Project 8 Design Report

Group number: 8

1.0 Project Description:

1.1 Project Description:

1.1.1 Project Aim:

This project's main aim is to develop a database to store music for a music streaming application. This database is mainly used for storing "songs", "albums", "library" and keeping track of users of the music streaming application. The users of this application can either be artists, listeners, or developers of the application.

1.1.2 Project Scope:

-In-Scope: The scope of the project includes interactions between artists and record labels, users and artists, users and songs, the creation of playlist and adding songs to it by the user, and maintaining songs in a library, so basically storing music/song metadata is within the scope of the project.

-Out-of-scope: The scope of the project excludes:

1- The ability to use the database to make song recommendations or find related songs.
2- The ability to add favorite songs to libraries.
3- The ability to add favorite artists to library.

1.1.3 Project Duration:

-Start date: October 25th, 2022.
-End date: January 2nd, 2023.

1.1.4 Platform used:

-Platform used to implement the database: Oracle Application Express (APEX).

1.1.5 Stakeholders:

The project is aimed for developers of the music streaming application. The stakeholder also includes the users of the application, which are users and listeners.

1.2 Entity sets:

The entity sets used for this project are:

1- *song*: This entity is created to hold the information of ***song***s in the music streaming platform. It consists of the following attributes:

- *id*: It is the primary key of the ***song*** entity.
- *name*: Indicates the name of the ***song***.
- *duration*: Indicates the duration of the ***song*** in minutes.
- *release_date*: Holds the date on which the ***song*** is released.
- *lyrics*: Holds the lyrics of the ***song***.
- *no_of_streams*: Indicates the number of streams that the ***song*** has.
- *{genre}*: This attribute indicates which genres the ***song*** belongs to. It is multivalued since a ***song*** can be included in more than one genre.

The ***song*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *contains*: It is a relationship between the ***song*** and ***album*** entities.
2- *holds*: It is a relationship between the ***playlist*** and ***song*** entities.

2- *user*: This entity is created to hold the information of people who will use the application. It consists of the following attributes:

- *user_id*: It is the primary key of the ***user*** entity.
- *username*: Indicates the username that the ***user*** chooses.
- *password*: Indicates the password that the ***user*** chooses.
- *email*: Holds the email address of the ***user***.
- *name*: It is a composite attribute that holds the first and last name of the user in *first_name* and *last_name* attributes respectively.
- *date_of_birth*: Holds the date of birth of the ***user***.
- *profile_image*: Holds an image that the ***user*** chooses to display in their profile.
- *age()*: It indicates the age of the ***user***, and it is derived from the *date_of_birth* attribute.
- *registration_date*: Holds the date of registration for a ***user***.

The ***user*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *follows*: It is a relationship between ***user*** entities.
2- *creates*: It is a relationship between the ***user*** and ***playlist*** entities. ***playlist*** is dependent on the ***user*** entity. ***user*** is the discriminator of ***playlist***.
3- *user_artist*: It is a relationship between the ***user*** and ***artist*** entities.
4- *user_library*: It is a relationship between the ***user*** and ***library*** entities

3- *artist*: This entity is created to differentiate an ***artist*** from a regular ***user***. An ***artist*** can have ***album***s which contain ***song***s. It consists of the following attributes:

- *artist_id*: It is the primary key of the ***artist*** entity.
- *name*: Indicates the name of the ***artist***.
- *monthly_streams*: Indicates the number of streams that the ***artist*** receives in a month.

The ***artist*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *contract*: It is a relationship between ***record_label*** and ***artist*** entities. It has an attribute which is *start_date*.
2- *has*: It is a relationship between the ***album*** and ***artist*** entities.
3- *user_artist*: It is a relationship between the ***user*** and ***artist*** entities.

4- *album*: This entity is created to depict an ***album*** which holds ***song***s. It consists of the following attributes:

- *album_id*: It is the primary key of the ***album*** entity.
- *name*: Indicates the name of the ***album***.
- *duration*: Indicates the duration of the ***album***.
- *release_date*: Holds the date on which the ***album*** is released.
- *image*: Holds the cover image of the ***album***.

The ***album*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *contains*: It is a relationship between ***album*** and ***song*** entities.
2- *has*: It is a relationship between the ***album*** and ***artist*** entities.
3- *lib_album*: It is a relationship between the ***library*** and ***album*** entities.

5- *playlist*: This entity is used to hold ***song***s selected by the ***user*** in a certain place. It is a weak entity that is dependent on the user entity, since it cannot exist unless the ***user*** decides to create a ***playlist***. It consists of the following attributes:

- *playlist_id*: It is the identifier of the ***playlist*** entity.
- *name*: Indicates the name of the ***playlist***.
- *duration*: Indicates the duration of the ***playlist***.
- *image*: Holds the image that the ***user*** chooses as the cover of the ***playlist***.

The ***playlist*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *creates*: It is a relationship between the ***user*** and ***playlist*** entities. ***playlist*** is dependent on the ***user*** entity. ***user*** is the discriminator of ***playlist***.
2- *holds*: It is a relationship between the ***song*** and ***playlist*** entities.
3- *lib_playlist*: It is a relationship between the ***library*** and ***playlist*** entities.

6- *library*: This entity is created to hold a ***user***'s ***playlist***s. Each ***user*** has only one ***library*** that is created automatically when the ***user*** signs up to the music streaming platform. It consists of the following attribute:

- *library_id*: It is the primary key of the ***library*** entity.

The ***library*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *user_library*: It is a relationship between ***user*** and ***library*** entities.
2- *lib_playlist*: It is a relationship between the ***library*** and ***playlist*** entities.
3- *lib_album*: It is a relationship between the ***library*** and ***album*** entities

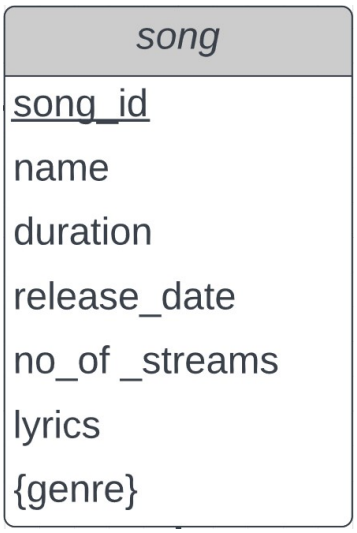
7- *record_label*: This entity is created to depict a *record_label* company with which an ***artist*** may be contracted. It consists of the following attributes:

- *record_label_id*: It is the primary key of the ***record_label*** entity.
- *record_label_name*: Indicates the name of the ***record_label***.
- *country*: Indicates the country in which the ***record_label*** is located.
- *{telephone_no}*: Holds the telephone number of the ***record_label***. It is a multivalued attribute since the company may have more than one telephone number.
- *email*: Holds the email address of the ***record_label*** company.

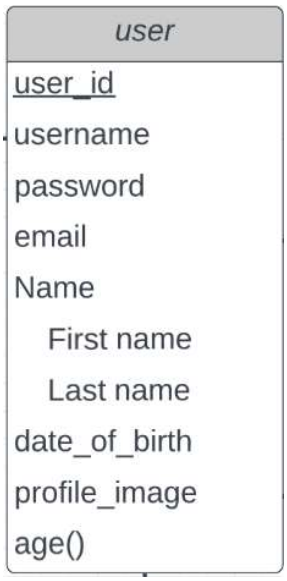
The ***record_label*** entity participates in more than one relationship and these relationships are explained in detail in section 1.3 Relationship Sets. The brief description of the relationships are as follows:

1- *contract*: It is a relationship between ***record_label*** and ***artist*** entities. It has an attribute which is *start_date*.

-Song entity set with attributes:



-User entity with attributes.

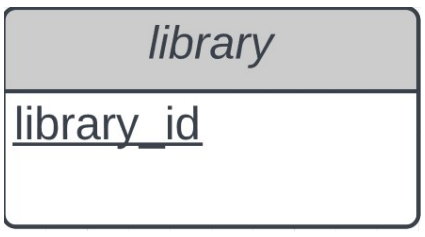


-Artist entity set with attributes:

-Album entity set with attributes:

-Playlist entity set with attributes:

-Library entity set with attributes:



-Record Label entity set with attributes:

1.3 Relationship sets:

The relationship sets used are:

1-Contract: *contract* relationship set is used in relationship between ***artist*** and ***record_label***. It became necessary to use it, as every artist may or may not sign a contract with a record label. The extra attribute *start_date* holds the information of when the artist signed the contract. As it can be seen in ERD, *contract* has partial participation on both sides, has one-cardinality on *record_label* side, and has many-cardinality on *artist* side.

- An artist may contract with at most one record label.
- A record label may contract with several artists.

2-User_artist: *user_artist* relationship set is used to denote that an ***artist*** is a ***user***, and artists can interact with users. *user_artist* has total participation on *artist* side, and has one-cardinality on both sides.

- An artist must be a user.
- A user may be at most one artist.

3-Has: *has* relationship set is used in the situation where *artist* has *album*. If an artist has only songs and has no official album, our application shows the singles as an album. For example, if the artist has 2 singles, it is kept as 2 albums in database. *has* relationship set has many-cardinality on both sides, and total participation on *album* side.

- An artist may have several albums.
- An album must be owned by an artist and may be owned by several artists.

4-Contains: *contains* relationship set is created to depict the *album*s containing *song*.

- An album must contain a song and may contain several songs.
- A song may be contained by several albums.

5-User_library: *user_library* relationship set is created to show the relationship provided between a user and their respective library. When user has been created, a library is also created belonging to that user, automatically. In short every user must have a library, that's why *user_library* has total participation on the *library* side. Relationship set has one-cardinality on both sides as the following:

- A user may have at most one library.
- A library must be owned by a user.

6-Follows: *follows* is a recursive relationship set with roles *user_id* and *following_id* as this relation holds when *user follows following_id* with *user_id*.

- A user may follow several another user.
- A user may be followed by several another user.

7-Creates: *creates* relationship set is used to denote user's ability to create a playlist by putting songs in it. There is no collaborator to a playlist as *creates* has the one-cardinality on ***user*** side. *creates* also has many-cardinality and total participation on ***playlist*** side.

- A user may create several playlists.
- A playlist may be created by at most one user and must be created by a user.

8-Holds: *holds* relationship set depicts the situation that *songs* are held by *playlists*. Holds has many-cardinality on ***song*** side, many-cardinality and total participation on ***playlist*** side.

- A playlist may hold several songs and must hold a song.
- A song may be held by more than one playlist.

9-Lib_album: *lib_album* relationship set denotes the case that user's ***library*** provides place for ***album***s. Both sides have partial participation and many-cardinality.

- A library may have several albums.
- An album may be in several libraries.

10-Lib_playlist: *lib_playlist* relationship set denotes the case that user's *library* provides place for ***playlist***s. Both sides have many-cardinality and ***playlist*** side has total participation.

- A playlist may be in several libraries and must be in a library.
- A library may have several playlists.

1.4 Users of the system:

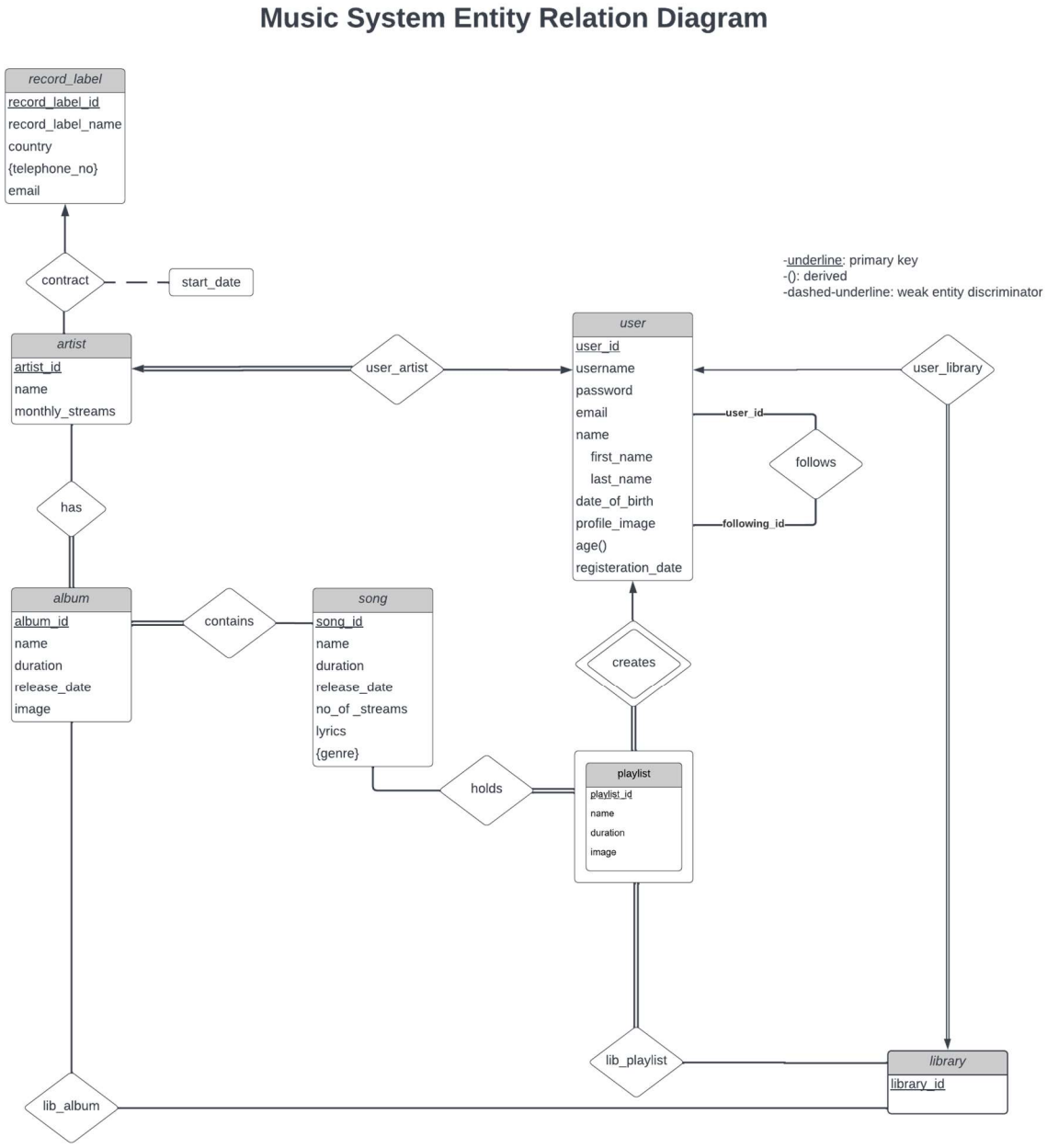
The main users of the system are:

1-Listeners.
2-Artists.

-**Note**: The other users of the system which will work on maintaining the system include the application developers, and record labels, where record labels can only view artists and add and update telephone numbers.

2.0 Entity Relation Diagram:

2.1 The diagram:



2.2 Assumptions and Business rules:

2.2.1 Assumptions:

The assumptions made while designing the ERD diagram were:

1- A song can belong to multiple albums.
2- An artist can contract with at most 1 record label.
3- An album must include at least 1 song.
4- An artist must have either a song or an album.

2.2.2 Business rules:

The business rules for the system are:

1- A *user* can only have **one** *email*.
2- A *user* can only have **one** *library* each.
3- *Artists* may have at most **one** *record label*.

4- A *record label* can have a contract with **many** *artists*.
5- Each *user* can be a **single** *artist*.
6- A *song* may be contained in **multiple** *albums*.
7- An *artist* cannot have a *song* **without** an *album*.
8- An *album* must contain at least **one** *song*.
9- A *playlist* must hold at least **one** *song*.
10- A *user* may follow **other** *users*.
11- A *library* may contain **one or more** *albums* or *playlists*.
12- Each *user* can create a *playlist*.
13- A *song* may be contained in **multiple** *playlists*.
14- A *telephone number* can only be added by the *administrator*.
15- A *record label* can view *artists*.

3.0 Relational schema:

A. *REPRESENTING AS A TABLE:*

Entities:

record_label(record_label_id, record_label_name, country, email)

artist(artist_id, name, monthly_streams)

user(user_id, username, password, email, first_name, last_name, date_of_birth, profile_image, registeration_date)

album(album_id, name, duration, release_date, image)

song(song_id, name, duration, release_date, no_of_streams, lyrics)

library(library_id)

playlist(user_id, playlist_id, name, duration, image)

Multivalued Attributes:

record_label_telephone_number(record_label_id, telephone_no)

song_genre(song_id, genre)

Relations:

contract(record_label_id, artist_id, start_date)

user_artist(user_id, artist_id)

has(artist_id, album_id)

contains(album_id, song_id)

holds(playlist_id, song_id)

lib_playlist(library_id, playlist_id)

lib_album(library_id, album_id)

follows(user_id, following_id)

user_library(user_id, library_id)

Since it is an identifying relationship, ‘creates’ does not become table. Instead, ‘user_id’ is added as a primary key of ‘playlist’.

B. *ELIMINATION OF REDUNDANCIES:*

Entities:

1>> record_label(record_label_id, record_label_name, country, email)

2>> artist(artist_id, name, monthly_streams, start_date, record_label_id, user_id)

3>> user(user_id, username, password, email, first_name, last_name, date_of_birth, profile_image, registeration_date)

4>> album(album_id, name, duration, release_date, image)

5>> song(song_id, name, duration, release_date, no_of_streams, lyrics)

6>> library(library_id, user_id)

7>> playlist(user_id, playlist_id, name, duration, image)

Multivalued Attributes:

record_label_telephone_number(record_label_id, telephone_no)

song_genre(song_id, genre)

Relations:

~~contract(record_label_id, artist_id, start_date)~~ (.2)

~~user_artist(user_id, artist_id)~~ (.2)

~~user_library(user_id, library_id)~~ (.6)

has(artist_id, album_id)

contains(album_id, song_id)

lib_playlist(library_id, playlist_id)

lib_album(library_id, album_id)

follows(user_id, following_id)

holds(playlist_id, song_id)

C. *RESULT (RELATIONAL SCHEMA):*

record_label(record_label_id, record_label_name, country, email)

artist(artist_id, name, monthly_streams, start_date, record_label_id, user_id)

user(user_id, username, password, email, first_name, last_name, date_of_birth, profile_image, registeration_date)

album(album_id, duration, release_date, image)

song(song_id, name, duration, release_date, no_of_streams, lyrics)

library(library_id, user_id)

playlist(user_id, playlist_id, name, duration, image)

record_label_telephone_number(record_label_id, telephone_no)

song_genre(song_id, genre)

has(artist_id, album_id)

contains(album_id, song_id)

lib_playlist(library_id, playlist_id)

lib_album(library_id, album_id)

follows(user_id, following_id)

holds(playlist_id, song_id)