Bill Ge (bge)

Meng Zhai (mzhai)

**1. Original Project Proposal**  
  
For our database, we have decided to populate it with information about music. We will be using the top 100 songs as judged by Billboard from 2006-2010. Listed below are the exact details of which tables and attributes we wish to have. We would have tables for the artist, song, and file and have the following relations: artist-song, file-song. The file information would come from the music file in iTunes.  
  
Some questions we are considering investigating into include:

* Which artist has been most popular according to Billboard? (based on the number of songs the artist has had in the top 100 over 5 years of data as well as the song’s ranking)
* Who is the most popular artist overall in the past 5 years? (based on the number of total songs on Billboard)
* What is the relation between gender (male/female) and the popularity of songs?
* What is the most common lengths of songs in these 500 popular songs?
* What genre of song is most popular?

The interface would be created in Java, and would allow the user to see the different tables, including the relational tables. The user would be able to submit SQLite queries and the result would be displayed, and the user can save the result as a file.  
  
Tables

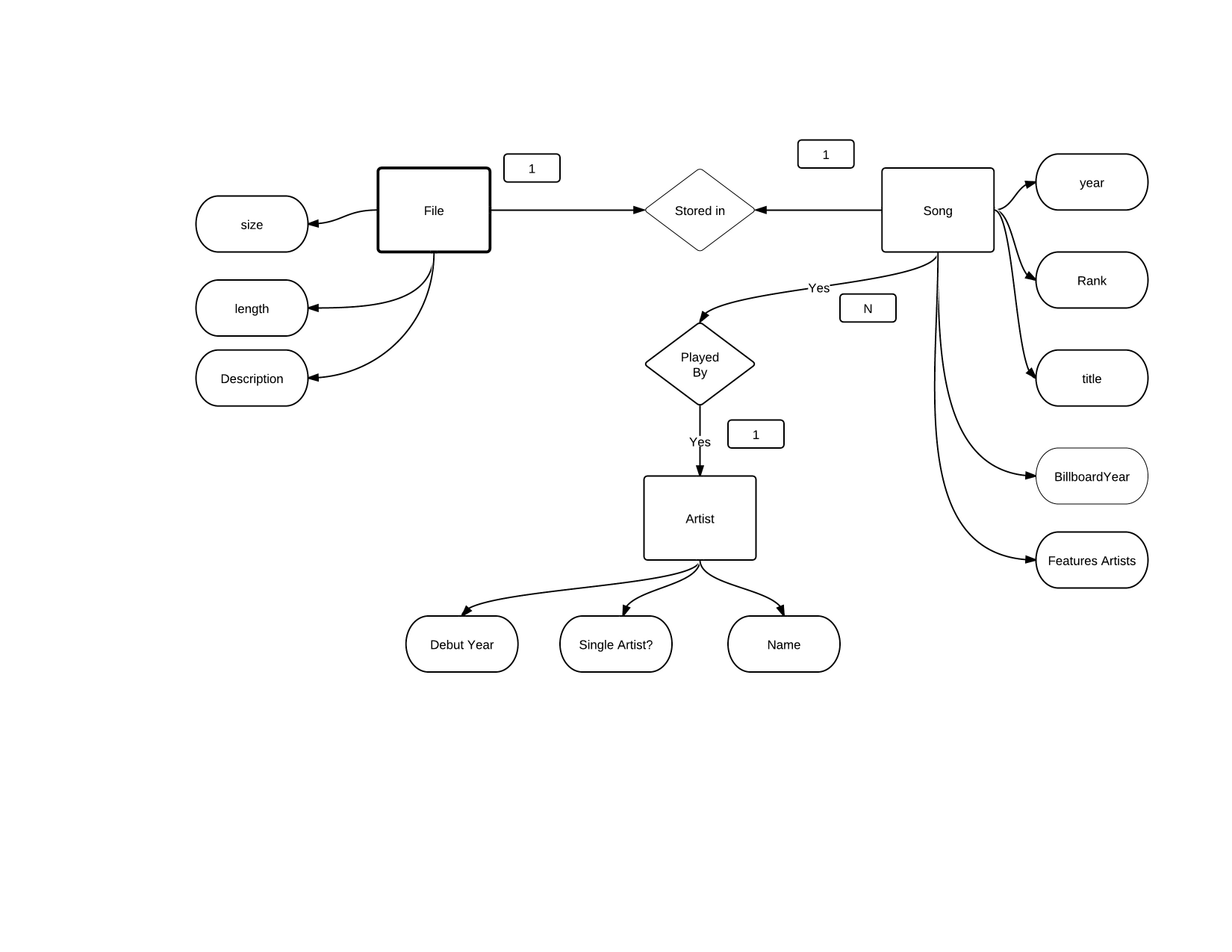
* File
  + Size of File in Bytes [int]
  + Description (artist-song-year) [string]
  + Length of File in seconds [int]
* Song
  + Title [string]
  + Year on billboard [int]
  + Position on billboard [int]
  + Features/multiple artists [boolean]
* Artist
  + Name [string]
  + Debut Year [int]
  + Group [boolean]

Relations

* Song-Artist
* Song-File

Informal Use Case Document

* Select: The user will be able to select a row or column, or multiple rows and columns by clicking on the heading of the row or column. Then, the user will be able to delete the record, etc.
* Views: After the user has selected some columns, they can press a view button that will let them see data only from those columns.
* Sorting: Given a view and by clicking on the columns, the table will be sorted by the column clicked. For example, if the user were to click on length of song, all the rows would be sorted by that particular attribute in ascending order.
* Delete: The user can select to delete either an artist, or a song, by selecting the row(s) and clicking the delete button. If the user deletes a song, the corresponding row will be removed from the songs table, and the corresponding row will also be removed from the file table. If the artist of the song only has that song, the row of that artist will be removed from the artist table. Otherwise, the artist will not be removed. If the user chooses to remove an artist, the corresponding row for that artist will be removed from the artist table, and all songs belonging to that artist will be removed from the songs and file tables.
* Add: There will be input fields where the user can add a record (the fields would be the same as the input fields) by filling in the fields and clicking on the add button. The user must fill out all the fields for the record to be added, so the application will check for that. The application will also check of make sure the domain is correct (ie, integer, or string, etc.) before adding the record. Once the record is added, the information will be shown on the corresponding tables.
* Save result: There will be a button that allows the user to save the current table that the user is seeing, as a csv file.
* Summarizing data: There will be text fields that display the average of appropriate fields (such as length of song in seconds, or size of file in bytes), and an “update” button. The average values will get updated when the button is clicked. The user could enter in a new record, click update, to see the updated averages.
* Export: The application will also have a function to allow the user to save the data from the current view as a .csv file.

ER Diagram  


Lower-Level Schema

* File Table
  + Size of File in Bytes [int]
  + Description (artist-song-year) [string]
  + Length of File in seconds [int]

CREATE TABLE File(

size INT,

length INT,

description TEXT

);

* Song Table
  + Title [string]
  + Year on billboard [int]
  + Position on billboard [int]
  + Features/multiple artists [boolean]

CREATE TABLE Song(  
    Title TEXT,  
    BillboardYear INTEGER  
    Rank INTEGER,  
    Features BOOLEAN  
);

* Artist Table
  + Name [string]
  + Debut Year [int]
  + Group [boolean]

CREATE TABLE Artist(

Name TEXT,

DebutYear INTEGER,

Group BOOLEAN

);  
  
Description of Input File Structure  
Our input files consists of text files exported from iTunes. We then use Microsoft Excel to read in both files and format them accordingly. The files are organized as listed below.  
  
Input File 1  
Field 1: Rank, integer (1-100)  
Field 2: Billboard Year, integer (2006-2010)  
Field 3: Title of song, text  
Field 4: Artist of song, text  
Field 5: Whether the song features other artists, boolean  
Field 6: Size of file of song in bytes, integer  
Field 7: Length of song in seconds, integer  
  
Input File 2  
Field 1: Name of artist, text  
Field 2: Debut Year of artist, int  
Field 3: Whether the artist is in a group/band, boolean

**2. Project Overview**

For our database, we have decided to populate it with information about music. We will be using the top 100 songs as judged by Billboard from 2006-2010. Listed below are the exact details of which tables and attributes we wish to have. We would have tables for the artist, song, and file and have the following relations: artist-song, file-song. The file information would come from the music file in iTunes.  
  
Some questions we are considering investigating into include:

* Which artist has been most popular according to Billboard? This is based on the number of songs the artist has had in the top 100 over 5 years of data as well as the song’s ranking. The user can select an artist and see how many top 100 songs that artist has.
* Which artists debuted in xxxx? The user can select a year and see which artists debuted in that year.
* Which artists are international? Which artists debuted in which states?
* What is the average length of songs in these 500 popular songs?
* What percent of these top 100 artists are bands as opposed to a single artist?

The interface would be created in Java, and would allow the user to see the different tables, including the relational tables. The user would be able to view lists of artists by year, country, and state, and view songs by artist. The user can import data through a csv file and export the data to a csv file. The user can also add entries and delete songs, artists, and files.

**3. Informal Use Case Document**

* Statistics: Relevant statistics are displayed: percent of songs that features multiple artists, percent of artists that is a group, average size of files in bytes, and average length of files in seconds. After the user manipulates the tables (add/delete entries), the user can press an “update all statistics” button to see updated statistics.
* Views: The user has 5 graphical views (5 tabs, with a table in each tab). Each view shows a different table: songs, artists, files, the relation between songs and artists, and the relation between songs and files. In the 6th (Main) tab, the user sees data that fits certain criteria (such as artists that debuted in a certain year, or songs that are by a certain artist). That data is displayed in a text box in csv format.
* Add: The user can add an entry in the same format as original entries. The user must enter in data into all fields to be able to add that entry (except for state, where “n/a” will be automatically entered if it’s left blank, if an artist is foreign). After the user enters in the data, they click a button to add the entry, and all tables will be updated. The application makes sure that the domain of each field is correct before adding the entry.
* Delete: The user can select to delete either an artist, or a song, by selecting the row and clicking the delete button. If the user deletes a song, the corresponding row will be removed from the songs table, and the corresponding row will also be removed from the file table. If the artist of the song only has that song, the row of that artist will be removed from the artist table. Otherwise, the artist will not be removed. If the user chooses to remove an artist, the corresponding row for that artist will be removed from the artist table, and all songs belonging to that artist will be removed from the songs and file tables.

Import and Export: The application includes a data file that we have already compiled to populate the database, but the user can import any csv file to populate the database, as long as it is in the correct format. The application imports from “input.csv”. The application will also have a function to allow the user to save the data from the current database to a csv file. After the user is done manipulating the tables, the user can export the database to “output.csv” where the file is a csv file in the format of the input file.  
  
**4. ER Diagram**

**5. Lower-Level Schema Documentation**

* File Table
  + Size of File in Bytes [int]
  + Description (artist-song-year) [string]
  + Length of File in seconds [int]

CREATE TABLE File(

Size INTEGER,

Length INTEGER,

Description TEXT

);

* Song Table
  + Title [string]
  + Year on billboard [int]
  + Position on billboard [int]
  + Features/multiple artists [boolean]

CREATE TABLE Song(  
    Title TEXT,  
    BillboardYear INTEGER  
    Rank INTEGER,  
    Features BOOLEAN  
);

* Artist
  + Name [string]
  + Debut Year [int]
  + Group [boolean]
  + State [string]
  + Country [string]

CREATE TABLE Artist(

Name TEXT,

DebutYear INTEGER,

Group BOOLEAN,

State TEXT

Country TEXT

);

* Song-Artist relation
  + Song Rank-Year [string]
  + Artist name [string]

CREATE TABLE Song-Artist (

Song TEXT,

Artist TEXT

);

* Song-File relation
  + Song Rank-Year [string]
  + File Description Artist-Song-Year [string]

CREATE TABLE Song-File (

Song TEXT,

File TEXT,

);

**6. Description of Input File Structure**  
Our input files consists of text files exported from iTunes. We then use Microsoft Excel to compile files from 2006 to 2010 and read in all files and formatted them accordingly. The resulting input.csv file is organized as follows:  
Rank, Year, Name, Artist, Features, Size, Time, Group, Debut Year, State, Country  
Field 1: Rank, integer (1-100)  
Field 2: Billboard Year, integer (2006-2010)  
Field 3: Title of song, text  
Field 4: Artist of song, text  
Field 5: Whether the song features other artists, boolean  
Field 6: Size of file of song in bytes, integer (positive)  
Field 7: Length of song in seconds, integer (positive)

Field 8: Whether the artist is in a group/band, Boolean

Field 9: Debut Year of artist, integer (positive)

Field 10: State of origin of the artist, text

Field 11: Country of origin of the artist, text  
 **7. How to run and build this project**

* + 1. The executable .jar file is included in MusicDatabase/MusicDatabase.jar. You can double-click and run the .jar file and interact with the program from there. Or, you can import the MusicDatabase project folder into your Eclipse environment, and run DatabaseProjectGUI.java (it contains the main class). MusicDatabase.jar MUST BE IN THE SAME DIRECTORY AS input.csv.
    2. Please be patient when you initially start up the application. It may take up to 20 seconds to load because it is populating the database from the input file.
    3. The input data file is located in MusicDatabase/input.csv. The current input.csv contains the original data we used for this project. If you would like to use different input data, you must change the contents of input.csv. DO NOT MOVE OR DELETE OR RENAME THIS FILE. You may change the contents of input.csv if it’s in the same format as the original input file.. When the “Import” button is clicked in the main panel of the application, the application will import from input.csv.
    4. The output data file is located in MusicDatabase/output.csv. There is no original output.csv included with the original application, but while running the application, the user can click “export” and the database will be created and saved in MusicDatabase/output.csv in the same format as the input.csv file. If output.csv already exists, it will be overwritten.

About this application:

* There are 6 tabs. The Main tab allows the user to add entries, update statistics, import, export, and see certain data.
  + The user can select an artist from “Artists” and click “Show Songs” to see all songs by the selected artist. The user can select a year from “Debut year” and click “Show Artists” to show all artists that debuted in the selected year. The user can select a country from “Country” and click “Show Artists” to show all artists originating from the selected country. Likewise with state. The output of these will be shown in the text box on the right in a csv format, for easy export (copy/paste).
  + The user can add a song by inputting all the data in the text boxes, and clicking on “Add Song” and the tables will be updated.
  + The user can input a database by changing the content in input.csv and clicking on the import button.
  + The user can export the current data by clicking on the export button. The output will be in output.csv in the same format as the input.
  + After the user manipulates the data, the user can click on “Update All Statistics” to update all the statistics in the other tabs.
* The Songs tab displays all the songs and the relevant song information. It also shows the percent of songs that feature multiple artists. You can select a row (a song) and delete that song by click on “delete song”. All the relevant tables will be updated. If that is the only song of an artist, that artist will be deleted.
* The Artists tab displays all the artists and their information. It also shows the percent of artists that is in a group. You can select a row of an artist and delete the artist by clicking on the “Delete Artist” button. All the relevant tables will be updated – all the songs of that artist will be deleted.
* The Files tab displays all the files and their information. It also shows the average size of files in bytes and the average length of files in seconds. You can delete a file by selecting the file and clicking on the “Delete File” button. All other relevant tables will be updated. If this is the only file of an artist, that artist will be deleted.
* The Song-Artist and Song-File tabs show which song corresponds to which artist and which song corresponds to which file respectively, by using foreign keys and primary keys.

**8. Citations**

The data we used were from our personal music collections. We obtained top 100 information from [www.billboard.com](http://www.billboard.com) Billboard Top 100. We used Apple ITunes to get information about song length and song size, and used ITunes to export all 500 music files as .csv files.