MUSIC LIBRARY MANAGEMENT SYSTEM

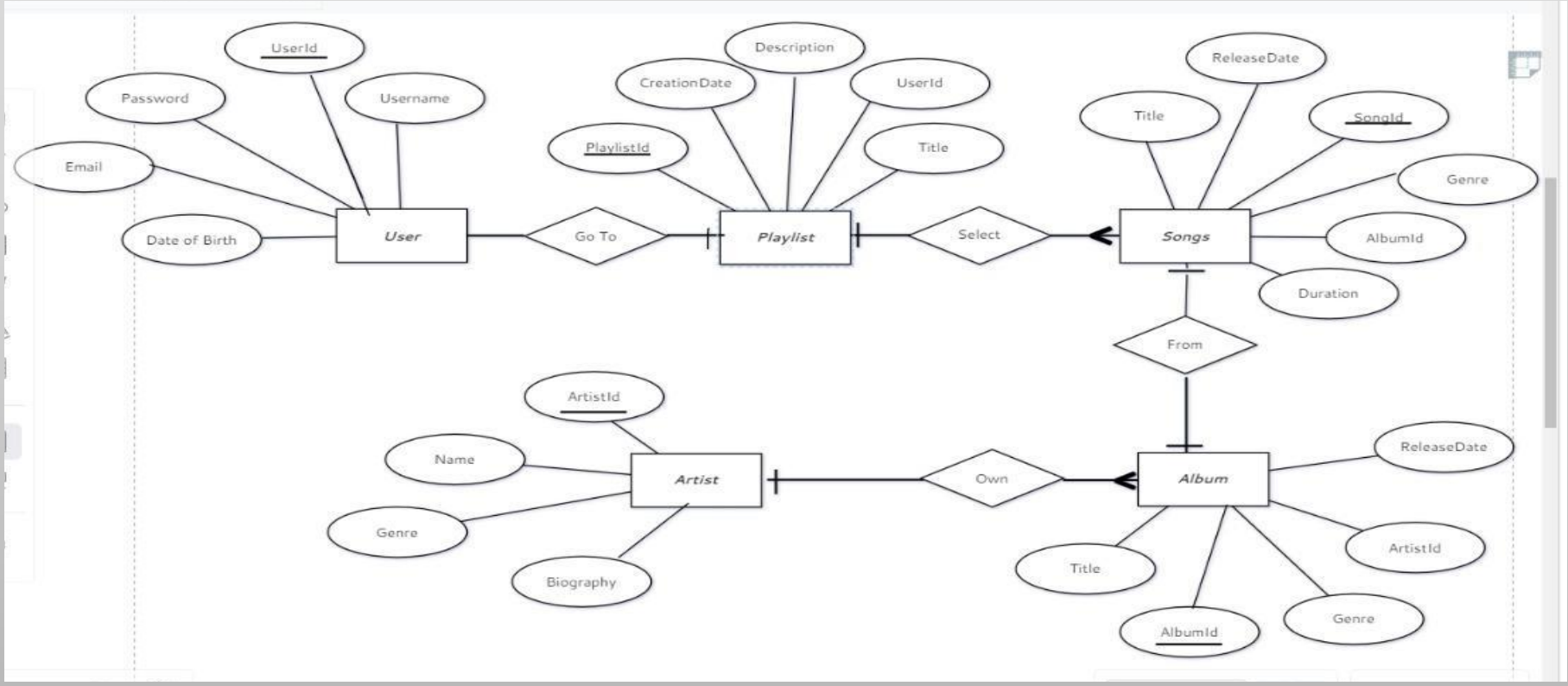
1.BACK GROUND

- The Music Library Management System aims to provide a user-friendly platform for organizing and managing music collections. It involves creating a database system to efficiently store and manage user accounts, songs, artists, albums, and playlists.

2.DESCRIPTION

- The project involves designing and implementing a database system for a music library. This includes creating an Entity-Relationship (ER) diagram, converting it into relational tables, normalizing the tables up to 3rd Normal Form (3-NF), populating the tables with sample data, writing SQL queries to retrieve information, and creating views for customized data perspectives.

3.ER DIAGRAM


UserId
Password
Username
Email
Date of Birth
User
Go To
Description
CreationDate
UserId
PlaylistId
Title
Playlist
Select
ReleaseDate
Title
SongId
Genre
Songs
AlbumId
Duration
From
ArtistId
Name
Genre
Biography
Artist
Own
ReleaseDate
Album
ArtistId
Title
AlbumId
Genre

4.DESCRIPTION FOR ER DIAGRAM

- Entities and Attributes:

1.USERS: Represents a person who uses the music platform.

Users have attributes **User ID,Username,Email,Date of Birth.**

2.PLAYLIST:Represents a curated list of songs.

Playlist have attributes **Playlist ID,Title,Description,User ID CreationDate.**

3.ALBUM:Represents a collection of songs released together by an artist or band.
Album have attributes **Album ID,Title,ReleaseDate,Genre,Artist ID.**
4.Artist:Represents a musician, band, or performer who creates music.
Artist have attributes **Artist ID,Title,ReleaseDate,Genre,Artist ID.**
5.SONG:Represents a musical composition typically consisting of lyrics and melody.
Song have attributes **Song ID,Title,Duration,ReleaseDate,Genre, Album ID.**

❖ Relationship Between these Entites:-

- Song – Album Relationship: One-to-many relationship
- Album – Artist Relationship: One-to-many relationship
- Playlist – Song Relationship: Many-to-many relationship

5. TABLES FROM ER DIAGRAM

SONG TABLE:

Song_ID	INT [PK]
Title	VARCHAR(100)
Duration	INT
ReleaseDate	DATE
Genre	VARCHAR(100)
Album_ID	INT[FK]

ALBUM TABLE:

Album_ID	INT [PK]
Title	VARCHAR(100)
Artist_ID	INT[FK]
ReleaseDate	DATE
Genre	VARCHAR(100)

ARTIST TABLE:

Artist_ID	INT [PK]
Name	VARCHAR(255)
Biography	TEXT
Genre	VARCHAR(100)

PLAYLIST TABLE:

Playlist_ID	INT [PK]
Title	VARCHAR(100)
Description	TEXT
CreationDate	DATE
User_ID	INT [FK]

USER TABLE:

User_ID	INT [PK]
Username	VARCHAR(100)
Email	VARCHAR(100)
Password	VARCHAR(100)
DateOfBirth	DATE

7. NORMALIZATION OF TABLE UP TO 3-Nf

Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like *Insertion, Update, and Deletion Anomalies.*

The normal form is used to reduce redundancy from the database table.

The main reason for normalizing the relations is removing anomalies.

- ***Insertion Anomaly***: Insertion Anomaly refers to when one cannot insert a new tuple into a relationship due to lack of data.
- ***Deletion Anomaly***: The delete anomaly refers to the situation where the deletion of data results in the unintended loss of some other important data.
- ***Updatation Anomaly***: The update anomaly is when an update of a single data value requires multiple rows of data to be updated.

- ***First Normal Form (1NF):***

  Each attribute contains atomic values, and there are no repeating groups.

- ***Second Normal Form (2NF):***

  All non-key attributes are fully functionally dependent on the primary key.

- ***Third Normal Form (3NF):***

  No transitive dependencies exist

  - **Atomicity** : Each attribute in all tables contains atomic values.
  - **No Partial Dependencies** : No partial dependencies exist in any table.
  - **No Transitive Dependencies** : no non-key attributes that depend on other non-key attributes.

FEW SQL QUERIES ON THE CREATED TABLES :

Retrieve all songs from a specific album:

```
SELECT * FROM Song
WHERE Album_ID = 1;
```

Retrieve all playlists created after a specific date:

```
SELECT * FROM Playlist
WHERE CreationDate
> '2024-01-01';
```

Retrieve the title and release date of albums released in 2023:

```
SELECT Title, ReleaseDate
FROM Album
 WHERE
YEAR(ReleaseDate) =
2023;
```

9. CREATION OF VIEWS USING THE TABLES

View 1: Songs with Album Titles

```
CREATE VIEW SongWithAlbum AS
 SELECT s.Title AS Song_Title, a.Title AS Album_Title FROM Song s
INNER JOIN Album a ON s.Album_ID = a.Album_ID;
```

View 2: Playlist Details

```
CREATE VIEW PlaylistDetails AS
SELECT p.Title AS Playlist_Title, p.CreaѲonDate, u.Username AS User_Username
FROM Playlist p
INNER JOIN Users u ON p.User_ID = u.User_ID;
```

View 3: Album Releases by Genre

```
CREATE VIEW AlbumReleasesByGenre AS
SELECT Genre, COUNT(*) AS Album_Count
FROM Album
GROUP BY Genre;
```

10.CONCLUSION

A well-designed music library system enhances the way people consume, organize, discover, and share music. It bridges the gap between users and their music collections, offering a more personalized and immersive listening experience.

PROJECT DONE BY

AP22110010007-K.Manaswi

AP22110010029-K.Swathi

AP22110010031-P.Sudhamai

AP22110010039-V. Sri Nidhi

AP22110010042-K.Madhavi