CS-GY 6083-A, Principles of Database Systems, Fall 2021
Course Project: A database application with a Web front-end

Manish Singh Saini (mss9238), Khalid Nizamuddin Ansari (kna2612)

[1] Overview

This web application has implications similar to prominent music applications of the time, such as Spotify, Amazon Music, and Apple Music in terms of its database. The application will hold data such as songs, artists, albums, bands, and users to answer a spectrum of questions that users ask, such as top songs, artists, and albums in disparate genres, when starting to use a certain music app. Each result then can be further sorted by genre, date, etc. according to the requirement of the user.

[2] ER Diagram constructs

Entity Sets:

Users(name, dob, since, country)
Songs(name, release_date, genre, language)
Artists (name, dob, other_skills, main_skill)
Albums(name, release_date)
Bands (name, since)
Awards(award_name, award_year)
Libraries(name)
Plays(play_ts)

Relationship Sets:

listen_to
customize
form
artists_create_songs (creation_date)
artists_create_albums (creation_date)
bands_create_songs (creation_date)
bands_create_albums (creation_date)
win
lists
Comprise (since)

[3] Business Rules and corresponding constraints in ER diagram:

- Business rule: *Artists* can “form” *bands*. Each *band* will have some *artist*. Corresponding constraint is a participation constraint of *Bands* entity in “form” relationship.
- *Artists/Bands* can “create” *Songs*. Each song is created by some artist/band. Corresponding constraints are participation constraints of *Artists/Bands* as well as *Songs* entities in “artists_create_songs/bands_create_songs” relationships.
- *Artists/Bands* can “create” *Albums*. Each album is created by some artist/band. Corresponding constraints are participation constraints of *Artists/Bands* as well as *Songs* entities in “artists_create_albums/bands_create_albums” relationships.
- *Albums* can “list” *songs*. *A*lbum will have some songs. Corresponding constraint is a participation constraint of *Albums* entity in “lists” relationship.
- *Users* can “customize” their *libraries* of *songs*. *Libraries* is a weak entity and *Users* is its defining entity set.
- *Artists* can “win” *awards*. *Award* is a weak entity and *Artists* is the defining entity set.
- *Users* can “listen_to” *Songs.* “listen_to” relationship is captured in *Plays* entity which records each play of any song by any user using “play_ts” which is a timestamp.

[4] Data Sources

- Wikipedia lists of artists and songs
- Reddit and Github lists
- Random list of usernames
- Automated (random) data for entities/relationships like “Plays” and “Libraries”.

[5] Questions answered by the app:

Few examples:

- Top played songs, i.e songs played the most number of times by a user (or by all the users combined).
- Most liked songs, by number of times the song has been played.
- Most active users, by number of times they’ve played the songs. Moreover, their recent and most listened songs.
- Most awarded artists, in terms of number of awards.
- Artists with most song releases filtered by year range and whether they have won any awards or not.
- Most featured albums by genre since any given year.
- Bands with most album releases and the ones with most songs played in any given year and genre.

[6] Code Structure & Data Loading Procedure

Code Structure:

Our code structure follows the guidelines given in the project document as below:

code/
- project.py (main Python code)
- schema.sql (SQL file with DROP + CREATE statements)

data/
- load.sql (SQL file with INSERT statements)

Data Loading Procedure:

Steps:

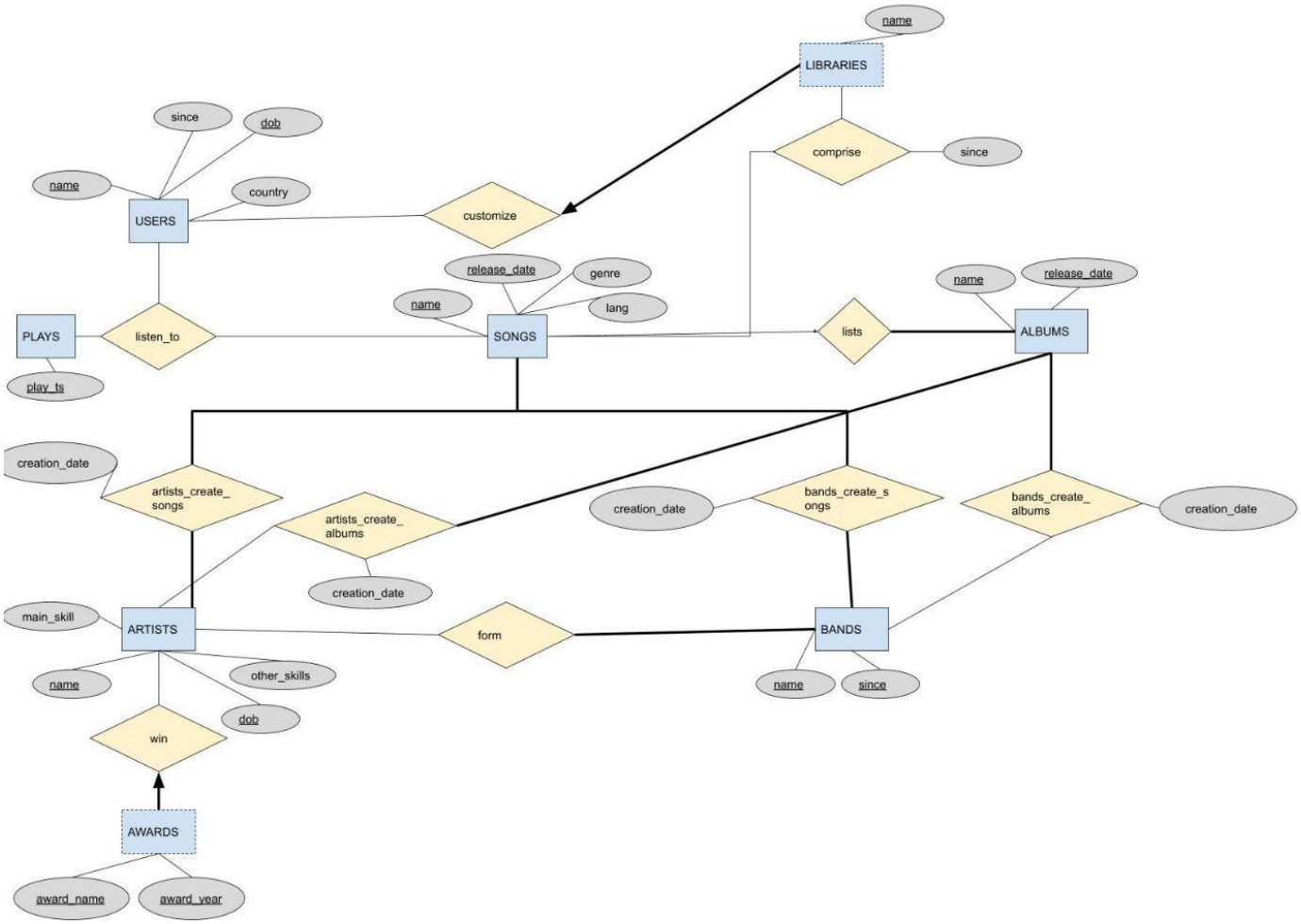
[1] Create Schema:

psql -d NETID_db -a -f code/schema.sql

[2] Loading Data:

psql -d NETID_db -a -f data/load.sql

[7] ER Diagram



[8] Create Statements

drop table if exists Users, Songs, Artists, Bands, Albums cascade;
drop table if exists Song_Plays, Artists_Form_Bands, Artists_Win_Awards, Albums_List_Songs, Artists_Create_Songs, Artists_Create_Albums, Bands_Create_Songs, Bands_Create_Albums, Users_Libraries cascade;

```
create table Users(
        name varchar(128),
        dob integer,
        since integer,
        country varchar(128),
        primary key(name, dob)
);
```

```
create table Songs(
        name varchar(128),
        release_date smallint,
        genre varchar(128),
        lang varchar(128),
        primary key(name, release_date)
);

create table Artists(
        name varchar(128),
        dob integer,
        main_skill varchar(128),
        other_skills varchar(128),
        primary key(name, dob)
);

create table Bands(
        name varchar(128),
        since smallint,
        primary key(name, since)
);

create table Albums(
        name varchar(128),
        release_date smallint,
        primary key(name, release_date)
);

create table Song_Plays(
        uname varchar(128),
        udob integer,
        sname varchar(128),
        srelease_date smallint,
        play_ts bigint,
        primary key(uname, udob, sname, srelease_date, play_ts),
        foreign key (uname, udob) references Users(name, dob),
        foreign key (sname, srelease_date) references Songs(name, release_date)
);

create table Artists_Form_Bands(
        bname varchar(128),
        bsince smallint,
        aname varchar(128),
```

```
        adob integer,
        primary key(bname, bsince, aname, adob),
        foreign key (bname, bsince) references Bands(name, since),
        foreign key (aname, adob) references Artists(name, dob)
);

create table Artists_Win_Awards(
        aname varchar(128),
        adob integer,
        award_name varchar(128),
        award_year smallint,
        primary key(aname, adob, award_name, award_year),
        foreign key (aname, adob) references Artists(name, dob) on delete cascade
);

create table Albums_List_Songs(
        aname varchar(128),
        arelease_date smallint,
        sname varchar(128),
        srelease_date smallint,
        primary key(aname, arelease_date, sname, srelease_date),
        unique (sname, srelease_date),
        foreign key(aname, arelease_date) references Albums(name, release_date),
        foreign key (sname, srelease_date) references Songs(name, release_date)
);

create table Artists_Create_Songs(
        aname varchar(128),
        adob integer,
        sname varchar(128),
        srelease_date smallint,
        creation_date smallint,
        primary key(aname, adob, sname, srelease_date),
        foreign key (aname, adob) references Artists(name, dob),
        foreign key (sname, srelease_date) references Songs(name, release_date)
);

create table Artists_Create_Albums(
        artist_name varchar(128),
        artist_dob integer,
        album_name varchar(128),
        album_release_date smallint,
        creation_date smallint,
        primary key(artist_name, artist_dob, album_name, album_release_date),
```

```
        foreign key (artist_name, artist_dob) references Artists(name, dob),
        foreign key (album_name, album_release_date) references Albums(name, release_date)
);

create table Bands_Create_Songs(
        bname varchar(128),
        bsince smallint,
        sname varchar(128),
        srelease_date smallint,
        creation_date smallint,
        primary key(bname, bsince, sname, srelease_date),
        foreign key (bname, bsince) references Bands(name, since),
        foreign key (sname, srelease_date) references Songs(name, release_date)
);

create table Bands_Create_Albums(
        bname varchar(128),
        bsince smallint,
        aname varchar(128),
        arelease_date smallint,
        creation_date smallint,
        primary key(bname, bsince, aname, arelease_date),
        foreign key (bname, bsince) references Bands(name, since),
        foreign key (aname, arelease_date) references Albums(name, release_date)
);

create table Users_Libraries(
        uname varchar(128),
        udob integer,
        lib_name varchar(128),
        sname varchar(128),
        srelease_date smallint,
        since integer,
        primary key(uname, udob, lib_name, sname, srelease_date),
        foreign key (uname, udob) references Users(name, dob) on delete cascade,
        foreign key (sname, srelease_date) references Songs(name, release_date)
);
```