BLG317E Database Systems
ER Diagram and Data Model Design

Berke Kurt 150210329

Efe Can Kırbıyık 150220768

December 17, 2024

1 ER Diagram

The ER diagram is shown in figure 1.

1.1 Relationships

The names, cardinalities, and modalities of the relations are shown below.

1. One-to-One Relationships

- **Entities:** Account ↔ User
  - **Name:** is a
  - **Cardinality:** 1:1
  - **Modality:** Mandatory on both sides (each account must have one user, and each user must belong to one account).
- **Entities:** Group ↔ User
  - **Name:** is a
  - **Cardinality:** 1:1
  - **Modality:** Mandatory on group side, (each group must be a user), optional on the user side (a user may not be a group).

2. One-to-Many Relationships

- **Entities:** User ↔ Playlist
  - **Name:** creates
  - **Cardinality:** 1:N
  - **Modality:** Mandatory on the playlist side (each playlist must be created by a user), optional on the user" side (a user may not be created any playlists).

3. Many-to-Many Relationships

- **Entities:** User ↔ Song
  - **Name:** listens
  - **Cardinality:** N:N
  - **Modality:** Optional on both sides (each user may listen many songs and each song may be listened by many users).
- **Entities:** Album ↔ Group

  - **Name:** has
  - **Cardinality:** N:N
  - **Modality:** Mandatory on the album side (each album must belong to a group), optional on the group side (a group may or may not has albums).

- **Entities:** Album $\leftrightarrow$ Song
  - **Name:** consist of
  - **Cardinality:** N:N
  - **Modality:** Mandatory on both sides (each song must belong to an album and each album must consist of one or more songs).

- **Entities:** Song $\leftrightarrow$ Playlist
  - **Name:** contains
  - **Cardinality:** N:N
  - **Modality:** Optional on both sides (each playlist may contain one or more songs and each song may be in several playlists).

- **Entities:** User $\leftrightarrow$ Playlist
  - **Name:** follows
  - **Cardinality:** N:N
  - **Modality:** Optional on both sides (each user may follow one or more playlists and each playlist may be followed by one or more users).

- **Entities:** Song $\leftrightarrow$ Genre (weak entity)
  - **Name:** has (weak relation)
  - **Cardinality:** N:N
  - **Modality:** Mandatory on genre side (each genre must belong to one or more songs), optional on song side(a song may or may has one or more genres).

- **Entities:** Group $\leftrightarrow$ Artist (weak entity)
  - **Name:** part of (weak relation)
  - **Cardinality:** N:N
  - **Modality:** Mandatory on both sides (each artist must belong to one or more groups and each group should contain at least one artist).

1.2 Entities And Their Attributes

- **Account:** account_id, mail, full_name, is_subscriber, registration_date, country, sex, language, birth_date
- **User:** user_id, nickname, favorite_genre, user_image
- **Playlist:** playlist_id, playlist_name, playlist_description, playlist_image, creator
- **Group:** group_id, group_name, number_of_members, creation_date, group_image
- **Album:** album_id, album_name, about, album_image
- **Song:** song_id, song_name, song_time, song_image, audio
- **Genre (weak entity):** genre
- **Artist (weak entity):** full_name, origin_country, instrument

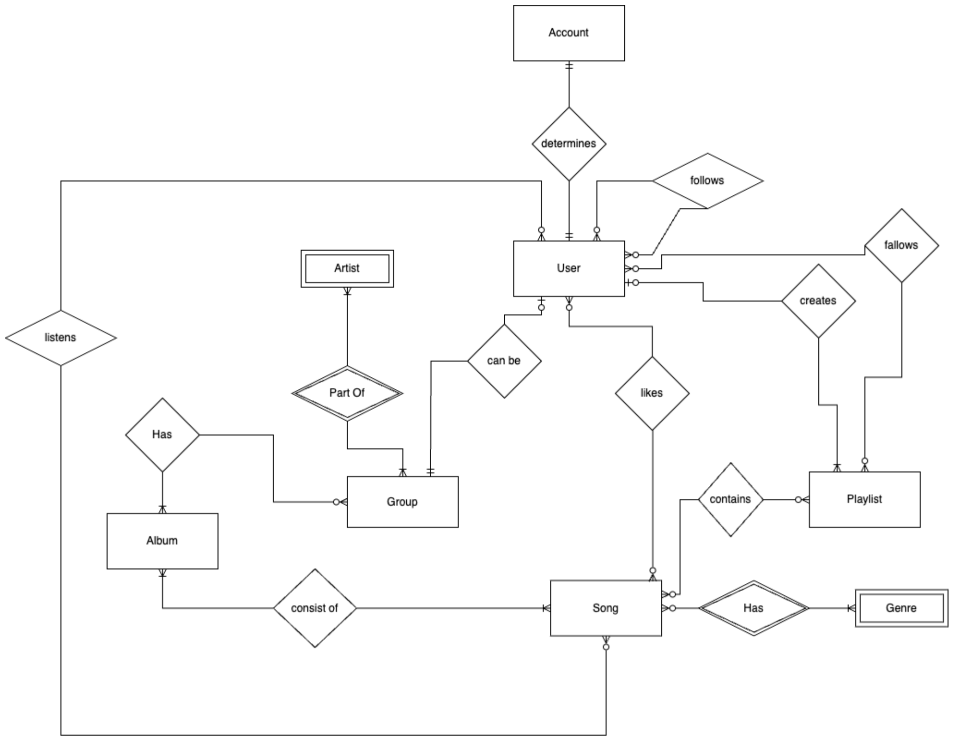

Figure 1: ER Diagram

2 Data Model

Data model is shown in figure 2. The descriptions and purposes of the tables are explained in the following.

- **Account:** Stores detailed account information such as email, subscription status, and personal details.
- **User:** Contains basic user information, including nickname and favorite genre.
- **Playlist:** Holds data on playlists created by users, including descriptions and images.
- **Group:** Stores details about music groups or bands, including their creation date and image.
- **Album:** Contains information on music albums, including album name and image.
- **Song:** Includes details about songs such as name, duration, and audio file.
- **Genre:** Associates songs with their genres.
- **Artist:** Contains information on individual artists, including their instruments.
- **Follower:** Manages user follow relationships.
- **History:** Logs user listening history.
- **Like:** Records which songs are liked by which users.
- **Playlist_Follower:** Tracks which users follow which playlists.
- **Playlist_Song:** Links songs to playlists, indicating which playlist contains which songs.
- **Album_Info:** Links songs to their respective albums, indicating which album contains which songs.
- **Album_Group:** Links albums to groups, indicating which group released which album.

2.1 Example Queries

- **Get follower count for each customer:**

```
SELECT User.user_id, f.fallower_count
FROM User
LEFT JOIN (
    SELECT user_id_1, COUNT(*) AS fallower_count
    FROM Fallower
    GROUP BY user_id_1
) AS f
ON User.user_id = f.user_id_1;
```

- **Get total listening time of each album in descending order:**

```
SELECT albums.album_name AS album_name,
       SUM(stream.total_listen_time) AS total_listen_time
FROM (
    SELECT Album.album_id, Album.album_name, Album.song_id
    FROM Album
    LEFT JOIN Album_Info
    ON Album.album_id = Album_Info.album_id
) AS albums
LEFT JOIN (
    SELECT Song.song_id, Song.song_name,
           SUM(History.duration) AS total_listen_time
    FROM History
    JOIN Song ON History.song = Song.song_id
    GROUP BY Song.song_id
) AS stream
ON albums.song_id = stream.song_id
GROUP BY albums.album_id, albums.album_name
ORDER BY total_listen_time DESC;
```

- **Find most listened genre in the last month:**

```
SELECT g.genre, SUM(h.duration) AS total_listen_time
FROM History h
JOIN Song s ON h.song = s.song_name
JOIN Genre g ON s.song_id = g.song_id
WHERE h.start_time >= GetDate() - INTERVAL "2 month"
  AND h.start_time <= GetDate() - INTERVAL "1 month"
GROUP BY g.genre
ORDER BY total_listen_time DESC
LIMIT 1;
```

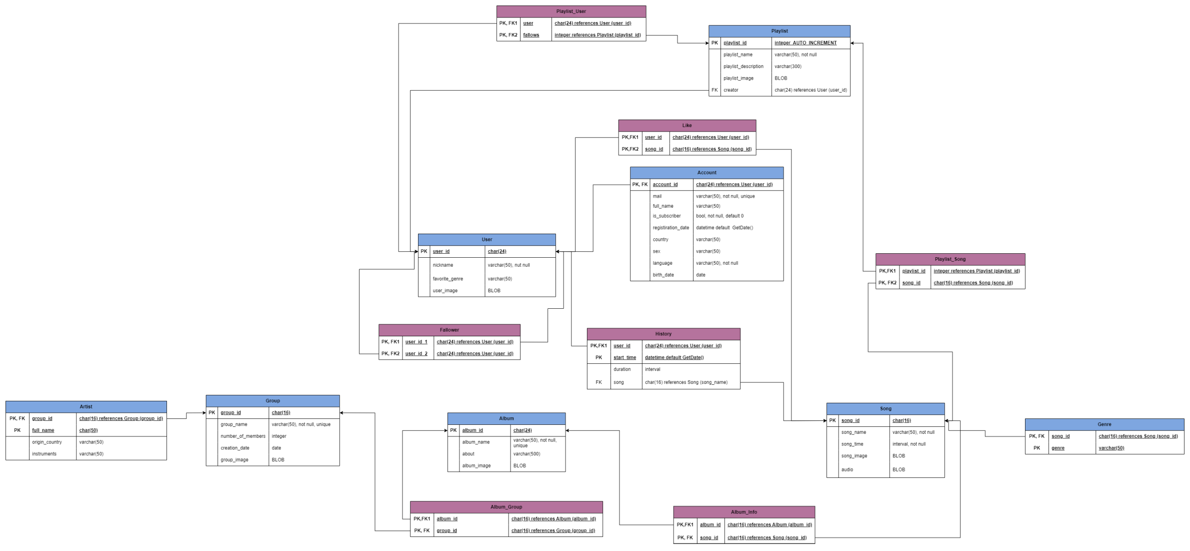
Playlist_User
PK, FK1 user char(24) references User (user_id)
PK, FK2 fallows integer references Playlist (playlist_id)
Playlist
PK playlist_id integer AUTO_INCREMENT
playlist_name varchar(50), not null
playlist_description varchar(300)
playlist_image BLOB
FK creator char(24) references User (user_id)
Like
PK,FK1 user_id char(24) references User (user_id)
PK,FK2 song_id char(16) references Song (song_id)
Account
PK, FK account_id char(24) references User (user_id)
mail varchar(50), not null, unique
full_name varchar(50)
is_subscriber bool, not null, default 0
registiration_date datetime default GetDate()
country varchar(50)
sex varchar(50)
language varchar(50), not null
birth_date date
User
PK user_id char(24)
nickname varchar(50), nut null
favorite_genre varchar(50)
user_image BLOB
Playlist_Song
PK,FK1 playlist_id integer references Playlist (playlist_id)
PK, FK2 song_id char(16) references Song (song_id)
Fallower
PK, FK1 user_id_1 char(24) references User (user_id)
PK, FK2 user_id_2 char(24) references User (user_id)
History
PK,FK1 user_id char(24) references User (user_id)
PK start_time datetime default GetDate()
duration interval
FK song char(16) references Song (song_name)
Artist
PK, FK group_id char(16) references Group (group_id)
PK full_name char(50)
origin_country varchar(50)
instruments varchar(50)
Group
PK group_id char(16)
group_name varchar(50), not null, unique
number_of_members integer
creation_date date
group_image BLOB
Album
PK album_id char(24)
album_name varchar(50), not null, unique
about varchar(500)
album_image BLOB
Song
PK song_id char(16)
song_name varchar(50), not null
song_time interval, not null
song_image BLOB
audio BLOB
Genre
PK, FK song_id char(16) references Song (song_id)
PK genre varchar(50)
Album_Group
PK,FK1 album_id char(16) references Album (album_id)
PK, FK group_id char(16) references Group (group_id)
Album_Info
PK,FK1 album_id char(16) references Album (album_id)
PK, FK song_id char(16) references Song (song_id)

Figure 2: Data Model