The Lost World: Jurassic Park|1997
Steven|Spielberg|Saving Private Ryan|1998
Steven|Spielberg|Schindler's List|1993
David|Lynch|Lost Highway|1997
David|Lynch|Twin Peaks: Fire Walk with Me|1992
Joel|Coen|Barton Fink|1991
Joel|Coen|The Big Lebowski|1998
Joel|Coen|Fargo|1996
Joel|Coen|The Hudsucker Proxy|1994
```

8)

```
sqlite> SELECT * FROM q8;
Boksuneun naui geot|2002
Chinjeolhan geumjassi|2005
Gongdong gyeongbi guyeok JSA|2000
Moon Is the Sun's Dream|1992
Oldboy|2003
Saibogujiman kwenchana|2006
Saminjo|1997
Simpan|1999
Thirst|2009
```

9)

```
sqlite> SELECT * FROM q9;
0|12
```

10)

```
sqlite> SELECT * FROM q10;
291|Cedric the Entertainer|m
584|Flea|m
908|Jack|m
1060|Kong-Guo-Jun|m
1587|Rain|m
1826|Sparky|m
1858|Sting|m
1871|Stromboli|m
2227|Bjork|f
2538|Jarah|f
2720|Nae|f
2926|Talila|f
```

11)

```
sqlite> SELECT * FROM q11;
Stan Adams|Amy Adams
Stan Adams|Margaret Adams
Richard Alexander|Markus Alexander
Gary Allen|Carl Allen
James Allen|Carl Allen

... [Observe that names will not repeat in reverse, so for example there is no Carl Allen|James Allen tuple] ...

Sean Young|Dick Young
Sean Young|Harrison Young
Sean Young|Ric Young
Sean Young|Richard Young
Sean Young|Ron Young
Sean Young|Ruben Young
Su-kyeong Yun|Jin-seo Yun
```

12)

```
sqlite> SELECT * FROM q12;
Gary Allen|Carl Allen
Gary Allen|James Allen
Gary Allen|Karen Allen
Gary Allen|Nancy Allen
Gary Allen|Sean Michael Allen
Gary Bullock|
Gary Busey|
Gary Cervantes|
Gary Epper|
Gary Hershberger|
Gary Houston|Robert Houston
Gary Marshal|
Gary Parker|Brad Parker
Gary Parker|F. William Parker
Gary Sefton|
```

Exercise 4 - SQLite 3 Type Quirks

Goal

To understand why SQLite is considered a 'light' database.

References

- The SQLite Frequently Asked Questions & Datatypes documentation:
- https://sqlite.org/faq.html
- https://sqlite.org/datatype3.html

Instructions

0) Download required files.

ex04_mysterious.sql

1) Load the file into SQLite using:

```
        sqlite> .read ex04_mysterious.sql
```

Observe there were no errors

2) Read the SQL file, and note that it would not be accepted in PostgreSQL. Ideally working with a classmate, and with reference to the SQLite FAQ, justify:

- Why SQLite accepted the type 'antidisestablishmentarianism'
- Why are equal inputs in the INSERT statement resulting in slightly different displayed values in the first SELECT statement?
- Why is there a 1 in the second SELECT statement
- Why Comparison 1 and 2 are so different from each other
- Why does there appear to be 2 newlines after some queries

3) These behaviours mostly stem from a single design choice in SQLite. Does this present a threat when JOINing relations? Run some experiments to confirm your hypothesis.

4) **Predict and check** the type of each instance of each attribute in Mysterious with variants of:

```
        sqlite> SELECT typeof(attr1) FROM Mysterious;
```

Outputs

Some discussion with other members of the lab and an understanding on SQLite's type affinity.