Computer Technologies in Telecommunications

Week 8/9
SQLite on Ubuntu

Victor Girard

1 Introduction

This week's work will focus on SQLite3 and some simple operations. the goal is to create and manipulate a database of movies. This report summerizes the work done.

2 Data generation

The goal is to deal with a database named movies.db, composed for each movies of : a unique Identification number, the name, release year, genre and rating of the movie. To simplify the entries in the database it is possible to use **Mockaroo** to generate a *.csv* file.

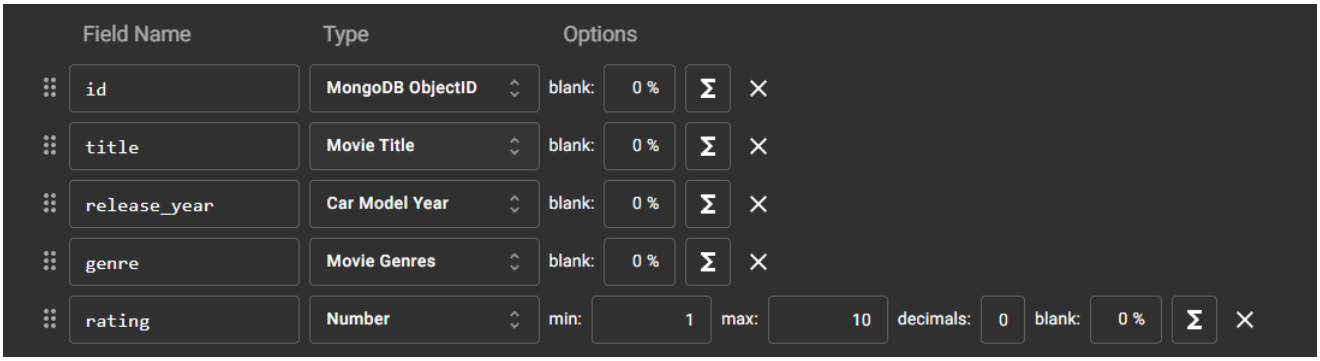

Figure 1: content of movies data file

3 ToDo(1/2) - Install SQLite and create DB

3.1 Install SQLite

To install SQLite on Ubuntu just use the command line :

```
sudo apt install sqlite3
```

```
victor@ubuntu:~$ sudo apt install sqlite3
[sudo] password for victor:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-5.11.0-27-generic linux-hwe-5.11-headers-5.11.0-27
  linux-image-5.11.0-27-generic linux-modules-5.11.0-27-generic
  linux-modules-extra-5.11.0-27-generic
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  sqlite3-doc
The following NEW packages will be installed:
  sqlite3
0 upgraded, 1 newly installed, 0 to remove and 68 not upgraded.
Need to get 860 kB of archives.
After this operation, 2 803 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 sqlite3 amd64 3.31.1-4ubuntu0
.2 [860 kB]
Fetched 860 kB in 5s (169 kB/s)
Selecting previously unselected package sqlite3.
(Reading database ... 236135 files and directories currently installed.)
Preparing to unpack .../sqlite3_3.31.1-4ubuntu0.2_amd64.deb ...
Unpacking sqlite3 (3.31.1-4ubuntu0.2) ...
Setting up sqlite3 (3.31.1-4ubuntu0.2) ...
Processing triggers for man-db (2.9.1-1) ...
```

Figure 2: SQLite install

3.2 Create and file movies.db

Once SQLite is install it's ready to use, first we need to create the table we're going to work with. So we first need to declare the database **movies.db** with it's dedicated column ID, name, year, genre, rate. Then declare that we're gonna upload a .csv datafile and finally import data with the csv file

```
sqlite> create table movies (movie_id int, title text, release_year int, genre text, rate i
sqlite> .mode csv movies
sqlite> .import MOVIES.csv movies
```

```
victor@ubuntu:~$ sqlite3
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> create table movies (movie_id int, title text, release_year int, genre text, rate int);
sqlite> .mode csv movies
sqlite> .import MOVIES.csv movies
```

Figure 3: movies.db create and fill

```
sqlite> select * from movies
   ...> ;
id,title,release_year,genre,rating
61883712fc13ae5f8800047e,Hardware,2011,Action|Horror|Sci-Fi,9
61883712fc13ae5f8800047f,"Ju-on: The Beginning of the End (Ju-on: Owari no hajimari)",1994,Horro
r,6
61883712fc13ae5f88000480,"Playing with Love (Puppy Love) (Maladolescenza)",1992,Drama,4
61883712fc13ae5f88000481,"Band Wagon, The",2012,Comedy|Musical,9
61883712fc13ae5f88000482,"Quartier Mozart",2000,Comedy,4
```

Figure 4: display movies.db

To display the db in a more lisible way we can activate the column mode with headers :

```
sqlite> .headers on
sqlite> .mode column
sqlite> select * from movies;
movie_id    title       release_year  genre       rate
----------  ----------  ------------  ----------  ----------
id          title       release_year  genre       rating
61883712fc  Hardware    2011          Action|Hor  9
61883712fc  Ju-on: The  1994          Horror      6
61883712fc  Playing wi  1992          Drama       4
61883712fc  Band Wagon  2012          Comedy|Mus  9
```

Figure 5: display movies.db in column mode

4 ToDo(2/2) - SQLite functions

4.1 retrieve records for all the movies

To get the records of all the movies we just need to select everything from movies.db, to select *everything* it is written as *. (cf section 3)

```
sqlite>select * from movies;
```

4.2 retrieve all the records for a single movie

To retrieve a single movie we can find it by it's name or ID number. So we need to specify an ID or name to select in movies db.

```
sqlite>select * from movies where title = "titanic";
```

```
sqlite> select * from movies where title = "Bait";
movie_id                  title       release_year  genre          rate
------------------------  ----------  ------------  -------------  ----------
61883715fc13ae5f88000801  Bait        1998          Action|Comedy  9
```

Figure 6: Select a single movie

4.3 update a single record

We're going to update the year of a movies by using the command update

```
sqlite>update movies set release_year = "1999" where title = "Bait";
```

```
sqlite> update movies set release_year = "1999" where title = "Bait";
sqlite> select * from movies where title = "Bait";
movie_id                 title       release_year  genre          rate
-----------------------  ----------  ------------  -------------  ----------
61883715fc13ae5f88000801  Bait        1999          Action|Comedy  9
```

Figure 7: Select a single movie

4.4 delete 10 records

To remove a certain amount of line in a db we need to use parameter **limit**. this time we're going to remove the 10 best rating drama movies. so we delete 10 row while selecting all the drama movies and sorting them descending.

```
sqlite>delete from movies where genre = "Drama" order by rate desc limit 10;
```

```
sqlite> select count(*) from movies where genre = "Drama" order by rate desc limit 10;
count(*)
----------
167
sqlite> delete from movies where genre = "Drama" order by rate desc limit 10;
sqlite> select count(*) from movies where genre = "Drama" order by rate desc limit 10;
count(*)
----------
157
```

Figure 8: Delete 10 row in Drama genre

4.5 update 20 records on your own choice