Team Monstars

Dr. Inclezan

CSE 385

April 4, 2014

Music Database Project - Early Stage Prototype

The goal of our application is to create a database storing a large variety of music. Our application will create playlists of songs based on the user’s mood and/or by their current activity. There are no applications at the moment that can make playlists for both mood and activities. Due to technological limitations we will have the user enter their mood and/or activity. For example, if the user was to enter that they were in a tired mood, the program would generate a playlist to play relaxing music. An example for the activity would be if the user enters sports, the program will play energetic music. This program will allow the user to not only save time by creating playlists in advance, but will also allow the user to discover new music.

This program is different from other playlist generators, pandora for example, because with pandora the playlists that the user is given is based off the traits of songs they liked for that playlist, which they have called the music genome project. With our program, the playlists are entirely based off of an activity and mood that the user chooses, so the playlist generated will have a wide variety of songs which can have many different traits but will all be related by the activity or the mood. This way the user is less likely to get random songs that don’t fit in with what they’re currently doing. Our program would be very useful on a day to day basis, by allowing the user to quickly choose music to play that is relevant to their current mood and/or activity. We would like to provide a variety of moods and activities to allow the application to be useful to a wide audience.

Our biggest difficulty will be designing a tagging system so that multiple tags can be added to a song, as well as have tags easily be added or removed by the user. Our database tables will need to be designed so that these tag changes are simple and straightforward while not compromising the integrity of the database. For example, the tags can be stored in a separate table with a foreign key to the songs table. A join can then link the songs with their tags, and the output table presented in a graphical manner to the user. We also might have trouble with combining the Java portion with the SQL database so that they can work together. So we will have at least five tables which include the songs, moods (weak entity), activities (weak entity), mood playlist (from joining mood and songs), and activity playlist (from joining activities and songs).

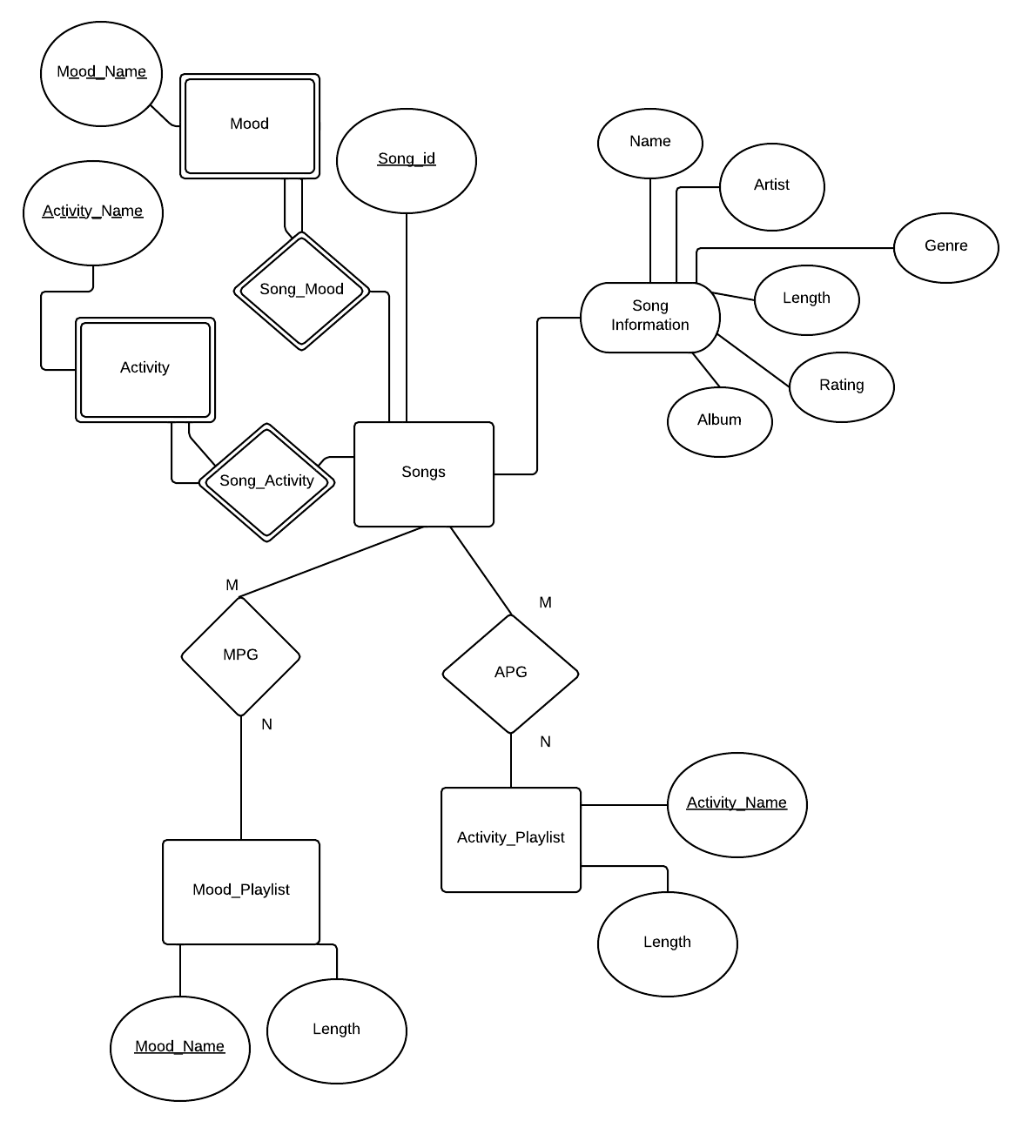
**List of Features**

* Create a playlist based on mood of the user (user input)
* Create a playlist based on activity of the user (user input)
* Create a playlist based on both activity and mood of the user (user input)

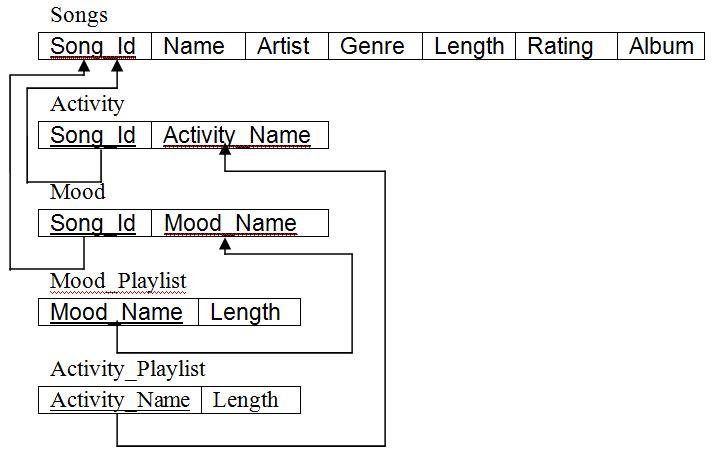
**Optional (if we have time) Features**

* User can create multiple different playlists for one mood or activity
* Have a username to track moods and activities specific users listen to
* User can add songs to playlists generated
* User can delete songs from playlists generated
* Filter specific genres and artists from playlist

**ER Diagram**



**ER - Relation Mapping**



**SQL Table Statements**

CREATE TABLE SONGS(

SONG\_ID INT PRIMARY KEY NOT NULL,

NAME CHAR(50) NOT NULL,

ARTIST CHAR(50) NOT NULL,

GENRE CHAR(50) NOT NULL,

LENGTH REAL NOT NULL,

RATING INT,

ALBUM CHAR(50) NOT NULL);

CREATE TABLE MOOD\_PLAYLIST(

MOOD\_NAME CHAR(30) PRIMARY KEY NOT NULL,

TIME\_LISTENED REAL NOT NULL);

CREATE TABLE ACTIVITY\_PLAYLIST(

ACTIVITY\_NAME CHAR(30) PRIMARY KEY NOT NULL,

TIME\_LISTENED REAL NOT NULL);

CREATE TABLE ACTIVITY(

SONG\_ID INT PRIMARY KEY NOT NULL,

ACTIVITY\_NAME CHAR(30) PRIMARY KEY NOT NULL,

FOREIGN KEY(ACTIVITY\_NAME) REFERENCES ACTIVITY\_PLAYLIST(ACTIVITY\_NAME),

FOREIGN KEY(SONG\_ID) REFERENCES SONG(SONG\_ID));

CREATE TABLE MOOD(

SONG\_ID INT PRIMARY KEY NOT NULL,

MOOD\_NAME CHAR(30) PRIMARY KEY NOT NULL,

FOREIGN KEY(MOOD\_NAME) REFERENCES MOOD\_PLAYLIST(MOOD\_NAME),

FOREIGN KEY(SONG\_ID) REFERENCES SONG(SONG\_ID));

**SQL INSERT/DELETE/UPDATE and Looking Into Multiple Tables Statement/Query**

INSERT INTO MOOD\_PLAYLIST(

SELECT SONG\_ID, MOOD\_NAME

FROM MOOD LEFT JOIN MOOD

ON MOOD\_PLAYLIST.MOOD\_NAME = moodName (we get this from user input through java)) ;

INSERT INTO ACTIVITY\_PLAYLIST(

SELECT SONG\_ID, ACTIVITY\_NAME

FROM ACTIVITY LEFT JOIN ACTIVITY

ON ACTIVITY\_PLAYLIST.ACTIVITY\_NAME = activityName (we get this from user input through java));

**SQL Computing Aggregate Query**

SELECT SUM(LENGTH)

FROM SONGS, MOOD

WHERE MOOD.SONG\_ID = SONGS.SONG\_ID;

SELECT SUM(LENGTH)

FROM SONGS, ACTIVITY

WHERE ACTIVITY.SONG\_ID = SONGS.SONG\_ID;

**Code Instructions**

Open the project in Eclipse, and add both JDBC .jar files to the build path, ensuring the javadoc and source are properly defined. Run the Window.java file, this is our GUI. From here you can select either a mood or activity from one of the drop down menus, and search for the one you desire. The songs for each selection are displayed in the area below, and the total playlist length is displayed at the bottom.