Introduction

Our team has proposed the creation of a MySQL database project - A Personal Library of Books to help individuals overcome the challenge of managing a personal library of books. The database will assist book lovers in managing various book-related tasks, namely keeping track of books they own, maintaining a history of book purchases, and managing a book review diary. Additionally, it makes it easier to keep track of who has borrowed their books.

This report will discuss the benefits of the Personal Library of Books database, its target audience, and the questions it can answer and include a detailed analysis of the database's features, how it works, and how it can be accessed. This database is a practical solution for individuals who want to manage their personal book collections more efficiently and make the most of their reading experience. Users can streamline their book-related tasks and share their reading experiences with others. It is a valuable tool for anyone who loves books and wants to organize their book collection effectively.

Database Design and Implementation

- Logical Design

  The purpose of this database is to enable the library owner to efficiently manage their collection of books and keep records of book loan transactions with friends and other contacts. Entities are:

  - Book: represents a book in a library. Contain attributes such as Title, Price etc.
  - Borrower: represents a person who has ever borrowed a book or can borrow a book in the future. Contain attributes such as First Name, Last Name, Contact No.,etc.
  - Author: represents a person who has written a book. Contain attributes such as First Name, Last Name, Nationality etc.
  - Language: represents a language in which a book is written.
  - Genre: represents the kind of book or the category to which a particular book belongs to.
  - Nationality: represents the country or nationality of an author or borrower.
  - Publisher: represents a company or organization that publishes the book.

- Relationship: represents a type of relationship borrower shares with the library owner such as Friend, Relative, Neighbour etc.
- Book Condition: represents the physical condition of a book, such as new etc.
- Book Cover Type: represents the type of cover material used for a book, which is either a hardcover or paperback

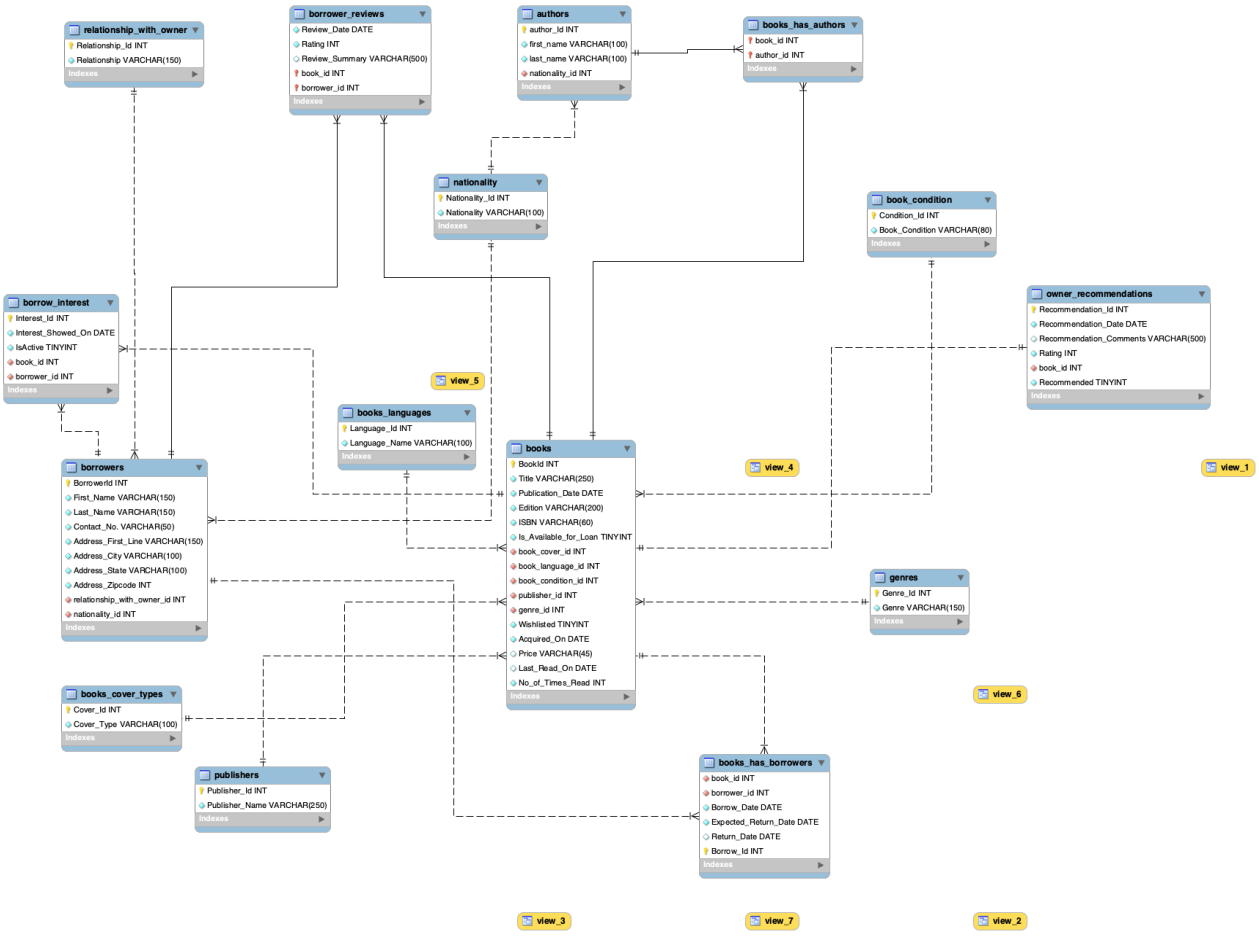
Relationships and Cardinalities

- Many to many relationship between Books and Authors. A book can have multiple Authors and likewise a single Author may have written multiple books. To resolve this, ERD includes a junction table called 'books_has_authors'.
- Many to many relationship between borrowers and books. For this, there is a junction table called 'books_has_borrowers'. There is another table which highlights this relationship between books and borrowers - 'Borrower_Reviews' which stores the reviews given by the borrowers for the books borrowed.
- One to one relationship between Book and Owner recommendation. Owner will write only one recommendation for each book.
- Many to one relationship between books and language. Many books can be written in one language but one book will be written in only one language. Similarly there is many to one relationship between books and genres
- Many to one relationship between books and publishers.
- Many to one relationship between borrowers and a relationship. For example, many borrowers can be Friends.
- Many to one relationship between borrowers and nationality and similarly many to one relationship between authors and nationality which means that multiple authors can have the same nationality and multiple borrowers can have the same nationality.

Database/ERD Assumptions

- One person (Author, Borrower) to have one nationality
- One person can borrow multiple books but the borrower can review one book only once
- Each borrower has only 1 type of relationship with the library owner.
- Borrow_Interest table is to capture the interest of people who want to borrow a book which is currently borrowed by someone else. If 2 people show interest, the person with the earlier 'interest_showed_on' date will get the book.

- 1 book to have only 1 genre



- **Physical Database**

We created the physical database by forward engineering the Entity Relationship Diagram in MySQL Workbench. It was a convenient way and ensured that all respective constraints are implemented.

- **Sample Data**

Our database consists of both true and hypothetical data. For instance, the names of books and their respective authors are real whereas the names of borrowers, their reviews and their relationship with the owner of the library is fictional. We did not import data from any pre-built database. We manually typed the data in MySQL Workbench. It has an efficient way of allowing the user to type data directly in the table rows and columns. When we save and apply the changes, the application automatically generates the corresponding INSERT and UPDATE queries and we can commit the changes.

- **Views Matrix and Requirement Grid**

View	Req A	Req B	Req C	Req D	Req E
View 1	Yes	Yes	Yes		
View 2				Yes	Yes
View 3	Yes	Yes	Yes		Yes
View 4	Yes	Yes (LIMIT keyword is used)	Yes		Yes
View 5	Yes	Yes	Yes		Yes
View 6	Yes	Yes	Yes		Yes
View 7	Yes	Yes	Yes		Yes

Specifications of each VIEW query:

View 1 - What is the list of all borrowers who have borrowed at least one book from the library and have a relationship type of "friend" with the owner of the library?

View 2 - What is the list of top 6 most read books (by the library owner) and their corresponding authors in the library?

View 3 - List the books whose average rating by borrowers is equal to or greater than the rating given by the Library owner?

View 4 - Which book is borrowed the most and which genre does it belong to?

View 5 - Display the book that has received the most expressions of interest for borrowing, along with a list of all the people who have expressed interest and whose interest is still active, and the details of their interest such as the date they showed interest, as well as their relationship with the library owner?

View 6 - What is the list of authors who have multiple books rated by the owner, but the owner's average rating for those books is less than 7? Include the author's name, the number of books rated, the average rating, and the author's ID?

View 7 - What is the list of languages in the library and the number of books available for loan in each language?

Changes From Original Design:

In our Project Proposal, we had proposed 10 tables for our database, namely, Books, Book_Sources, Genres, Authors, Reading_History, Reviews_by_borrowers, Owner_wishlist, Library_Owner_recommendations, Loans and borrowers.

However, in our final project submission, we have a total of 15 tables, which are book_condition, authors, relationship_with_owner, borrowers, books, books_cover_types, borrow_interest, books_has_borrowers, books_has_authors, nationality, owner_recommendations, borrower_reviews, books_languages, genres and publishers. The increase in number of tables and the purpose of those respectively is attributed to normalization to 3NF of the tables stated in the project proposal and review and feedback from the professors and the teaching assistant.

After receiving feedback and deliberating on the Draft ERD and our progress report, we decided to remove Owner_reading_History and Books_Sources tables. For attributes such as 'recommended' in Owner_recommendations, we changed the data type to tinyint instead of bit.

Issues Experienced During Development

- a. Issue encountered: The challenge of avoiding duplicate primary keys during manual data entry was exacerbated by the large volume of data.

  b. Solutions considered: We evaluated practical solutions, such as using database constraints to prevent duplicates, organizing data with spreadsheets prior to entry, employing meticulous data entry practices to verify the data before insertion.

  c. Solutions chosen: We decided to implement validation checks and constraints within the database, as well as use spreadsheets for data organization. These methods were chosen to prevent duplicates and for streamlining the data entry process.
- a. Issue encountered: Ensuring valid foreign key entries when primary keys were used as foreign keys in other tables.

  b. Solutions considered: We examined practical approaches, such as setting up database constraints for automatic referential integrity enforcement, manual cross-referencing and validation, and scripting an automated validation process.

c. Solutions chosen: We opted to set up foreign key constraints within the database, as this method efficiently maintained referential integrity and prevented invalid foreign key values, keeping table relationships consistent.

- a. Issue encountered: We faced challenges with typing errors in column names and incorrect data types, such as adhering to the YYYY-DD-MM date format.

  b. Solutions considered: We evaluated practical solutions, including using validation rules or check constraints, maintaining detailed schema documentation with spreadsheets, and implementing a two-person verification to ensure accuracy in data entry.

  c. Solutions chosen: We implemented validation rules and check constraints, and used a spreadsheet for schema documentation to effectively ensure correct data entry and removing errors in column names and data types.

Lessons Learned

While working on our project, we faced a few challenges & also learned some valuable lessons: The first lesson we learned was the importance of proper memory allocation while loading data into the workbench. We initially assigned strict memory allocations to each variable, but during the data loading process, we realized that some of the memory allocations were too tight, causing difficulties. We had to go back and reallocate the memory of most variables to accommodate the data comfortably, which was a time-consuming process. This experience taught us to be more cautious when assigning memory allocations in the future. We also faced challenges while importing data through a CSV file on Mac, which taught us the importance of having a backup plan. After spending a few hours trying to diagnose the issue, we decided to manually update the tables instead of wasting more time on the problem. This experience showed us the importance of being adaptable and flexible when working on projects. Finally, we encountered an issue while exporting data from the workbench. One team member had named the database "Personal Library," which caused problems when exporting the data due to a known issue in the workbench model on Mac. [Issue was that if the database name has any capital letter, workbench does not give an option to export that particular database] .We had to redo the data loading for at least 10 tables because of this issue. However, we learned a valuable lesson from this experience and became more cautious while naming our database in future attempts.

Future Scope:

Enhancements within the current technology and design (MySQL):

As the scope increases, following enhancements steps should ideally be taken -

- Integrating Payment System to enable the borrowers to pay the rent and owner to be able to manage the cash flows.
- User Interface: There should be a web-based application or mobile application that interacts with the database. This would allow users to browse the library catalog, search for books, add reviews, and more.
- Security: security of the database can be improved upon by implementing additional authentication and authorization mechanisms, such as using SSL/TLS encryption for database connections, or implementing role-based access control to limit access to sensitive data.
- Automation: certain tasks should be automated through events and triggers, such as generating reports or sending notifications when a book is due for return.

Feasibility of alternative implementations:

As the scope of our project grows and the number of books in our library expand, the amount of data will tremendously increase and the following problems may crop up -

- It may become slower and less responsive, queries may become slower. As a user interface gets built and every user gets an opportunity to interact with the database in some form, problem of performance & security and may become a pressing concern

But, considering the next immediate advancement should be having a payment system integrated in our project, it would be essential to ensure transactional consistency and reliability, which is best achieved through a relational database solution. However, to address the potential scalability and performance issues associated with relational databases, a combination of both relational and non-relational databases can be used.

We think we could use a relational database like MySQL to manage transactional data, such as user accounts, payment transactions, and library inventory, while a non-relational database like MongoDB or Cassandra can be used to store data such as user activity logs, user preferences, or user-generated content like reviews.