Apple Music

Dio Minott

4/28/17

Table of Contents

Executive Summary

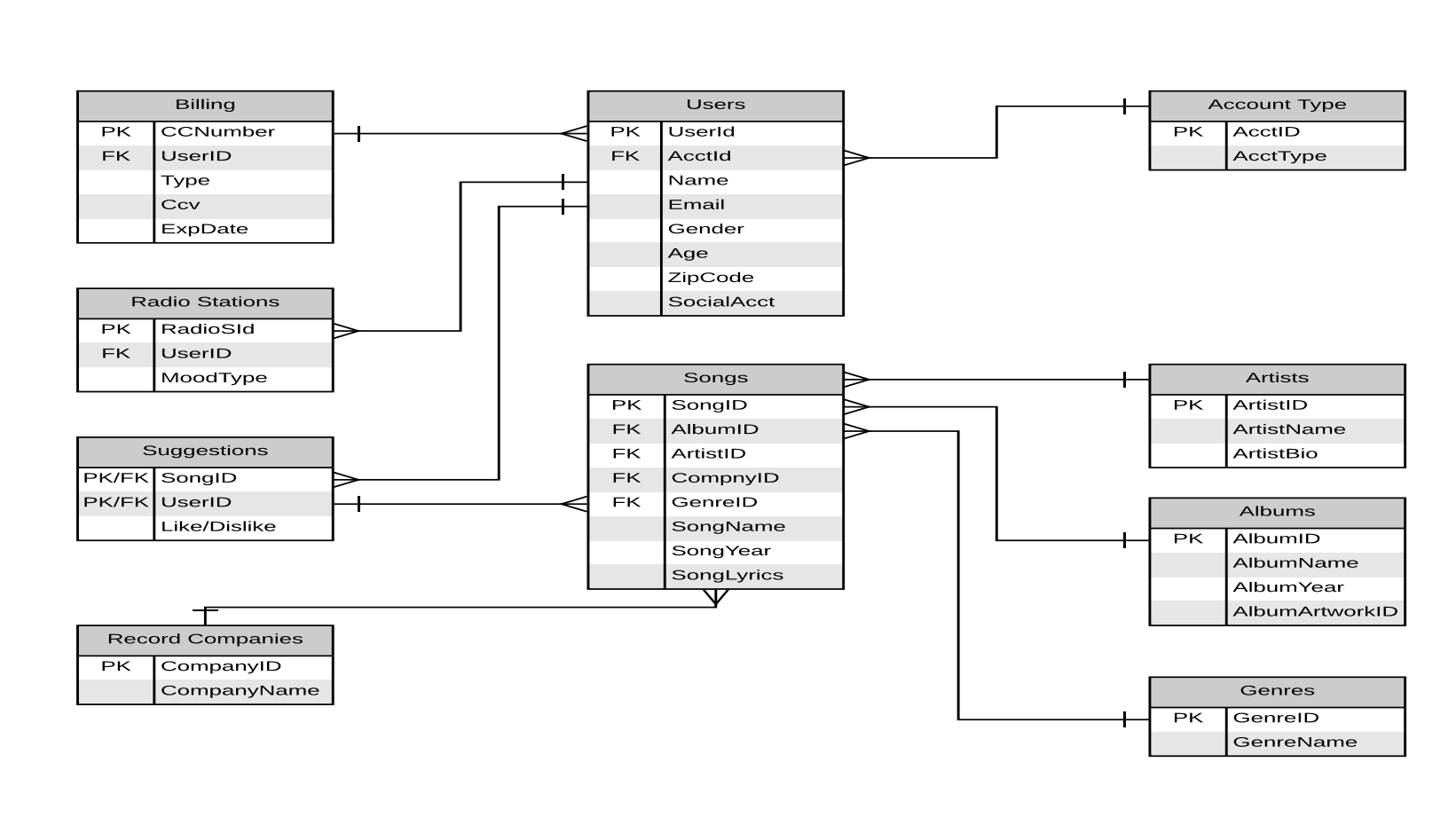
Overview

This is a database for consumers who stream music or listen to online radio of their choice. It can be challenging to capture every song or album out now for customers to stream, but It can be done. This setup will help organize and practice simplicity for the customer to stream music on the go.

Objectives

The purpose of this project is to create a database system to be able to enter and expose records of music artists, albums, songs, genres, Billing information, and what kind of music does the consumer likes to here. This will allow users to search what kind of music they want to listen to and be able to select a radio station and listen away all day long. Do to the simplicity of this database, it will also allow music artists to see their demographic and what's streaming most. This document will provide an overview of the database, along with functional dependencies, triggers, store procedures, and views.

Entity-Relationship Diagram



Tables

Artist Table

This table list all possible recording artist with legitimate music.

Functional Dependencies

artistID → artistName, artistBio

```
CREATE TABLE Artists (
        artistID char(3) not null,
        artistName text not null,
        artistBio text,
        primary key(artistID)
);
```

Albums Table

This table displays the artist album name, year, and artwork.

Functional Dependencies

albumID → albumName, albumYear, albumArtworkID.

```
CREATE TABLE Albums (
        albumID int not null,
        albumName text,
        albumYear int,
        albumArtworkID varchar(6)
        primary key(albumID)
);
```

Genres Table

This table displays the genre name of the music the user is listening to.

Functional Dependencies

genreID → genreName,

```
CREATE TABLE Genres (
        genreID int not null,
        genreName text not null,
        primary key(genreID)
);
```

Songs Table

The table displays the name of the song and info

Functional Dependencies

songID → albumID, artistID, companyID, genreID, songName, songYear, songLyrics.

```
CREATE TABLE Songs (
        songID varchar(5) not null,
        artistID char(3) text, not null,
        companyID text not null,
        genreID int not null,
        songName text,
        songYear int not null,
        songLyrics text,
        primary key(songID)
);
```

Billing Table

This table displays the consumers credit card info which includes the card number, type of card, ccv number, and the card expiration date.

Functional Dependencies

ccNumber → userID, type, ccv, expDate.

```
CREATE TABLE Billing (
        ccNumber int not null,
        userID varchar(6) not null,
        type text not null,
        ccv int not null,
        expDate int not null,
        primary key (ccNumber)
);
```

Suggestions Table

This table lets users select if the like or dislike the song they're listening to. This will be helpful for the companies focus group.

Functional Dependencies

songID,userID → likeDislike.

```
CREATE TABLE Suggestions (
        songID  varchar(5) not null,
        userID varchar(6) not null,
        likeDislike text
        primary key(songID,UserID)
);
```

Radio Stations Table

This table displays what radio stations the user is listening to and gives users suggestions depending what mode they're in.

Functional Dependencies

radioSID → userID, moodType.

```
CREATE TABLE RadioStations (
        radioSID varchar(3),
        userID varchar(6) not null,
        moodType text,
        primary key(radioSID)
);
```

Account Type Table

This table displays if consumer has a trial or paid account.

Functional Dependencies

acctID → acctType

```
CREATE TABLE AccountType (
        acctID text not null ,
        acctType text not null,
        primary key(acctID)
);
```

Users Table

This table displays information about the user also what kind of social media account they're connected too.

Functional Dependencies

userID → acctID, name, email, gender, age, zipCode, socialAcct .

```
CREATE TABLE Users (
        userID varchar(6) not null,
        acctID text not null ,
        name text not null,
        email text,
        gender text not null,
        age int,
        zipCode int not null,
        socialAcct text,
        primary key(userID)
);
```

Record Companies Table

This table displays the record company that has the copyrights to the song that's being displayed

Functional Dependencies

companyID → companyName.

```
CREATE TABLE RecordCompanies (
        companyID text not null,
        companyName text not null
        primary key (companyID)
);
```

Stored Procedures

Purpose

This procedure is used if the artist changes his/hers name.

```
CREATE OR REPLACE FUNCTION changeAlias (text,text, REFCURSOR)
RETURNS REFCURSOR AS
$$
        DECLARE
                old text: = $1;
                new text: = $2;
                resultset REFCURSOR: = $3;
        BEGIN
                open resultset for
                UPDATE Artists
                SET artistName = new
                WHERE artistName = old;
                RETURN resultset;
        END;
$$ LANGUAGE plpgsql;
```

Triggers

Purpose

The purpose of this trigger is for when the user changes they're name.

```
CREATE TRIGGER userChange
AFTER UPDATE ON Users
FOR EACH ROW EXECUTE PROCEDURE changeAlias();
```

Security

For Administrators:

GRANT ALL PRIVILAGES ON ALL TABLES IN SCHEMA public TO admin;

For Users:

GRANT INSERT, SELECT, UPDATE, DELETE ON Billing TO Users;

Implementation Notes

There are some suggestions for this database. Its best for the administrator using this database to focus on the user table first before looking at any other table. This will give the user a better understanding what they re looking at. Also, for security users are able to add another card or delete card information. Artists are able to observe they're demographic of like and unlike songs.

Known Problems

Mostly everything in this report was correct except for the display of my artist table. My output only displayed the first artist in the database table. Even though I executed a good input, I still couldn't’t figure out the problem.

Future Improvements

In the future I would like to see the following improvements:

- A way to add and update artist artwork.

- Away for the artist and record label to view users mood type as they select as a track list to listen too.