SWDV 691 - Professor Gradecki - Benjamin Erkan - 03.26.25

Capstone - Database Design, Updated

Overview

This document supplements the redesign work for the 'Recommend' project by providing an updated database schema and representative SQL queries. These updates reflect the shift to a follow-based user model and expanded discovery features such as genre-based user recommendations and media suggestions from followed users.

Technology Justification

PostgreSQL (via Supabase) was chosen as the database technology. The following are the main factors that lead to the choice:

- The application involves complex relationships between users, media items, genres, and social connections—ideal for a relational database.
- Core features rely on many-to-many relationships, such as users saving multiple media items across different lists, or following many users.
- SQL-based queries allow for efficient generation of personalized recommendation feeds, which involve joining data across multiple tables.
- Supabase provides a hosted PostgreSQL solution with built-in authentication, row-level security, and real-time updates, simplifying integration and access control.

Database Access and Roles

Each authenticated user can read their own data and public content from others. Specific rules:

- Users can only write to their own lists and genre interactions.

- Media items and genres are managed centrally (read-only for users).
- Lists marked as public are readable by all users.
- Follow relationships are only modifiable by the follower.

Updated Database Schema

High-level outline of the updated database schema using PostgreSQL:

Table Name	Description
users	Stores user credentials and profile information (ID, email, name, bio, etc).
media_items	Stores books, movies, or TV shows with associated data such as title, type, genre, date.
lists	Custom lists created by users containing media items.
list_items	Join table linking media_items to specific lists.
follows	Stores follower-followed user relationships (follower_id, followed_id).
genre_interactions	Logs user interactions with genres (user_id, genre_id, interaction_type, timestamp).
genres	List of genre categories.
recommendation_feed	Aggregated content stream for each user derived from followed users and genre-based logic.

Detailed Schema

users

Column	Type	Description
id	ID	Primary key
email	TEXT	Unique user email
name	TEXT	User’s display name
bio	TEXT	Short profile bio
avatar_url	TEXT	URL to avatar image

media_items

Column	Type	Description
id	UUID	Primary key
title	TEXT	Title of the media
type	TEXT	‘book’, ‘movie’, or ‘tv’
genre	TEXT	Genre name (optional FK to genres)
author_director	TEXT	Varies by media type (e.g., director or author)

lists

Column	Type	Description
id	UUID	Primary key
user_id	UUID	Foreign key to users.id
title	TEXT	User-defined list title
is_public	BOOLEAN	Controls list visibility

list_items

Column	Type	Description
id	UUID	Primary key
list_id	UUID	Foreign key to lists.id
media_item_id	UUID	Foreign key to media_items.id
created_at	TIMESTAMP	Timestamp when the item was added

follows

Column	Type	Description
follower_id	UUID	Foreign key to users.id
followed_id	UUID	Foreign key to users.id
created_at	TIMESTAMP	Timestamp when follow was created

Primary Key: (follower_id, followed_id)

genre_interactions

Column	Type	Description
id	UUID	Primary key
user_id	UUID	Foreign key to users.id
genre_id	UUID	Foreign key to genres.id
interaction_type	TEXT	e.g., ‘view’, ‘like’, ‘save’
timestamp	TIMESTAMP	When interaction occurred

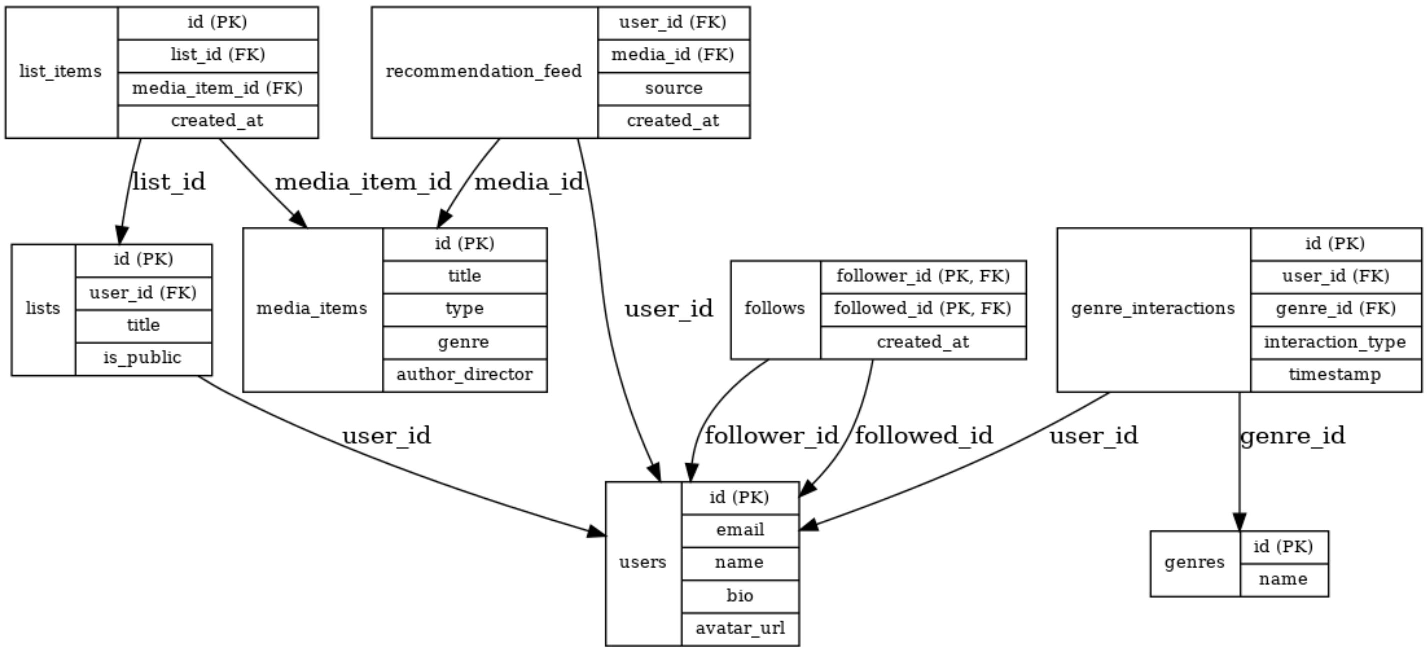
genres

Column	Type	Description
id	UUID	Primary key
name	TEXT	Genre label

recommendation_feed

Column	Type	Description
user_id	UUID	Foreign key to users.id
media_id	UUID	Foreign key to media_items.id
source	TEXT	‘followed_user’, ‘genre_match’, etc.
created_at	TIMESTAMP	Timestamp when recommendation was added

Entity Relationship Diagram



Sample Queries

Find media recently saved by followed users

```
SELECT media_items.title, media_items.type, media_items.genre,
users.name

FROM follows

JOIN lists ON lists.user_id = follows.followed_id

JOIN list_items ON list_items.list_id = lists.id

JOIN media_items ON media_items.id = list_items.media_item_id

JOIN users ON users.id = follows.followed_id

WHERE follows.follower_id = :current_user_id

ORDER BY list_items.created_at DESC

LIMIT 10;
```

Recommend users with overlapping genre interests

```
SELECT DISTINCT users.id, users.name, COUNT(*)

FROM genre_interactions AS interactions_1

JOIN genre_interactions AS interactions_2 ON interactions_1.genre_id =
interactions_2.genre_id

JOIN users ON users.id = interactions_2.user_id

WHERE interactions_1.user_id = :current_user_id

  AND interactions_2.user_id != :current_user_id

  AND users.id NOT IN (

      SELECT follows.followed_id FROM follows WHERE follows.follower_id
= :current_user_id
```

```
  )
GROUP BY users.id, users.name
ORDER BY shared_genres DESC
LIMIT 5;
```

Get list of media in user's public list

```
SELECT media_items.title, media_items.genre, lists.title
FROM lists
JOIN list_items ON list_items.list_id = lists.id
JOIN media_items ON media_items.id = list_items.media_item_id
WHERE lists.user_id = :target_user_id
  AND lists.is_public = TRUE;
```