ESCUELA
SUPERIOR DE
ARTE Y
TECNOLOGÍA

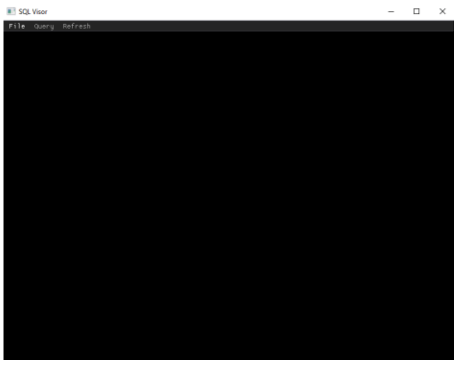
# DATABASE MEMORY

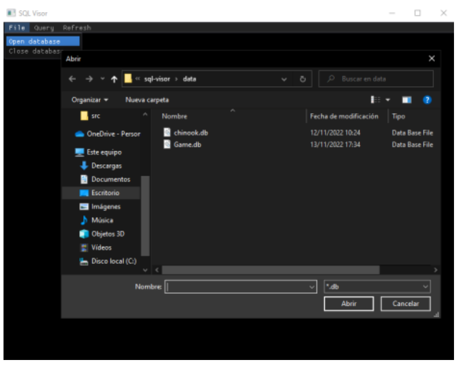
Course: Prog. Avanzada
Year: 2022/2023
Author: Manuel Alcázar López

## Index

## 1.- User Documentation

Once we start our executable, a black window will appear with an ImGui menu bar at its top. To open a database, we need to go to "**File**" and "**Open Database**", on the **top left of the menu bar**. Once we click and "Open" a **native Window's explorer window** will be open, from there we **need to search for our database** and select it to open it and show it in our window.



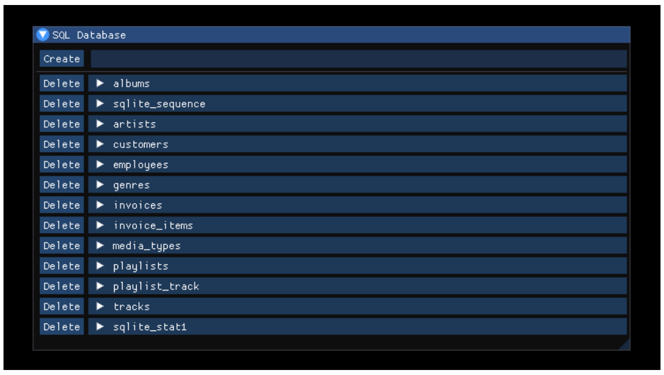


Notice that we **cannot click on** the "**Close Database**" until we **have opened a database**, but **we do not need to close a database to re-open a new one**. **Also**, once we have a database opened, the "**Query**" option in the menu bar **will be available**.

**After opening** a database, **the structure of the database**, its tables**, will be shown** in a new window. This window will show:

- **The tables of the database.**
- **An option to create new tables,** located at the top side of the window**.**
- **An option to delete the tables,** located next to the table we wish to delete.

The **delete option does not show any warning,** so **be careful.**

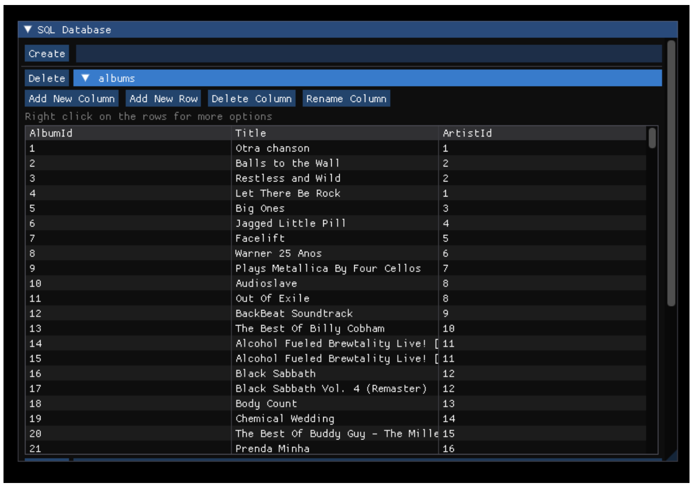


To see the information of the tables we just need to **click on the collapsing header**, to **see the structure and data** of the **table**. This will display a table with the current columns, rows, and data of the table, with options to add more columns and rows.

- **Change**, that will open a window to introduce a new value.
- **Delete**, **to delete the entire row**, **not just the value**.

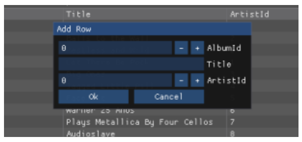
The options available when we open a table are:

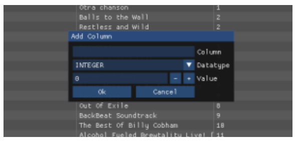
- **Add new column.**
- **Add new row.**
- **Delete column.**
- **Rename column**.

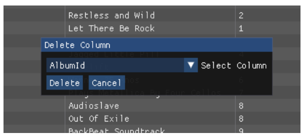


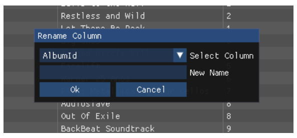
| AlbumId | Title | ArtistId |
|---|---|---|
| 1 | Otra chanson | 1 |
| 2 | Balls to the Wall | 2 |
| 3 | Restless and Wild | 2 |
| 4 | Let There Be Rock | 1 |
| 5 | Big Ones | 3 |
| 6 | Jagged Little Pill | 4 |
| 7 | Facelift | 5 |
| 8 | Warner 25 Anos | 6 |
| 9 | Plays Metallica By Four Cellos | 7 |
| 10 | Audioslave | 8 |
| 11 | Out Of Exile | 8 |
| 12 | BackBeat Soundtrack | 9 |
| 13 | The Best Of Billy Cobham | 10 |
| 14 | Alcohol Fueled Brewtality Live! [ | 11 |
| 15 | Alcohol Fueled Brewtality Live! [ | 11 |
| 16 | Black Sabbath | 12 |
| 17 | Black Sabbath Vol. 4 (Remaster) | 12 |
| 18 | Body Count | 13 |
| 19 | Chemical Wedding | 14 |
| 20 | The Best Of Buddy Guy - The Mille | 15 |
| 21 | Prenda Minha | 16 |

All these options are **focus on user friendliness**, and will display a little window, with all the **necessary steps and information** needed **to correctly create or delete** a column or row, to **ensure data validation and normalization** and avoid further issues on the database.

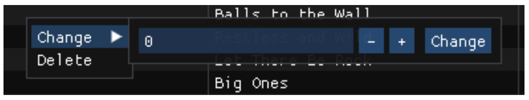






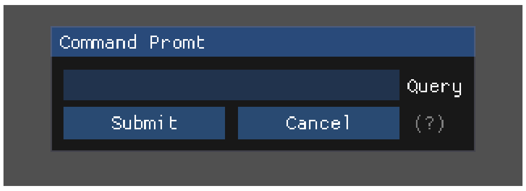


If we want to **change the value stored in the table**, we need to **right click on the value** we want to change, this opens a popup that show two options:



The options don't need any explanation because they are very intuitive and easy to understand, and the user has no option to miss any necessary step, make any mistake or introduce wrong data.

In the other hand, the "**Query**" option in the menu bar, will display a little window when the user will have more freedom. From here, we can execute queries directly to SQLite. The window has a help marker, that will display the maximum size of the query the user can introduce, and a brief cheat sheet of SQLite queries.



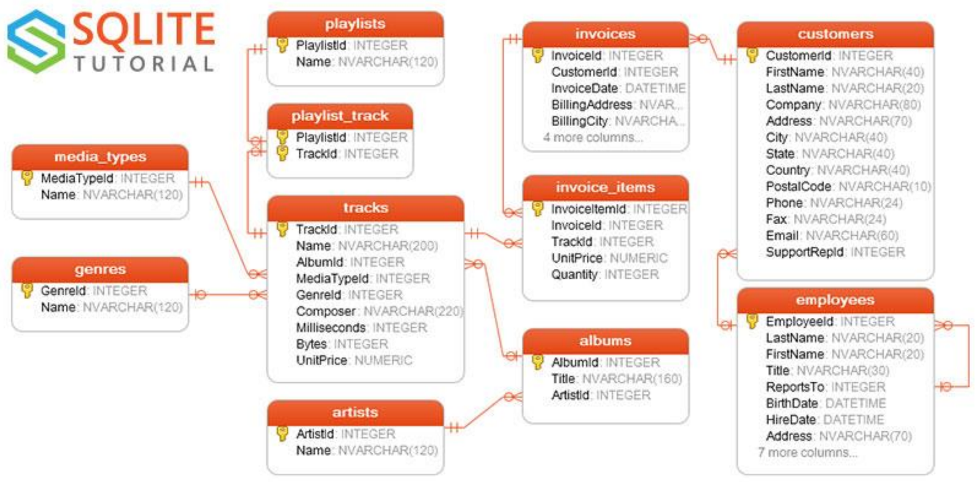
## 2.- Development Log

- 10 October: Started project, completed fully executable loop without any loaded data.
- 11 October: Create first version of data container using std vector. Finished first display of database structure, only tables.
- 12 October: First version of table layout, with minor issues, such as rows unaligned from columns.
- 13 October to 17 October: Working on memory allocation using std vector and finishing SQL basic layout.
- 18 October: Fixed an issue with imgui tables, added nfd library to search database using Window's native explorer.
- 19 October to 31 October: Working on fixing memory leaks, by implementing different approaches to load the data from the sqlite callback. Fixed memory leaks day 31 and change the SQL layout to display the information correctly according to the new code. Fixed the layout distribution, now is aligned.
- 2 November: Change the code to the final repository.
- 3 November: Added delete and create table options. Minor code sanitation.
- 6 November: Added create column option, working on more options and data validation.
- 8 November: Finished add new row option, working on more options, and sanitized code.
- 9 November: Finished delete column. Add row option fixed. Fixed memory issues related to data loading.
- 10 November: Fixed some minor bugs, changed table loading, changed memory allocation again and finished change value option.
- 13 November: Finish code, fixed minor bugs and actualized some options. Finish doxygen and documentation.

## 3.- Flowchart.

Due to the necessity to have a big database to fully test the project and speed up the workflow by not wasting time creating our own database, I have used a sample database used to learn SQLite queries. The database can be found here: https://www.sqlitetutorial.net/sqlite-sample-database/

The database sample flowchart is:



## 4.- Postmortem.

Although the database does not show any known memory leak, change it to std vector and std string will prevent any possible leak. Also changing the approach in which the information is loaded from the database can be improved to be lightweight and less complex.

The options implemented in the program may be enough to test if the values extracted and submitted to the database are correct, but there is certainly a lack of options when we want it to use it in deeper level. For example: there is no option to add keys to the columns, not primary or unique; there is a lack of formats in which the rows and columns can be built, using only the basic datatypes provided by SQLite. Also, the options do not allow to realize multiple operations in a single query. For example: the user cannot add multiple rows at the same time or delete multiple columns at the same time, only one at the time, which can be frustrating.

The performance or speed of the source code can be improve using better store containers and avoid certain expensive operations such as strcat or srtcpy. But in general, these function calls have been avoid using instead sprint or memcpy, which are faster a easier to use and with less undefined behaviors probabilities.

The query prompt does not return the output of the query, this is actualized in the SQL visor instead, but the data is not validated nor normalized, which could lead to issues. It also does not work with the SELECT statement.

The code underneath can be reduced avoiding code redundance and unnecessary instructions that makes the code bigger and complex for no reason. With a correct base allocation of the data retrieved from the database the implementation and the data validation can be easily do, otherwise it just makes the code hard to read and debug.

Overall, the program achieves its purpose efficient and correctly, but it cannot be compared to a fully database editor, and it does not have a great scalability.