Czech University of Life Sciences

Faculty of Economics and Management

Database Systems project:

Database for Music Streaming Service

Autor: Adina Sarsenbayeva

Introduction

This work designs a database for music streaming service. It stores basic information, and it supports the functionality of the streaming service. It's based on following assumptions:

- Each user can create one or more playlist.
- Each album can contain one or more song.
- Each artist can perform one or more songs.
- Subscription can be a subject of payment
- each playlist can contain or many songs

Outline

1. Possible use cases for the model

- Find all names and dates of birth of artists who have released at least one song in the pop genre
- Find all titles and album names of songs that were released by a particular artist
- Get the titles and name of playlists that contain a particular song
- Find the names and subscription periods of users who have subscribed to the service using a particular payment method

- Find the titles and album names of songs that were released by a particular artist and have a length greater than 3 minutes

2. Entity relationship diagrams

Following section captures the proposed structure of database using entity relationship diagram. The diagrams were created in Lucidchart.

2.1. Conceptual ERD

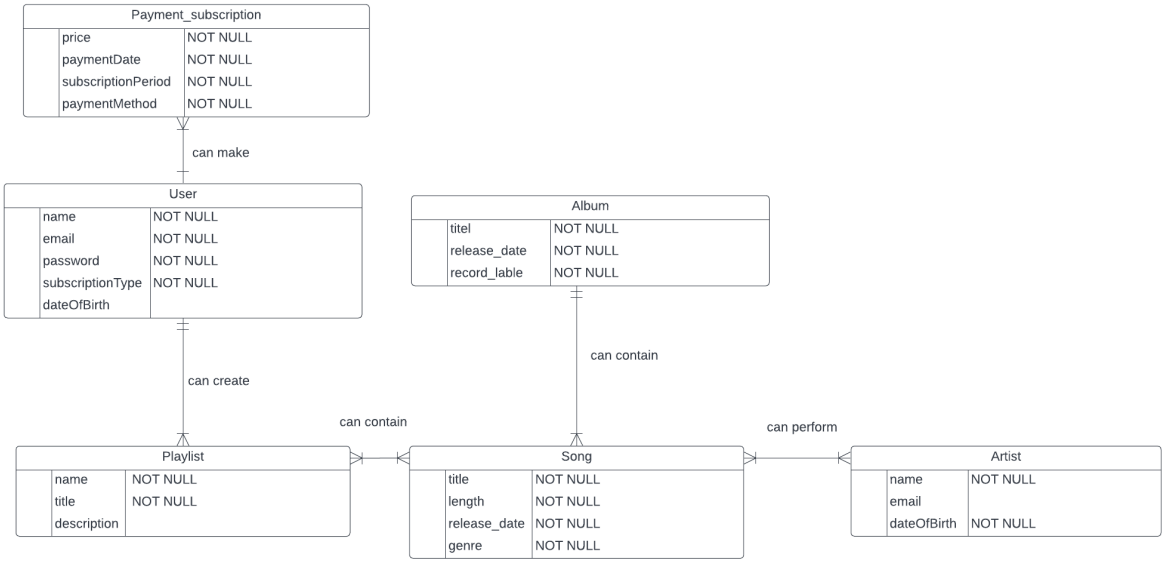

Figure 1: Conceptual ERD

2.2. Logical ERD

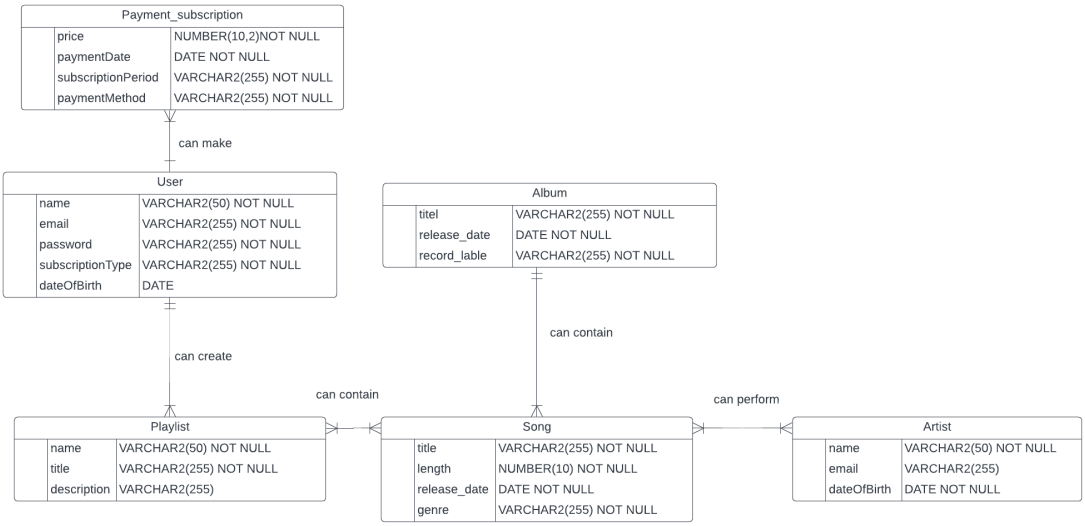

Figure 2: Logical ERD

2.3. Physical ERD

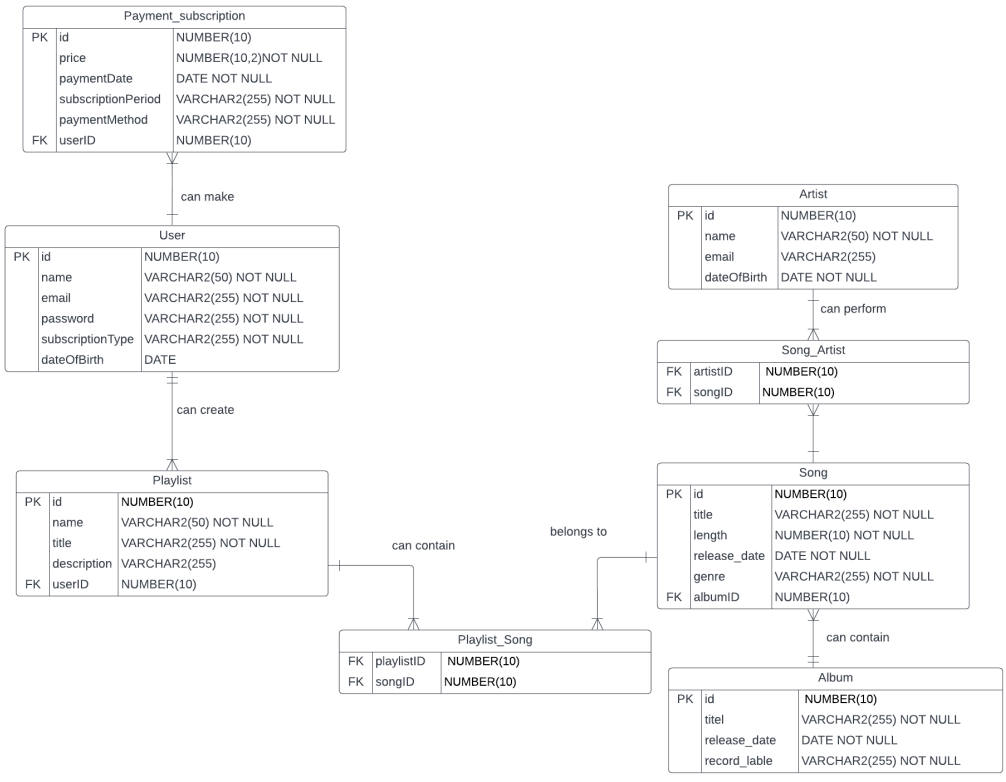

Figure 3: Physical ERD

3. SQL Implementation

The database was implemented in Oracle Application Express, which uses Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production as a DBMS.

3.1. DDL : Defining the database objects

CREATE TABLE Payment_subscription (id NUMBER(10), price NUMBER(10,2)NOT NULL, paymentDate DATE NOT NULL, subscriptionPeriod VARCHAR2(255) NOT NULL, paymentMethod VARCHAR2(255) NOT NULL, userID NUMBER(10), PRIMARY KEY (id));

CREATE TABLE Artist (id NUMBER(10), name VARCHAR2(50) NOT NULL, email VARCHAR2(255), dateOfBirth DATE NOT NULL, PRIMARY KEY (id));

CREATE TABLE Song (id NUMBER(10),title VARCHAR2(255) NOT NULL,song_length NUMBER(10) NOT NULL, release_date DATE NOT NULL, genre VARCHAR2(255) NOT NULL, albumID NUMBER(10), PRIMARY KEY (id));

CREATE TABLE Playlist (id NUMBER(10), name VARCHAR2(50) NOT NULL, title VARCHAR2(255) NOT NULL, description VARCHAR2(255), userID NUMBER(10), PRIMARY KEY (id));

CREATE TABLE Users (id NUMBER(10), name VARCHAR2(50) NOT NULL, email VARCHAR2(255) NOT NULL, password VARCHAR2(255) NOT NULL, subscriptionType VARCHAR2(255) NOT NULL, dateOfBirth DATE, PRIMARY KEY (id));

CREATE TABLE Album (id NUMBER(10),title VARCHAR2(255) NOT NULL, release_date DATE NOT NULL, record_lable VARCHAR2(255) NOT NULL, PRIMARY KEY (id));

CREATE TABLE Playlist_Song (playlistID NUMBER(10), songID NUMBER(10));

CREATE TABLE Song_Artist (artistID NUMBER(10),songID NUMBER(10));

Constraints:

ALTER TABLE Payment_subscription ADD FOREIGN KEY(userID) REFERENCES Users (id);

ALTER TABLE Playlist ADD FOREIGN KEY(userID) REFERENCES Users (id);

ALTER TABLE Playlist_Song ADD FOREIGN KEY(playlistID) REFERENCES Playlist (id);

ALTER TABLE Playlist_Song ADD FOREIGN KEY(songID) REFERENCES Song (id);

ALTER TABLE Song ADD FOREIGN KEY(albumID) REFERENCES Album (id);

ALTER TABLE Song_Artist ADD FOREIGN KEY(artistID) REFERENCES Artist (id);

ALTER TABLE Song_Artist ADD FOREIGN KEY(songID) REFERENCES Song (id);

3.2. DML: Inserting the data (examples)

INSERT INTO Users(id, name, email, password, subscriptionType, dateOfBirth) VALUES(1, 'James', 'james@example.com', '123Jj', 'Premium', '08/15/1980');

INSERT INTO Users(id, name, email, password, subscriptionType, dateOfBirth) VALUES(2, 'Jenny', 'Jenny@example.com', 'Jenn123', '7 days trial, Premium', '12/13/2001');

INSERT INTO Payment_subscription(id, price, paymentDate, subscriptionPeriod, paymentMethod, userID) VALUES(1, 200, '08/01/2019', 'Monthly', 'Visa', 1);

INSERT INTO Payment_subscription(id, price, paymentDate, subscriptionPeriod, paymentMethod, userID) VALUES(2, 200, '01/03/2023', 'Montly', 'Mastercard', 2);

INSERT INTO Album (id, title, release_date, record_lable) VALUES(1, 'When We All Fall Asleep, Where Do We Go?', '03/29/2019', 'Interscope Records');

INSERT INTO Artist (id, name, dateOfBirth) VALUES(1, 'Billie Eilish', '12/18/2001');

INSERT INTO Song(id, title, song_length, release_date, genre, albumID) VALUES(1, 'Bad guy', 194, '03/29/2019', 'Pop', 1);

INSERT INTO Song(id, title, song_length, release_date, genre, albumID) VALUES(2, 'Xanny', 134, '03/29/2019', 'Pop',1);

INSERT INTO Song_Artist(artistID, songID) VALUES(1,1);

INSERT INTO Song_Artist(artistID, songID) VALUES(1,2);

INSERT INTO Playlist(id, name, title, userID) VALUES(1, '<3', 'BestSong', 1);

INSERT INTO Playlist(id, name, title, userID) VALUES(2, 'my Playlist', 'Best Playlist', 2);

INSERT INTO Playlist_Song(playlistId, songID) VALUES(1,1);

INSERT INTO Playlist_Song(playlistId, songID) VALUES(2,2);

3.3. SQL Queries

- Find all names and dates of birth of artists who have released at least one song in the pop genre

  SELECT DISTINCT ar.name, ar.dateOfBirth FROM Artist ar INNER JOIN Song_Artist sa ON ar.id = sa.artistID INNER JOIN Song s ON sa.songID = s.id WHERE s.genre = 'Pop';

- Find all titles and album names of songs that were released by a particular artist
-

  SELECT s.title, al.title FROM Song s INNER JOIN Song_Artist sa ON s.id = sa.songID INNER JOIN Artist ar ON sa.artistID = ar.id INNER JOIN Album al ON s.albumID = al.id WHERE ar.name = 'Billie Eilish';

- Find the titles and name of playlists that contain a particular song

  SELECT p.title, p.name FROM Playlist p INNER JOIN Playlist_Song ps ON p.id = ps.playlistID INNER JOIN Song s ON ps.songID = s.id WHERE s.title = 'Bad guy';

- Get the names and subscription periods of users who have subscribed to the service using a particular payment method

SELECT u.name, ps.subscriptionPeriod FROM Users u INNER JOIN Payment_subscription ps ON u.id = ps.userID WHERE ps.paymentMethod = 'Mastercard';

- Get the titles and album names of songs that were released by a particular artist and have a length greater than 3 minutes

  SELECT s.title, al.title FROM Song s INNER JOIN Song_Artist sa ON s.id = sa.songID INNER JOIN Artist ar ON sa.artistID = ar.id INNER JOIN Album al ON s.albumID = al.id WHERE ar.name = 'Billie Eilish' AND s.song_length > 180;

4. Conclusion

This project contains a basic proposal for a database, which can be used in a music streaming service management system. It contains definitions of essential database objects, and examples of possible use cases realized in the form of SQL queries. This project can serve as a basis for further improvements. Therefore, it is an essential part of the possible implementation of full-scale systems.