System Architecture Design

## Anthony Combs

7/31/2023

**Introduction and Problem Definition**

In an increasingly digitized world, libraries stand as one of the earliest forms of knowledge sharing and hold a vital role in today's information landscape. However, the management of a library is a complex process, encompassing not only the tracking, lending, and returning of books but also maintaining a detailed inventory of books, authors, publishers, genres, and other related information.

Traditional manual methods of tracking these processes are time-consuming and prone to error, particularly as the size of the library increases. The possibility of human error, such as mis recording a transaction or failing to update an item's status, can lead to significant inconsistencies over time. These challenges highlight the need for a comprehensive and automated system that can streamline library management processes and ensure accurate and up-to-date tracking of library transactions.

This is why I though it would be a good idea to create the Library Database Management System – a tailored solution designed to handle the unique requirements of library management. Leveraging the power of relational databases and SQL, this system aims to automate library processes and improve data integrity. The system not only caters to the library's primary requirement of tracking book lending and returns but also includes advanced features such as managing book reviews, handling complex book-genre and book-author relationships, and facilitating staff and patron data management.

The following sections will delve into the specific design considerations, system architecture, and implemented functionalities of this Library Database Management System. The report will also evaluate the system's reliability, scalability, and maintainability, ensuring it stands as a robust solution capable of effectively supporting a library's operations.

**Solution Approach**

The architecture of this Library Database Management System (DBMS) relies on the principles of Relational Database Management Systems (RDBMS) and the use of Structured Query Language (SQL). The approach I used was to create a database model that accurately reflects the real-world system it represents, including the entities (e.g., books, publishers, users, staff) and their relationships.

First, I designed a comprehensive ER (Entity-Relationship) diagram to model the complex relationships between the entities. The key entities I identified were Books, Authors, Publishers, Genres, Users (Patrons), Staff, BookCopies, BookLoans, and BookReviews. The relationships between these entities were established through identifying one-to-many, many-to-many, and one-to-one associations as needed, resulting in additional tables like BookAuthors and BookGenres.

I then translated this ER model into a relational schema, creating a structured database. One design principle of this database I used was normalization to reduce data redundancy and improve data integrity.

**Adding Constraints, roles, and testing**

Each entity was represented as a table in the SQL database with fields that correspond to the entity's attributes. Constraints were applied to fields to enforce data integrity and prevent invalid data entry. These constraints included, among others, primary and foreign key relationships, 'NOT NULL' constraints, and check constraints to maintain valid data ranges. Along with these constraints I also added Security and access controls. I did this by implementing a user role with specific permissions to access data allowing the database to prevent unauthorized actions. Lastly, I populated the database with sample data, considering real-world scenarios that the system would encounter. This step was crucial to facilitate testing and ensure the system could handle the library's daily operations. In the next sections I will dive deeper into the design process explaining the Reliability, Scalability and Maintainability of this design.

**System Design**

The fundamental components of this system are tables the tables: Books, Authors, Publishers, Genres, Users, Staff, BookCopies, BookLoans, and BookReviews. These entities encompass the main functions and interactions within a library.

Each table contains fields representing the attributes of the corresponding entity. The 'Books' table, for instance, has fields like 'Title', 'PublicationYear', and 'PublisherID'. These fields capture the properties of a book that the system needs to store and manipulate.

Each entity's identifier, like 'BookID' or 'AuthorID', serves as the primary key in the corresponding table. This primary key is unique for each record, ensuring that every piece of data can be precisely identified.

The relationships between entities are represented using foreign keys. A foreign key in one table references the primary key of another, linking records across tables. For example, the 'BookID' field in the 'BookCopies' table is a foreign key that references the 'BookID' primary key in the 'Books' table. This relationship allows tracing every book copy back to the specific book.

Certain associations between entities required me to use intermediary tables due to the nature of their relationships. For instance, a book can have multiple authors, and an author can write multiple books. This many-to-many relationship is represented using the 'BookAuthors' table, which includes two foreign keys: 'BookID' and 'AuthorID'.

Similarly, to represent the various genres a book can belong to, I used the 'BookGenres' table. This table enables storing a book's multiple genres without repeating the other details of the book.

To further ensure the reliability of our DBMS, we have included constraints at the table and field levels. Constraints like 'NOT NULL' ensure that certain critical fields must always have a value. Check constraints enforce specific conditions on the values that a field can have. For example, we ensured that the 'Rating' field in 'BookReviews' is always an integer between 1 and 5 as well as ensuring each user can only leave one review for a book.

The power of this RDBMS comes into play when we want to extract information from our database. Complex SQL queries can be used on the DB to pull data from multiple tables based on various conditions. It can also perform aggregate calculations, such as finding the average rating of a book or the total number of loans by a user.

Finally, the data security and integrity are maintained by creating different user roles and granting them appropriate privileges. This restricts unauthorized access and modifications to the database.

**Experiment Setup and Results**

To demonstrate the effectiveness of the Library Database I conducted a series of experiments. The aim was to test both the functional and non-functional aspects of the system, encompassing areas such as data integrity, query performance, and system scalability.

The setup for these experiments involved creating and populating the database tables using SQL scripts. I simulated a real-world library scenario by inserting records into each table. The test data covered all possible relationships between entities and ensured a wide distribution of values across all attributes.

The experiments involved the execution of the SQL queries provided inside ***SampleQueries.docx***. These queries were designed to test a wide range of database operations, including simple data retrieval, complex joins, aggregate calculations, and subqueries. They also covered various aspects of SQL such as the creation of views, use of functions and procedures, and trigger execution.

All queries executed successfully, demonstrating that the DBMS could handle a variety of data manipulation and retrieval operations. The results returned by the queries were as expected, indicating the integrity of the data and the correctness of the relationships between entities.

**Conclusion**

In conclusion, this Library Database Management System exhibits performance in the key aspects of Reliability, Scalability, and Maintainability. Its robust design ensures high reliability, executing complex operations without errors and maintaining data integrity under all tested conditions. The scalability of the system is demonstrated by its ability to handle increasing data volumes without a significant impact on performance. The maintainability aspect is manifested in the simplicity of the design and the use of standard SQL for operations, ensuring that the system can be easily managed and modified as needed. With these capabilities, I believe this system provides a comprehensive solution that efficiently handles the diverse data management needs.