MusicDB Project

CS-GY 6083

Yanjie Xu (yx2304)
Hanqing Zhang (hz2758)
Deployed port number: 8690

Description:

In this project, we created a music database to get some interesting information behind the data. The database contains information about songs, albums, artists, releasing companies and more . Users can type in or select keywords to explore our database from the website.

Entity sets, relationship sets, and business rules:

Entities sets:

Artist (aid: string, name: string);

_User (User_id: integer, name:string, email :string);

Company (company_id: integer, name :string);

Album(album_id String, name :string, release_date :date);

Song(sid :string, album_id :string, name :string, duration_ms :integer, release_:date : date);

- album_id is a foreign key from album

Playlist(pid:integer, pname :string, playtimes:integer, update_time: date);

Rating (cid:integer, sid:string, rating : integer, _time: date);

Relationship sets:

PlaylistStrores(pid:integer, sid:string)

- pid is a foreign key from Playlist
- sid is a foreign key from Song

PlaylistCreated(pid:integer, uid:string)

- pid is a foreign key from Playlist
- uid is a foreign key from _user

Album_Artist_Map(album_id:string, aid,string)

- album_id is a foreign key from album
- aid is a foreign key from Artist

Song_Artist_Map(sid:string, aid,string)

- sid is a foreign key from Song
- aid is a foreign key from artist

ArtistOwned(aid: string, comany_id:integer)

- aid is a foreign key from Artist
- Company_id is a foreign key from Company

ArtisitSubscribed (aid: string, uid:integer)

- aid is a foreign key from Artist
- Company_id is a foreign key from Company

Ratingby(cid: integer, uid:integer)

- cid is a foreign key from Rating
- uid is a foreign key from _user

Business rules:

Song is performed by artists.

- a song must has a name
- can be produced by one or more artists
- can have a rating given by users
- must have duration and release date

Album is composed of songs and belongs to artists.

- an album can have one or more artists,
- has at least one song.

Artists perform songs.

- artists must have one or more songs.
- artists must have one or more albums.
- artists can belong to a company or not, but only one company max.

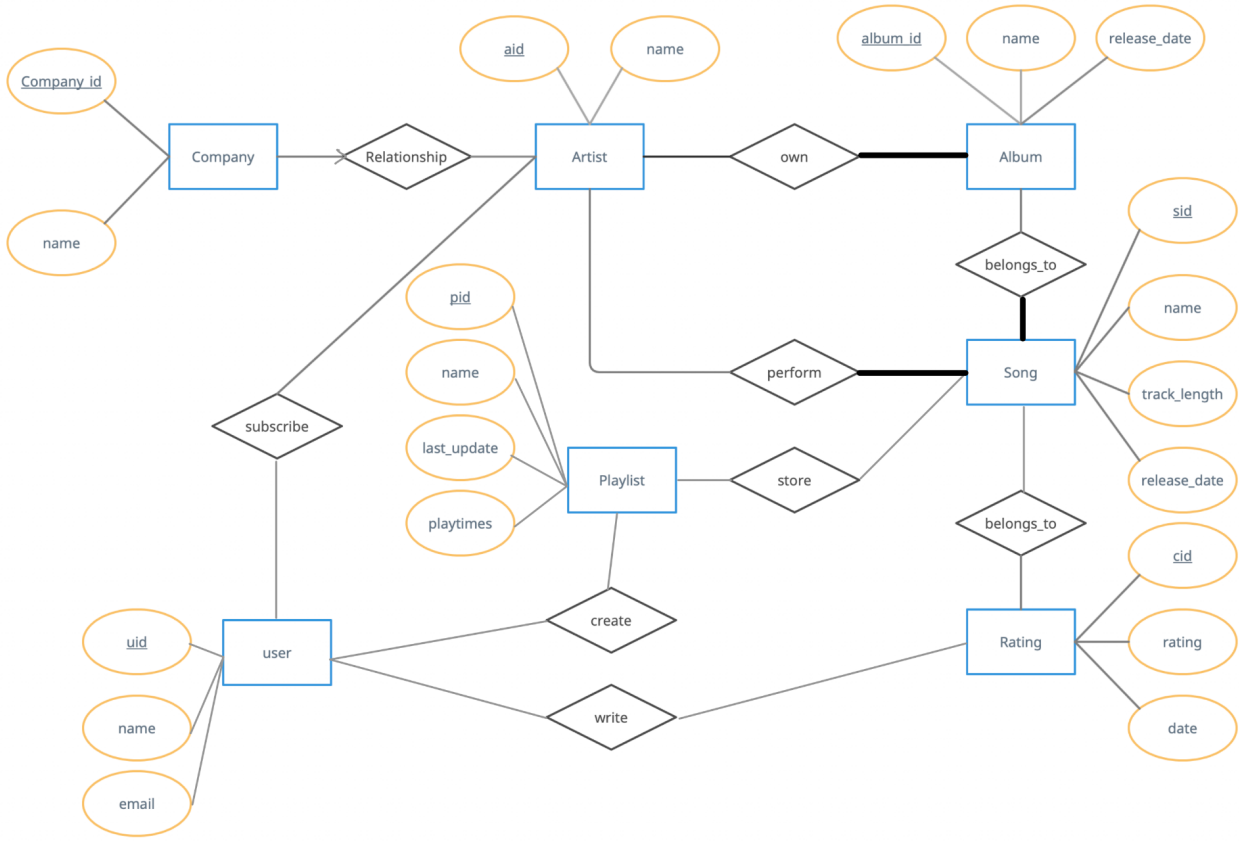
Company owns artists and composers.

Users can own and create playlists, subscribe to artists or composers and give ratings

Playlist: users can store songs in the playlists or not.

Ratings: are given by users, users can give on songs.

ER Diagram



Data loading procedure

1. Change directory to project directory and set the your database name
2. Create table
   `psql -d {DatabaseName} -a -f code/db_create.sql`
3. `cd data`
4. Insert values using bash. (Must run in projects' /data folder)
   `bash load_csv.sh`

Data Source

There are plenty of free music databases from the internet, we can just download them and use them as our needed datasets. Here are a few example urls.

Spotify API
https://developer.spotify.com/documentation/web-api/

Kaggle Spotify Music Database
https://www.kaggle.com/rodolfofigueroa/spotify-12m-songs/code

And we used https://www.onlinedatagenerator.com/ to generate some user related information and ratings.

User interaction

Our application will have a user interface implemented using Streamlit and Psycopg2. The users can search for songs, albums, artists, companies, users, and playlist if they type in or select the required keywords, and they will get their results with their included information such as songs' release date, name and the corresponding artist, can also give out some interesting results from built in queries such as the songs with specific rating etc.

schema.sql

```
create table Artist(
  aid varchar(128) primary key ,
  name varchar(128)
);

create table Album(
 album_id varchar(128) primary key,
 name varchar(128),
 release_date date
);


create table Album_Artist_Map(
  album_id varchar(128),
  aid varchar(128),
   primary key (aid, album_id),
   foreign key (aid) references Artist(aid),
   foreign key (album_id) references Album(album_id)
);

create table Song(
 sid   varchar(128)    primary key,
 album_id varchar(128) not null,
 name varchar(128) not null,
 duration_ms integer not null,
 release_date date not null,
 foreign key (album_id) references Album(album_id)
);

create table Song_Artist_Map(
   sid varchar(128),
   aid varchar(128),
   primary key (sid, aid),
   foreign key (aid) references Artist(aid),
   foreign key (sid) references Song(sid)
);

create table _User
(
  user_id     serial primary key,
  name    varchar(128) not null,
  email     varchar(128) not null
);
```

```
create table Company(
  company_id serial primary key ,
  name varchar(128)
);


create table Rating(
  cid serial primary key,
  sid varchar(128),
  rating integer not null,
  _time date,
   foreign key (sid) references Song(sid)
);


create table Playlist(
  pid serial primary key ,
  pname varchar(128),
  playtimes integer default 0,
  update_date timestamp not null
);

create table PlaylistStores(
  pid integer,
  sid varchar(128),
  primary key (pid,sid),
  foreign key (pid) references Playlist(pid),
  foreign key (sid) references Song(sid)
);


create table ArtistOwned(
  aid varchar(128) primary key,
  company_id integer,
  foreign key (aid) references Artist(aid),
  foreign key (company_id) references Company(company_id)
);


create table PlaylistCreated(
  pid integer primary key ,
  uid integer not null ,
  foreign key (pid) references Playlist(pid),
  foreign key (uid) references _User(user_id)
);

create table ArtistSubscribed(
  aid varchar(128) primary key ,
```

```
  uid integer,
  foreign key (aid) references Artist(aid),
  foreign key (uid) references _User(user_id)
);


create table ratingby(
  cid integer,
  uid integer,
  primary key (cid, uid),
  foreign key (cid) references Rating(cid),
  foreign key (uid) references _User(user_id) on delete cascade
);
```