EECS 447 Course Project: Goop

Zai Erb, Nicholas Nguyen, Chinh Nguyen

Contents

1 Introduction **2**

2 Project Compoonents **2**
2.1 SQL . . . 2
2.2 PHP backend server . . . 2
2.3 Python scripts . . . 2
2.4 React App . . . 3

3 Requirements Analysis **3**
3.1 Constraints . . . 3
3.1.1 Entity-Relationship Constraints . . . 3
3.1.2 Referential Integrity Constraints . . . 3
3.1.3 Data Integrity Constraints . . . 4
3.1.4 Performance Constraints . . . 4
3.1.5 Consistency Constraints . . . 4
3.2 Operations . . . 4
3.2.1 Create Operations . . . 4
3.2.2 Read Operations . . . 4
3.2.3 Update Operations . . . 4
3.2.4 Delete Operations . . . 4
3.2.5 Search Operations . . . 4
3.2.6 Aggregate Operations . . . 4
3.2.7 Transactional Operations . . . 5
3.2.8 Reporting Operations . . . 5

4 Conceptual Design - ER Diagrams **5**
4.1 Original ER Diagram . . . 5
4.2 Current ER Diagram . . . 6

5 Logical Design - Relational Schemas **7**

6 Project Log **8**

1 Introduction

Goop is an application that allows users to view, save, share, and find songs, albums, DJ Mixes, and artists. Its primary feature is its fine-tuned searching and cataloging of music based on any user-given parameters. The database allows users to see specific information about music such as release date, record label, performing artists, DJ mixes, and more. For example, if a user wants to display every DJ mix that contains a specific song or artist–or if they want to see a record label's discography from a specified timeframe.

Each user account with their profiles and account information are stored in the database. Additionally, users are able to save and organize their music. Users are allowed to follow each other in addition to artists, labels, genres. This service is primarily useful for DJs/performers who are "crate digging" or finding new music to play in their mixes. Being able to fine-tune search results based on adjacent information (such as which mixes a song was played in or the labels an artist is on) in a collection will assist in organizing record libraries in external applications like Rekordbox, Serato, or VirtualDJ.

The concept for Goop was born from frustrations we have had with the existing options for music cataloging. There are lots of options but all of them have significant drawbacks, the biggest of which is that they all have a narrow scope, designed to focus on one specific aspect of music cataloging and categorization, the other issue that is ubiquitous with these sites is a severely outdated, often hard to navigate UI. Rate Your Music and Album of the Year exist for rating, reviewing, and cataloging tracks and albums but neither of them provide tracklists for DJ mixes and while Rate Your Music has more data, depth, and flexibility the UI is far worse than the simpler but cleaner Album of the Year site. For getting tracklists of DJ mixes there are a few options such as 1001Tracklists, TracklistsDB and Tracklists.net all provide similar functionality but their interfaces are outdated and the integration of the tracklists with other information such as the album tracks are from, their release dates, their associated record labels, the genre of the individual tracks etc. For finding events in your area and finding electronic music recommendations Resident Advisor is fantastic however it is limited mostly to electronic music and the mixes posted on the site do not have tracklists. All of this makes for a frustrating and tedious experience for those that want to dig for music using one platform with a database that connects Albums, Genres, Mixes, Labels, Artists, Events and more.

The biggest challenge for creating an application like this is populating the database. A lot of the aforementioned sites rely on community contribution and/or integrate with streaming services such as spotify to ensure new music is added to the database. Seeing as implementing solutions like this would not be feasible for our timeline we relied on scraping data from some of these existing sites to test our database schema and user interface. If we were to continue with this project we would enable to user contributions and could potentially integrate with streaming services to constantly update the database with newly released music.

In this report we will detail what we cover what our app is capable of currently, our methodology including how we constructed our database and schema as well as how we built the UI. We will also cover plans to expand the functionality in the future.

2 Project Compoonents

2.1 SQL

code to create tables and views in database hosted on phpMyAdmin

2.2 PHP backend server

to handle interaction with the database

2.3 Python scripts

for webscraping:

- Scraper to collect data from resident advisor using GraphQL SQL Queries
- Scraper to collect tracklists from tracklist.net
- Script to parse json files containing genres and subgenres into our tables

2.4 React App

- Handle front end for displaying information from the backend
  - Search functionality allows user to query on any of the tables
  - Filters:
  - Varies by page allowing users to implement filters on the page they are on.
- Genre page:
  - Displays genres as clickable dropdown menus to display subgenres belonging to a genre.
- Artists page:
  - Displays artists according to filters set by the user: options to filter by genre, year and label.
- Mixes Page:
  - Displays DJ Mixes according to filters set by the user: options to filter by recommended, genre, year and label.
- Albums Page:
  - Displays Albums according to filters set by the user: options to filter by recommended, genre, year and label.
- Current Features:
  - Interactive pages built on Genres, Albums, Artists, Mixes and Labels tables.
  - Filter and search by album, genre, artist, track, label, year and more.
- Future Features:
  - Users database that allows users to follow other users as well as any other category within the database.
  - Ability for users to add entries.
  - Integration with streaming services to regularly and automatically update the database.
  - Tree view to more easily visualize connections.

3 Requirements Analysis

3.1 Constraints

3.1.1 Entity-Relationship Constraints

- Each user must have a unique username.
- Each artist, genre, and label should have a unique identifier.
- Albums, tracks, and DJ mixes should be uniquely identified.

3.1.2 Referential Integrity Constraints

- Ensure that foreign key constraints are in place to maintain referential integrity. For example, an album should belong to a specific artist, genre, and label.
- Tracks within an album should be linked to the respective album they belong to.
- Users can only follow existing entities in the database (users, artists, labels, and genres).

3.1.3 Data Integrity Constraints

- Implement constraints to enforce data integrity, such as NOT NULL constraints for mandatory fields.
- Define constraints to ensure that publication dates are valid and reasonable.
- Implement check constraints to validate data against specific criteria, such as ensuring publication dates are not in the future.

3.1.4 Performance Constraints

- Consider indexing columns frequently used in search operations to improve query performance.
- Optimize database schema and queries to minimize response times, especially for operations involving large datasets.

3.1.5 Consistency Constraints

- Ensure consistency across related entities. For example, if an artist's name is updated, it should reflect consistently across albums, tracks, and DJ mixes associated with that artist.

3.2 Operations

3.2.1 Create Operations

- Add new users, artists, genres, labels, albums, tracks, and DJ mixes.
- Establish relationships between entities (e.g., users following artists, labels, or other users).

3.2.2 Read Operations

- Retrieve information about users, artists, genres, labels, albums, tracks, and DJ mixes.
- Fetch lists of albums, tracks, and DJ mixes belonging to specific artists, genres, or labels.
- Retrieve the followers/following lists of users and entities they follow.

3.2.3 Update Operations

- Modify user profiles, artist information, genre details, label information, album details, track details, and DJ mix details.
- Update relationships, such as users following/unfollowing other users, artists, genres, or labels.

3.2.4 Delete Operations

- Remove users, artists, genres, labels, albums, tracks, and DJ mixes from the database.
- Delete relationships between entities, such as users unfollowing artists, genres, or labels.

3.2.5 Search Operations

- Search for users, artists, genres, labels, albums, tracks, and DJ mixes based on various criteria (e.g., name, genre, publication date).
- Perform advanced searches, such as finding tracks by artist or album name, or finding users with similar tastes.

3.2.6 Aggregate Operations

- Calculate statistics, such as the number of users following a particular artist, label, or genre.
- Aggregate data, such as counting the total number of albums, tracks, or DJ mixes in the database.

3.2.7 Transactional Operations

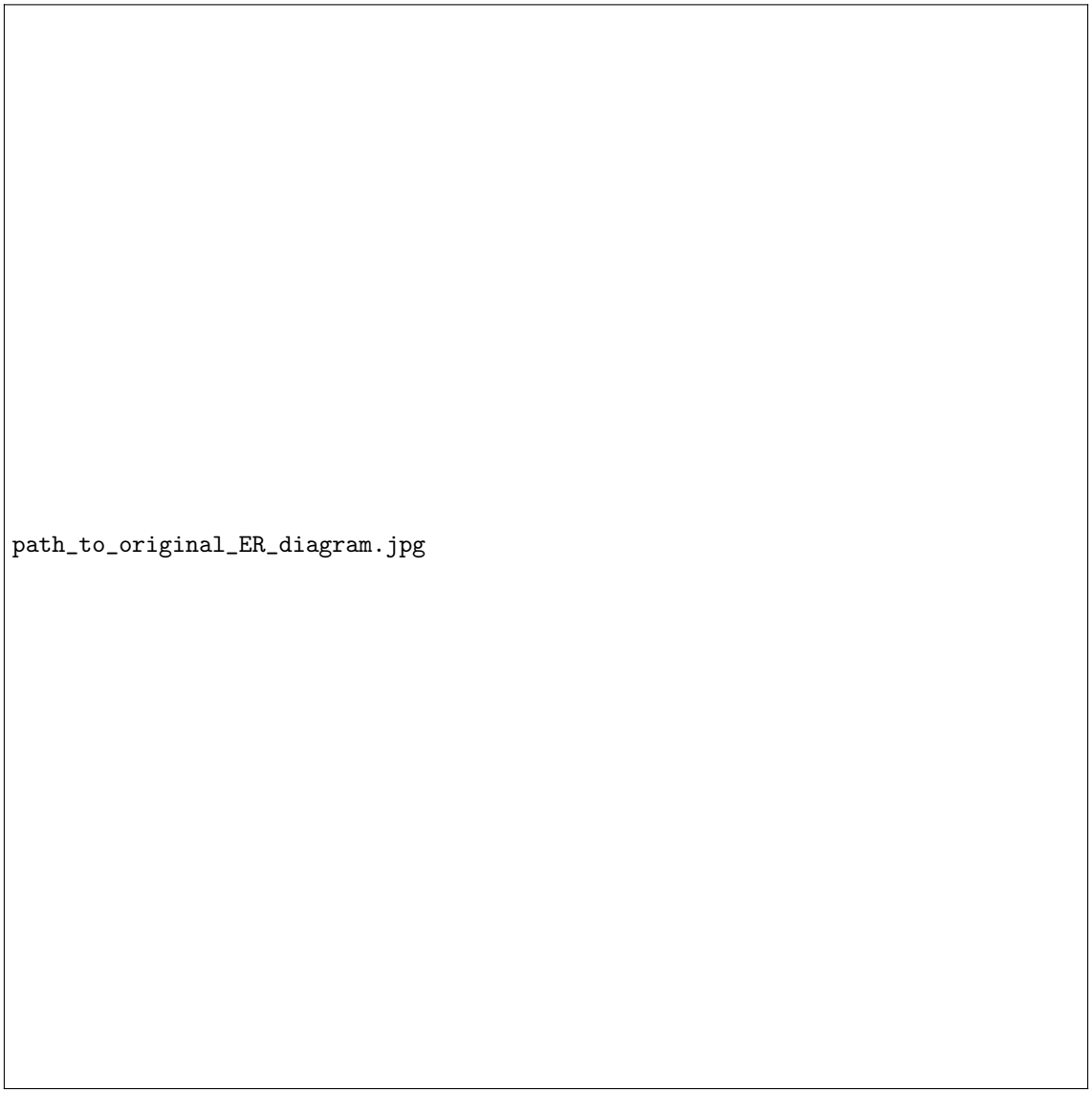
- Execute transactions to maintain data consistency and integrity (e.g., updating multiple entities atomically).
- Ensure that operations such as following/unfollowing entities are performed reliably and consistently.

3.2.8 Reporting Operations

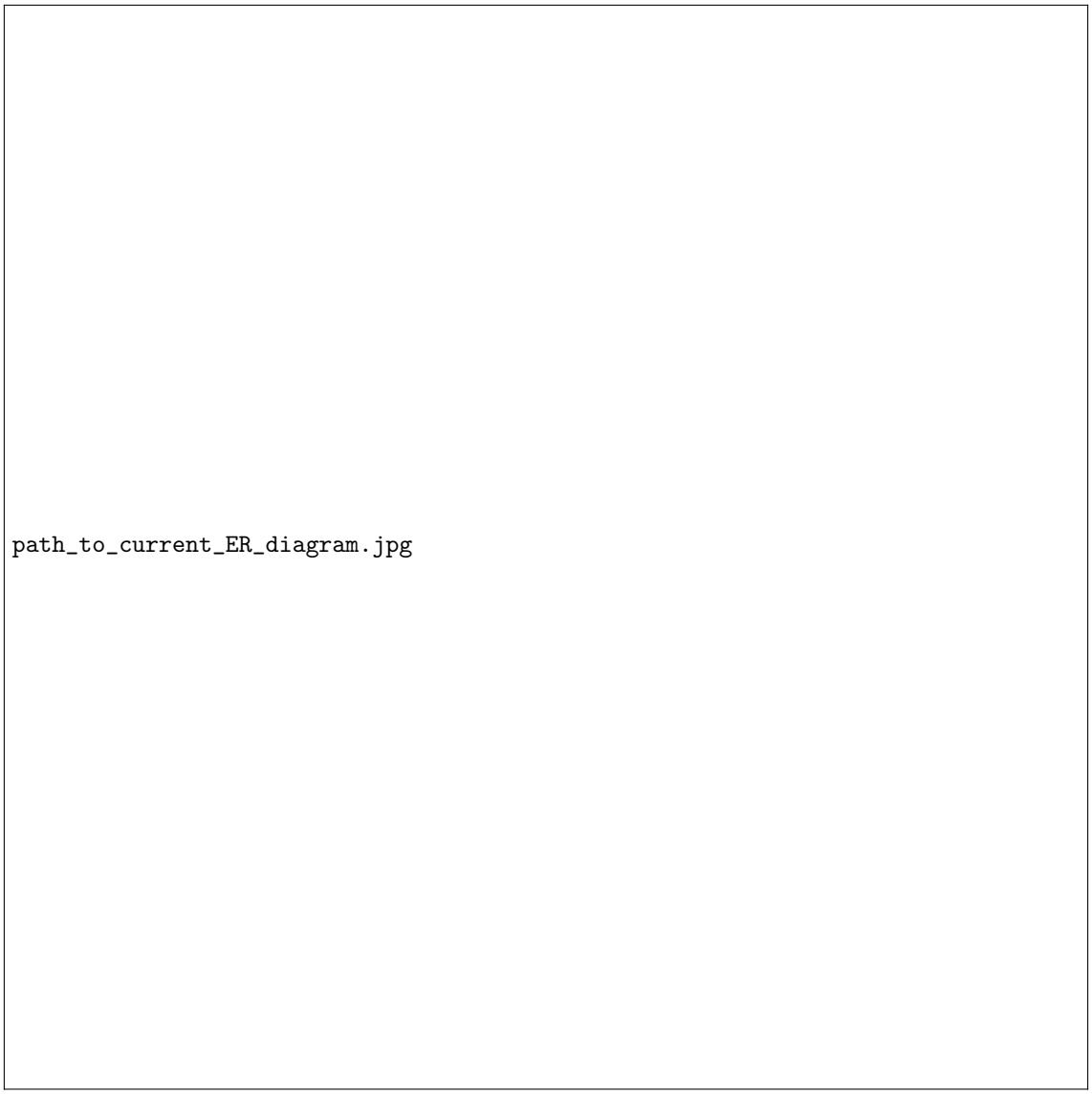
- Generate reports on user activity, popular artists/genres/labels, trending albums, tracks, or DJ mixes.
- Provide insights into user preferences and behavior based on their interactions with the platform.

4 Conceptual Design - ER Diagrams

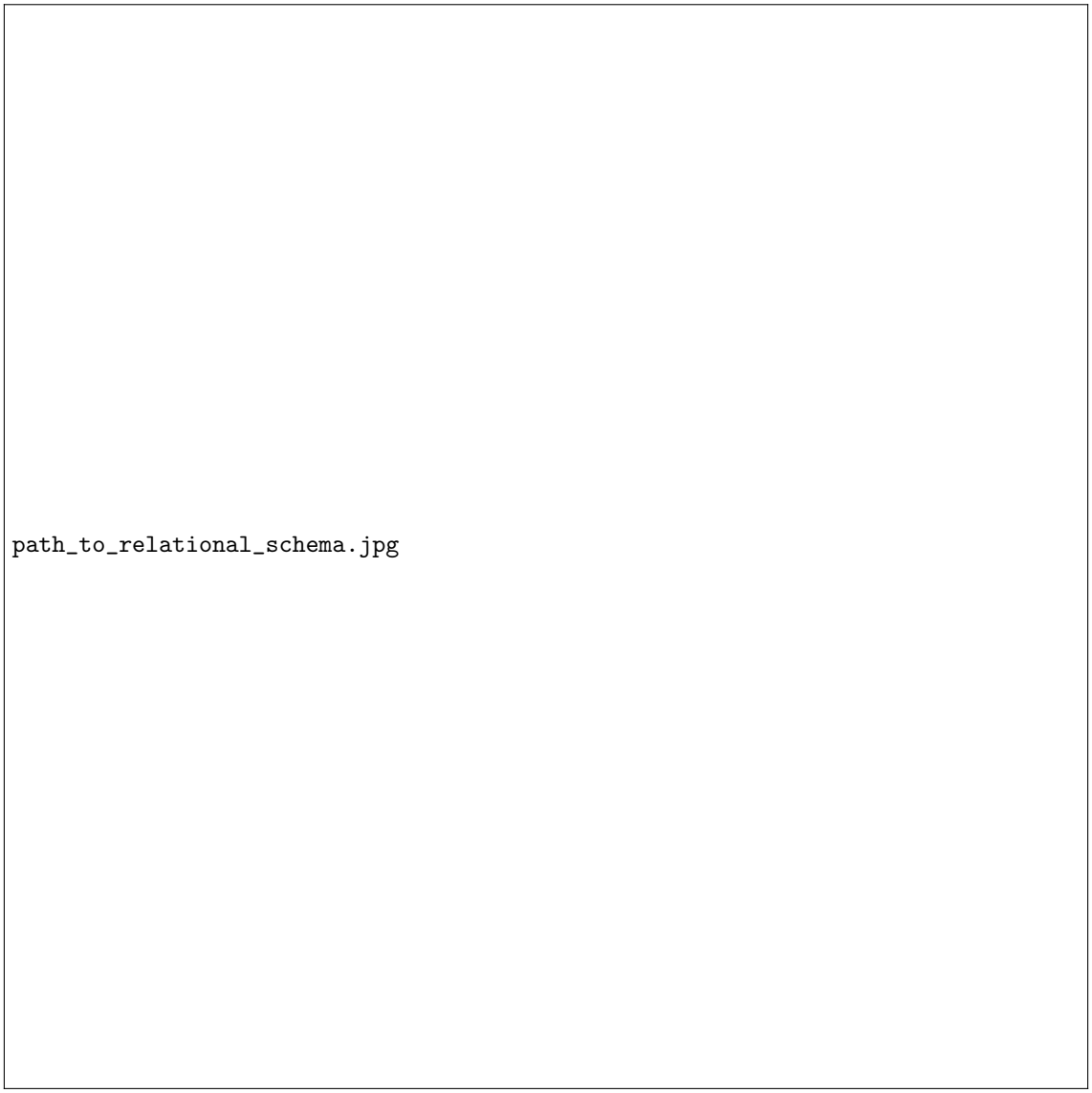
4.1 Original ER Diagram



4.2 Current ER Diagram



5 Logical Design - Relational Schemas



6 Project Log

Author	Date	Message
Zai Erb	2024-05-02	Added SQL views to simpli
Zai Erb	2024-05-02	Improved Scraper
Zai Erb	2024-04-29	Updated Styles
Zai Erb	2024-04-29	Added resident advisor s
Zai Erb	2024-04-18	styling
Chinh Nguyen	2024-04-18	Navigation bar now reflects the current pa
Zai Erb	2024-04-18	Search Styling
Zai Erb	2024-04-17	Styling
Chinh Nguyen	2024-04-17	STYLING!
Chinh Nguyen	2024-04-17	Fixed fetching tracks by artistName, doesn't h
Zai Erb	2024-04-17	Styling
Chinh Nguyen	2024-04-17	Cleaned up CSS, made scrollable
Chinh Nguyen	2024-04-17	Added search for tracks by a
Chinh Nguyen	2024-04-17	Added fetching tracks by ar
Zai Erb	2024-04-17	Updated styling
Zai Erb	2024-04-17	Added GenreClick functionality
Zai Erb	2024-04-17	Added all pages to sort items
Chinh Nguyen	2024-04-17	Implemented search artists b
Chinh Nguyen	2024-04-17	Added 5th query with JOIN: Get artis
Chinh Nguyen	2024-04-17	Results Textarea
Zai Erb	2024-04-17	Genre Components
Chinh Nguyen	2024-04-17	Updated to include Searc
Chinh Nguyen	2024-04-17	Two new queries: Search Tracks based
Chinh Nguyen	2024-04-17	Basic Search Page
Zai Erb	2024-04-17	Added 1001 scraper. Not
Chinh Nguyen	2024-04-17	Added more functio
Chinh Nguyen	2024-04-17	Added Subgenre fetching using INNERJOIN o
Chinh Nguyen	2024-04-17	Subgenre Fetch Function + Dro
Chinh Nguyen	2024-04-17	Added query for Subgenre table: Subgenr
Chinh Nguyen	2024-04-17	Oops. Fixed reading of cor
MistrrrMedia	2024-04-17	Fixed Genre alignment (along with other elements) fixed weird redundancies too sha
Chinh Nguyen	2024-04-17	Necessary Packages Required, Added Localhost with Port 3001
Chinh Nguyen	2024-04-17	Connected to the backend and displays "Genres" table w