CS3200
Teddy Davidson and Nico Gaspar

CS3200 Project Report

Technical Specifications

This application is fairly simple to run, requiring only Python 3 and the PyMySQL library to run. Additionally, the database can be created from the included database dump or create schema, both of which are SQL text files, requiring that the user also has an application such as MySQL Workbench. The database is then hosted on the user's local server, and the client application connects to it using the user's login for the localhost account.

Lessons Learned

This project was great at improving our familiarity with PyMySQL and connecting to and modifying an SQL database using external coding language (Python). We also learned a ton on how to use Tkinter, a very powerful and applicable Python GUI library. We also used Git a bit to collaborate on code and keep track of past versions, so this project was beneficial in improving our abilities with that as well.

We definitely ran into some issues in communication and planning, mostly regarding breaking down the work into smaller bits for us to work on simultaneously and not right at the end of the semester. We both let school get in the way, so going forward we agree that keeping communication up is crucial even when, and especially when, we get bogged down by other work.

We also found ourselves needing to work around some shortcomings/limitations from our schema. Because of how we wanted to store our data (for example saving the names of all genres in the songs table without duplicates but removing them if all songs of a particular genre were deleted), we struggled to find a straightforward way to add, remove, and update our tuples and resorted to a bit more hard-coding instead. Ultimately the way we finally decided to go about using the database worked the way we wanted to overall, but we admit that there were definitely simpler methods that we just weren't quite able to figure out in time to implement.

Regardless of how optimal, suboptimal, or just barely functional our code was at times, as of now, all of our code works (or we at least have considerable reason to believe it does after testing almost everything we could think a user could do). The user shouldn't be able to run into errors often at worst or ever at best, since we were sure to fill the code with plenty of try-except blocks to catch whatever we thought they could throw at us. We also had our fair share of typing errors, so we were able to root out some issues from simple errors as well. However, we never reached a good solution for deleting tuples from the artist, album, or genre tables the way we wanted, so tuples in these tables aren't removed after they're added. This works fine with the application the way we have it now, but it isn't completely consistent with our goal.

There shouldn't be bugs, but if the application windows behave weirdly at any point (rarely, but usually only if the program somehow runs into an error) closing them (ideally by logging out or returning to previous menus) and re-running the Python script should clear up anything.

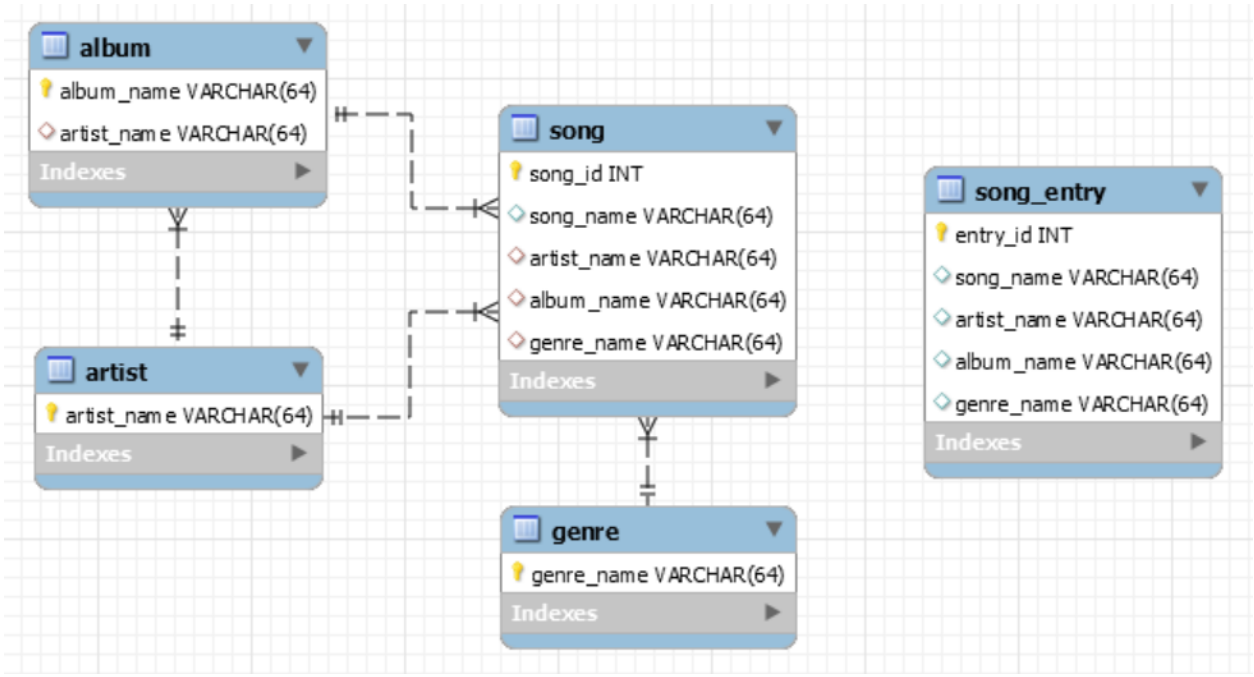
Future Work

In the future, we'd love to make the database compatible with .mp3s or other audio files to turn it from a database of information about songs into a database of songs themselves. If we chose to take the project a different way, we also could make the server public and use data on what songs users frequently enter to create a recommendation engine of sorts. For both of these uses, an additional table for Playlists or something similar would be useful, and probably one for Users as well.

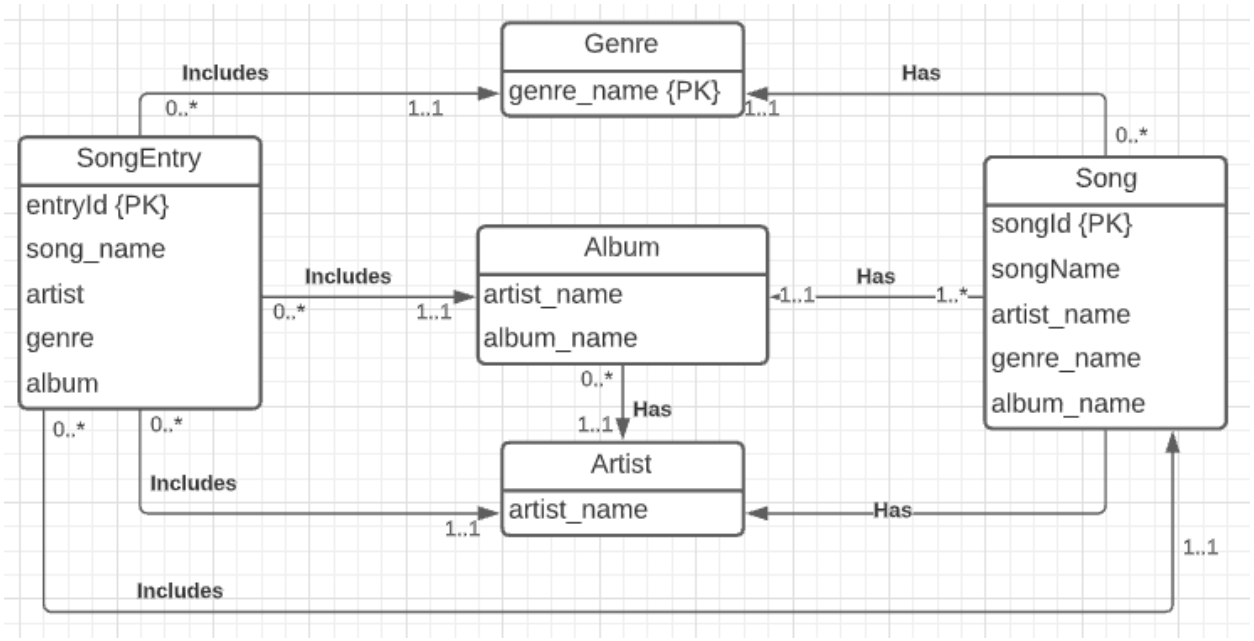
We really wish we could improve the search function. It's great that it's flexible enough to match the search to songs, artists, and albums, but it'd be great and way more user friendly if the search didn't work on exact string matches. Another great quality of life change would be to take advantage of the Song, Artist, Album, and Genre tables and use the data store in them (or even just a few rows from them) in a drop-down menu to simplify user inputs for the fields. A last change like this would be to find a way to navigate the whole application from one window instead of creating new ones for new parts of the application. We feel what we have now is easy enough to follow and lines up nicely with how the code is laid out, but we think it might make for a simpler user experience to only have to worry about one changing window rather than multiple static windows.

CS3200
Teddy Davidson and Nico Gaspar

EER of Database Schema



UML Diagram of Conceptual Design



CS3200
Teddy Davidson and Nico Gaspar

User Flow