SOEN 363: Data Systems for Software Engineers

Final Report: Books Database Project

Project Overview

The Books Database project is an in-depth exercise in database management and migration, aiming to showcase the application of relational and NoSQL database systems to efficiently handle real-world data. The project's scope includes designing and implementing a relational database using PostgreSQL (Phase 1) and later transitioning to MongoDB for flexibility and scalability (Phase 2). This end-to-end approach demonstrates the adaptability of database systems to evolving data needs.

Phase 1: Relational Database Design and Implementation

Objective

To create a normalized relational database system that organizes and queries information about books, their authors, genres, and publications. The system was designed to integrate data from external APIs, focusing on practical database principles and advanced SQL features.

Database Design and Schema

- **Entity Relationships**: The schema linked entities like Publications, Books, Authors, and Genres, ensuring logical and intuitive associations.
- **Normalization**: The database was normalized to 3NF, eliminating redundancy and maintaining data consistency.
- **IS-A Relationship**: Books were modeled as a specialization of Publications, inheriting shared attributes and adding unique ones.
- **Constraints**: Referential integrity was enforced using primary and foreign keys, along with triggers for cascading updates.

Data Sources and Integration

- **Google Books API**: Provided metadata such as titles, descriptions, and ratings.
- **Open Library API**: Supplemented data with author biographies and detailed publication information.

Implementation Highlights

1. **Schema Creation**: Developed using PostgreSQL scripts defining tables, keys, and constraints.
2. **Data Population**: Python scripts fetched, cleaned, and validated data from APIs, ensuring accurate integration.
3. **Query Demonstrations**: SQL queries showcased database capabilities, including complex joins, aggregations, and subqueries.

Phase 2: Migration to MongoDB

Objective

To transition from a relational model to a document-oriented NoSQL model, addressing the need for scalable design and faster queries for large datasets.

Migration Process

1. **Data Extraction**: Retrieved data from PostgreSQL using SQL queries.
2. **Transformation**: Reformatted data into JSON-like structures suitable for MongoDB.
3. **Embedding Relationships**:
    - **Genres**: Migrated as a collection with `genre_id` as `_id`.
    - **Authors**: Migrated with `author_id` as `_id`.
    - **Books**: Embedded genres and authors directly for efficient querying.
4. **Batch Insertion**: Inserted data in batches of 1,000 for optimal performance.

Implementation Steps

- Established connections to PostgreSQL and MongoDB.
- Developed Python scripts for data migration, handling schema transformations and maintaining data integrity.

Key Benefits

- **Simplified Queries**: Embedded relationships reduced the need for joins.
- **Scalable Design**: MongoDB's document-oriented structure supported larger datasets.
- **Improved Performance**: Optimized schema and batch processing enhanced query speed.

Challenges and Solutions

1. **Data Integration**:
    - **Challenge**: Aligning distinct identifiers from APIs.
    - **Solution**: Implemented ISBN-based mapping for accurate data merging.

2. **Rate Limits**:
    - **Challenge**: API request limits and pagination complexities.
    - **Solution**: Scripts handled pagination and incorporated delays to stay within limits.
3. **Data Validation**:
    - **Challenge**: Ensuring consistent data for optional fields like ratings.
    - **Solution**: Validation rules were integrated into scripts to reject invalid entries.

Project Insights

This project highlights the evolution of database management from traditional relational systems to modern NoSQL models.

- **Phase 1 Achievements**:
    - Built a comprehensive relational database that integrated data from multiple sources.
    - Demonstrated advanced SQL features, including joins, aggregations, and subqueries.
    - Ensured data integrity and normalization, making it ideal for structured queries.
- **Phase 2 Transition**:
    - Adapted to the growing need for scalability and faster access.
    - Emphasized efficient data organization by embedding relationships directly into documents.
    - Improved query performance and reduced complexity for developers.

The Books Database project exemplifies a seamless blend of database design principles, practical problem-solving, and technological adaptability. It serves as a testament to the power of combining relational and NoSQL models to meet diverse data management needs.

Conclusion

The Books Database project demonstrates the application of database principles in managing complex datasets. Phase 1 illustrated the capabilities of relational databases, while Phase 2 highlighted the advantages of migrating to a NoSQL model for scalability and performance. This comprehensive project serves as a foundation for further enhancements, showcasing the synergy between relational and document-oriented database systems.