SOEN 363: Data Systems for Software Engineers

Final Report: Books Database Project

Project Overview

The Books Database project is an exercise in designing, implementing, and querying a relational database system that captures details about books, including their authors, genres, and publication data. The main objective of this project was to construct a database capable of efficiently handling and organizing data retrieved from external APIs, showcasing the practicality and versatility of relational databases in managing real-world data. The database not only stores information but also supports complex queries and demonstrates advanced SQL concepts through its implementation.

Database Design and Schema Details

The design of the database was driven by the need to achieve normalization, maintain data integrity, and facilitate efficient querying. Key elements of the design include:

- **Entity Relationships:** The database schema is structured to highlight the relationships between major entities such as Publications, Books, Authors, and Genres. The Entity-Relationship Diagram (ERD) clearly outlines these associations, making sure it is a logical and intuitive design.
- **IS-A Relationship:** The Books table represents an IS-A relationship with the Publications table, allowing books to inherit common attributes like title, description, and language while adding specific attributes such as ratings and review counts.
- **Normalization**: The schema is fully normalized up to the third normal form (3NF), eliminating redundancy and ensuring consistency across the database.
- **Weak Entities:** Certain relationships like BookGenres are implemented as weak entities, linking publications to their genres through a foreign key relationship.
- **Constraints and Referential Integrity:** Primary keys, foreign keys, and check constraints are implemented to maintain the integrity of the data. Complex referential integrity is enforced through triggers, making sure that changes in one table cascade appropriately to related tables.

Data Sources and Integration

To populate the database, data was sourced from the following public APIs:

1. **Google Books API:** This API provides a wide range of metadata about books, including their titles, descriptions, authors, categories, and ratings. The API also supplies unique ids such as Google Books IDs and ISBNs which are crucial for linking records.
2. **Open Library API:** Supplementary information, including author biographies, detailed publication dates, and page counts, was retrieved from the Open Library API. This data was integrated into the database by mapping ISBNs between the two sources.

The integration process involved handling discrepancies between the data structures of the

two APIs. For example, while Google Books API provides author names as plain text, Open Library API offers additional details like birth dates and biographies.

Implementation Process

The implementation of the Books Database project involved multiple steps, each addressing a critical aspect of database design and functionality:

1. Database Schema Creation:

- The schema was designed using PostgreSQL, incorporating tables for Publications, Books, Authors, Genres, and their relationships. The schema was made in the `dbcreation.sql` script, which includes detailed instructions for creating tables, keys, constraints, and indexes.

2. Data Population:

- A Python script (`script.py`) was developed to fetch data from the APIs, process it, and insert it into the database. The script includes functionality for cleaning and validating data, such as making sure that ratings are within a valid range (0-5) and handling missing values.

3. Data Integrity Mechanisms:

- Triggers were implemented to maintain referential integrity, such as updating related records when a primary record is modified. For example, when a book's genre is updated, the corresponding entries in the BookGenres table are automatically adjusted.

SQL Queries and Demonstrations

The functionality and robustness of the database were tested using a variety of SQL queries, which also demonstrate advanced database concepts:

- Basic Queries: Simple SELECT statements with WHERE clauses were used to filter data based on attributes such as genre, rating, and publication year.

- Join Operations: Various join types (INNER, LEFT, RIGHT, FULL) were applied to retrieve data spanning multiple tables, such as listing all books along with their authors and genres.

- Aggregations: GROUP BY and HAVING clauses were utilized to calculate statistics, such as the average rating of books in each genre or the total number of books by an author.

- Null Handling: Queries were crafted to handle NULL values which makes sure that incomplete data did not lead to inaccurate results.

- Set Operations: UNION, INTERSECT, and EXCEPT queries were used to identify overlapping or distinct records between different subsets of data.

- Nested Queries: Not only correlated but also uncorrelated subqueries were used to derive sophisticated results like finding the highest-rated book in each type of genre.

- Division Operations: One way to search for the desired information is division queries, implemented by using NOT IN and NOT EXISTS, which allow getting records meeting the multiple criteria (complex) that are actually the operations with the division subqueries. Such statements may be formed by utilizing division subqueries introducing all.

- Views: The database supports well-defined views like full_access_books and low_access_books, which restrict the columns and rows shown or exposed based on <user level>. These views exhibit the additional use of hard-coded conditions for access checks.

Challenges and Solutions

Numerous challenges appeared throughout this project. Nevertheless, we successfully went through the obstacles of a project through these creative solutions:

1. Data Integration: Combining data from different APIs with various structures was a huge stumbling block. For example, Google Books API and Open Library API utilize distinct identifiers for the identical book.

- Solution: A mapping mechanism was implemented in the Python script to align records based on ISBNs, ensuring accurate integration.

2. Rate Limits and Pagination: The APIs set limitations on the number of requests that could be completed in a specific period.

- Solution: The script was designed to handle pagination and include respectful delays between requests to avoid exceeding rate limits.

3. Data Validation: Making sure the data is consistent and correct, especially when dealing with optional fields like publication dates and ratings.

- Solution: Validation rules were incorporated into the script, such as rejecting future publication dates and ensuring ratings fall within the 0-5 range.

Future Prospects and Enhancements

The Books Database project serves as a foundation for future developments. Potential enhancements include:

- Advanced Analytics: Introducing recommendation algorithms to suggest books based on user preferences

- Front-End Development: Users should be able to have the possibility of building up a web application as the final result.

- Integration with Additional Data Sources: Extending the database to contain information from APIs that contribute reviews, sales statistics, or users' demographics.

Conclusion

The Books Database project displays the real-world use of relational database principles in dealing with and getting information from complicated data. The project, through combining data from multiple APIs and the use of advanced SQL, shows the potential of PostgreSQL and Python in the development of a strong and scalable database system. The Project phase 1 work has been completed successfully, but this project is the starting point for possible future extensions and real-world applications.