# 1103 Project Final Paper

# - Music Database -

Taryn Cail

Winter 2024

## 1 Abstract

This project involved creating a database for a set of music and some of its relations. This included me creating ER diagrams for the Music database, creating DDL statements for each of the database's entities and relations and connecting my diagrams to a JDBC. I also went on to create an application for the database that contained several methods for data manipulation within the database.

Upon implementing this project, many new findings were discovered about SQL implementation into Java, such as:

- How to create a Prepared Statement which allowed me to set up the SQL statement before retrieving the input from the user and then re-running the statement with the values from the user.
- Becoming more familiar with the try-catch-finally method in Java.
- How to work with Statement's in Java and the benefits of setting Auto Commit to false to preserve the integrity of the Database and not allow SQL statements to execute without explicitly allowing them to.
- How to use Meta Data to print out the result of the newly changed database.
- How to better alter and manipulate databases through SQL.
- How to connect a SQL data set to a Java IDE and use it through the Java IDE.

# 2 Introduction

## 2.1 Requirements Met

I met the ***challenge requirements*** for this project by creating:

- An ER diagram for the database.
- DDL statements for each entity.
- DDL statements for each relationship.
- Explaining the relation between the DDL and ER diagram.
- Writing sample insert statements for each table.
- Connecting the database using JDBC.
- Creating database schema using JDBC.
- Inserting records to each table.
- Having a function that finds and prints the largest playlist in the database.
- Having functions to insert/delete songs, artists, and playlists.
- Having a function to change the number of songs in a playlist.
- Having a function to change the playlist a song is in.
- Having a function to change an artist's net worth.

## 2.2 The System

The system involves three entities and two relationships, they are as following:

*Entities:*

- Song
- Artist
- Playlist

*Relationships:*

- has_song
- sings_song

I then created data for each table and relationship, which is attached in the Appendix.

The methods created were as follows:

*Welcome method:*

- This method created the layout for the application and detailed each of the methods that were available to the user to manage the database system.
- This method was used in the main method repeatedly and is called before the user runs any other method.

*Print Song Method:*

- This method printed out the entire song table formatted within 30 spaces.
- This method was used throughout other methods such as the insert/delete song methods and the change a songs playlist method.

*Print Playlist Method:*

- This method printed out the entire playlist table formatted within 30 spaces.
- This method was used throughout other methods such as the insert/delete playlist methods, the change a playlists number of songs method and finding the biggest playlist.

*Print Artist Method:*

- This method printed out the entire artist table formatted within 30 spaces.
- This method was used throughout other methods such as the insert/delete artist methods and the change an artist's net worth method.

*Insert/Delete Song, Artist, and Playlist Methods:*

- These methods allow the user to insert a song/playlist/artist to the database given the data required.
- These methods allow the user to delete and song/playlist/artist from their respective databases given the data required.

*Change a Songs Playlist:*

- This method allows the user to identify a song within the database given the song's ID number and change the playlist it is apart of given a playlists ID number.

*Change an Artists Net Worth:*

- This method allows the user to identify an artist within the database given the artist's ID number and change the net worth attribute.

*Change the number of songs in a Playlist:*

- This method allows the user to identify a playlist within the database given the playlist's ID number and change the number of songs attribute.

*Find the biggest Playlist:*

- This method allows the user to search the playlist database and find the one with the largest number of songs and print it to the screen.

# 3 Conclusion

This project was incredibly helpful in expanding my understanding of creating and implementing a JDBC. It gave me hands on experience in creating methods to manipulate data using Java and SQL. I will be further expanding this database and the methods provided in the future for my own personal use and exploration.

# 4 Bibliography

**DatabaseMetaData (Java Platform SE 8 ). (2024, January 8).** https://docs.oracle.com/javase/8/docs/api/java/sql/DatabaseMetaData.html

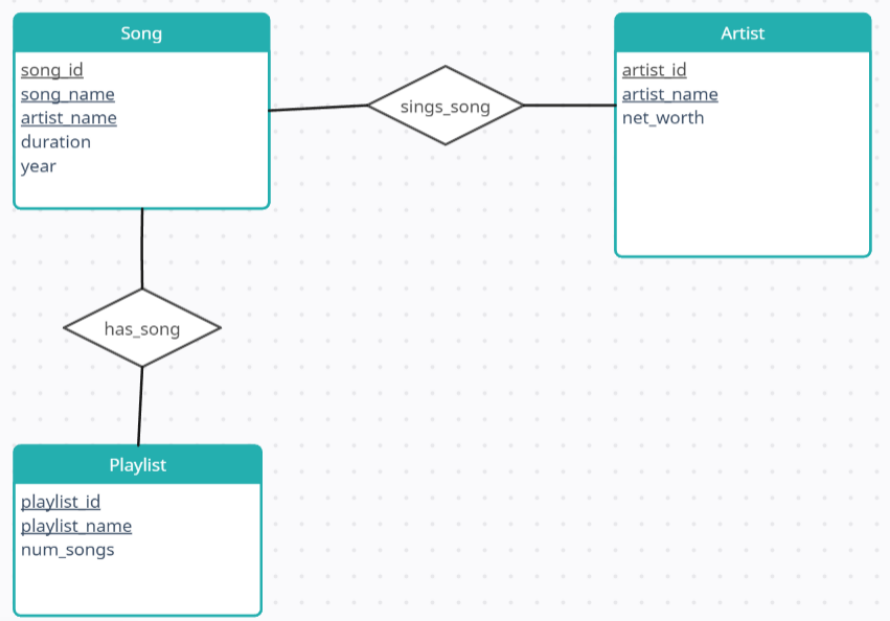
**Fadatare, R. (2020, December 12).** ***Java JDBC preparedstatement interface example with mysql database*****. Java Guides.** https://www.javaguides.net/2018/10/jdbc-preparedstatement-interface.html

***Package java.sql*****. java.sql (Java Platform SE 8 ). (2024, January 8).** https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html

***Using prepared statements*****. Using Prepared Statements (The JavaTM Tutorials > JDBC Database Access > JDBC Basics). (n.d.).** https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html

# 5 Appendix

## 5.1 ER Diagram



## 5.2 DDL Statements

```
create table artist
        (artist_id      varchar(5),
         artist_name    varchar(50) not null,
         net_worth      varchar(15),
         primary key (artist_id, artist_name)
        );

create table song
        (song_id        varchar(3),
         song_name      varchar(50) not null,
         artist_name    varchar(50) not null,
         duration       numeric(4,2),
         playlist_id    numeric(3),
         primary key(song_id, song_name)
         foreign key(artist_name) references artist(artist_name),
         foreign key(playlist_id) references playlist(playlist_id)
        );

create table playlist
        (playlist_id    varchar(3),
         playlist_name  varchar(50) not null,
         num_songs      numeric(10,0),
         primary key(playlist_id, playlist_name)
        );
```

```
create table has_song
       (song_id        varchar(10),
        song_name      varchar(50),
        playlist_id    varchar(5),
        playlist_name  varchar(50) not null,
        primary key(song_id, playlist_id),
        foreign key(song_id, song_name) references song(song_id, song_name),
        foreign key(playlist_id, playlist_name) references playlist(playlist_id, playlist_name)
       );


create table sings_song
       (song_id        varchar(10),
        song_name      varchar(50),
        artist_id      varchar(5),
        artist_name    varchar(50) not null,
        primary key(song_id, artist_id),
        foreign key(song_id, song_name) references song(song_id, song_name),
        foreign key(artist_id, artist_name) references artist(artist_id, artist_name)
       );
```

# 6 User's Guide

Step 1: Open the Java Project and run the main method, this may be different depending on what IDE you are using.

Step 2: When running the program, you will be greeted with 13 different options of ways you can manipulate the database. Read through the options and select which option you'd like by entering the corresponding number.

Step 3: Once your selection is made, the program will perform the desired task or walk you through the necessary steps to perform the task. Input any required data.

Step 4: Once the task is completed the program will ask whether you'd like to complete another task, input 'yes' if you would or 'no' if you'd like to end the program.

- If you accidentally say, 'yes' when you meant to say 'no' and exit the program, just select option 14 to close the program.

Step 5: Loop through the program as many times as needed to complete the desired tasks.

Step 6: Exit the program, your database has been successfully manipulated.