Music Streaming Service Database Project

Yanjie Xu (yx2304)
Hanqing Zhang (hz2758)

Description:

Our project is to create a music database for a music streaming service. The database contains information about songs, albums, artists, and releasing companies. Users can have their own account, search, listen, comment, add their favorite songs to playlists and subscribe to their favorite artists.

Entity sets, relationship sets, and business rules:

Entities sets:

Artists(aid, dob, gender, name, country, genre, last_updated)
Composer(cpid, dob, gender, name, country, genre, last_updated)
_User(**uid,** User_name, Email, Fav_genre)
Company(company_id, name)
Song(sid, track_length, release_date, name, song_size, qong_quality)
Album(album_id, name, release year, language, genre)
Playlist(pid, genre, playtimes, visibility, update_timestamp)
Comment(cid, content, _time, rating)

Relationship set and business rule:

Song is written by a composer and performed by artists.
- a song must belong to an album
- can be produced by one or more artists
- can have one or more comments.

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.

Composers write songs.
- composers must have one or more songs.
- artists can be composers.

Company owns artists and composers.
- Company must have at least one artist or composer.

Users can own and create playlists, subscribe to artists or composers and leave comments.
- Users have at least one playlist called favorite
- Each user has exact one playlist_collection

Playlists are created by the users, users can store songs in the playlists or not.
- Playlist is a weak entity set.

Comments are written by users, users can comment on songs and albums.
- Comment is a weak entity set.

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.
https://musicbrainz.org/doc/MusicBrainz_Database/Download
http://millionsongdataset.com/
https://github.com/mdeff/fma

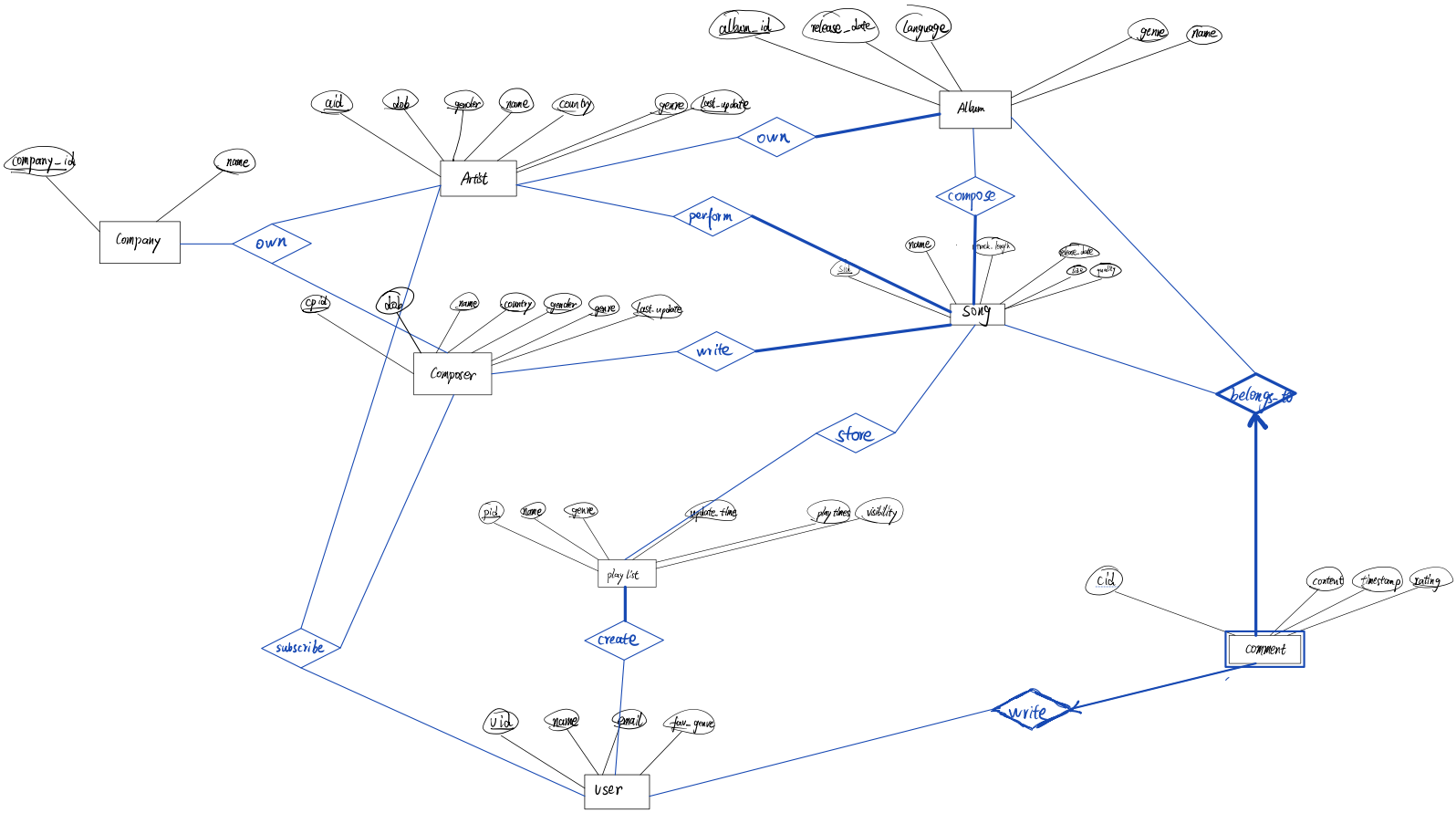
User interaction

Our application will have a user interface. The users can search for songs, albums, artists, composers, companies, users, and playlist if they type in the required keywords, and they will get their results with their included information such as songs' release date, the gender of an artist or composer, the genre of an album etc. The users can create their own playlists as well (will have one called my fav by default), along with the options to adjust their playlists' visibilities. The users can also make comments about the songs and the albums.

```
create table Artist(
aid serial primary key ,
dob date not null,
gender varchar(5) not null,
name varchar(128),
country varchar(128),
genre varchar(32),
last_update date
);

create table Composer(
cpid serial primary key ,
dob date not null,
gender varchar(5) not null,
name varchar(128),
country varchar(128),
genre varchar(32),
last_update date
);

create table _User(
user_id serial primary key,
name varchar(128) not null,
email varchar(128) not null,
fav_genre varchar(32)
);

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

create table Album(
album_id serial primary key,
name varchar(128),
aid integer not null,
release_date date,
language varchar(32),
genre varchar(32),
foreign key (aid) references Artist(aid)
);

create table Song(
sid serial primary key,
aid integer not null,
cpid integer,
album_id integer,
name varchar(128) not null,
track_length integer not null,
release_date date not null,
song_size integer not null,
song_quality integer not null,
foreign key (aid) references Artist(aid),
foreign key (cpid) references Composer(cpid),
foreign key (album_id) references Album(album_id)
);

create table Playlist(
pid serial primary key ,
name varchar(128),
genre varchar(32),
playtimes integer default 0,
visibility bool not null,
update_date timestamp not null
);

create table Comment(
cid serial primary key ,
content varchar(1024) not null,
rating integer not null,
_time timestamp
);

create table ArtistOwned(
aid integer primary key,
company_id integer,
foreign key (aid) references Artist(aid),
```

```
foreign key (company_id) references Company(company_id)
);

create table ComposerOwned(
cpid integer primary key,
company_id integer,
foreign key (cpid) references Composer(cpid),
foreign key (company_id) references Company(company_id)
);

create table AlbumOwned(
album_id integer primary key ,
aid integer not null ,
foreign key (album_id) references Album(album_id),
foreign key (aid) references Artist(aid)
);

create table SongPerformed(
sid integer primary key ,
aid integer not null ,
foreign key (sid) references Song(sid),
foreign key (aid) references Artist(aid)
);

create table SongComposedOf(
sid integer primary key ,
album_id integer not null ,
foreign key (sid) references Song(sid),
foreign key (album_id) references Album(album_id)
);

create table SongWroteby(
sid integer primary key ,
cpid integer not null ,
foreign key (sid) references Song(sid),
foreign key (cpid) references Composer(cpid)
);

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

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

create table ComposerSubscribed(
cpid integer primary key ,
uid integer,
foreign key (cpid) references Composer(cpid),
foreign key (uid) references _User(user_id)
);

create table CommentWroteby(
cid integer,
uid integer,
primary key (cid, uid),
foreign key (cid) references Comment(cid),
foreign key (uid) references _User(user_id) on delete cascade
);

create table CommentToSongs(
cid integer,
sid integer,
primary key (cid, sid),
foreign key (cid) references Comment(cid),
foreign key (sid) references Song(sid) on delete cascade
);

create table CommentToAlbum(
cid integer,
album_id integer,
primary key (cid, album_id),
```

```
foreign key (cid) references Comment(cid),
foreign key (album_id) references Album(album_id) on delete cascade
);
```


album_id
release_date
language
genre
name
Album
aid
dob
gender
name
country
genre
last_update
Artist
own
company_id
name
Company
own
perform
compose
name
track_lists
genre_info
size
quality
Song
cp_id
name
country
gender
genre
Last_update
Composer
write
belongs_to
store
pid
name
genre
update_time
play times
visibility
play list
create
subscribe
cid
content
timestamp
rating
comment
write
Uid
name
email
fav_genre
User