Wojciech Musiatowicz
Matteo Gilardi
Jakub Bielawski

Databases Report #1

1.ER Diagrams

Before creating the diagrams we made following assumptions:

- song can be in multiple albums
- user can share only files
- concert can exist without artists (some festivals announce artists after some time)
- group can't exist without at least one user
- playlists are always public and user can add someone else's playlist

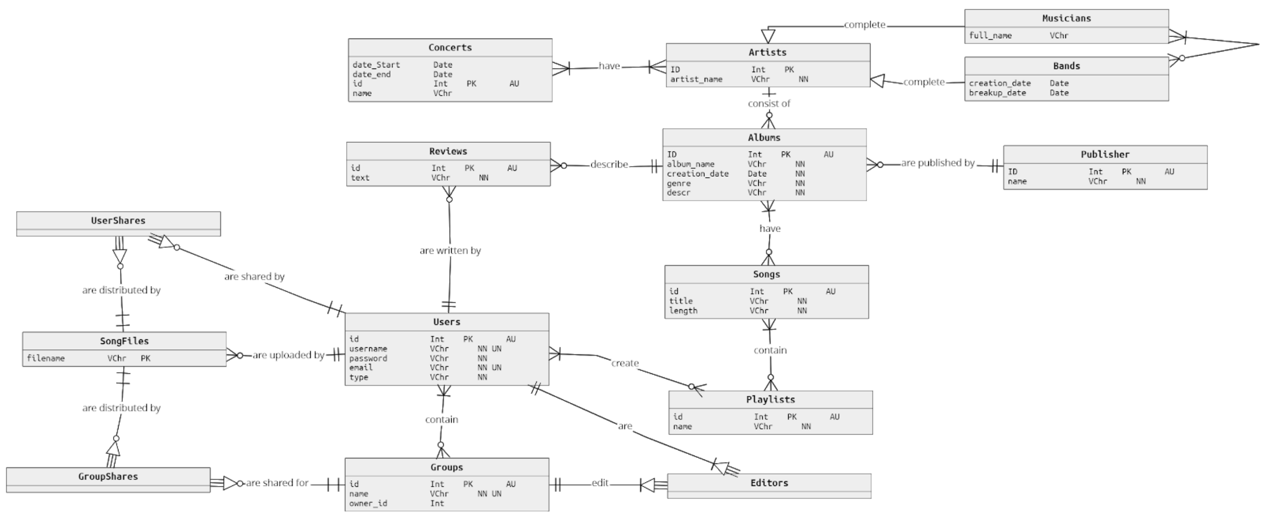
Also, as Erasmus students, we had to translate the assignment document using Google Translate. Any doubts were explained by the teacher, but still some parts of the text could have resulted in bad translation leading to wrong perception of the details.

The following points explain relations between entities:

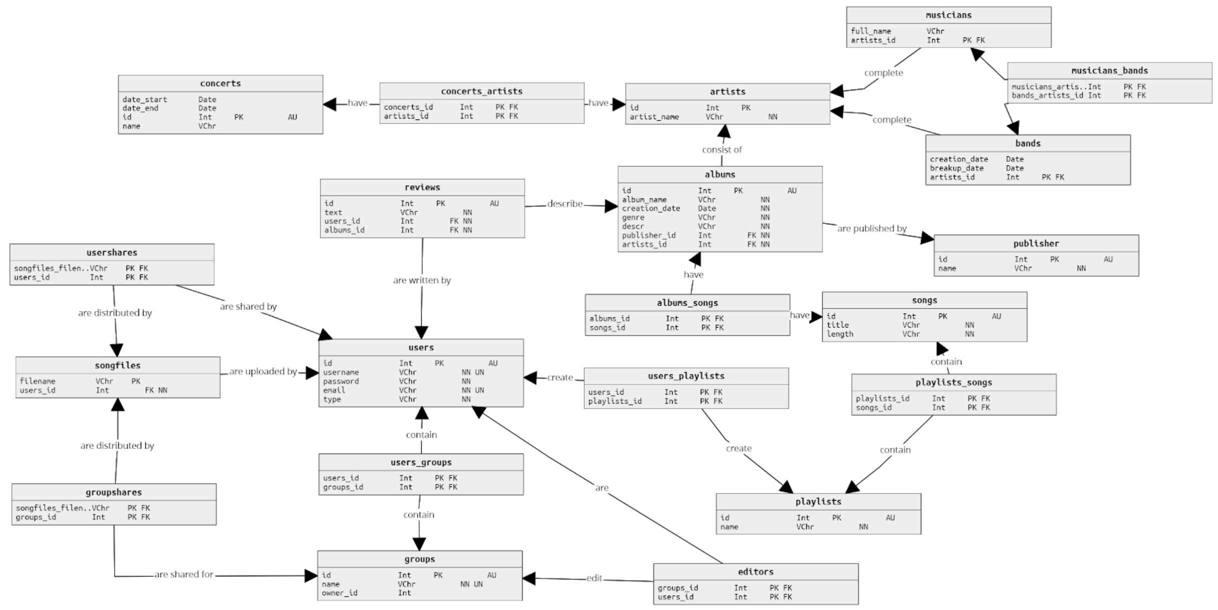
- Musicians complete Artists - Artists entity is the one which Musicians inherit from;
- Bands complete Artists - Artists entity is the one which Bands inherit from.
- Artists create Albums - there can be one artist that creates multiple albums
- Albums are published by Publishers - Publishers have another table, because they don't depend on albums. Multiple albums can have one publisher
- Songs have Albums (and vice versa) - the reason for many to many relation is that one song can actually be in multiple albums (i.e. best of albums).
- Reviews describe Albums - many reviews can describe one album and only one album and album must exist before review is created (review doesn't)
- Reviews are written by Users - self explanatory, one user writes multiple reviews
- Playlists contain Songs (and vice versa) - many to many relationship, because a song can be in many playlists, and a playlist can have multiple songs.
- Users create Playlists - many to many relationship, because user should be able to create multiple playlists and playlists are added by many users (assuming they are always public and any playlist can be added by any user)
- Editors are Users - Editors are weak entity, doesn't need its own key, because it foreign keys group id and user id identify each editor. User can be an editor in many groups, and one editor status cannot be shared by many users.
- Editors edit Groups - There can be many editor in a group, but one editor status doesn't apply for many groups.
- Groups contain Users - many to many relationship, because user can be in multiple groups (but always exists in the public one) and one group can contain many users
- Groupshares are distributed by SongFiles - weak entity, because there is no need of having a primary key for this one, as each share is identified by 2 foreign keys: file owner, the file (file is connected to one user, so there is no need for the user id to be in that table) and the group. one file can can be distributed in multiple shares

- Groupshares are shared for Groups - multiple shares (filename, group) can be in a group but one share (filename,group) have one group)Usershares are shared for Users - weak enitity, same case for Groupshares.
- Usershares are distributed by SongFiles
- SongFiles are uploaded by Users - one user can upload multiple files, but each one is assgned to one user.
- Groups have Users and Users belong to Groups - many to many relationship

  The diagram:



2.Physical Diagram:



3.DB creation script:

```
CREATE TABLE artists (
                id                              INTEGER,
                artist_name VARCHAR(512) NOT NULL,
                PRIMARY KEY(id)
);

CREATE TABLE musicians (
                full_name       VARCHAR(512),
                artists_id INTEGER,
                PRIMARY KEY(artists_id)
);
```

```
CREATE TABLE bands (
            creation_date DATE,
            breakup_date            DATE,
            artists_id      INTEGER,
            PRIMARY KEY(artists_id)
);

CREATE TABLE albums (
            id                      INTEGER,
            album_name VARCHAR(512) NOT NULL,
            creation_date DATE NOT NULL,
            genre           VARCHAR(512) NOT NULL,
            descr           VARCHAR(512) NOT NULL,
            publisher_id  INTEGER NOT NULL,
            artists_id      INTEGER NOT NULL,
            PRIMARY KEY(id)
);

CREATE TABLE publisher (
            id              INTEGER,
            name VARCHAR(512) NOT NULL,
            PRIMARY KEY(id)
);

CREATE TABLE songs (
            id              INTEGER,
            title           VARCHAR(512) NOT NULL,
            length VARCHAR(512) NOT NULL,
            PRIMARY KEY(id)
);

CREATE TABLE reviews (
            id              INTEGER,
            text            VARCHAR(512) NOT NULL,
            users_id        INTEGER NOT NULL,
            albums_id INTEGER NOT NULL,
            PRIMARY KEY(id)
);

CREATE TABLE users (
            id              INTEGER,
            username VARCHAR(512) UNIQUE NOT NULL,
            password VARCHAR(512) NOT NULL,
            email           VARCHAR(512) UNIQUE NOT NULL,
            type            VARCHAR(512) NOT NULL,
            PRIMARY KEY(id)
);

CREATE TABLE playlists (
            id              INTEGER,
            name VARCHAR(512) NOT NULL,
            PRIMARY KEY(id)
);

CREATE TABLE groups (
            id              INTEGER,
            name            VARCHAR(512) UNIQUE NOT NULL,
            owner_id INTEGER,
            PRIMARY KEY(id)
);

CREATE TABLE editors (
            groups_id INTEGER,
            users_id        INTEGER,
            PRIMARY KEY(groups_id,users_id)
);

CREATE TABLE songfiles (
            filename VARCHAR(512),
            users_id INTEGER NOT NULL,
            PRIMARY KEY(filename)
);

CREATE TABLE groupshares (
            songfiles_filename VARCHAR(512),
            groups_id                       INTEGER,
            PRIMARY KEY(songfiles_filename,groups_id)
);

CREATE TABLE usershares (
            songfiles_filename VARCHAR(512),
            users_id                        INTEGER,
```

```sql
		PRIMARY KEY(songfiles_filename,users_id)
);

CREATE TABLE concerts (
		date_start DATE,
		date_end	DATE,
		id		INTEGER,
		name		VARCHAR(512),
		PRIMARY KEY(id)
);

CREATE TABLE concerts_artists (
		concerts_id INTEGER,
		artists_id	INTEGER,
		PRIMARY KEY(concerts_id,artists_id)
);

CREATE TABLE musicians_bands (
		musicians_artists_id INTEGER,
		bands_artists_id		INTEGER,
		PRIMARY KEY(musicians_artists_id,bands_artists_id)
);

CREATE TABLE users_groups (
		users_id	INTEGER,
		groups_id INTEGER,
		PRIMARY KEY(users_id,groups_id)
);

CREATE TABLE playlists_songs (
		playlists_id INTEGER,
		songs_id	INTEGER,
		PRIMARY KEY(playlists_id,songs_id)
);

CREATE TABLE users_playlists (
		users_id	INTEGER,
		playlists_id INTEGER,
		PRIMARY KEY(users_id,playlists_id)
);

CREATE TABLE albums_songs (
		albums_id INTEGER,
		songs_id	INTEGER,
		PRIMARY KEY(albums_id,songs_id)
);

ALTER TABLE musicians ADD CONSTRAINT musicians_fk1 FOREIGN KEY (artists_id) REFERENCES artists(id);
ALTER TABLE bands ADD CONSTRAINT bands_fk1 FOREIGN KEY (artists_id) REFERENCES artists(id);
ALTER TABLE albums ADD CONSTRAINT albums_fk1 FOREIGN KEY (publisher_id) REFERENCES publisher(id);
ALTER TABLE albums ADD CONSTRAINT albums_fk2 FOREIGN KEY (artists_id) REFERENCES artists(id);
ALTER TABLE reviews ADD CONSTRAINT reviews_fk1 FOREIGN KEY (users_id) REFERENCES users(id);
ALTER TABLE reviews ADD CONSTRAINT reviews_fk2 FOREIGN KEY (albums_id) REFERENCES albums(id);
ALTER TABLE editors ADD CONSTRAINT editors_fk1 FOREIGN KEY (groups_id) REFERENCES groups(id);
ALTER TABLE editors ADD CONSTRAINT editors_fk2 FOREIGN KEY (users_id) REFERENCES users(id);
ALTER TABLE songfiles ADD CONSTRAINT songfiles_fk1 FOREIGN KEY (users_id) REFERENCES users(id);
ALTER TABLE groupshares ADD CONSTRAINT groupshares_fk1 FOREIGN KEY (songfiles_filename) REFERENCES songfiles(filename);
ALTER TABLE groupshares ADD CONSTRAINT groupshares_fk2 FOREIGN KEY (groups_id) REFERENCES groups(id);
ALTER TABLE usershares ADD CONSTRAINT usershares_fk1 FOREIGN KEY (songfiles_filename) REFERENCES songfiles(filename);
ALTER TABLE usershares ADD CONSTRAINT usershares_fk2 FOREIGN KEY (users_id) REFERENCES users(id);
ALTER TABLE concerts_artists ADD CONSTRAINT concerts_artists_fk1 FOREIGN KEY (concerts_id) REFERENCES concerts(id);
ALTER TABLE concerts_artists ADD CONSTRAINT concerts_artists_fk2 FOREIGN KEY (artists_id) REFERENCES artists(id);
ALTER TABLE musicians_bands ADD CONSTRAINT musicians_bands_fk1 FOREIGN KEY (musicians_artists_id) REFERENCES musicians(artists_id);
ALTER TABLE musicians_bands ADD CONSTRAINT musicians_bands_fk2 FOREIGN KEY (bands_artists_id) REFERENCES bands(artists_id);
ALTER TABLE users_groups ADD CONSTRAINT users_groups_fk1 FOREIGN KEY (users_id) REFERENCES users(id);
ALTER TABLE users_groups ADD CONSTRAINT users_groups_fk2 FOREIGN KEY (groups_id) REFERENCES groups(id);
ALTER TABLE playlists_songs ADD CONSTRAINT playlists_songs_fk1 FOREIGN KEY (playlists_id) REFERENCES playlists(id);
ALTER TABLE playlists_songs ADD CONSTRAINT playlists_songs_fk2 FOREIGN KEY (songs_id) REFERENCES songs(id);
ALTER TABLE users_playlists ADD CONSTRAINT users_playlists_fk1 FOREIGN KEY (users_id) REFERENCES users(id);
ALTER TABLE users_playlists ADD CONSTRAINT users_playlists_fk2 FOREIGN KEY (playlists_id) REFERENCES playlists(id);
ALTER TABLE albums_songs ADD CONSTRAINT albums_songs_fk1 FOREIGN KEY (albums_id) REFERENCES albums(id);
ALTER TABLE albums_songs ADD CONSTRAINT albums_songs_fk2 FOREIGN KEY (songs_id) REFERENCES songs(id);
```