

SQLite Introduction

Hugo Pires

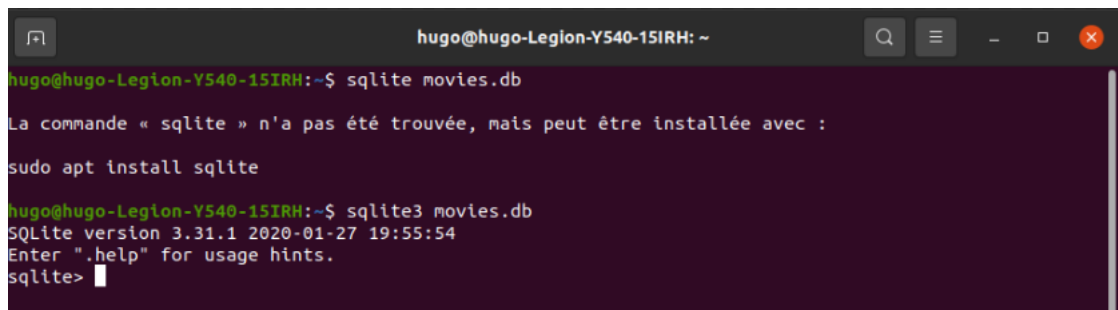
November 2021

1 Introduction

This document will describe the work done for week 9. The goal was to get started with SQLite3, to do some request to manipulate a database. For this, I have first download a random database on mockaroo. The database is about movies and their attributes. There are 100 elements (movies). I worked on linux terminal and visualized operations in SQLBrowser.

2 Create database : movies.db

First step was to install and start sqlite3 in order to create the database "movies.db".

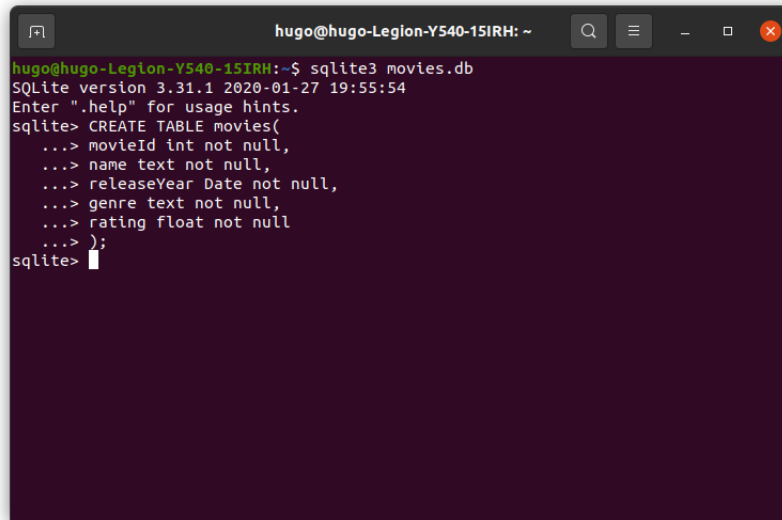
A terminal window with a dark background and light green text. The window title is 'hugo@hugo-Legion-Y540-15IRH: ~'. The user enters 'sqlite movies.db' and receives a message in French: 'La commande « sqlite » n'a pas été trouvée, mais peut être installée avec :'. They then enter 'sudo apt install sqlite'. After the installation, they enter 'sqlite3 movies.db' and see the output: 'SQLite version 3.31.1 2020-01-27 19:55:54', 'Enter ".help" for usage hints.', and a prompt 'sqlite>'.

```
hugo@hugo-Legion-Y540-15IRH:~$ sqlite movies.db
La commande « sqlite » n'a pas été trouvée, mais peut être installée avec :
sudo apt install sqlite
hugo@hugo-Legion-Y540-15IRH:~$ sqlite3 movies.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite>
```

Figure 1: create movies.db

3 Create a table : movies

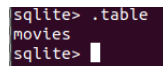
Next step is to get some data in our database and create a table. Our table is composed of 5 filters (5 columns) : movieID,name, releaseYear, genre and rating.

A terminal window titled 'hugo@hugo-Legion-Y540-15IRH: ~' with search, menu, and window control icons. The terminal shows the execution of 'sqlite3 movies.db', displaying 'SQLite version 3.31.1 2020-01-27 19:55:54' and 'Enter ".help" for usage hints.' The user enters 'CREATE TABLE movies(' followed by column definitions: 'movieId int not null,', 'name text not null,', 'releaseYear Date not null,', 'genre text not null,', and 'rating float not null'. The command is completed with ');' and the prompt returns to 'sqlite>'.

```
hugo@hugo-Legion-Y540-15IRH:~$ sqlite3 movies.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> CREATE TABLE movies(
...> movieId int not null,
...> name text not null,
...> releaseYear Date not null,
...> genre text not null,
...> rating float not null
...> );
sqlite>
```

Figure 2: create movies table

We can verify if our table is well created with the command .table.

A terminal snippet showing the command '.table' being entered, which outputs 'movies' to confirm the table's existence.

```
sqlite> .table
movies
sqlite>
```

Figure 3: Verifying table creation

I have loaded data from a csv file generated on mockaroo and created the table "movies". We can visualise this table in the terminal with the command 'SELECT * from movies'(figure 4). It is better to use .mode columns view(figure 5). We can select just one records if we want and show his attributes (figure 6).

```

hugo@hugo-Legion-Y540-15IRH: ~
1|Return to Horror High|6/24/2021|Horror|7.888
2|Islander|7/21/2021|Drama|1.626
3|Zandalee|8/1/2021|Drama|Thriller|1.457
4|Ranma ½: Nihao My Concubine (Ranma ½: Kessen Tôgenkyô! Hanayome o torimodose!!)|7/26/2021|Action|Adventure|Comedy|Fantasy|Romance|9.161
5|LOL (Laughing Out Loud)|5/28/2021|Comedy|4.515
6|New Kids Turbo|8/25/2021|Action|Comedy|9.508
7|Come Back to the Five and Dime, Jimmy Dean, Jimmy Dean|6/23/2021|Drama|0.012
8|Toy Story Toons: Partysaurus Rex|10/10/2021|Animation|Children|Comedy|2.517
9|Out for a Kill|8/12/2021|Action|Crime|Thriller|0.073
10|Cloudy with a Chance of Meatballs|5/16/2021|Animation|Children|Fantasy|IMAX|4.632
11|Rust and Bone (De rouille et d'os)|10/28/2021|Drama|Romance|1.862
12|War|12/12/2020|Action|Thriller|3.326
13|Take My Eyes (Te doy mis ojos)|2/26/2021|Drama|Romance|Thriller|5.617
14|Autobiography of Miss Jane Pittman, The|3/15/2021|Drama|0.363
15|American Pie 2|11/30/2020|Comedy|4.078
16|Performance|6/25/2021|Crime|Drama|Thriller|0.676
17|Rent-a-Cat|12/13/2020|Comedy|Drama|9.857
18|Pete Seeger: The Power of Song|4/15/2021|Documentary|2.073
19|Mon oncle d'Amérique|3/5/2021|Drama|8.024
20|Tonight and Every Night|5/28/2021|Musical|5.656
21|The Russian Novel|3/7/2021|Drama|4.701
22|Deadly Blessing|10/24/2021|Horror|9.043

```

Figure 4: basic view

```

hugo@hugo-Legion-Y540-15IRH: ~
sqlite> .mode column
sqlite> SELECT * FROM movies;
4,Ranma ½: Nihao My Concubine (Ranma ½: Kessen Tôgenkyô! Hanayome o torimodose!!),7/26/2021,Action Adventure Comedy Fantasy Romance,9.161
68,Dragon Ball Z the Movie: The World's Strongest (a.k.a. Dragon Ball Z: The Strongest Guy in The Adventure Animation Sci-Fi Thriller,4.26
82,"Glamorous Life of Sachiko Hanai, The (Hatsujô kateikyôshi: sensei no aijiru)",11/5/2020,Action Comedy Drama Fantasy Mystery
movieID      name      releaseYear  genre      rating
1            Return to  6/24/2021    Horror      7.888
2            Islander   7/21/2021    Drama      1.626
3            Zandalee   8/1/2021     Drama|Thri  1.457
4            Ranma ½: N 7/26/2021    Action|Adv  9.161
5            LOL (Laugh 5/28/2021    Comedy     4.515
6            New Kids T 8/25/2021    Action|Com  9.508
7            Come Back  6/23/2021    Drama      0.012

```

Figure 5: mode columns view

```

sqlite> SELECT * FROM movies WHERE movieID = 10;
10      Cloudy with a Chance of Meatballs  5/16/2021  Animation|Children|Fantasy|IMAX  4.632

```

Figure 6: one record

4 Update records

With the command 'UPDATE movies SET rating = 10 WHERE movieID = 10', I changed the rating of the 10 th records and turned it into 10. With the same command I replaced the 20 last movie's name by "mymovie".

```
sqlite> SELECT * FROM movies WHERE movieID = 10;
10      Cloudy with a Chance of Meatballs  5/16/2021  Animation|Children|Fantasy|IMAX  4.632

sqlite> UPDATE movies SET rating = 10 WHERE movieID = 10;
sqlite> SELECT * FROM movies WHERE movieID = 10;
10      Cloudy with a Chance of Meatballs  5/16/2021  Animation|Children|Fantasy|IMAX  10.0
```

Figure 7: update one record

```
sqlite> UPDATE movies SET genre = "Drama" WHERE movieID > 80;
```

Figure 8: update 20 records

```
sqlite> SELECT * FROM movies Where movieID > 80;
movieID  name          releaseYear  Drama  rating
81       Make Belie  1/1/2021    Drama  5.77
82       Glamorous  11/5/2020   Drama  3.953
83       Adventures 9/6/2021    Drama  9.901
84       Sadie Thom  7/30/2021   Drama  4.461
85       Thérèse    3/15/2021   Drama  1.521
86       Patriot Ga 1/30/2021   Drama  1.075
87       Kadosh      5/30/2021   Drama  3.518
88       eXistenZ   9/4/2021    Drama  8.79
89       Parineeta  8/30/2021   Drama  2.821
90       Don't Tell 5/6/2021    Drama  8.203
91       Children o 10/28/2021  Drama  3.628
92       Priceless 9/23/2021   Drama  6.892
93       Savage Inn 8/22/2021   Drama  1.011
94       El Escarab 8/6/2021    Drama  1.852
95       Holy Smoke 4/3/2021    Drama  2.109
96       Kung Fu Pa 9/6/2021    Drama  8.28
97       King Solom 5/4/2021    Drama  5.059
98       Karan Arju 6/24/2021   Drama  0.262
99       Cinderella 6/18/2021   Drama  7.044
100      Europa Rep 7/6/2021    Drama  0.845
```

Figure 9: update 20 records

5 Delete 10 records

With the command 'DELETE FROM movie WHERE rating < 3' I have deleted all the movies with a grade inferior to 3. We can see there are now just 65 rows instead of 100 with the command SELECT count(*) FROM movies.

```
sqlite> DELETE FROM movies WHERE rating<3;
sqlite> SELECT count(*) FROM movies ;
65
```

Figure 10: delete records

6 Conclusion

In this homework we saw basics commands for manipulation of databases on sqlite3. It's interesting to see how it's easy to manage a database and how it could be implemented in bigger project. Next level of command could be to do some group by or joins.