**Music Streaming Database: Technical Report**

1. Keval Patel
2. Miten Patel
3. Shikhar Shah

Web Design and Development, Conestoga College

PROG 8320: Database Design and Maintenance

Prof. Dr. Angelina Ziesemer

August 16, 2021

**INTRODUCTION**

The goal of this project was designing a music streaming database where we can find the data for memberships, songs listening history of the users, artists, singers, albums, and different genres. To achieve this goal, our group divided the tasks among the members.

**GOALS:**

* Design the database for the Music Streaming scenario.
* Define the realistic constraints for all the tables.
* Defining the relationships between tables.
* Design basic ER model and then go through normalization process to make 3rd normal form.
* Populate the database.
* Solve the queries including Stored Procedure.

**GENERAL SCENARIO**

For music store, a database should be designed taking care of customers, singers, albums, and keeping track of the products sold by the store.

**SPECIFIC SCENARIO**

Our specific scenario was creating a model to support a streaming service for that music store, so that users can buy different types of membership and, the database should keep record of most popular membership in general and particular by age.

**DOCUMENTATION**

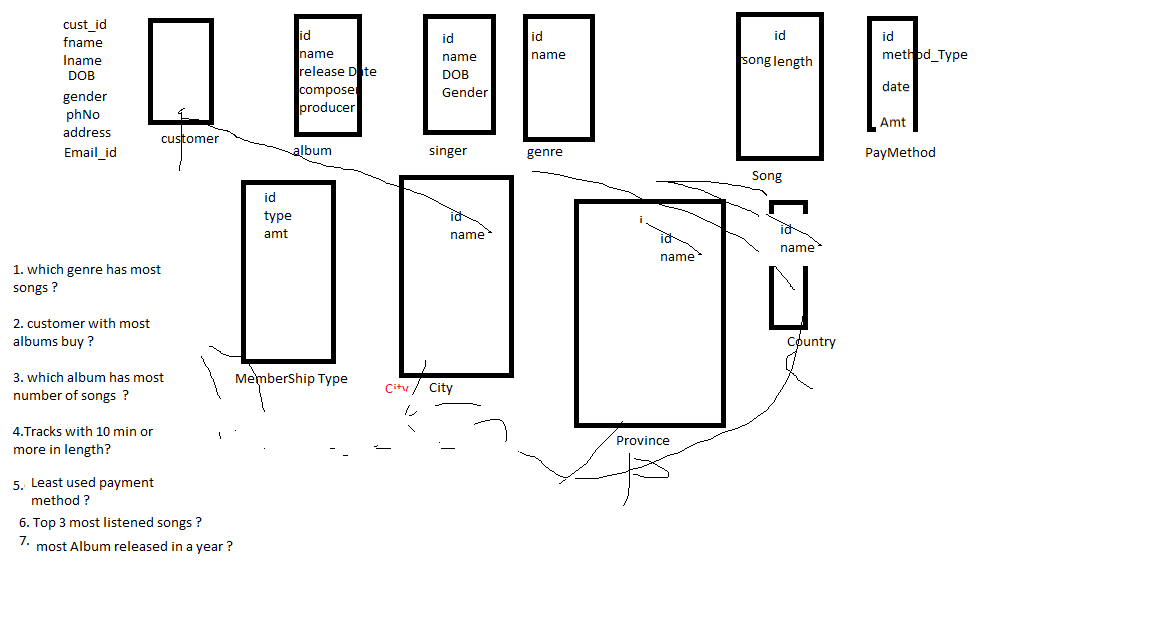


Fig 1. Discussion about questions, table name and column name

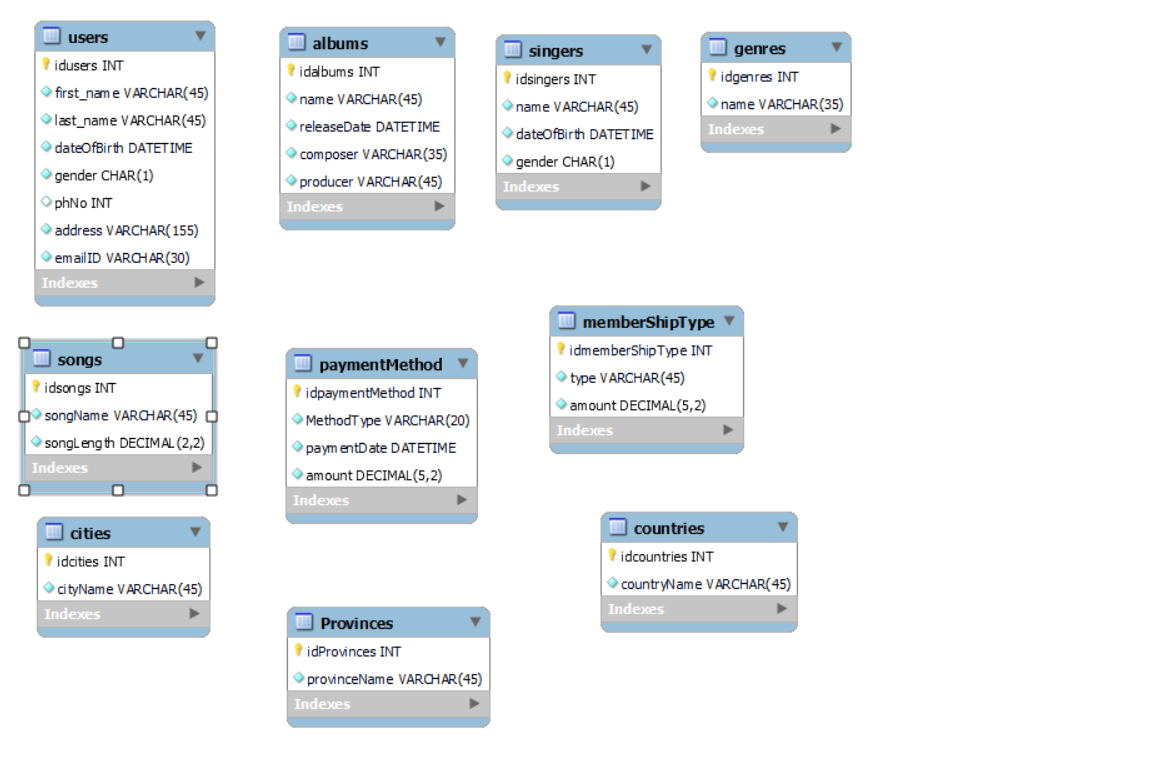
****

Fig 2. Implementing table name and column name (rough)

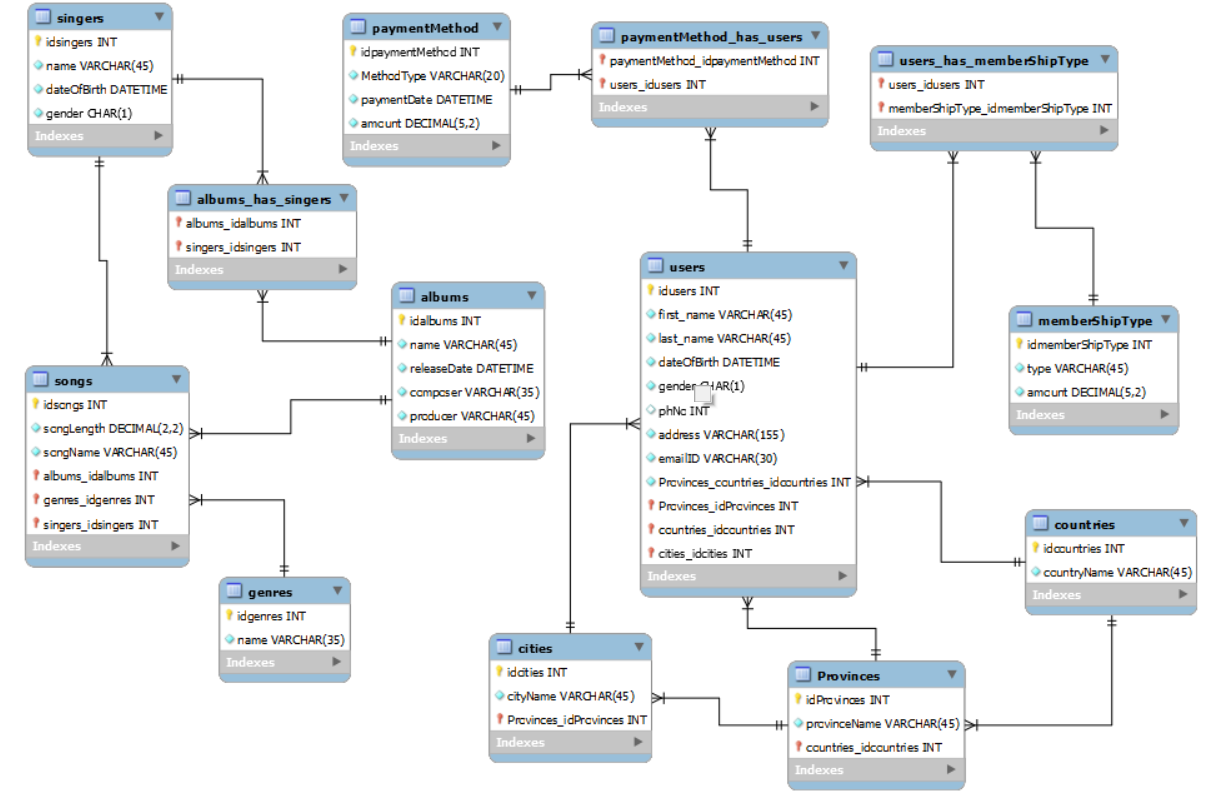
****

Fig 3. ER diagram with relationship (rough)

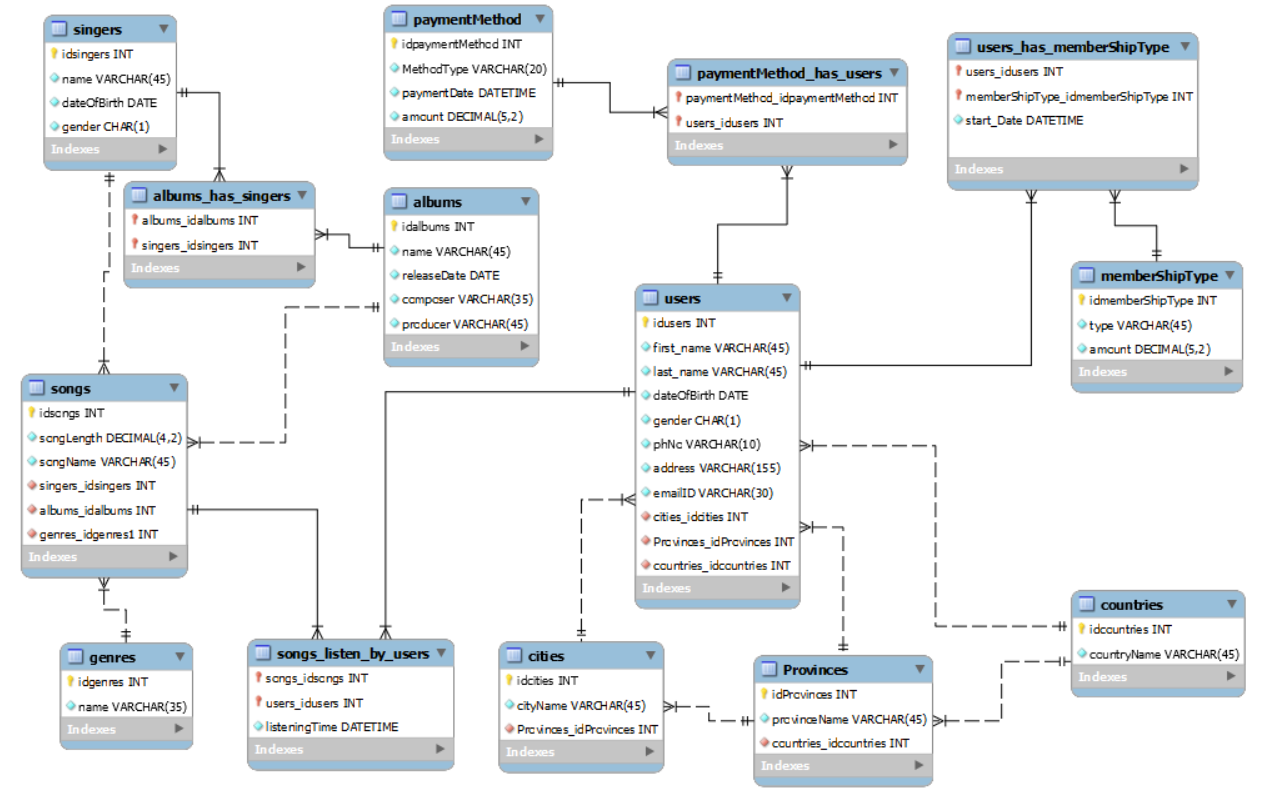
****

Fig 4. Final ER Diagram with relationship

**Solved proposed questions**

Question 1: Which gender has purchase most number of membership? (Stored Procedure)

The following tables, columns and function were necessary to answer this question:

* users (gender).
* COUNT() function.

Question 2: List how many users the system has by location (Country, Province, and City), and then sort them.

The following tables, columns and function were necessary to answer this question:

* users (cities\_idcities).
* cities (cityName, idcities, Provinces\_idProvinces).
* provinces (provinceName, idProvinces, countries\_idcountries).
* countries (countryName, idcountries).
* COUNT() function.

Question 3: List how many distinct albums each singer has.

The following tables, columns and function were necessary to answer this question:

* singers (name, idsingers).
* albums\_has\_singers (singers\_idsingers).
* COUNT() function.

Question 4: Top 5 most listened songs.

The following tables, columns and function were necessary to answer this question:

* songs (songName, idsongs).
* songs\_listen\_by\_users (songs\_idsongs).
* COUNT() function.

Question 5: Showing all genres and how many songs by each genre.

The following tables, columns and function were necessary to answer this question:

* songs (songName, genres\_idgenres1).
* genres (name, idgenres).
* COUNT() function.

Question 6: Most and less popular payment method and how many times they were used? (Stored Procedure)

The following tables, columns and function were necessary to answer this question:

* paymentmethod (MethodType).
* CONCAT(), COUNT(), MIN(), MAX() function.

Question 7: Most album released by singer in a year.

The following tables, columns and function were necessary to answer this question:

* albums (releaseDate, idalbums).
* singers (name, idsingers).
* albums\_has\_singers (albums\_idalbums, singers\_idsingers).
* COUNT(), YEAR() function.

Question 8: Most popular membership.

The following tables, columns and function were necessary to answer this question:

* membershiptype (type, idmemberShipType).
* users\_has\_membershiptype (memberShipType\_idmemberShipType).
* MAX(), COUNT() function.

Question 9: Which Membership is most popular by age?

The following tables, columns and function were necessary to answer this question:

* membershiptype (type, idmemberShipType).
* users (first\_name , idusers, dateOfBirth)
* users\_has\_membershiptype(users\_idusers, memberShipType\_idmemberShipType).
* YEAR(), CURDATE(), COUNT() function.

Question 10: Top 3 singers with least number of songs.

The following tables, columns and function were necessary to answer this question:

* singers (name, idsingers).
* songs (idsongs, singers\_idsingers).
* COUNT() function.

Question 11: Country with most subscriber.

The following tables, columns and function were necessary to answer this question:

* users (cities\_idcities).
* cities (idcities, Provinces\_idProvinces).
* provinces (idProvinces, countries\_idcountries).
* countries (countryName, idcountries).
* COUNT() function.

**CONCLUSION**

During the duration of the project, we accomplished our goal of designing and developing a Streaming database for the music store that would be able to track the data for memberships, listening history of the users, songs by genres and album of singers.

Main challenge was to decide how many tables are going to be in the database and relation between each of them. While generating database script some errors occur due to different version of software.

We were glad that we were able to complete this project on time with cooperation of all team members.

**APPENDIX**

**TABLE BREAKDOWN**

**TABLE: Album**

**Columns:**

* **idalbums**
* Data type: int(11).
* Constraints: Primary key (PK), Auto Increment (AI), Not Null.
* **name**
* Data type: varchar(45).
* Constraints: Not Null.
* **releaseDate**
* Data type: date.
* Constraints: Not Null.
* **composer**
* Data type: varchar(35).
* Constraints: Not Null.
* **producer**
* Data type: varchar(35).
* Constraints: Not Null.

**Relationships:**

* **“albums”** table has **1:M** relationship with **“songs”** table.
* **“singers”** table has **M:M** relationship with **“albums”** table.

**TABLE: GENRES**

**Columns:**

* **idgenres**
* Data type: int(11).
* Constraints: Primary key (PK), Auto Increment (AI), Not Null.
* **name**
* Data type: varchar(35).
* Constraints: Unique , Not Null.

**Relationships:**

* **“genres”** table has **1:M** relationship with **“songs”** table.