PANDORA

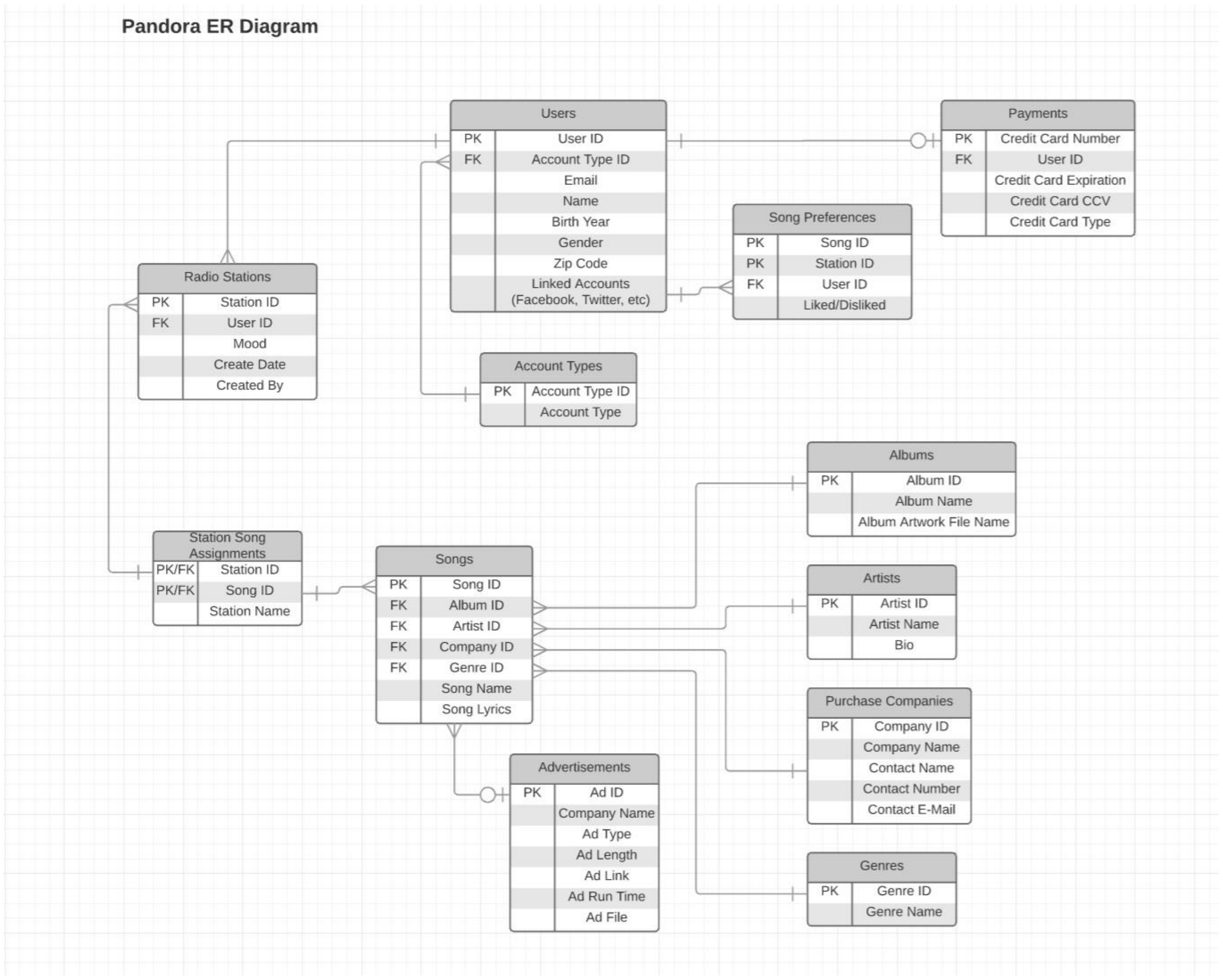
internet radio

Kelly Jones

Database Management

4/20/2016

Entity Relationship Diagram



Tables

Users

Purpose

This table contains the information of all of the user accounts

Create Statement

```
CREATE TABLE users (
        UserID text not null,
        AccountTypeID text not null references AccountType(AccountTypeID),
        Email text not null,
        Name text not null,
        Birth Year varchar (4) not null,
        Gender text not null,
        Zip Code varchar (6),
        Primary key (UserID)
);
```

Functional Dependencies

UserID → AccountTypeID, Email, Name, Birth Year, Gender, Zip Code

Sample Data

User ID	**Account Type ID**	**Email**	**Name**	**Birth Year**	**Gender**	**Zip Code**
11000	1	Kelly@email.com	Kelly	1996	Female	06110
21000	1	Jake@email.com	Jake	1982	Male	04416
31000	2	Henry@email.com	Henry	2001	Male	03911
41000	1	Hannah@email.com	Hannah	1999	Female	21224

Account Types

Purpose

This table stores the different types of accounts available on Pandora, Free users and paid users

Create Statement

```
CREATE TABLE accountTypes (
        AccountTypeID text not null,
        AccountType text not null,
        Primary key (AccountTypeID)
);
```

Functional Dependencies

AccountTypeID → AccountType

Sample Data

AccountTypeID	**AccountType**
1	Free
2	Paid

Payments

Purpose

This table stores the information of the user that they use to pay for the service, such as credit card information

Create Statement

```
CREATE TABLE payments (
        CreditCardNumber text not null,
        UserID text not null references users (UserID),
        CreditCardExpiration text not null,
        CreditCardCCV text,
        CreditCardType text,
        Primary key (CreditCardNumber)
);
```

Functional Dependencies

CreditCardNumber → UserID, CreditCardExpiration, CreditCardCCV, CreditCardType

Sample Data

CreditCardNumber	**UserID**	**CreditCardExpiration**	**CreditCardCCV**	**CreditCardType**
1234567887654321	31000	05/15	670	MasterCard
2948270198472193	69320	07/15	290	Visa

*This is completely made up credit card information and is not real information for anyone's real credit card as far as I am aware

Song Preferences

Purpose

This table stores the information of what songs a user has liked or disliked on which station

Create Statement

```
CREATE TABLE songPreferences (
        SongID text not null references Songs (SongID),
        UserID text not null references Users (UserID),
        LikedDisliked text,
        StationID text not null references Radio Stations (StationID),
        Primary key (SongID, StationID)
);
```

Functional Dependencies

Song ID, Station ID → UserID, LikedDisliked

Sample Data

SongID	**UserID**	**LikedDisliked**	**StationID**
827394	31000	Liked	87
182973	11000	Disliked	849
298374	41000	Liked	293

Radio Stations

Purpose

This table stores the information of the different stations made by the users

Create Statement

```
CREATE TABLE radioStations (
        StationID text not null,
        UserID text not null references Users (UserID),
        CreateDate date,
        CreatedBY text,
        Primary key (StationID)
);
```

Functional Dependencies

StationID → UserID, CreateDate, CreatedBy

Sample Data

StationID	**UserID**	**CreateDate**	**CreatedBy**
923	31000	04/19/2015	31000
392	11000	02/17/2015	31000

Station Song Assignments

Purpose

This table connects specific songs to specific stations

Create Statement

```
CREATE TABLE stationSongAssignments (
        StationID text not null references RadioStations (StationID),
        SongID text not null references Songs (SongID),
        StationName text,
        Primary key (StationID)
);
```

Functional Dependencies

StationID → SongID, StationName

Sample Data

StationID	**SongID**	**StationName**
827	37042	Green Day Radio
290	28932	Matchbook Romance Radio
472	78320	Bruises Radio

Songs

Purpose

This table stores the information of all of the songs

Create Statement

```
CREATE TABLE songs (
        SongID text not null,
        AlbumID text not null references Albums (AlbumID),
        ArtistID text not null references Artists (ArtistID),
        CompanyID text not null references PurchaseCompanies (CompanyID),
        GenreID text not null references Genres (GenreID),
        SongName text,
        SongLyrics text,
        Primary key (SongID)
);
```

Functional Dependencies

SongID → AlbumID, ArtistID, CompanyID, GenreID, SongName, SongLyrics

Sample Data

SongID	AlbumID	ArtistID	CompanyID	GenreID	SongName	SongLyrics
32431	534	1233	1235	523	Bruises	Haven't seen you since high school Good to see you're still beautiful Gravity hasn't started to pull Quite yet I bet you're rich as hell One that's five and one that's three Been two years since he left me Good to know that you got free That town I know was keeping you down on your knees Etc…**

**This would continue for the rest of the lyrics as well, but I've cut it off for special reasons.

Advertisements

Purpose

This table stores the information of the different advertisements that they show to the free users

Create Statement

```
CREATE TABLE advertisements (
        AdID text not null,
        CompanyName text,
        AdType text,
        AdLength text,
        AdLink text,
        AdRunTime text,
        AdFile text,
        Primary key (AdID)
);
```

Functional Dependencies

AdID → CompanyName, AdType, AdLength, AdLink, AdRunTime, AdFile

Sample Data

AdID	CompanyName	AdType	AdLength	AdLink	AdRunTime	AdFile
3972	Google	Video	00:30	http://www.google.com	3 months	gad.mp4
2811	Gatorade	Picture	NULL	http://www.gatorade.com/	1 month	gaad.jpeg

Albums

Purpose

This table stores the information of the different albums the songs belong to

Create Statement

```
CREATE TABLE albums (
        AlbumID text not null,
        AlbumName text,
        AlbumArtwork text,
        Primary key (AlbumID)
);
```

Functional Dependencies

AlbumID → AlbumName, AlbumArtwork

Sample Data

AlbumID	**AlbumName**	**AlbumArtwork**
534	California 37	Cali37art.jpeg
367	Voices	Voices.jpeg

Artists

Purpose

This table stores the information of all of the artists of the songs

Create Statement

```
CREATE TABLE artists (
        ArtistID text not null,
        ArtistName text,
        Bio text,
        Primary key (CreditCardNumber)
);
```

Functional Dependencies

ArtistID → ArtistName, Bio

Sample Data

ArtistID	**Artist Name**	**Bio**
8397	Shawn Mendes	Canadian singer-songwriter and model
1928	Matchbook Romance	American rock band from Poughkeepsie, New York and was formed in 1997. They are signed to Epitaph Records.

Purchase Companies

Purpose

This table stores the information of the companies that purchase the rights to the songs

Create Statement

```
CREATE TABLE purchaseCompanies (
        CompanyID text not null,
        CompanyName text,
        ContactName text,
        ContactEmail text,
        Primary key (CompanyID)
);
```

Functional Dependencies

CompanyID → CompanyName, ContactName, ContactEmail

Sample Data

CompanyID	**CompanyName**	**ContactName**	**ContactEmail**
29837	Music Acquire	Melissa	Melissa@email.com

Genres

Purpose

This table stores the information of the different genres

Create Statement

```
CREATE TABLE genres (
        GenreID text not null,
        GenreName text,
        Primary key (GenreID)
);
```

Functional Dependencies

GenreID → GenreName

Sample Data

GenreID	**GenreName**
12	Classical
13	Country
14	Rock & Roll

Views

Songs Played

Purpose

This view shows what songs have already been played during the current session in order to avoid repeated songs

Create Statement

```
CREATE VIEW songsPlayed AS
    SELECT songs.SongName
    FROM songs
    WHERE songs.songID in (SELECT stationSongAssignments.songID
                           FROM StationSongAssignments
                           WHERE stationSongAssignments.stationID= *currentStationID);
```

Sample Output

Songs.SongName
Monsters
Spongebob Theme Song
Twinkle Twinkle Little Star
Bruises
Hello
Bang Bang

Reports

TotalLikes

Purpose

This reports on the total amount of likes a song has

Create Statement

```
SELECT count(songPreferences.songID)
FROM songPreferences
WHERE LikedDisliked = ‘liked’);
```

TotalDislikes

Purpose

This reports on the total amount of dislikes a song has

Create Statement

```
SELECT count(songPreferences.songID)
FROM songPreferences
WHERE LikedDisliked = ‘disliked’);
```

Stored Procedures

likeSong

Purpose

Let a user like a song and insert their preference into the Song Preferences table.

Create Statements

```
CREATE OR REPLACE FUNCTION likeSong(INT, INT, INT, REFCURSOR) RETURNS refcursor
AS

$$

DECLARE

    song        INT             := $1;

    station     INT             := $2;

    user        INT             := $3;

    resultset   REFCURSOR       := $4;

BEGIN

    open resultset FOR

        INSERT INTO songPreferences (songID, stationID, userID, likedDisliked)

        VALUES (song, station, user, ‘liked’);

    RETURN resultset;

END;

$$

LANGUAGE plpgsql
```

dislikeSong

Purpose

Let a user like a song and insert their preference into the Song Preferences table.

Create Statements

```
CREATE OR REPLACE FUNCTION dislikeSong(INT, INT, INT, REFCURSOR) RETURNS
refcursor AS

$$

DECLARE

    song        INT             := $1;

    station     INT             := $2;

    user        INT             := $3;

    resultset   REFCURSOR       := $4;

BEGIN

    open resultset FOR

        INSERT INTO songPreferences (songID, stationID, userID, likedDisliked)

        VALUES (song, station, user, ‘disliked’);

    RETURN resultset;

END;

$$

LANGUAGE plpgsql
```

Security

The part of this database that needs to be the best protected is the Credit Card information, and only those with who absolutely needs to have access to it to avoid the possibilities of the important information getting into the wrong hands. After that, all of the personal information on the users is very important to protect as well, and should only be visible to whoever the user themselves decides they want to share it with. The information of the companies should also only be given at the discretion of the company. The songs, artists, albums, genres, and ads are all information that anyone can have access to.

Implementation Notes

This database is already implemented into the web application in a very elegant way. The user interface is extremely well-designed and easy to use.

Known Problems

The algorithm used to find a song to play based on the user input is flawed since the song that a user puts in to create a station is often never played on that station.

Future Enhancements

In the future, I would find a way to change the algorithm to make it so the inputted song is the first one played on the station since if a user likes the song enough to make a whole station around it, they must want to hear it.