Team Sigma – Portfolio

Reg Rotator

Functionality

As outlined in the proposal, all functionality that is available with the paper card system will be brought to the new, digital system.

These functions are:

1. Add new albums, with descriptions, RIYL, and Try tracks
2. Create a playlist for a shift
3. Keep track of the number of plays each song receives
4. Obtain Top 30 list for albums each week
5. View albums currently in rotation
6. View albums that were previously in rotation and sort by date released
7. View albums by artist
8. Browse albums, artists, songs, playlists in a sensible fashion

Evolution of Our Scenario

Throughout the process of this term project, our scenario has evolved very little. Because we are extremely familiar with the domain of our scenario, we already had an essentially perfect concept of what we needed to create.

Small changes were made to our database in which we created helper tables to store multiple links between different data members. Additionally, after the design phase we also moved some attributes to other tables for easier comprehension and organization. Though we did not drastically change anything in previous phases, we have decided to limit the visual elements and front-end features of our prototype. We had originally intended for this project to be a user-friendly web app, but due to time constraints, some these features will be added at a later date after the semester ends. The above functions are all that is guaranteed.

In terms of our stack, we had to change our database to MySQL, as MongoDB was not SQL-based. Our final stack choice is M(MySQL), E(Express.js), R(React.js), and N(Node.js).

Proposal

Scenario:

When it comes to college radio, music directors use a system to ensure that that the collection of recent releases played on air get a somewhat equal amount of air time. Albums are separated into the following rankings: Hots, Mediums, Lights and A-Picks. Disc jockeys pick a certain amount of songs from each category, and the albums those songs are on are sent to the back of the queue. This system is called “regular rotation,” and, at U92, this is all done using paper cards and sheets. A digitized version of this would be helpful for many college radio stations across the country.

Album:

An Album is a singular collection of music released by an artist. Every album contains multiple songs. Because our radio station does not accept “Singles,” all albums must contain at least 4 songs. An album is the primary entity in the database and is what rotation focuses on. Every album is placed into one of the 4 categories outlined in the scenario.

Essential Attributes: Album ID, Album Title, Release Date, Category, Date Added (to rotation), X tracks (music director picks), Description, Rotation Status?, RIYL (helper table)

Relationships: Song - 1:N

Artist:

An Artist is the creator of an album, EP, or other, self-contained collection of music. Artist is the second highest level entity in the database. Artists can have multiple albums, and there can be multiple artists on a single album.

Essential Attributes: Artist ID, Artist Name, City, State, Country, Debut Date, Genre (IDv3 metadata)

Relationships: Album - M:N, 1:N

Song:

A song is one of the many pieces that make up an album. Once chosen by a disc jockey, the date the song is played is marked down on a playlist, and the album the song is on is sent to the back of the queue. Songs must be linked to an album.

Essential Attributes: Song ID, Song Name, Artist Name, Track Number, Length, Request Flag, X/D Pick Flag, Album_ID, Explicit?

Relationships: Artist - N:M, Album - 1:N

Playlist:

A Playlist is a list, recorded by a DJ during their shift, of songs played during each hour. The playlist is structured by the categories outlined in the scenario. For the purposes of this system, D-Picks and X-Picks will not be stored in any location other than the Playlist as plain text (and these songs will also have no attributes, as they are not entities).

Essential Attributes: Playlist ID, DJ Name, Date, Time, Song Selections (helper table)

Relationships: Song - 1:N

Prototype:

Reg Rotator will be a web application designed to allow radio stations a means of keeping their music rotation records digitally.

To achieve this, we must provide a UI with all functions that are present on our current paper medium. We must also provide a database that allows DJs to view what albums are currently in rotation, as well as what has been in rotation in the past. This database should be easy to filter and search through.

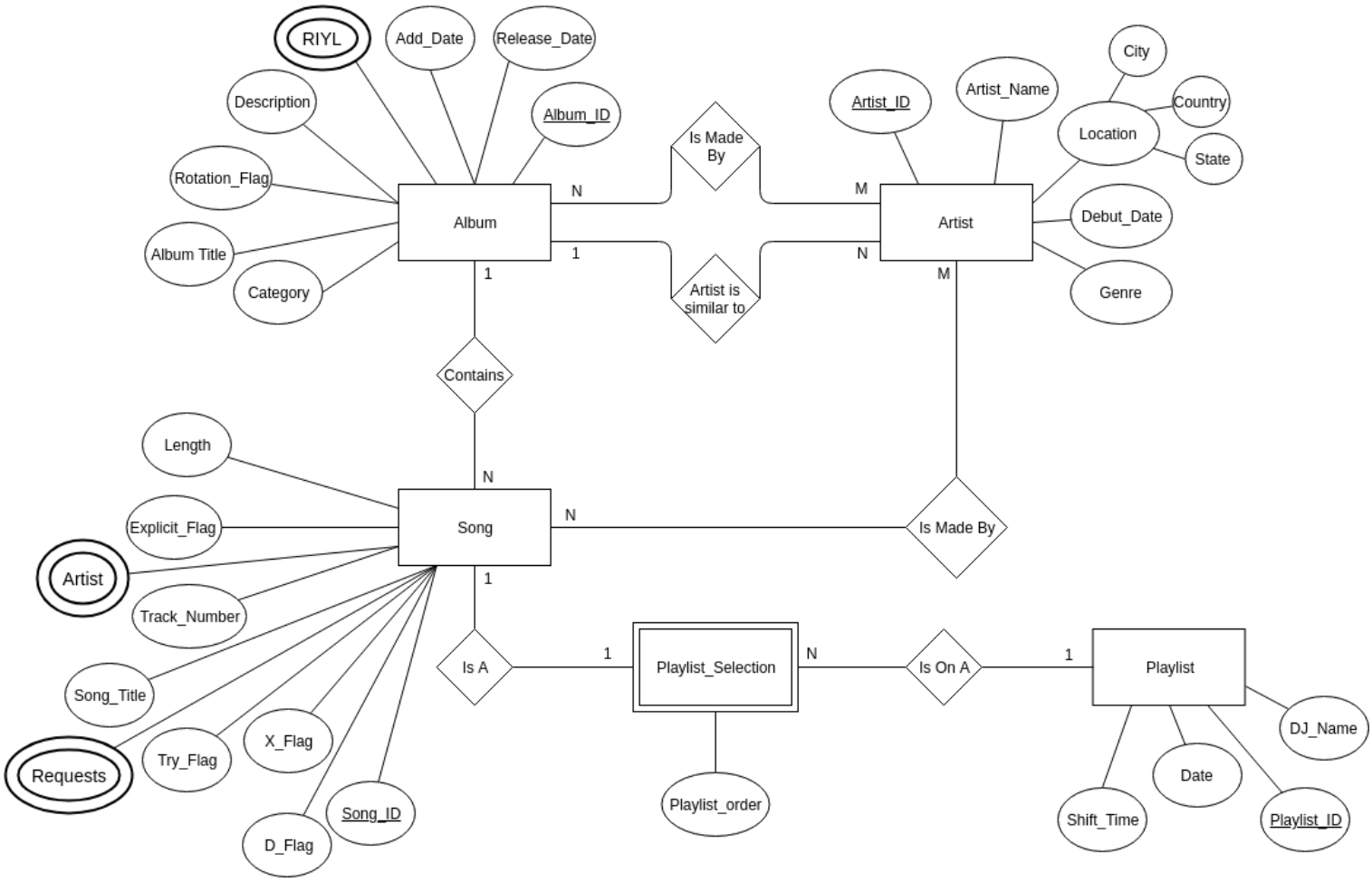
The target users of this project will be college radio stations, and specifically DJs & music directors at those stations. These users will not be represented as entities.

Roles:

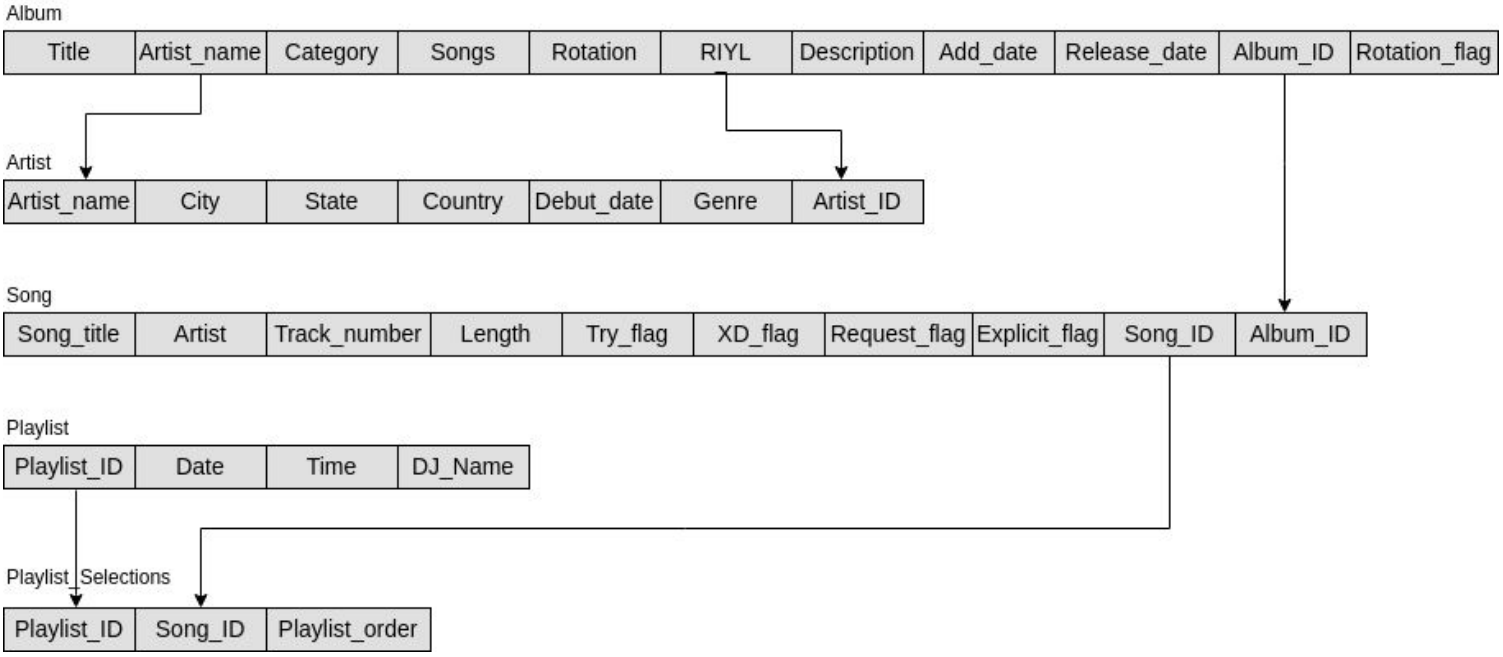
Victor Perez - Front-End engineer, Jon Montesano - Back-End engineer

Design

ER Diagram:



ER Schema


Album
Title
Artist_name
Category
Songs
Rotation
RIYL
Description
Add_date
Release_date
Album_ID
Rotation_flag
Artist
Artist_name
City
State
Country
Debut_date
Genre
Artist_ID
Song
Song_title
Artist
Track_number
Length
Try_flag
XD_flag
Request_flag
Explicit_flag
Song_ID
Album_ID
Playlist
Playlist_ID
Date
Time
DJ_Name
Playlist_Selections
Playlist_ID
Song_ID
Playlist_order

ALBUM

Album_id	Album_title	Artist_name	Category	Release_date	Add_date	Rotation_flag	Description
a1	Some New Form of Life	Devi McCallion, Katie Dey	H	2018-12-15	2019-02-10	Y	collab album...
a2	The C.I.A.	The C.I.A.	M	2018-11-30	2019-01-20	Y	debut album...
a3	Far Out Dust	Talos	L	2019-02-08	2019-02-10	Y	record from..
a4	Mutant	Arca	A	2015-11-20	2019-2-10	Y	second studio...
a5	The River Strumming	Cotton Jones	A	2008-09-23	2018-12-02	N	Cotton Jones...

SONG

Song_id	Song_title	Artist_name	Track_num	Length	Request_flag	Try_flag	X_D_flag	Album_id	Explicit_flag
a5-s1	It Comes To Me Now	Cotton Jones	1	5:39	N	Y		a5	N
a5-s2	To Death With You	Cotton Jones	2	3:34	N	N		a5	N
x-s1	Computer Love	Kraftwerk	5	7:19	N	N	X		N
d-s1	Mag11 P82	Venetian Snares	1	4:06	N	N	D		N
a1-s10	Pigs Don't Feel Pain	Devi McCallion, Katie Dey	10	4:24	N	Y		a1	N
r-s1	Pro Radii	Autechre	3	8:42	Y				N

ARTIST

Artist_id	Artist_name	City	State	Country	Debut_year	Genre
m1	Cotton Jones	Cumberland	Maryland	USA	2008	Psychedelic
m2	Arca	London		UK	2011	IDM
m3	Devi McCallion	Toronto	Ontario	CA	2011	Industrial
m4	Katie Dey	Melbourne		AU	2011	Experimental

PLAYLIST

Playlist_id	DJ_name	Time	Date
p1	Hank R.	8:00 AM	2019-01-25
p2	Hank R.	9:00 AM	2019-01-25
p3	Tyler C.	12:00 AM	2019-02-12
p4	Tyler C.	1:00 AM	2019-02-12
p5	Tyler C.	2:00 AM	2019-02-12

ALBUM_OWNERSHIP

Album_id	Artist_id
a1	m3
a1	m4
a2	m5
a3	m6
a4	m2
a5	m1

SONG_OWNERSHIP

Song_id	Artist_id
a5-s1	a5
a5-s2	a5
a1-s10	m3
a1-s10	m4

RIYL

Album_id	Artist_id
a1	m3
a2	m5
a3	m6
a4	m2
a5	m1

PLAYLIST_SELECTION

Playlist_id	Song_id	Playlist_order
p1	a5-s2	1
p1	d-s1	2
p1	a1-s10	3

Creation Queries

```
CREATE TABLE ALBUM
        ( Album_id              VARCHAR(10)             NOT NULL,
          Album_title           VARCHAR(30)             NOT NULL,
          Category              CHAR                    NOT NULL,
          Release_date          DATE,
          Add_date              DATE,
          Rotation_flag         BOOLEAN                         NOT NULL,
          Description           TEXT,

        PRIMARY KEY (Album_id) );

CREATE TABLE ALBUM_OWNERSHIP
        ( Album_id              VARCHAR(10)             NOT NULL,
          Artist_id             VARCHAR(10)             NOT NULL,

        PRIMARY KEY (Album_id, Artist_id),
        FOREIGN KEY (Album_id) REFERENCES ALBUM(Album_id),
        FOREIGN KEY (Artist_id) REFERENCES ARTIST(Artist_id) );

CREATE TABLE SONG
        ( Song_id               VARCHAR(10)             NOT NULL,
          Song_title            VARCHAR(30)             NOT NULL,
          Artist                VARCHAR(30)             NOT NULL,
          Album_id              VARCHAR(10),
          Track_num             INT(3)                  NOT NULL,
          Length                FLOAT                   NOT NULL,
          Request_flag          BOOLEAN                 NOT NULL,
          Try_flag              BOOLEAN                 NOT NULL,
          XD_flag               CHAR(1)                 NOT NULL,
          Explicit_flag         BOOLEAN                 NOT NULL,

        PRIMARY KEY (Song_id),
        FOREIGN KEY (Album_id) REFERENCES ALBUM(Album_id));
```

```
CREATE TABLE SONG_OWNERSHIP
        ( Song_id                   VARCHAR(10)                 NOT NULL,
          Artist_id                 VARCHAR(10)                 NOT NULL,

        PRIMARY KEY (Song_id, Artist_id),
        FOREIGN KEY (Song_id) REFERENCES SONG(Song_id),
        FOREIGN KEY (Artist_id) REFERENCES ARTIST(Artist_id) );

CREATE TABLE ARTIST
        ( Artist_id                 VARCHAR(10)                 NOT NULL,
          Artist_name               VARCHAR(30)                 NOT NULL,
          City                      VARCHAR(30),
          State                     VARCHAR(30),
          Country                   VARCHAR(30),
          Debut_year                DATE,
          Genre                     TEXT,

        PRIMARY KEY (Artist_id) );

CREATE TABLE PLAYLIST
        ( Playlist_id               VARCHAR(10)                 NOT NULL,
          DJ_name                   VARCHAR(30)                 NOT NULL,
          Time                      TIME                        NOT NULL,
          Date                      DATE                        NOT NULL,

        PRIMARY KEY (Playlist_id) );

CREATE TABLE PLAYLIST_SELECTION
        ( Playlist_id               VARCHAR(10)                 NOT NULL,
          Song_id                   VARCHAR(10)                 NOT NULL,
          Playlist_order            INT(2)                      NOT NULL,

        FOREIGN KEY (Playlist_id) REFERENCES PLAYLIST(Playlist_id),
        FOREIGN KEY (Song_id) REFERENCES SONG(Song_id) );

CREATE TABLE RIYL
        ( Album_id                  VARCHAR(10)                 NOT NULL,
          Artist_id                 VARCHAR(10)                 NOT NULL,

        FOREIGN KEY (Album_id) REFERENCES ALBUM(Album_id),
        FOREIGN KEY (Artist_id) REFERENCES ALBUM(Album_id) );
```

Retrieval Queries

Q1	Select the latest album_id by the artist ‘Aphex Twin’ as well as any albums by the RIYL artists for that album.
Purpose	Create a list of similar albums to choose tracks from on a specialty show.
Relational Algebra	$AFX\_NEW \leftarrow \pi_{album\_id}((\sigma_{release\_date = \textbf{MAXIMUM}\ release\_date}(\sigma_{artist = \text{“Aphex Twin”}}(ALBUM)))$ $RIYL\_ARTISTS \leftarrow \pi_{artist\_id}(\sigma_{album\_id = \textbf{AFX\_NEW}}(RIYL))$ $RIYL\_ALBUMS \leftarrow \pi_{album\_id}(\sigma_{artist\_id = \textbf{RIYL\_ARTISTS}}(ALBUM\_OWNERSHIP))$ $AFX\_NEW \cup RIYL\_ALBUMS$
DML	**SELECT** ALBUM.Album_id **FROM** ALBUM **WHERE** ALBUM.release_date = (**SELECT MAX**(Release_date) **FROM** ALBUM **WHERE** ALBUM.Artist = "Aphex Twin") **AND** ALBUM.Artist = "Aphex Twin" **UNION** **SELECT** ALBUM_OWNERSHIP.Album_id **FROM** ALBUM_OWNERSHIP, ALBUM, RIYL **WHERE** RIYL.Album_id = (**SELECT** Album_id **FROM** ALBUM **WHERE** ALBUM.Release_date = (**SELECT MAX**(release_date) **FROM** ALBUM **WHERE** ALBUM.Artist = "Aphex Twin") **AND** ALBUM.Artist = "Aphex Twin") **AND** ALBUM_OWNERSHIP.Artist_id = RIYL.Artist_id;

Q2	Select the name and city of all catalogued punk artists from Pittsburgh and West Virginia that have released an album in the past 2 years.
Purpose	Create a list of local punk bands to share with radio listeners.
Relational Algebra	$\pi_{\text{artist\_name, city}}((\sigma_{\text{(city = 'Pittsburgh' OR state = 'West Virginia') AND genre = 'Punk'}}(\text{ARTIST})) \cap (\sigma_{\text{release\_date <= '2017-3-17'}}(\text{ARTIST, ALBUM}))$
DML	**SELECT DISTINCT** ARTIST.Artist_name, ARTIST.City **FROM** ARTIST, ALBUM **WHERE** (ARTIST.City = "Pittsburgh" OR ARTIST.State = "WV") **AND** ARTIST.Genre = "Punk" **AND** ARTIST.Artist_name **IN** (**SELECT** ARTIST.Artist_name **FROM** ARTIST, ALBUM **WHERE** ALBUM.Release_date >= '2017-03-17' **AND** ALBUM.Artist = ARTIST.Artist_name);

Q3	Select all X, D, and request picks played on a specific shift (playlist date and time)
Purpose	Verifying if the tracks selected and played by DJs fit the station's format rules
Relational Algebra	$\pi_{song\_title, XD\_flag}(\sigma_{song\_title}(\sigma_{song.XD\_flag = 'X' \ OR \ song.XD\_flag = 'D' \ OR \ song.request\_flag = 'Y'}(SONG))$ $\cap \sigma_{song\_id}(\sigma_{time = '8:00 \ AM' \ AND \ date = '2019-01-25'}(PLAYLIST)) \ AND$ $\sigma_{playlist\_id}(PLAYLIST\_SELECTION) \ AND \ \sigma_{playlist\_id}(PLAYLIST))$
DML	**SELECT** Song_title, XD_flag **FROM** SONG **WHERE** SONG.XD_flag = "Y" **AND SONG**.Song_id IN (**SELECT** Song_id **FROM** PLAYLIST_SELECTION, PLAYLIST **WHERE** PLAYLIST_SELECTION.Playlist_id = PLAYLIST.Playlist_id **AND** PLAYLIST.Time = "12:00" **AND** PLAYLIST.Date = "2019-03-11");

Q4	Select the albums currently in rotation 4 months older than the current date and display the album name, category, and amount of songs on that album.
Purpose	Keeping track of albums that have been in rotation too long that might need to be taken out of rotation
Relational Algebra	$\pi_{album\_title,\ category,\ (\sigma_{COUNT(*)}(\sigma_{album\_id}(SONG) = \sigma_{album\_id}(ALBUM))}$ $(\sigma_{add\_date <= '2018-01-19' \text{ and } rotation\_flag = 'Y'}(ALBUM))$
DML	**SELECT** Album_title, Category, (**SELECT** **COUNT(*)** **FROM** SONG, ALBUM **WHERE** SONG.Album_id = ALBUM.Album_id) **AS** "Song Count" **FROM** ALBUM **WHERE** ALBUM.add_date <= '2018-01-19' **AND** ALBUM.Rotation_flag = 1;

Q5	Select the album name and number of plays each song has, from 3/10/19 through 3/16/19
Purpose	Generate weekly NACC album rank charts.
Relational Algebra	$\pi_{album\_title, (\sigma_{COUNT(*)}(\sigma_{album\_id}(SONG) = \sigma_{album\_id}(ALBUM)\ AND\ \sigma_{song\_id}(PLAYLIST\_SELECTION) = \sigma_{song\_id}(SONG)\ AND\ \sigma_{playlist\_id}(PLAYLIST\_SELECTION) = \sigma_{playlist\_id}(PLAYLIST)\ AND\ \sigma_{date <= '2019-0-16'\ AND\ date >= '2019-03-10'}(PLAYLIST)))}(\sigma_{rotation\_flag = 'Y'}(ALBUM))$
DML	**SELECT** Album_title, (**SELECT** **COUNT(*)** **FROM** SONG, ALBUM, PLAYLIST, PLAYLIST_SELECTION **WHERE** SONG.Album_id = ALBUM.Album_id **AND** PLAYLIST_SELECTION.Song_id = SONG.Song_id **AND** PLAYLIST_SELECTION.Playlist_id = PLAYLIST.Playlist_id **AND** PLAYLIST.Date <= "2019-03-16" **AND** PLAYLIST.Date >= "2019-03-10") **AS** "Plays" **FROM** ALBUM **WHERE** ALBUM.Rotation_flag = 1;

Q6	Select every album in the database that was released at least 10 years ago by its title and artist.
Purpose	Determine which albums are now eligible as X picks.
Relational Algebra	$\pi_{album\_title, artist\_name}(\sigma_{release\_date >= '2009\text{-}3\text{-}17'}(ALBUM))$
DML	**SELECT** ALBUM.Album_title, ALBUM.Artist **FROM** ALBUM **WHERE** ALBUM.Release_date <= CURRENT_TIMESTAMP