Database Design

April 20, 2016

David Emory

Table of Contents

Executive Summary

Overview

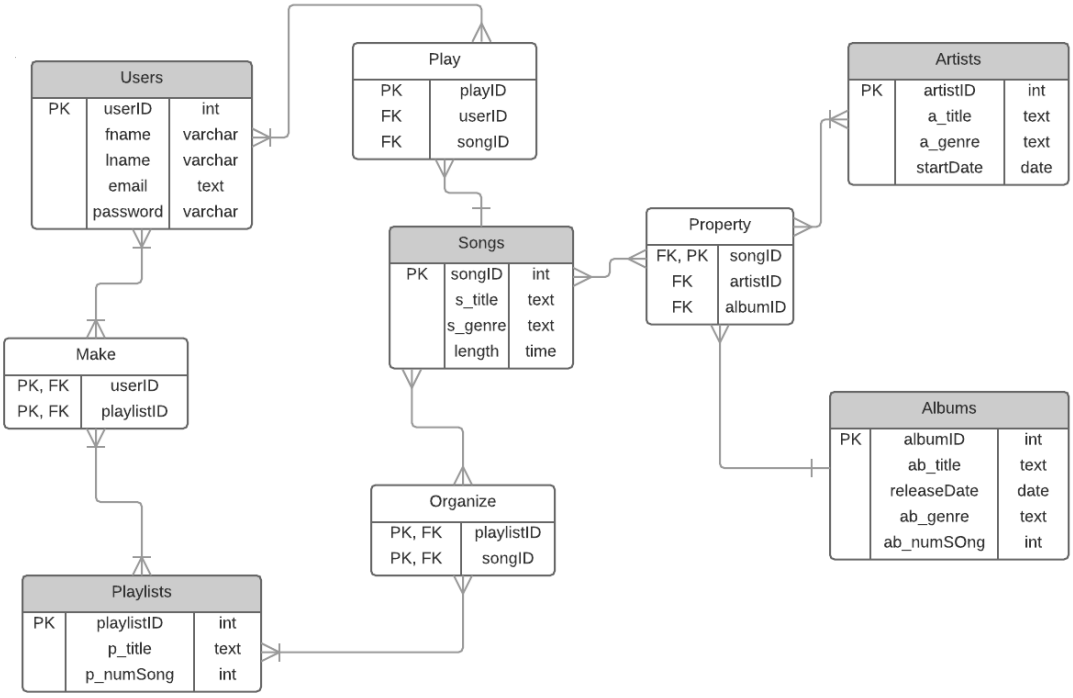
The American population revolves highly around the music industry for daily entertainment. Throughout history sources of music have evolved from cassettes to digital downloads. However, with the ever evolving music industry it can be difficult to keep track and listen to all of your favorite bands in one convenient place. Music lovers can now utilize the incredibly expedient and resourceful music application, Spotify. Users can join for free or pay a reasonable price to become premium members and have access to millions of artists, songs, and albums all at the touch of their fingers.

Objectives

This document describes the structure and entities involved in the creation of the music player/music provider Spotify. Spotify provides a variety of music to people around the world at a low monthly cost through a client that access a database.

The organization of its database Spotify has been able to provide features to its users through its client software. With this software users are able to listen to songs, albums, artists, personal playlists, public playlists, and the radio. Spotify's mission is to provide a wide variety of music to its customers on an organized and easily navigable web-based application that gets its data from a normalized and beautiful database

Entity Relationship Diagram



Tables

Songs

Purpose

This table contains the songID, title, and genre of all the songs in this database.

Create Statement

```
CREATE TABLE Songs(
     songID int NOT NULL,
     s_title text NOT NULL,
     s_genre text NOT NULL CHECK
          (s_genre in ('Country', 'Rap', 'Rock', 'Alternative',
           'Pop', 'Hip-Hop', 'Other')),
  PRIMARY KEY(songID)
);
```

Functional Dependencies

songID → s_title, s_genre, length

Sample Data

Data Output | Explain | Messages | History

	songid integer	s_title text	s_genre text
1	1	Roses	Other
2	2	Kanye	Other
3	3	Stronger	Rap
4	4	Cigarette Daydreams	Alternative
5	5	Candyman	Other
6	6	Welcome to the Jungle	Rock
7	7	Tangerine Girl	Rap
8	8	The Home	Pop
9	9	Gold on the Ceiling	Rock
10	10	Ms.Jackson	Hip-Hop
11	11	Red Eye	Hip-Hop
12	12	Life at the Outpost	Country

Users

Purpose

This table contains the userID, first and last name, email (if provided), password, and gender of a person's user account.

Create Statement

```
CREATE TABLE Users(
     userID int NOT NULL,
     fname varchar(36) NOT NULL,
     lname varchar(36) NOT NULL,
     email text DEFAULT 'None',
     password varchar(40) NOT NULL,
     gender text NOT NULL CHECK
        (gender in ('Male', 'Female')),
 PRIMARY KEY(userID)
);
```

Functional Dependencies

userID → fname, lname, email, password, gender

Sample Data

Data Output | Explain | Messages | History

	userid integer	fname character varying(36)	lname character varying(36)	email text	password character varying(40)	gender text
1	1	David	Emory	David.emory@marist.edu	carrots	Male
2	2	Dom	Emory		pineapple	Male
3	3	Lauren	Emory	lauren.emory@marist.edu	chestnuts	Female
4	4	Bob	Emory	Remory@gmail.com	jesus	Male
5	5	Andrew	Steere		captncrunch	Male
6	6	Danielle	Carreri	DeeDee@gmail.com	puppies	Female
7	7	Rosa	Emory	rosa.emory@marist.edu	interlude	Female
8	8	Nick	Esposito	nicholas.esposito3@marist.edu	Trop	Male
9	9	Chris	Fushcetti	crazy8@hotmail.com	nigel	Male
10	10	Emma	Fern		trees	Male
11	11	Bob	Marley	420blazeit@hotmail.com	blazeit	Male
12	12	Billy	Emory		belle	Male
13	13	Charlie	Bruce	None	Farquad	Male

Playlists

Purpose

This table contains the playlists that have been made, what songs are in them and the playlistID.

Create Statement

```
CREATE TABLE Playlists(
     playlistID int NOT NULL,
     p_title text NOT NULL,
     p_numSong int NOT NULL,
  PRIMARY KEY(playlistID)
);
```

Functional Dependencies

playlistID → p_title, p_numSong

Sample Data

Data Output | Explain | Messages | History

	playlistid integer	p_title text	p_numsong integer
1	1	Bump	32
2	2	LIT	25
3	3	Pump up	20
4	4	Party	50
5	5	Hell	666
6	6	Codd	102
7	7	interlude	4
8	8	Trop	15
9	9	Get It	10
10	10	Work Out	25
11	11	blazeit	56
12	12	Driving	80

Artists

Purpose

This table contains the artistID, title of the artist's band or his/her name, the genre of the artist, and the start date of the artist's career.

Create Statement

```
CREATE TABLE Artists(
     artistID int NOT NULL,
     a_title text NOT NULL,
     a_genre text NOT NULL CHECK
          (a_genre in ('Country', 'Rap', 'Rock', 'Alternative',
           'Pop', 'Hip-Hop', 'Other')),
     startDate date NOT NULL,
  PRIMARY KEY(artistID)
);
```

Functional Dependencies

artistID → a_title, a_genre, startDate

Sample Data

Data Output | Explain | Messages | History

	artistid integer	a_title text	a_genre text	startdate date
1	1	Action Bronson	Rap	2001-10-15
2	2	Chance the Rapper	Rap	1999-12-01
3	3	Fall Out Boy	Alternative	1997-06-20
4	4	Asher Roth	Rap	1999-07-30
5	5	Kanye West	Rap	2000-10-25
6	6	Saitin	Other	0666-12-03
7	7	Galantis	Other	2007-11-01
8	8	Caged the Elephant	Alternative	2005-03-15
9	9	Beyonce	Pop	2000-05-11

Albums

Purpose

This table contains the albumID, title of the album, release date of the album and the number of songs in the album.

Create Statement

```
CREATE TABLE Albums(
    albumID int NOT NULL,
    ab_title text NOT NULL,
    releaseDate date NOT NULL,
    ab_genre text NOT NULL CHECK
        (ab_genre in ('Country', 'Rap', 'Rock', 'Alternative',
         'Pop', 'Hip-Hop', 'Other')),
    ab_numSong int NOT NULL,
 PRIMARY KEY(albumID)
);
```

Functional Dependencies

albumID → ab_title, releaseDate, ab_genre, ab_numSong

Sample Data

Data Output | Explain | Messages | History

	albumid integer	ab_title text	releasedate date	ab_genre text	ab_numsong integer
1	1	Mr. Wonderful	2001-10-15	Rap	13
2	2	Acid Drip	1999-12-01	Rap	15
3	3	From Under the Cork Tree	1997-06-20	Alternative	20
4	4	RetroHash	1999-07-30	Rap	12
5	5	Graduation	2000-10-25	Rap	15
6	6	Saitin	0666-12-03	Other	666
7	7	Galantis	2007-11-01	Other	10
8	8	Melophobia	2005-03-15	Alternative	14
9	9	Beyonce	2000-05-11	Pop	12

Play

Purpose

This table displays the songs people have played in the past.

Create Statement

```
CREATE TABLE Play(
   playID int NOT NULL,
   uID int NOT NULL REFERENCES Users(userID),
   sID int NOT NULL REFERENCES Songs(songID),
 PRIMARY KEY(playID)
);
```

Functional Dependencies

playID → uID, sID

Sample Data

Data Output | Explain | Message

	playid integer	uid integer	sid integer
1	1	1	1
2	2	1	4
3	3	5	1
4	4	4	10
5	5	3	5

Organize

Purpose

This table show which songs are organized into playlists and what playlists the songs are in.

Create Statement

```
CREATE TABLE Organize(
    p_ID int NOT NULL REFERENCES Playlists(playlistID),
    s_ID int NOT NULL REFERENCES Songs(songID),
 PRIMARY KEY(p_ID, s_ID)
);
```

Functional Dependencies

p_ID, s_id →

Sample Data

Data Output | Explain

	p_id integer	s_id integer
1	1	1
2	2	1
3	3	5
4	2	5
5	4	3
6	4	5
7	4	1
8	5	7
9	3	4
10	7	4
11	6	4
12	5	3

Make

Purpose

This table shows which users have mad playlists of music.

Create Statement

```
CREATE TABLE Make(
    u_ID int NOT NULL REFERENCES Users(userID),
    pID int NOT NULL REFERENCES Playlists(playlistID),
 PRIMARY KEY(u_ID, pID)
);
```

Functional Dependencies

u_ID, pID →

Sample Data

Data Output | Explain

	u_id integer	pid integer
1	1	1
2	3	1
3	5	5
4	5	8
5	1	3
6	3	5
7	4	11
8	5	7
9	3	4
10	10	4
11	6	4
12	5	3

Property

Purpose

This table displays which songs belong to a certain artist, which song belongs to a specific album, and which artist owns a specific album.

Create Statement

```
CREATE TABLE Property(
    sid int NOT NULL REFERENCES Songs(songID),
    aid int NOT NULL REFERENCES Artists(artistID),
    abid int NOT NULL REFERENCES Albums(albumID),
 PRIMARY KEY(sid)
);
```

Functional Dependencies

sID → aID, abID

Sample Data

Data Output | Explain | Message

	sid integer	aid integer	abid integer
1	1	1	1
2	3	1	6
3	2	5	2
4	5	8	2
5	6	3	3
6	7	5	4
7	4	9	9
8	11	7	4
9	8	4	7
10	10	4	1
11	9	4	8
12	12	3	8

Views

Descriptive Albums

Purpose

This allows a user to see what artist owns an album and what songs are on the album.

Create Statement

```
CREATE VIEW descriptiveAlbums AS
   SELECT al.ab_title,
          s.s_title AS Song,
          ar.a_title AS Artist
   FROM   Albums al,
          Songs s,
          Artists ar,
          Property p
   WHERE s.songID = p.sid
     AND al.albumID = p.abid
     AND ar.artistID = p.aid
   ORDER BY al.ab_title DESC;
```

Sample Data

Data Output | Explain | Messages | History

	ab_title text	song text	artist text
1	Mr. Wonderful	Roses	Action Bronson
2	Saitin	Stronger	Action Bronson
3	Acid Drip	Kanye	Kanye West
4	Acid Drip	Candyman	Caged the Elephant
5	From Under the Cork Tree	Welcome to the Jungle	Fall Out Boy
6	RetroHash	Tangerine Girl	Kanye West
7	Beyonce	Cigarette Daydreams	Beyonce
8	RetroHash	Red Eye	Galantis
9	Galantis	The Home	Asher Roth
10	Mr. Wonderful	Ms.Jackson	Asher Roth
11	Melophobia	Gold on the Ceiling	Asher Roth
12	Melophobia	Life at the Outpost	Fall Out Boy

Playlists and their Songs

Purpose

This view will allow users to see what songs are in what playlists.

Create Statement

```
CREATE VIEW playlistSongs AS
     SELECT pl.p_title AS Playlist,
            s.s_title AS Song
     FROM   Playlists pl,
            Songs s,
            Organize o
     WHERE  s.songID = o.s_ID
       AND  pl.playlistID = o.p_ID
     ORDER BY pl.p_title DESC
```

Sample Data

Data Output | Explain | Messages | History

	playlist text	song text
1	Pump up	Candyman
2	Pump up	Cigarette Daydreams
3	Party	Stronger
4	Party	Candyman
5	Party	Roses
6	LIT	Roses
7	LIT	Candyman
8	interlude	Cigarette Daydreams
9	Hell	Stronger
10	Hell	Tangerine Girl
11	Codd	Cigarette Daydreams
12	Bump	Roses

Reports

Average Playlists Created

Purpose

This provides the administration with the amount of playlists currently residing in the database. The administration can then take further action if playlists need to be deleted.

Query

```
SELECT ROUND(AVG(p.playlistID)) AS playlistAmount
FROM  Playlists p;
```

Sample Data

Data Output | Explain

	playlistamount numeric
1	7

Rap Listeners

Purpose

This allows administration to further identify whom their demographic is and how to increase and diversify their customers.

Query

```
SELECT AVG(u.userID) AS RapListeners
FROM Users u,
     Play p,
     Songs s,
     Property pr,
     Artists a
WHERE u.userID = p.uID
 AND  p.sid = s.songID
 AND  pr.sid = s.songID
 AND  pr.aid = a.artistID
 AND a_genre = 'Rap';
```

Sample Data

	raplisteners numeric
1	3.3333333333333333

Stored Procedures

Insert Song Titles

Purpose

The purpose of this is to allow for song titles to be inserted into songs that have null values after the table is updated. This ensure that the column for song titles is not deleted from the Songs table.

Query

```
CREATE OR REPLACE FUNCTION insertSongTitle()
RETURNS trigger AS
$$
  BEGIN
    IF NEW.s_title IS NULL THEN
       RAISE EXCEPTION 'The song needs a title yo';
    END IF;
    INSERT INTO Songs(s_title)
          VALUES (NEW.s_title);
    RETURN NEW;
  END;
$$ LANGUAGE plpgsql;
```

Triggers

Song Title

Purpose

This trigger will force a stored procedure (Insert Song Title) to run, assuring that the Songs table is updated correctly.

Query

```
CREATE TRIGGER songTitle
AFTER UPDATE ON Songs
    FOR EACH ROW EXECUTE PROCEDURE insertSongTitle();
```

Security

There is only one primary user on the front end of this system and those are the customers. We cannot allow users to do too much, however, we can give them a few basic permissions. Also, we must include the Database Administrator, who will have all privileges.

Customers

Customers will be granted privileges to select data from most tables in the database. Additionally the users will be granted insert and update privileges to the playlists they have created, as well as, on their Users account.

```
GRANT SELECT, UPDATE, INSERT ON Playlists TO Customers;
GRANT SELECT, UPDATE ON Users TO Customers;
GRANT SELECT ON Songs TO Customers;
GRANT SELECT ON Artists TO Customers;
GRANT SELECT ON Albums TO Customers;
```

Database Administrator

This DBA will have Codd-like powers over the database.

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO dbAdmin;
```

Implementation Notes

The following is for the implementation of this system.

- First, a customer will need to install the Spotify client. This allows for the customer to access the music database. However, a customer can access the system through the internet, though there will be features that a customer will not be able to use on the web.
- When a customer creates a playlist or searches for a song, artist, album, or a genre of music he/she will be using a search bar or navigating the database through a GUI (graphical user interface). This way customers will not be able to run complex queries that may present threats to the system.
- Only the database administrator may add data to the database or modify/update the database as needed.

Known Problems

This is a list of issues that may be persistent with the current system:

- Customers are not able to favorite songs, artists, or albums. They can still listen to the songs they would like, however, the convenience of a list that contains all of a customer's favorite songs, which can be updated easily, is not possible.
- A customer cannot have the same song in a playlist (based off of songID, not song name).

- The stored procedures and triggers may be buggy.

Future Enhancements

Here are aspirations for further improving this system.

- The availability of a radio that randomly plays music based on a specific genre or artist a customer likes.
- A premium option for customer user accounts, this option will provide customers with more features than a non-premium customer.
- The addition of more music genres.
- Advertisements for artists' performances.