University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

Essay / Assignment Title: **Database System Design for a Library**

Programme title: **Enterprise Data Warehouses and Database Management Systems**

Name: **Kanchan Bansode**

Year: **2024**

CONTENTS

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

Kanchan Bansode

………

Date: 04/06/2024

1. INTRODUCTION

In the world we live in now where data rules the day knowing how to set up control and ask questions to databases is key for any organization to thrive. This paper takes a deep dive into the basic ideas behind database management systems and how to use them to create a database for a public library with the help of MySQL. First it sheds light on the essential elements of creating managing and overseeing databases. Next it molds into the art of extracting data for business insights. Lastly it focuses on enhancing skills in crafting complex SQL queires for the analysis and manipulation of extensive data collections. By working on this project, I will undertake the construction of an Entity-Relationship (ER) diagram. Following this step the database will be brought to life in MySQL. I have also created two store procedures and some complex queries that showcases my SQL skills.

This system handles everything from keeping track of all the books to who borrows them and even the library's daily workings. This system is like a big web made up of different parts. There are bits for the writers, the books, who published them, where they're kept, the library users, the different library locations, the people working there, and all the borrowing that goes on. Plus, it has special links that connect authors and the records of who did what. All this works together to make the library run better, work faster, and feel easier to use. On top of that, it's a big help for the librarians when they need to dig deep into the numbers.

2. ERD DIAGRAM-ENTITY & RELATIONSHIP

ERD (Entity relationship diagram) is an initial stage and practical method commonly applied in the field of database design aimed at data organization. The ER modeling has its key components which comprises of entities, attributes, relationships, primary key, cardinality and foreign keys. All of the entities are described by a set of features which are called attributes, including name and ID numbers, contact information, and other specifics important for the functioning of the entities integrated into the library environment.

Later different entities are related to each other, creating relationships with the help of primary keys and foreign keys (marked as FK), cardinality (1:1) or (M:M) meaning a many to many relationships. More often than not they comprise of factors such the magnet/optional characteristic of the participation, also something about weak entities and a process of generalization and specialization.

ENTITIES

- Author: Refers to people who compose books or authors of books. Has fields like author_id, name, email and contact_no to store information of authors.
- Books: Stores all the information about each book that is in the library such as book_id, title of book, genre, edition number, publisher_id, location id and availability.
- Publisher: Has information concerning the different publishers who do publish books, with the information including; publisher_id, name, address and contact details.
- Location: Proper physical locations within the library that hold books are defined with location_id, shelf_no and section.
- Members: This table is used to record members information who are registered in the library which includes members ID, name, address, e-mail address and phone number.
- Branch: Stores branch details including the branch ID, location and contact details since a library can have many branches.
- Staff: Includes records of employees working in the library: staff_id, name, position, contact details, supervising staff ID, and branch ID.
- Record: Records borrowing and returning of the books through a record containing the record_id, date of transaction, book_id, type of transaction.

These entities are the main parts of the library system since they help manage the books, records all transactions, and stores data about members and the staff, as well as the location.

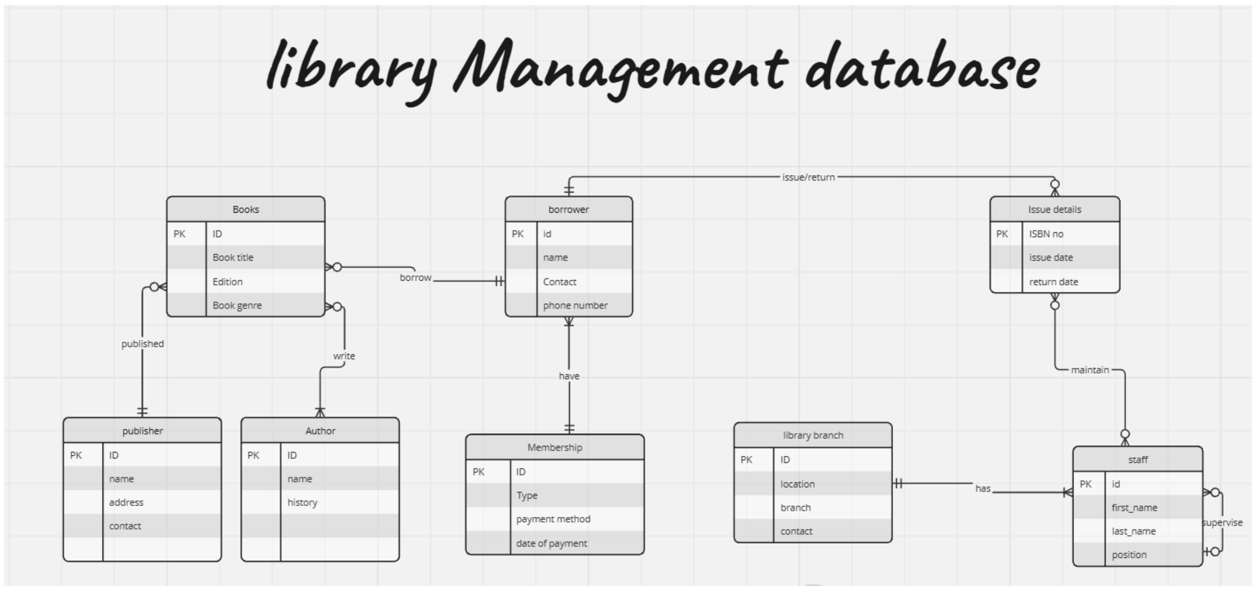
❖ RELATIONSHIPS & CARDINALITIES

1. Books and Publisher (1:M)
   - Each book is published by one publisher, and one publisher can publish many books.
2. Books and Location (1:M)
   - Each book is stored in one location, and one location can store many books.
3. Record and Books (1:M)
   - Each record is related to one book, and one book can have many records.
4. Record and Members (1:M)
   - Each record involves one member, and one member can have many records.
5. Record and Branch (1:M)
   - Each record is related to one branch, and one branch can have many records.
6. Branch and Staff (1:M)
   - Each branch has many staff members, and each staff member is assigned to one branch.
7. Staff (supervise relationship) (1:M)
   - Each staff member can supervise many staff members and each staff member can be supervised by one staff member.
8. Books and Author (through Author_fk) (M:M)
   - Each book can be written by many authors, and each author can write many books.
   - Junction Table: Author_fk which includes write_id and written_by.
9. Record and Staff (through Record_duty) (M:M)
   - Each record can be maintained by many staff members, and each staff member can maintain many records.
   - Junction Table: Record_duty which includes maintain and maintain_by.

❖ SCHEMA

Schema is a structure that depicts how data is arranged and how the interactions between the entities are carried out in the database. It works as a guide in administarting and linking database.

- Author: author_id, name, email, contact_no
- Books: book_id, title, genre, edition, publisher_id, location_id, availability
- Publisher: publisher_id, name, address, contact
- Location: location_id, shelf_no, section
- Members: members_id, name, address, email, contact
- Branch: branch_id, location, contact
- Staff: staff_id, name, position, contact, supervise_id, branch_id
- Record: record_id, date, book_id, type, amount, member_id, branch_id
- Author_fk: write_id, written_by
- Record_duty: maintain, maintain_by

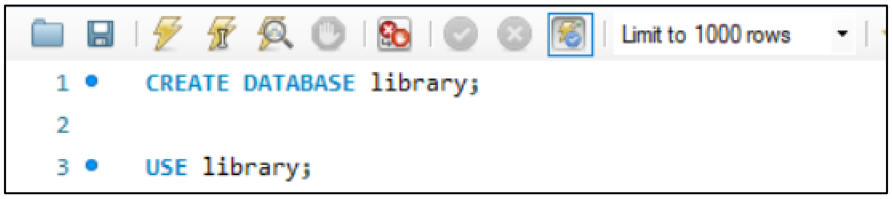


A manually created ERD

3. Database Implementation

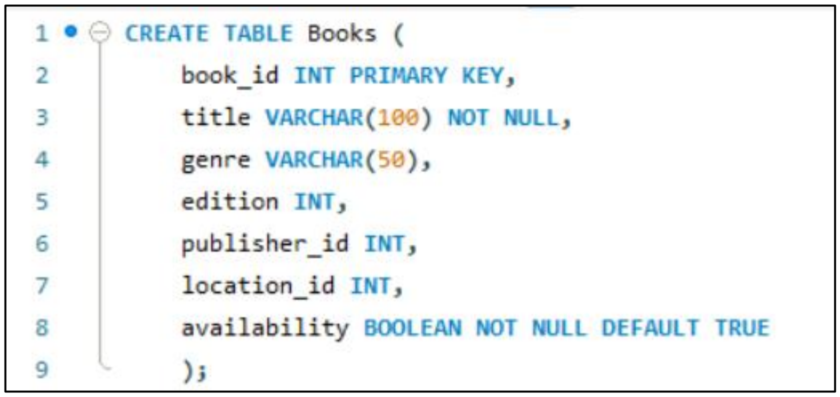
Implementation of database is basically an act of putting operations or establishing a real working database right from the schema that we have created. In the context of our library database, we have to ensure that the database serves the library and run all operations of the database secuerly like handling of book stocks, detials of each member, all transactional records and staff activities. Everything will come to life in MySql as we follow the process. Below are the detailed queries for each and every step:

1. To create schema/database: 'library'



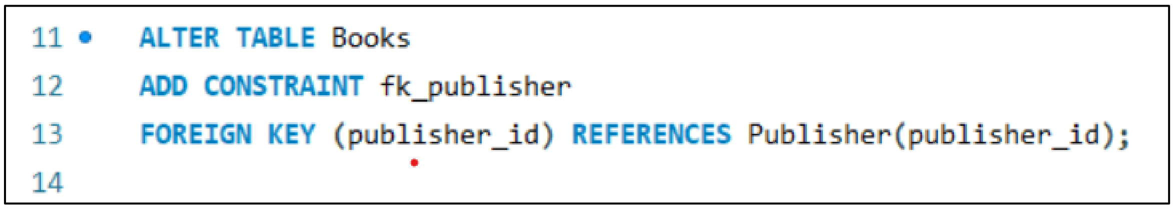
```sql
CREATE DATABASE library;

USE library;
```

2. To create table: For each of the defined entitis we create tables. It refers to the identification of the structure of the database so that it is possible to store, retrieve, or even maintain the data effectively. Then we add set of attributes that describe its properties and data type.



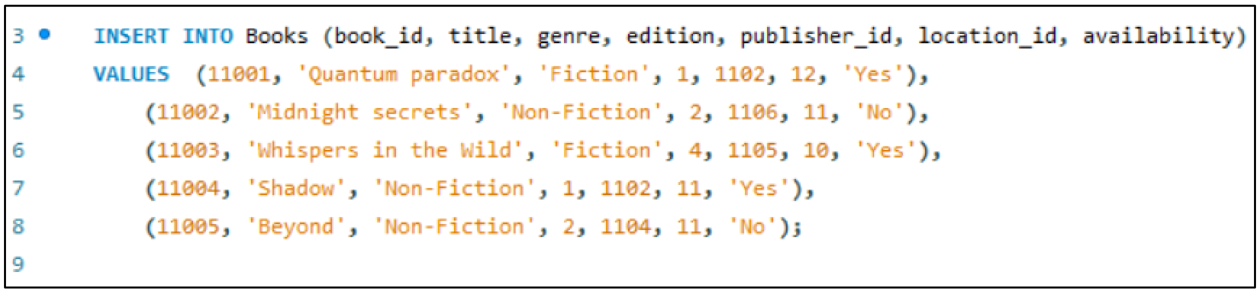
```sql
CREATE TABLE Books (
        book_id INT PRIMARY KEY,
        title VARCHAR(100) NOT NULL,
        genre VARCHAR(50),
        edition INT,
        publisher_id INT,
        location_id INT,
        availability BOOLEAN NOT NULL DEFAULT TRUE
        );
```

3. To add foreign keys: Adding foreign keys to the Books table establishes relational integrity by linking publisher_id to the Publisher table and location_id to the Location table. This ensures that each book is correctly associated with its publisher and location within the library. For that we will first Alter the table.



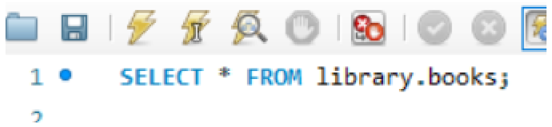
```sql
ALTER TABLE Books
ADD CONSTRAINT fk_publisher
FOREIGN KEY (publisher_id) REFERENCES Publisher(publisher_id);
```

4. To insert data into the tables:



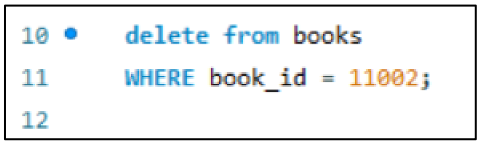
```sql
INSERT INTO Books (book_id, title, genre, edition, publisher_id, location_id, availability)
VALUES  (11001, 'Quantum paradox', 'Fiction', 1, 1102, 12, 'Yes'),
    (11002, 'Midnight secrets', 'Non-Fiction', 2, 1106, 11, 'No'),
    (11003, 'Whispers in the Wild', 'Fiction', 4, 1105, 10, 'Yes'),
    (11004, 'Shadow', 'Non-Fiction', 1, 1102, 11, 'Yes'),
    (11005, 'Beyond', 'Non-Fiction', 2, 1104, 11, 'No');
```

5. To use select command: The 'SELECT' command in SQL is used to retrieve data from one or more tables in a database.



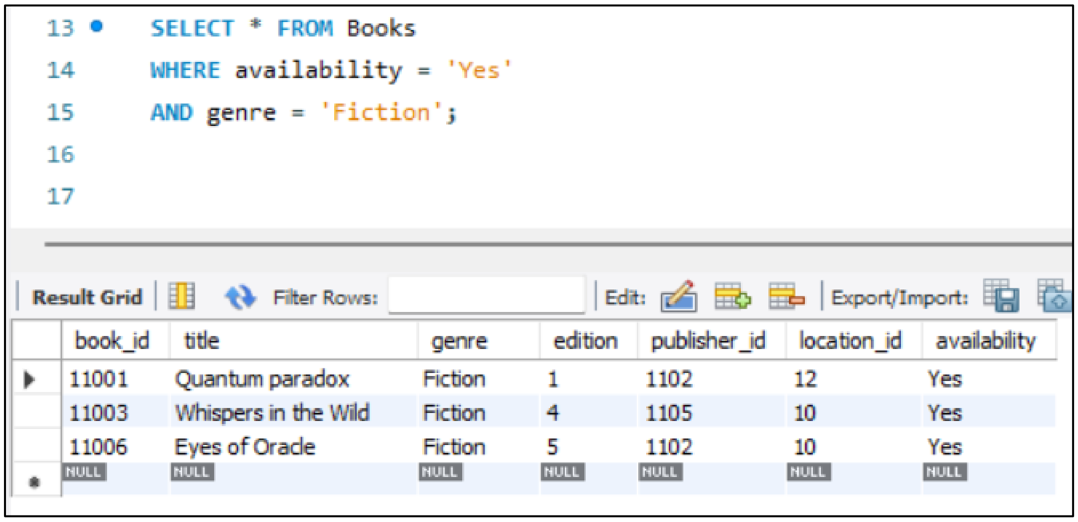
```sql
SELECT * FROM library.books;
```

6. To use delete command for duplicate entries:



```sql
delete from books
WHERE book_id = 11002;
```

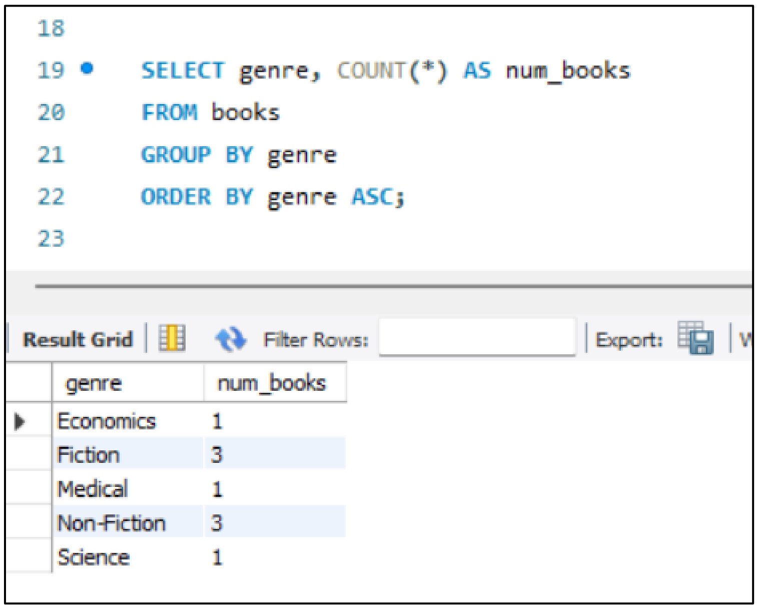
7. To filter data: To filter data based on our requirement we can use 'WHERE' clause combined with 'AND' operator. For example, here we just want to check the available books in the specific genre.



```sql
SELECT * FROM Books
WHERE availability = 'Yes'
AND genre = 'Fiction';
```

book_id	title	genre	edition	publisher_id	location_id	availability
11001	Quantum paradox	Fiction	1	1102	12	Yes
11003	Whispers in the Wild	Fiction	4	1105	10	Yes
11006	Eyes of Oracle	Fiction	5	1102	10	Yes
NULL	NULL	NULL	NULL	NULL	NULL	NULL

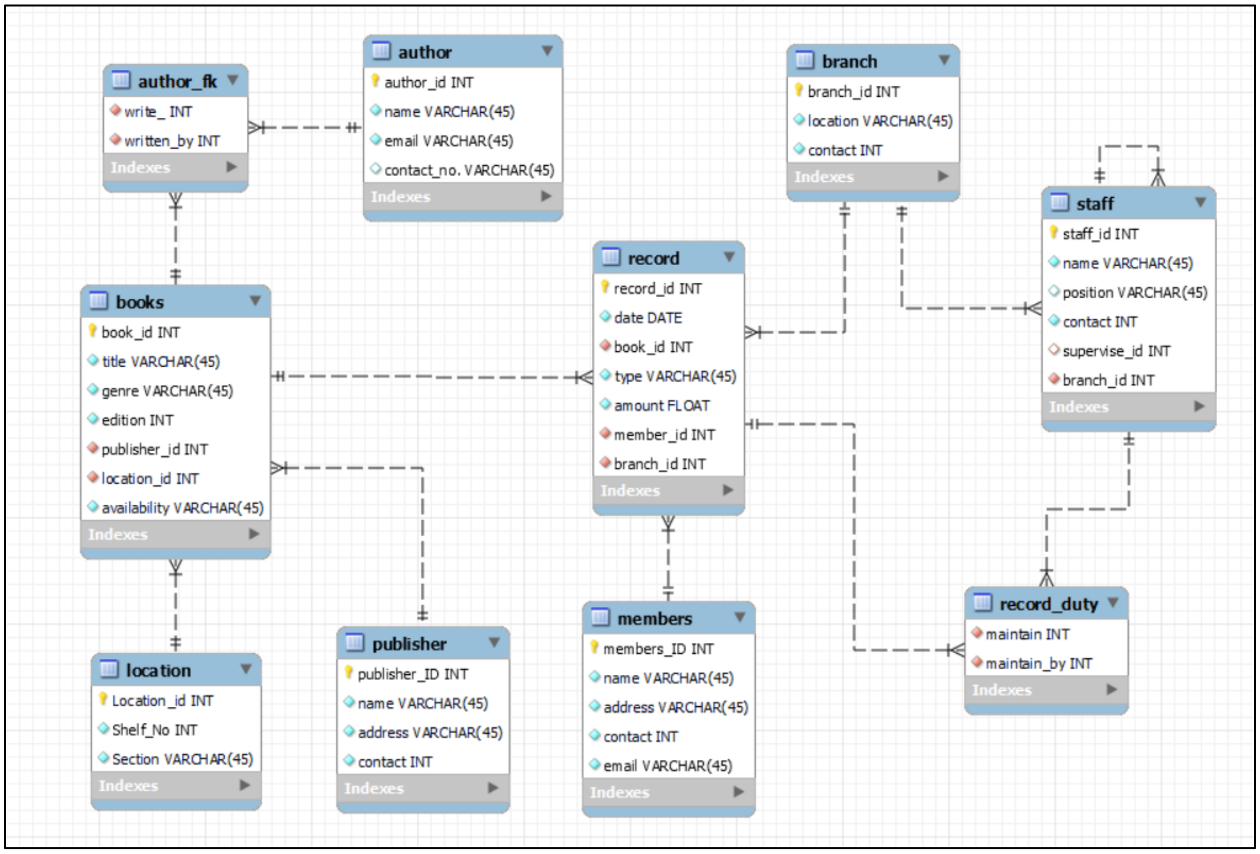
8. To use different functions: In a database we often use functions like 'COUNT' which aggregates data to tally occurences, then we use 'Aliases' for simplifying column references, next we use 'GROUP BY' basically for organizing data into subsets and lastly, we have 'ORDER BY' to arrange results by specified criteria to enhance readability and flexibilty.



```
SELECT genre, COUNT(*) AS num_books
FROM books
GROUP BY genre
ORDER BY genre ASC;
```

genre	num_books
Economics	1
Fiction	3
Medical	1
Non-Fiction	3
Science	1

9. To generate ERD: For generating an Entity Relationship Diagram we will be using a tool named as 'Reverse Engineering' which in one click automatically extract schema information from our database.



ERD generated by Workbench for Library Database

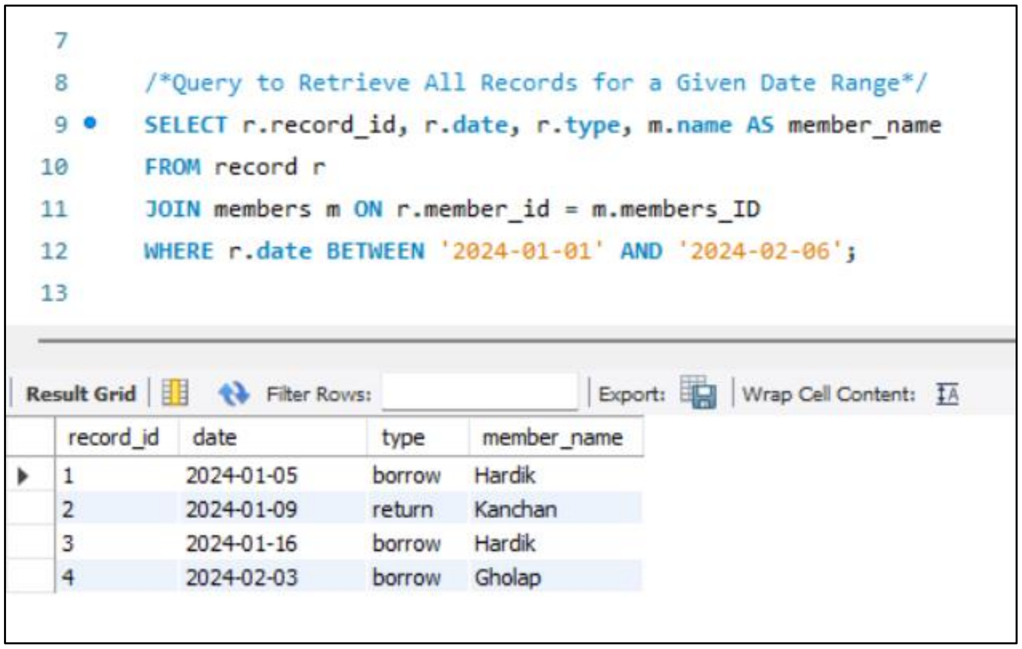
4. SQL QUERIES AND STORED PROCEDURES

SQL queries are all about involving multiple tables and various SQL functionalities to retrieve specific data sets. These queries often utilize joins to combine data from different tables based on related columns, aggregate functions like SUM, COUNT, or AVG to calculate values across groups, and filtering conditions using WHERE clauses to narrow down results based on specified criteria. Subqueries are employed to nest queries within queires, providing more refined data manipulation or filtering.

❖ SQL QUERIES

Below are some SQL queries I used to manipulate our library database and get some filtered data out of it.

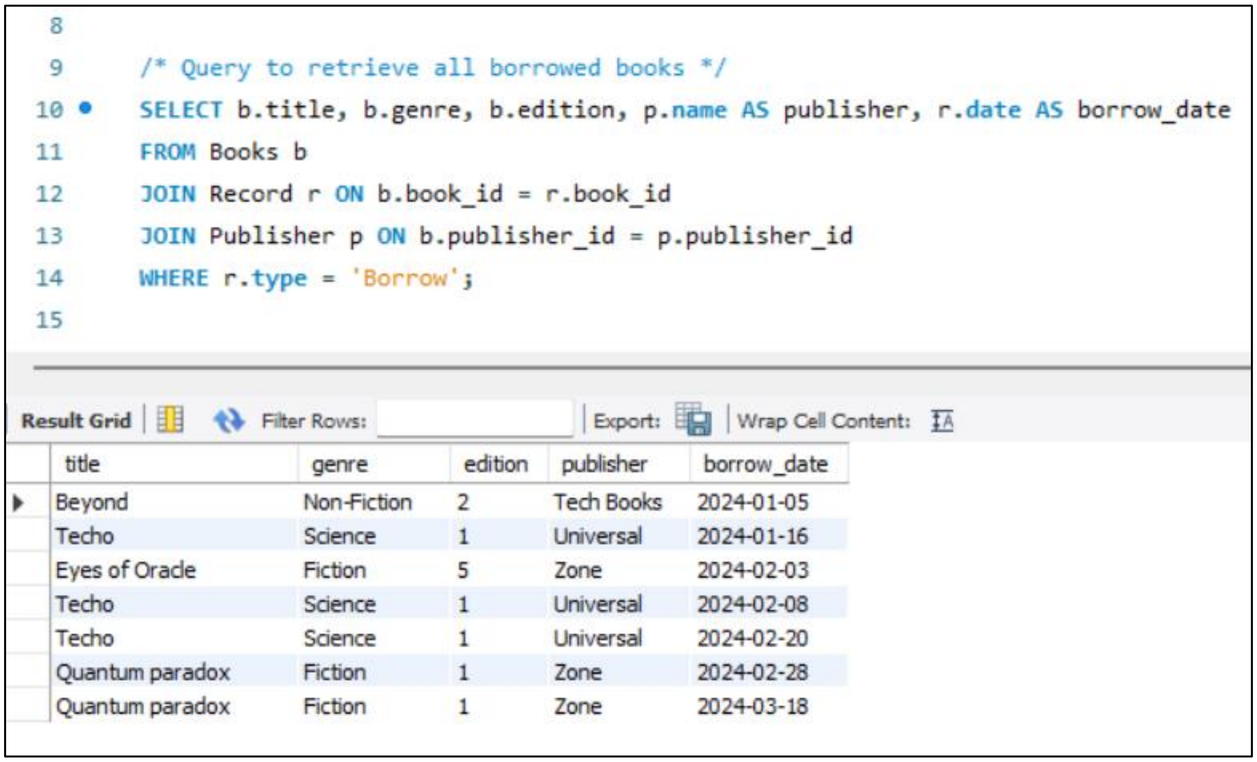
1. Query to retrieve all records for a given date range: I used the query to get all records from the given date range. To achieve this, I have joined the 'Record' table with the 'Members' table on the member_ID. I used where clause to specify the date range and we got the desired outcome.



```sql

/*Query to Retrieve All Records for a Given Date Range*/
SELECT r.record_id, r.date, r.type, m.name AS member_name
FROM record r
JOIN members m ON r.member_id = m.members_ID
WHERE r.date BETWEEN '2024-01-01' AND '2024-02-06';

```

record_id	date	type	member_name
1	2024-01-05	borrow	Hardik
2	2024-01-09	return	Kanchan
3	2024-01-16	borrow	Hardik
4	2024-02-03	borrow	Gholap

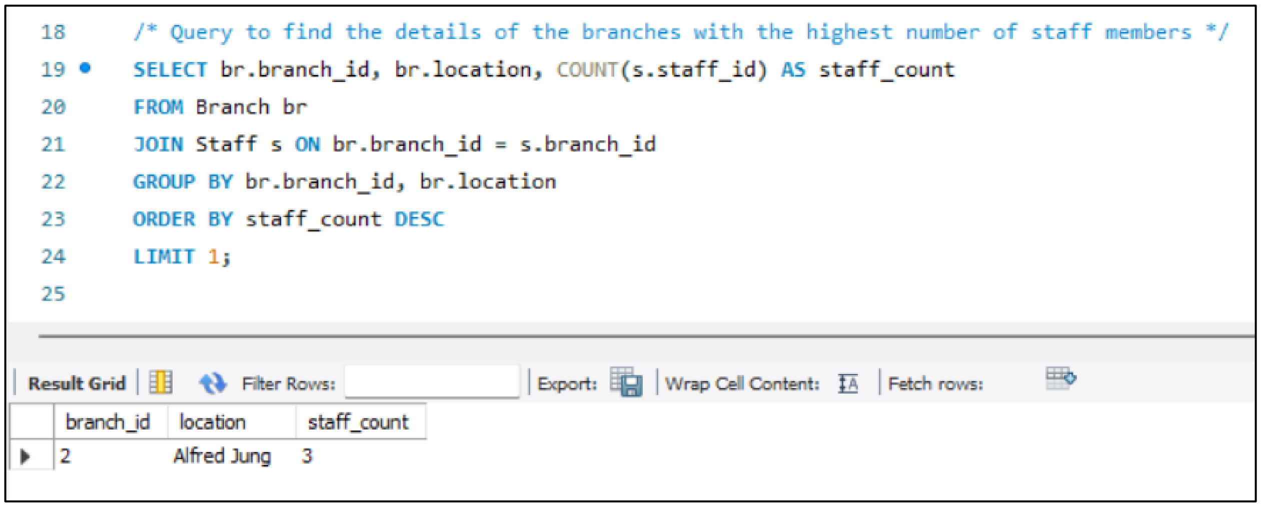
2. <u>Query to retrieve all borrowed books</u>: This is used to get all borrowed books from the dataset. for that we had to join 'books', 'record' and 'publisher' table. The query filters the records and shows the ones with 'borrow' transaction type. The result includes the book title, genre, edition, publisher name and borrow date.



```
/* Query to retrieve all borrowed books */
SELECT b.title, b.genre, b.edition, p.name AS publisher, r.date AS borrow_date
FROM Books b
JOIN Record r ON b.book_id = r.book_id
JOIN Publisher p ON b.publisher_id = p.publisher_id
WHERE r.type = 'Borrow';
```

title	genre	edition	publisher	borrow_date
Beyond	Non-Fiction	2	Tech Books	2024-01-05
Techo	Science	1	Universal	2024-01-16
Eyes of Oracle	Fiction	5	Zone	2024-02-03
Techo	Science	1	Universal	2024-02-08
Techo	Science	1	Universal	2024-02-20
Quantum paradox	Fiction	1	Zone	2024-02-28
Quantum paradox	Fiction	1	Zone	2024-03-18

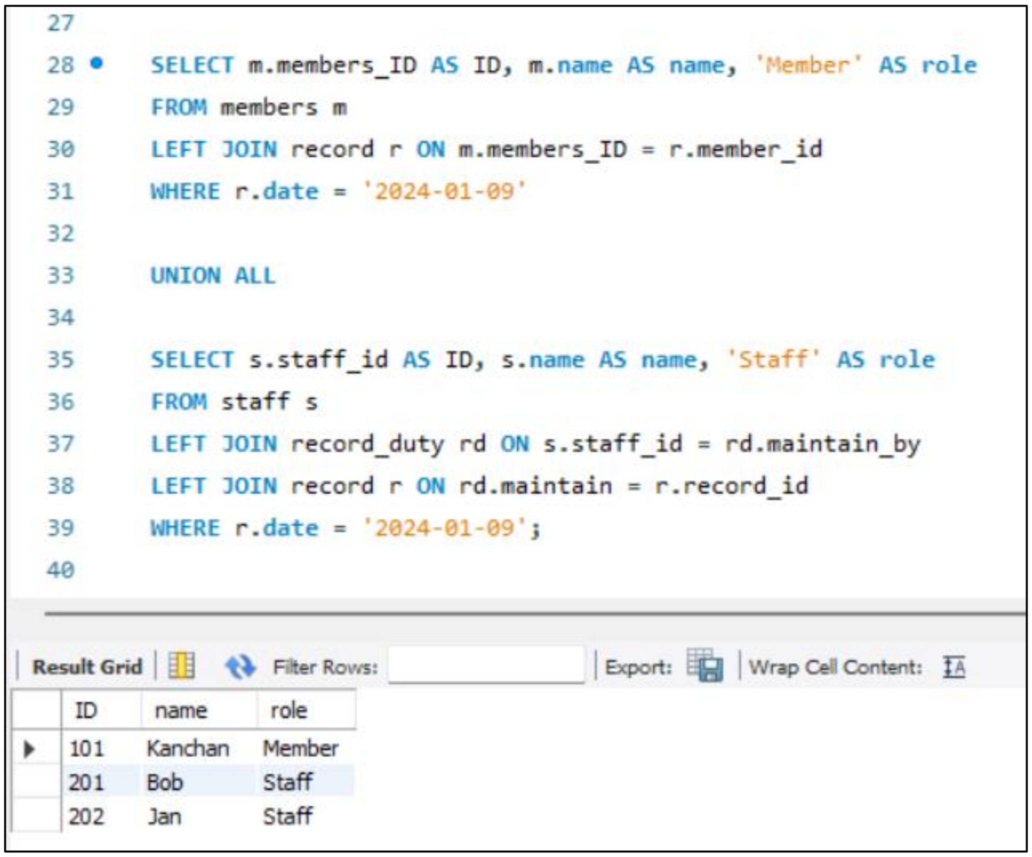
3. <u>Query to find the details of the branches with the highest number of staff members</u>: This SQL query retrieves details of the branch with the highest number of staff members. It joins the Branch and Staff tables on the branch_id, counts the staff members for each branch, and groups the results by branch ID and location.



```
/* Query to find the details of the branches with the highest number of staff members */
SELECT br.branch_id, br.location, COUNT(s.staff_id) AS staff_count
FROM Branch br
JOIN Staff s ON br.branch_id = s.branch_id
GROUP BY br.branch_id, br.location
ORDER BY staff_count DESC
LIMIT 1;
```

branch_id	location	staff_count
2	Alfred Jung	3

4. <u>To get staff involved in a specific day record</u>: In this query, I used subqueries to selectively retrieve information about members and staff based on their involvement in specific records on given dates. By filtering members and staff using a different subquery linked to records dated '2024-01-09', I ensured precise data extraction. The `UNION` operator is used.



```
SELECT m.members_ID AS ID, m.name AS name, 'Member' AS role
FROM members m
LEFT JOIN record r ON m.members_ID = r.member_id
WHERE r.date = '2024-01-09'

UNION ALL

SELECT s.staff_id AS ID, s.name AS name, 'Staff' AS role
FROM staff s
LEFT JOIN record_duty rd ON s.staff_id = rd.maintain_by
LEFT JOIN record r ON rd.maintain = r.record_id
WHERE r.date = '2024-01-09';
```

ID	name	role
101	Kanchan	Member
201	Bob	Staff
202	Jan	Staff

❖ STORED PROCEDURES

Stored procedures are precompiled SQL statements that are stored in a database server where we can find them easily by their title and use them multiple times. They are basically series of qureies and logic into a reusable unit allowing efficient execution and reducing network traffic. They enhance the SQL using experience and user performance by reducing the need to send multiple queries.

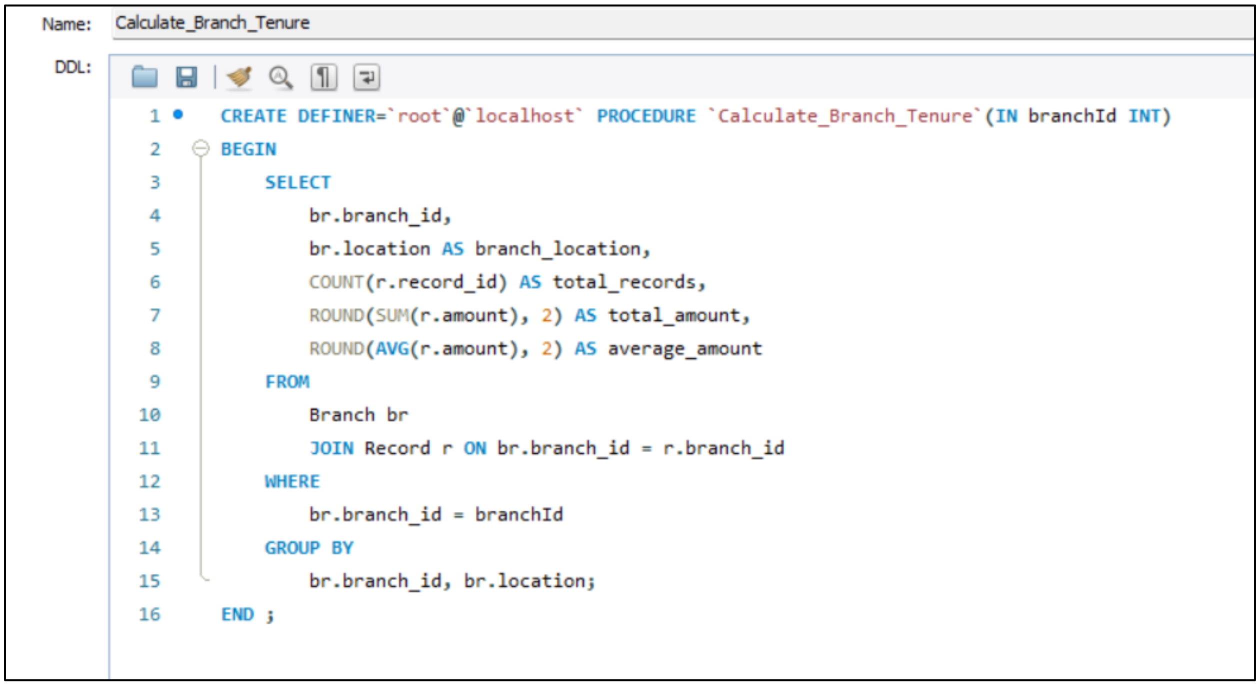
Some Key Features include:

- **Reusability**: Stored procedures are like ready-made sets of instructions stored in our database that can be accessed over and over again for common tasks.
- **Performance**: They make our database run faster by storing and optimizing frequently used queries directly on the server.

- **Security**: They help keep your data safe by controlling who can access and manipulate it, ensuring only authorized users can execute specific procedures.
- **Manageability**: They simplify database management by centralizing complex SQL logic, making it easier to maintain and update.
- **Parameterization**: They can take inputs (parameters) and return outputs, making them flexible for different scenarios and reducing the need for repetitive code.

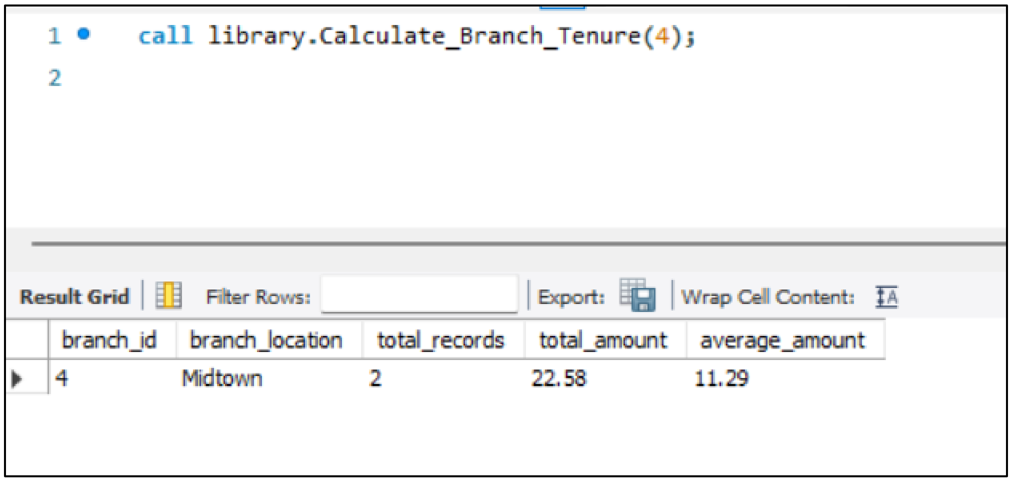
Store procedures that I created in our library database:

1. **Store procedure to calculate branch tenure:** The stored procedure is designed to calculate and display summary statistics for a specified branch. It takes a single input parameter branch_Id to identify the branch of interest. Inside the procedure, a SELECT statement retrieves the branch ID, branch location, total number of records, total transaction amount, and average transaction amount for the specified branch. The JOIN clause and WHERE clause are used.



```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Calculate_Branch_Tenure`(IN branchId INT)
BEGIN
    SELECT
        br.branch_id,
        br.location AS branch_location,
        COUNT(r.record_id) AS total_records,
        ROUND(SUM(r.amount), 2) AS total_amount,
        ROUND(AVG(r.amount), 2) AS average_amount
    FROM
        Branch br
        JOIN Record r ON br.branch_id = r.branch_id
    WHERE
        br.branch_id = branchId
    GROUP BY
        br.branch_id, br.location;
END ;
```

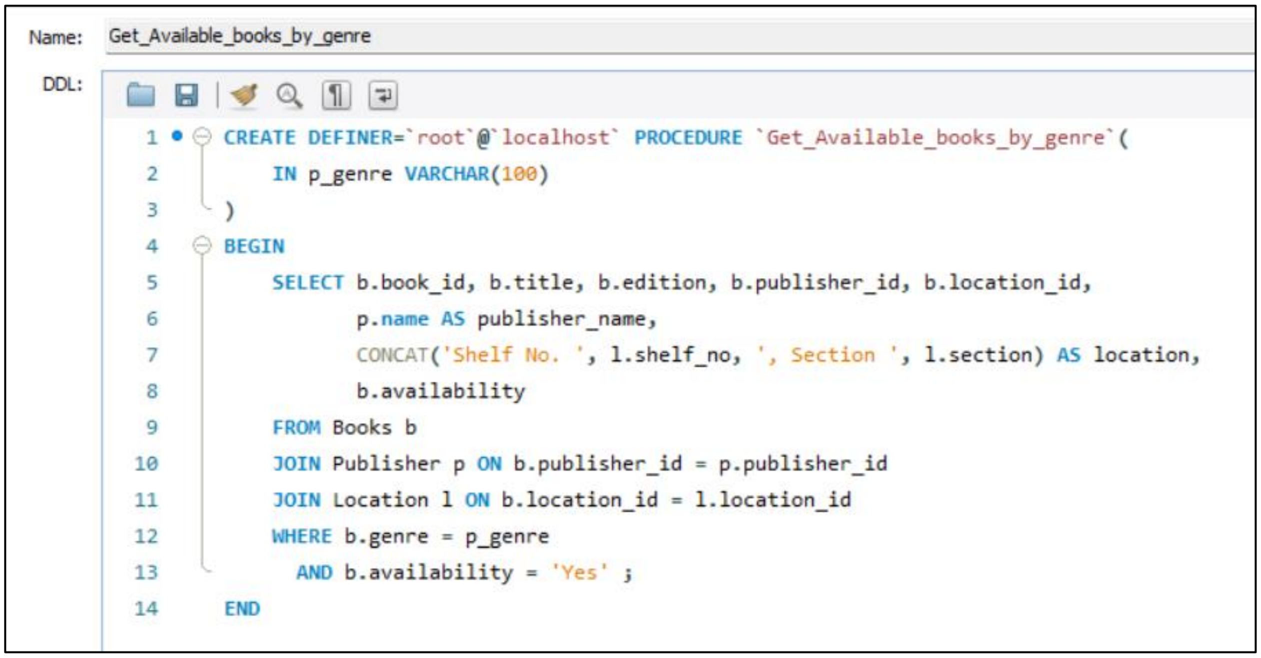
Then we call our procedure meaning using it in our query and mentioning the input parameter. And the generated output is mentioned right below.



```
call library.Calculate_Branch_Tenure(4);
```

```
branch_id  branch_location  total_records  total_amount  average_amount
4          Midtown          2              22.58         11.29
```

2. **Stored procedure to get available book by genre:** This store procedure is designed to retrieve details of books based on a specified genre that are currently marked as available. It operates within the context of a database where book information is stored across multiple tables. The procedure uses SQL joins to connect these tables and fetch comprehensive data and the physical location of the book within the library (shelf_no and section). The WHERE clause filters the results to only include books where availability is set to 'Yes'.

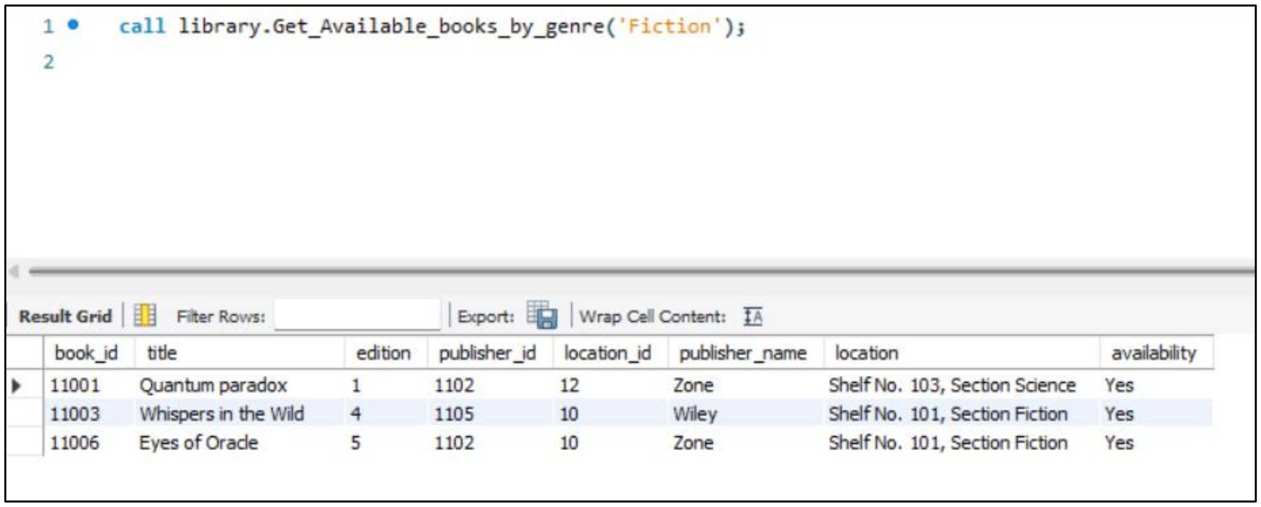


Name: Get_Available_books_by_genre

DDL:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `Get_Available_books_by_genre`(
    IN p_genre VARCHAR(100)
)
BEGIN
    SELECT b.book_id, b.title, b.edition, b.publisher_id, b.location_id,
           p.name AS publisher_name,
           CONCAT('Shelf No. ', l.shelf_no, ', Section ', l.section) AS location,
           b.availability
    FROM Books b
    JOIN Publisher p ON b.publisher_id = p.publisher_id
    JOIN Location l ON b.location_id = l.location_id
    WHERE b.genre = p_genre
      AND b.availability = 'Yes' ;
END
```

Then we call our procedure meaning using it in our query and mentioning the input parameter. And the generated output is mentioned right below.



```
call library.Get_Available_books_by_genre('Fiction');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

book_id	title	edition	publisher_id	location_id	publisher_name	location	availability
11001	Quantum paradox	1	1102	12	Zone	Shelf No. 103, Section Science	Yes
11003	Whispers in the Wild	4	1105	10	Wiley	Shelf No. 101, Section Fiction	Yes
11006	Eyes of Oracle	5	1102	10	Zone	Shelf No. 101, Section Fiction	Yes

5. CAP THEOREM

The CAP theorem, which was introduced by Eric Brewer is fundamental, in the design of distributed data systems, such as the databases used in libraries. It suggests that in any distributed system it's not feasible to achieve all three aspects Consistency, Availability, and Partition Tolerance.

Components of CAP Theorem:

- Consistency ensures that all nodes in a distributed system possess the data at the same time. In a library setting this means ensuring that changes like book borrowing or returning are instanlty reflected in all queries.

- Availability guarantees that every request receives a response even if it’s an error response. In libraries this translates to ensuring users can access the catalog check book availability and conduct transactions without facing system downtime

- Partition Tolerance deals with the system’s ability to function despite network partitions that may occur in a distributed environment. Libraries have branches and partition tolerance ensures each branch’s system can operate independently during network disruptions.

Within a library’s database framework each aspect of the CAP theorem influences decisions related to data management where each component has its dignified role:

- Consistency ensures data representation for operations like checking book availability and processing member transactions.

- Availability is vital for providing service to library users by enabling access, to information and seamless transaction processing.

- Partition Tolerance is crucial, for the distributed structure of the library enabling branches to function in case of network issues while upholding the systems integrity as a whole.

When creating the database structure, for a library setup it is crucial to consider the CAP components. In cases libraries choose to go with an AP (Availability Partition Tolerance) system due to its benefits, in maintaining operations during network issues and ensuring system reliability. Availability is vital, for providing service to library users allowing them to access information and conduct transactions promptly.

Impact on Database Design and Implementation

In creating the database layout for a library system, choices need to focus on elements of the CAP theorem. Generally, libraries tend to choose an AP (Availability Partition Tolerance) system, involves specific considerations to ensure reliability, availability, and efficient management of data. Here's a clear breakdown of how these principles impact the database design and implementation for our library:

❖ Database Schema Considerations:

- Books: Maintain a field (availability) to indicate whether a book is currently available for borrowing or is it already borrowed. This allows users to check the real-time availability of books across different branches.
- Members: Manage user data including personal information and borrowing history. The system should prioritize availability to ensure that members can access their accounts, view their borrowing history, and borrow books seamlessly.
- Records: Track transactions such as book borrowings and returns. Immediate availability of transaction records ensures accurate and up-to-date information for both users and administrators.

❖ Designing for an AP System:

1. Eventual Consistency:
    - Explanation: Updates to the database propagate asynchronously across different nodes (servers or branches). This approach ensures that the system remains available to users even if there are delays in data synchronization across branches.
    - Example: If a book is borrowed at Branch 1, Branch 3 might not immediately reflect the updated availability status due to network delays. However, eventually, the system synchronizes and ensures that all branches have consistent data.
2. Redundancy and Load Balancing:
    - Implementation: Deploy redundant servers and employ load balancing mechanisms. Redundancy ensures that if one server fails, another can take over seamlessly, preventing downtime and maintaining continuous availability for users.
    - Example: If the server handling book reservations goes down, another server automatically handles incoming requests, ensuring users can still check out books without disruption.
3. Partitioning Data:
    - Strategy: Partition data based on branch or geographic location. Each branch manages its own subset of data locally, which enhances system availability and resilience against network partitions.

  - Example: If there’s a network issue between Branch 1 and the central server, Branch 1 can still operate independently. Once the network is restored, data synchronization occurs to update the central database with local changes.

❖ Example scenario for AP system:

Scenario 1 - Book Availability:

- Issue: A user checks the availability of a book at Branch 1, but there are network issues delaying the response from the central server.
- Resolution: The AP system allows Branch 1 to operate independently, showing the last known availability status of the book. Once network connectivity is restored, the system synchronizes to update the availability status across all branches.
- Benefits: Branch 1 maintains local operations, displaying the last known availability status. this ensures high availability and resilience ensuring independent operations.
- Trade Offs**:** consistent delay, waiting for outdated information until synchronization. Synchronization complexity as it requires immediate process across branches.

Scenario 2 - Concurrent Book Borrowing:

- Issue: Two users attempt to borrow the last available copy of a popular book from different branches simultaneously.
- Resolution: The AP system permits both transactions to proceed. Each branch updates its availability records independently and eventually synchronizes to ensure all branches reflect consistent availability status.
- Benefits: Both Transactions proceeds independently which enhances user experience without centralized constraints. Scalability as it handles simultaneous transactions.
- Trade Offs: Eventual consistency as it takes time to synchronize. Data conflicts occurs due to delayed updates

CONCLUSION

In conclusion, this report navigated the essential aspects of database management through the creation of a MySQL database for a public library. From ER modeling to SQL query development and considerations of the CAP theorem, the project underscored the importance of structured data systems in optimizing library operations and user experience.

REFERENCES

MySQL. (Year). *CREATE DATABASE Statement*. Retrieved July 4, 2024, from MySQL Documentation: https://dev.mysql.com/doc/refman/8.4/en/create-database.html

W3Schools. (Year). *SQL Quick Reference*. Retrieved July 4, 2024, from W3Schools: https://www.w3schools.com/sql/sql_quickref.asp

SQL Shack. (Year). *Learn MySQL: The Basics of MySQL Stored Procedures*. Retrieved July 4, 2024, from SQL Shack: https://www.sqlshack.com/learn-mysql-the-basics-of-mysql-stored-procedures/