Jose Cabrera

Nicholas Pavao

Joshua Tavares

Application Description and Architecture

Our application allows the user to access a database that stores information about a music library. This includes details about songs, albums, artists, genres of music, and record labels. Users are able to search the database based on a specific song, album, artist, genre or record label and get matching outputs. When a song name is searched the user can review songs in the database that match the entered string. Each song lists its name, playtime, album, artist, label, and contributing artists where applicable. When an album name is searched, the user can review the albums that match the entered string. Each album details the album's name, its artist, the release date, genre, track count, and playtime. When an artist is searched, the user can review the artists that match the entered string. Each particular artist's name, date of birth, list of albums, and current label are displayed where applicable. The user is also capable of searching by label name. Here, each label that matched the entered string will be displayed along with its name, established date, and currently signed artists. Lastly, the user may search by genre. Each matching genre to the entered string will present its name, a short description, and all of the albums that contain songs matching that genre currently stored in the database.

The song table records an integer primary key for identification as songs can have the same name/album/artist as another song. Also included is the ID for the album name, the artist's name, the ID for the label name, and the ID for the genre name as foreign keys. The songs table records foreign keys for all other tables as it is the most specified object in our database and requires information from all tables. This could have been avoided if we decided to make separate tables for foreign keys. However, this would have been less efficient as we would end up joining multiple tables when performing a query on song. Lastly, the song's length is stored in seconds as an integer, the song's name is stored as a string, and the song's contributing artists as a string or as NULL if there are none.

The album table records an integer primary key for identification. As the album is expected to display its artist's name, release date, genre, track count, and playtime, those are also stored in the table. There are no foreign keys in this table as they are unnecessary.

The artist table also records an integer primary key for identification. The artist's current label is stored as an integer referring to the label table as a foreign key. Otherwise, the artist's first, last, and artistic names are all recorded as strings and their date of birth is stored as a char of size 11 for specific formatting.

The label table has its own integer primary key as well as the label name and establish date saved. There is no foreign key to determine which artists are currently signed. Instead, the system checks the artist's table for their current label's ID. If it matches the searched label we will then display the artist's name.

Finally, the genre table which also has its unique ID as a primary key. There isn't much information to store here other than the genre name and its short description which are both stored as strings. To display the albums that contain songs that follow the current genre we investigate the song table. If the song matches the genre, we take the albumID and associate it with a name to display to the user.