Project Wrap Up Document

Destin Krepps

This project was a great way to get hands on practice with SQL which is a heavily sought-after experience qualification in a lot of jobs after graduating. The course itself was good at getting design and theoretics to help build the database and schema properly, and I felt this project was the thing motivating us to put it to use. I was completing this project alone so all the duties that came with my project idea fell upon me. Duties including coming up with a project idea, choosing a database system, finding a dataset that can accomplish it, designing a database schema based upon the dataset found, and creating a user interface to interact with the database created. I also decided to present so creating a presentation for the project as well.

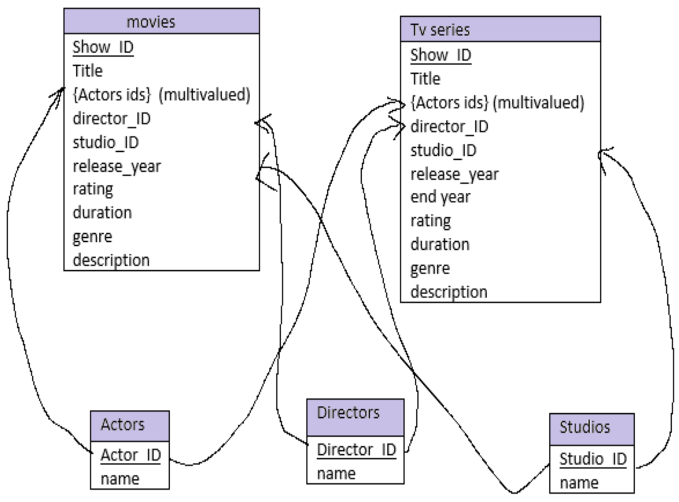
When coming up with a project idea I remembered back to when I first started college and was initially learning about arrays and such, I was tasked with creating a data structure that could optimally hold movies from a text file. So, when coming up with a project idea I figured what better way to end my college career than with a more optimal and practical movie database and search engine. To find a dataset that would help me accomplish my goal I originally thought to use the files offered from IMDB, they had so many tables and each of them had millions of rows and they also needed to be cleaned (structuring to fit my dataset into the schema and removing mistakes in the data that could cause issues with my program) so it was more effort than it was worth, especially with the time constraints given and lack of experience designing a database system. I eventually came across my current dataset from a website called 'Kaggle.com' which had like 9000 movies from Netflix and only like 35000 actors for them. When I designed the original schema, I had the idea of two tables for movies and tv series, with actor ids as multivalued attributes. But as I realized that multivalued attributes were probably not the most efficient way of storing actor data for the movie and that I didn't need two tables for the shows I just needed a type of attribute to differentiate the two. The new schema combines the two and tables and uses a casts table to associate actors with shows so we can remove the actors attribute from the shows table entirely (for images of the old and new schema see below).

The dataset from 'Kaggle.com' was really similar to what I had envisioned for my project, so it was great, but it still needed a bit of forming and there were a few mistakes that needed to be taken care of before inserting it into my database system (MySQL). For example, when I created my actor table there was no way to differentiate duplicate actors because the only thing that identified them was the movie they were in, so unless I wanted to verify each actor by their movie by hand it wasn't feasible to allow duplicates initially. So, the only realistic solution was to remove duplicates from the actor table and then once the actor table is created from the initial dataset, they will have unique ids to identify them and if we wanted to add a new one, they can have the same name just different ids. This wasn't a big issue as my project didn't have more attributes to actors other than their name so duplicates wouldn't make sense anyway and in search engines if a user wants to watch a movie with an actor named john doe, most search engines will show all movies with an actor named john doe in general and not a specific one. Also, a problem with my dataset I had was that I didn't have any studio data and I really wanted it in my database, so I chose 10 studios and manually cross referenced about 150 shows to associate studios with shows in the database. Then just added the Ids of the studios to their respective shows. I wrote a couple of scripts to pull names from actors and directors and fill a new file with Ids and replaced the names with ids in the original file as it makes for a more structured database. After that I used a MySQL script to insert the data into database tables that I created using a python cursor.

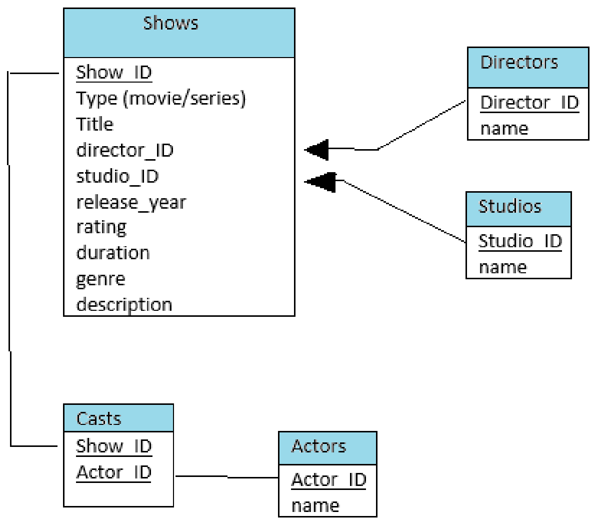
For the user interface I prompted the user for around 13 different ways to search through show data. Most of the ways are just attributes from the original show table including title, genre, show type, rating, and release year. But for my natural joins I joined the actor, cast, and show table for a search by actors, joined the show and director table for search by directors, and show and studio table for searches by studios. I also allowed for a few combination searches such as actor, studio, director, with genre. And actors and directors to be searched by. Actors and directors were a little weird to search by because natural joins with the 4 tables it would take interacted oddly because the name attribute, but I found a trick with two different queries that can cross reference each other instead. The last thing I wanted to talk about code wise was the use of the LIKE keyword for MySQL and that most of the code was just repetition and error checking user input. The LIKE keyword in MySQL allows me to give the user leeway in their searches instead of user needing to type an exact title, they can instead search parts of the name and get results with the search in the name. I used this keyword in almost all my queries.

My feelings on the project after everything is now completed is that it was really fun when interacting with a completed database in my user interface, but the most unfun portion was fixing mistakes in my data and formatting my data from an excel like structure to my database structure especially because the data came from someone else, I had to rely on their completeness and validity for the sake of my project.

OLD



NEW


Shows
Show_ID
Type (movie/series)
Title
director_ID
studio_ID
release_year
rating
duration
genre
description
Directors
Director_ID
name
Studios
Studio_ID
name
Casts
Show_ID
Actor_ID
Actors
Actor_ID
name