Databases and Data Mining

Music, artist, and concert database

Ibrahim Tancredi, ibrahim.tancredi@insa-lyon.fr
Oluwateniola Dania, oluwateniola.dania@insa-lyon.fr

1. Introduction

This report details the design and creation of a database detailing the discography and performances of a musical artist for the course Databases and Data Mining (DBM1). For this project, the aim was to create a database that stores and relates songs, artists, albums and labels, as well as storing data about concerts, such as their dates and venues, and relating said data to relevant performances.

2. Database design

The database consists of six entities:

Artist
Artist is an entity that describes a musician that has written and released music. This means that it describes bands or solo performers, but not the members of a band. Including the members of a band as an entity connecting to the band was ultimately discarded as it did not appear to be interesting for the purposes of our queries. This means that individual performers can be part of multiple "artist" entities depending on the artist name at the time of release. Harry styles and One Direction will both be entities, where the Harry Style entity only relates to songs from his solo career. Multiple artists can collaborate on songs, which is why the relationship between Artist and Song is a many-to-many relationship. The chosen attributes are name, country, number of members, starting year, ending year (with zero if still performing).

Song
Song is the core entity of the schema with the attributes SongLength, and SongName. Songs can be part of one or multiple albums, made by one or multiple artists, and performed at concerts by one or multiple artists.

Album
An album has one or many songs. An album with only one song is meant to model a single. Every album must be produced by a label (label would be specified as the artist themselves in the case of a self-published album). Every song is part of at least one album. This creates a path between each song and its respective label. The album also has an attribute for its release date.

Concert
A concert takes place at one venue. A concert relates to the rest of the database through the relationship *performs* (represented by concert_song_artist). *Performs* is a relationship between a concert, a song that was performed at said concert and the artist that performed said song. The attributes chosen for this entity are name and start date.

Venue
Venues have one or multiple concerts held at them. They also contain information on the venue's capacity, country and city.

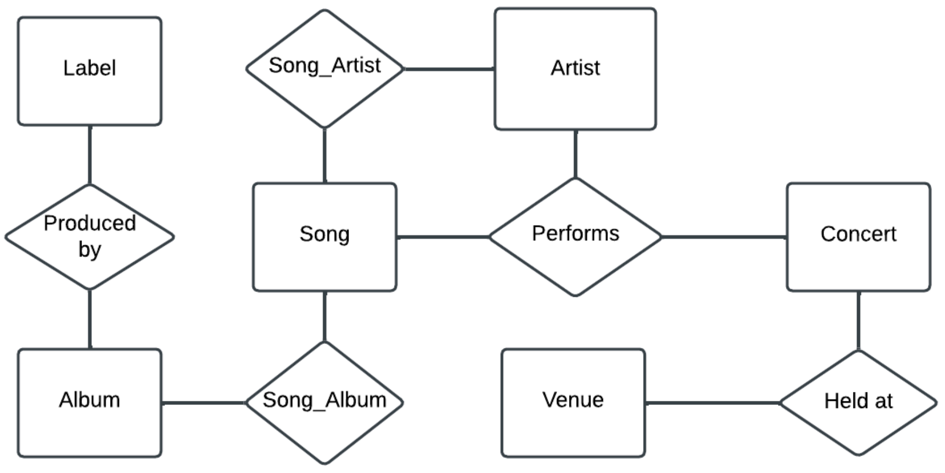

Figure 1: ER diagram, music database

3. Database implementation

The database was initially implemented on PostgreSQL, but later implemented on MySQL.
We started by creating the relevant tables for the entities, relationships for the many to many relationships, as well as identifying relevant foreign keys for the one to many relationships. This was used as the basis for the ER diagram.
The schema was then translated into various python Django models, which were then migrated to a MySql database. Data entry was then performed using python, .csv files and the Django Admin page.

4. Dataset and preprocessing

The data used in this project is real data gathered using ChatGPT.
Requests for data were made using prompts like:

"Can you give me information on Radiohead with the values ArtistName, Country, NoMembers, yearStarted, yearEnded. yearEnded can be 0 if they're still active"

The given data was then placed into various .csv files on VS Code and formatted using python. The new formatted .csv files were then inserted into the database using another python script with statements such as[1]:

```
data = {field: row[field] for field in row if field in [field.name for
field in model._meta.fields]}
model.objects.create(**data)
```

[1]. The full python script is not provided here

5. Queries

Implemented queries are given with the SQL and relational algebra for two of them.

- "What artists have made each song?"
  SELECT a.ArtistName, s.SongName
  FROM artist_song AS asg
  JOIN song AS s ON asg.SongID = s.SongID
  JOIN artist AS a ON asg.ArtistID = a.ArtistID;

$\pi_{A.ArtistName,S.SongName}((ASG \bowtie_{ASG.SongID=S.SongID} S) \bowtie_{ASG.ArtistID=A.ArtistID} A)$

- “Which artist has done the most performances at concerts?”
```
SELECT a.ArtistID, a.ArtistName, COUNT(csa.SongID) as SongsPlayed
FROM concert_song_artist csa
JOIN artist a on a.ArtistID = csa.ArtistID
Group By a.ArtistID, a.ArtistName
ORDER BY SongsPlayed DESC
LIMIT 1;
```

πA.ArtistID,A.ArtistName,SongsPlayed(ρ1(τSongsPlayed DESC(γA.ArtistID,A.ArtistName,COUNT(CSA.SongID)→SongsPlayed(CSA⋈CSA.ArtistID=A.ArtistIDA))))

- What is the longest song played at each concert?
- Who are the artists in the concert with the most artists?
- What is the most performed song?
- What country has the most venues and which artists have played in those venues?
- What is the longest song released by each artist?
- Which artist has performed the most songs at a concert?
- For each country, who is the most performing artist?

6. Fullstack implementation

The Fullstack application was developed using the python web framework Django. The models in the Django implementation were modeled after the schema identified above.

The Django application communicates with a database hosted on MySQL server, on which the Django models were migrated and to which queries were sent and performed.

7. Conclusion

The database and web application were implemented successfully and allow us to interact with them the way we initially intended. The use of real data also made the use of the system more engaging. The project however wasn’t without complications. Modeling an artist group and solo artists as one entity took some reflection. The use of PostgreSQL with Django also caused great confusion. In hindsight, the entities and relationships could have been modeled in a way that would require less joins. This would be a focus point in a re-attempt or similar project.
All in all, the project was fun and interesting and the added task of a web application was a welcome challenge.