



**BERLIN SCHOOL OF
BUSINESS & INNOVATION**

Assignment Title: Database System Design for a Library

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

Name: Mihir Pandey

Year: 2024

CONTENTS



1. Introduction
2. Chapter 1 – Database Design
3. Chapter 2- Implementation (LO1, LO2)
4. Chapter 3- CAP theorem discussion (LO1, LO3)
5. Concluding Remarks
6. Reference

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):

MIHIR PANDEY

Date: 28/06/2024

INTRODUCTION

In the face of a rapidly shifting technological landscape, many businesses are prioritizing digital metamorphosis to stay ahead of the curve. Technologies like cloud computing, mobile platforms, and advanced data analysis are fundamentally reshaping how organizations function, granting them a sustained competitive edge. The library domain, a meticulously crafted database catering to the specific requirements of librarians, scholars, and patrons establishes a well-organized infrastructure for swiftly and precisely classifying, indexing, and retrieving the immense volume of information.

An open source, MySQL is a software program adept at managing databases. It empowers users to seamlessly insert, modify, erase, and retrieve data. When faced with mountains of information, constructing a linked database using MySQL reigns supreme as a prevalent solution. Building upon the foundation of MySQL, this exploration ventures into the uncharted territory of library database design and its real-world applications. It underscores the paramount importance of efficient systems that elevate the art of information organization. By adhering to the bedrock principles of database design, we unveil a meticulous methodology for constructing a multifaceted library management database.

In this report, we see how Implementing and thorough testing of a database, a robust and user-friendly library database system can be established, ensuring the efficient management of library resources and fostering a positive user experience. Finally, the report shows the CAP Theorem compels libraries to carefully analyze their priorities. By understanding the trade-offs, librarians and database administrators can design and implement a library database system that offers a balance between data consistency and system availability, while still ensuring optimal user experience and efficient resource management.

CHAPTER ONE - Database Design



Figure1 - The Staatsbibliothek Zu Berlin

The Staatsbibliothek Zu Berlin, also known as the Berlin State Library or Staatsbibliothek is a renowned library in Germany. The library is located in two locations the first one is in Unter den Linden and the other in Potsdamer Platz in Berlin. This Library was built in 1903-1914. The Staatsbibliothek Unter den Linden is one of the largest buildings in Berlin and one of the most important libraries in the world.[1] In this report, I took some data and references from this library. In this library, a person should book his seat prior and scan the QR code before 30 minutes of arrival.



Figure 2- Library Card Front



Figure 3- Library Card Back

The scope and objectives of the database for a public library, considering stakeholders, requirements, and functionalities.

A library database acts as a digital treasure trove, offering a searchable index of trustworthy published resources. It unlocks a universe of valuable research materials, encompassing scholarly journals, current newspapers, and in-depth magazines. Many databases even extend their reach to electronic books, relevant online resources, and diverse multimedia content. A well-designed public library database acts as the lifeblood of its operations, streamlining workflows, enriching user experiences, and bolstering the library's core mission. Here's a deeper dive into its scope, considering the key players, essential needs,

and functionalities.

Library staff rely on it daily to manage resources, patron accounts, circulation, and generate reports.

Library patrons interact with it indirectly through the public catalog, searching for knowledge and

borrowing materials. Library administrators use the data for budgeting, resource allocation, and

evaluating the library's performance. Security, scalability, and integration with other library systems are

crucial. Most importantly, accessibility ensures everyone can benefit from the library's resources.

Scope of Database for a Public Library :

- **Data Management:**
 - Cataloging and indexing library materials (books, eBooks, audiobooks, DVDs, etc.) efficiently.
 - Maintaining accurate inventory levels to ensure resource availability.
- **Seamless Search and Accessibility:**
 - A user-friendly interface for patrons to search the catalog by title, author, subject, keyword, or availability.
 - Advanced search options for researchers, including publication date and format filtering.
- **Streamlined Circulation Management:**
 - Efficient borrowing and returning processes, including holds and renewals.
 - Tracking overdue materials and generating fines (if applicable).
- **Data-Driven Insights:**
 - Generating reports on usage statistics, popular materials, and patron demographics to inform decision-making.
 - Analyzing data to optimize collection development and service planning.

The objective of a public library database, considering stakeholders, requirements, and functionalities, is to create a centralized, efficient, and user-friendly system that supports the library's mission of promoting literacy, lifelong learning, and equitable access to information.

1. Empower Users and Improve User Experience

For Library Patrons (Users): Offer a user-friendly public access catalog (PAC) for efficient searching by title, author, subject, keyword, or availability. Enable indirect interaction with the database through the PAC and public access computers to locate and access materials both physical and digital (eBooks, online databases). Allow patrons to manage their accounts for renewals and holds.

2. Enhance Efficiency and Streamline Operations

The database empowers staff with streamlined operations (resource management, automated circulation, patron tracking) and insightful reports. Library administrators benefit too, using data for budgeting, resource allocation, and performance evaluation to optimize the library's resources.

3. Support the Library's purpose

The library database fosters knowledge sharing, resource utilization, and equitable information access for all (including those with disabilities), solidifying its role as a community hub. It empowers lifelong learning by providing a wealth of materials and research assistance, supporting both formal education and self-directed exploration.

ER diagram representing the database structure, incorporating appropriate entities, attributes, relationships, and cardinalities.

The conceptual design stage lays the groundwork for a powerful and efficacious system. During this phase, the goal is to translate the relationships, attributes, and requirements into a standardized database architecture. This can be accomplished by crafting an Entity-Relationship (ER) diagram. This diagram serves as a visual blueprint, encapsulating the core structure of the database and the intricate connections between its constituent elements (Figure 4). The specific format and methodology employed here draw inspiration from the seminal text 'Database Systems'[2].

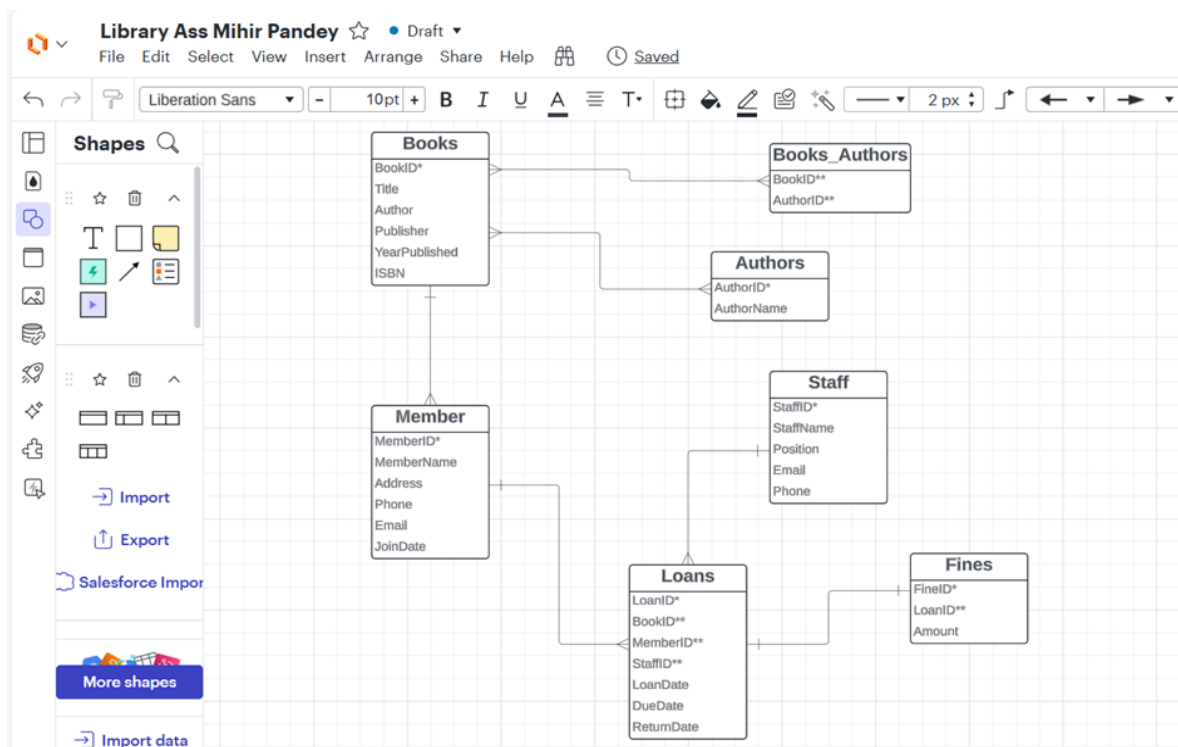


Figure 4- ER Diagram

Relationships: The database reflects real-world connections: Books and Authors (many-to-many via Books Authors), Members and Loans (one-to-many), Books and Loans (one-to-many), Staff and Loans (one-to-many), Loans and Fines (one-to-one). This structure prioritizes efficient data management and retrieval, ensuring data integrity, consistency, and availability – essential principles for a well-functioning library database.

Design choices based on the specific needs of a library, aligning with the principles of database design.

Data Modeling

The foundation of a well-designed library database lies in its data organization. Separate entities are established for Library Materials (books, eBooks, etc. with details like title, author, and availability), Patrons (library cardholders with information like borrowing history and fines), Staff (librarians with department and role details), Collections (groupings like children's fiction), Loans (tracking who borrows what), and even Digital Resources (online databases with access credentials). This structured approach ensures efficient data management and retrieval.

Implementing the database design involves translating the Entity-Relationship diagram into tables. For "one-to-many" relationships, a table representing the "many" side gets a foreign key referencing the primary key of the "one" side's table. For complex "many-to-many" scenarios, a junction table acts as the go-between, containing a composite primary key made from both entities' primary keys, and additional foreign keys referencing each entity[5].

Effective Library database design prioritizes user needs. Patrons manage personal information, library cards, reservations, and appointments. Staff information is linked to departments, and appointment scheduling fosters collaboration. The database comprehensively captures resources: branches, departments, events, courses, collections (with publisher and contributor details), and physical item copies, ensuring efficient resource management and user interactions.

Design Principles

1. Normalization

Minimizing data redundancy is paramount in database design to ensure data integrity. The first step is achieving the First Normal Form (1NF), where each table cell contains a single value and each record is unique. Imagine a Book table; attributes like Title, Author, and ISBN each hold only one value per record. The second Normal Form (2NF) goes further, requiring all non-key attributes to depend solely on the primary key. In the Books table example, Title, Author, and Publisher all depend on the unique Book ID. Finally, the Third Normal Form (3NF) eliminates transitive dependencies, meaning attributes rely only on the primary key. This might involve separating Authors into a distinct table and using a junction table to manage many-to-many relationships between Books and Authors.

Illustration:

The Books table breaks down the Author into an associated Authors table, so there is no redundancy of author information on several records for multiple books.

2. Entity-Relationship Design

This will model the relationship structure of the entities properly with real-world entities and their relationships. Books and authors: Since a book may have more than one author, and an author writes more than one book, this is a many-to-many relationship to be implemented by way of the junction table Books_Authors. Members and loans: It is a one-to-many relationship since one member can get many books on loan but a loan to one member is specific.

3. Data Integrity and Constraints

Database integrity is safeguarded by constraints: Primary Keys (PK) uniquely identify each record in a table (e.g., Book ID in Books). Foreign Keys (FK) maintain relationships between tables (e.g., Book ID in Loans references Books table's Book ID). Finally, Unique Constraints ensure specific fields hold unique values (e.g., ISBN in Books, Email in Members and Staff)

Example:

The Loans table enforces referential integrity through foreign keys and makes sure there are valid book and member IDs.

4. Flexibility and Scalability

The flexible design readily scales for future needs. New entities (like DVDs) can be incorporated seamlessly, and the system can handle complex relationships, as exemplified by the "Books Authors" table managing the many-to-many connection between books and authors.

CHAPTER TWO - Implementation (LO1, LO2)

To implement the database design I used My SQL Workbench. Firstly, I created a database by the name of Assignment. Then by the command to use this Database, I created seven Tables (Books, Authors, Books_Authors, Fines, Loans, members, Staff). For creating Tables in My SQL Workbench I used the CREATE TABLE command and added Data Types to the particular attribute. Moreover, For a unique ID in the table, I mentioned it by Primary Key. A few tables also require Foreign Key consequently I used the function Foreign Key for that. The most frequent data types I used are INT, VARCHAR, and DATE. Furthermore, If an attribute has no null values I used a NOT NULL function.

Now, Below are the codes for how I create databases and tables on My SQL workbench.

1. DATABASE Creation :

Create database Assignment;

use Assignment;

2. TABLE Creation :

CREATE TABLE Books (

BookID INT AUTO_INCREMENT PRIMARY KEY,

Title VARCHAR(200) NOT NULL,

Author VARCHAR(200) NOT NULL,

Publisher VARCHAR(200),

YearPublished YEAR,

ISBN VARCHAR(20) UNIQUE);

CREATE TABLE Authors (

AuthorID INT AUTO_INCREMENT PRIMARY KEY,

AuthorName Varchar(50) NOT NULL);

CREATE TABLE Books_Authors (

BookID INT AUTO_INCREMENT PRIMARY KEY,

AuthorID Int);

CREATE TABLE Members (

MemberID INT AUTO_INCREMENT PRIMARY KEY,

MemberName VARCHAR(50) NOT NULL,

Address VARCHAR(50),

Phone VARCHAR(15),

Email VARCHAR(100) UNIQUE,

JoinDate DATE);

CREATE TABLE Loans (

LoanID INT AUTO_INCREMENT PRIMARY KEY,

BookID INT,

MemberID INT,

StaffID INT,

LoanDate DATE,

DueDate DATE,

ReturnDate DATE,

FOREIGN KEY (BookID) REFERENCES Books(BookID),

FOREIGN KEY (MemberID) REFERENCES Members(MemberID));

CREATE TABLE Fines (

FineID INT AUTO_INCREMENT PRIMARY KEY,

LoanID INT,

Amount INT);

CREATE TABLE Staff (

StaffID INT AUTO_INCREMENT PRIMARY KEY,

StaffName VARCHAR(50) NOT NULL,

Position VARCHAR(50),

Email VARCHAR(100) UNIQUE,

Phone VARCHAR(15));

**** Provide sample data reflecting typical library scenarios...**

To insert the values I used INSERT INTO command. Here we have to be careful the values of the primary key should be unique. In every table there should be only one primary key and there can be more than one foreign key in a table. The string should be inside closed brackets. These rules are not applied for numerical values or int.

Given below are the codes for inserting values in the tables:

INSERT INTO Books (BookID, Title, Author, Publisher, YearPublished, ISBN) VALUES

(1,'Talk to Strangers', 'Matt Dahlia', 'Derin Emre', 2023, '978-0-06-112408-4'),

(2,'To Kill a Mockingbird', 'Harper Lee', 'J.B. Lippincott & Co.', 1960, '978-0-06-112008-4'),

(3,'1984', 'George Orwell', 'Secker & Warburg', 1949, '978-0-452-28423-4'),

(4, 'The Last Horizon', 'Rachel Adams', 'Horizon House', 2019, '978-1-76543-210-3'),

(5, 'Whispers in the Wind', 'David Foster', 'Blue Sky Publishing', 2017, '978-1-54321-678-9');

INSERT INTO Authors (AuthorID, AuthorName) Values

(101, 'Matt Dahlia'),

(102, 'Harper Lee'),

(103, 'Secker & Warburg'),

(104, 'Rachel Adams'),

(105, 'David Foster');

INSERT INTO Books_Authors (BookID, AuthorID) Values

(1,101),

(2,102),

(3,103),

(4,104),

(5,105);

INSERT INTO Members (MemberID, MemberName, Address, Phone, Email, JoinDate) Values

(901,'Johnny Doe', '163 Main St', '555-1234', 'johnny.doe@example.com', '2022-01-15'),

(902,'Wane Smith', '436 Elm St', '555-5678', 'wane.smith@example.com', '2021-02-20'),

(903,'Slice Johnson', '782 Oak St', '555-8765', 'slice.johnson@example.com', '2023-03-10'),

(904,'Bobby Williams', '111 Pine St', '555-4321', 'bobby.williams@example.com', '2024-04-05'),

(905,'Carolina Brown', '272 Maple St', '555-3456', 'carolina.brown@example.com', '2019-05-15'

);

INSERT INTO Staff (StaffID, StaffName, Position, Email, Phone) Values

(1001, 'Rohit Joshi', 'Librarian', 'rohit23@gmail.com', 9140967823),

(1002, 'Vivek Pandey', 'Assistant Librarian', 'vivekp456@gmail.com', 7302192070),

(1003, 'Darshan Singh', 'Library Clerk', 'darshansng23@gmail.com', 7654893247);

INSERT INTO Loans (LoanID, BookID, MemberID, StaffID, LoanDate, DueDate, ReturnDate) Values

(11123, 1, 901, 1001, '2023-03-01', '2023-03-15', NULL),

(11194, 2, 902, 1003, '2023-03-05', '2024-03-19', '2023-03-18'),

(11175, 3, 903, 1001, '2023-04-01', '2023-04-15', '2023-04-14'),

(11154, 4, 901, 1001, '2023-05-01', '2024-05-15', NULL),

(11185, 5, 902, 1002, '2023-06-01', '2023-06-15', NULL);

INSERT INTO Fines (FineID, LoadID, Amount) Values

(56403, 11154, 550),

(56512, 11185, 200);

*** Develop a series of SQL queries demonstrating manipulation of library system data. This includes queries for user information retrieval, account management, activity history, as well as complex queries involving JOIN operations and subqueries for advanced operations.

```

118 1. Retrieve all books available in the library:
119
120 SELECT * FROM Books;

```

Result Grid

BookID	Title	Author	Publisher	YearPublished	ISBN
1	Talk to Strangers	Matt Dahila	Derin Emre	2023	978-0-06-112408-4
2	To Kill a Mockingbird	Harper Lee	J.B. Lippincott & Co.	1960	978-0-06-112008-4
3	1984	George Orwell	Secker & Warburg	1949	978-0-452-28423-4
4	The Last Horizon	Rachel Adams	Horizon House	2019	978-1-76543-210-3
5	Whispers in the Wind	David Foster	Blue Sky Publishing	2017	978-1-54321-678-9

Books 1 x

Output

Figure 5- Solution for query all books available in library

```

126 3. Find the loan history for a specific member:
127
128 SELECT Loans.LoanID, Books.Title, Loans.LoanDate, Loans.DueDate, Loans.ReturnDate
129 FROM Loans
130 JOIN Books ON Loans.BookID = Books.BookID
131 WHERE Loans.MemberID = 903;
132

```

Result Grid

LoanID	Title	LoanDate	DueDate	ReturnDate
11175	1984	2023-04-01	2023-04-15	2023-04-14

Result 4 x

Figure 6 - Solution for query to find loan history for a specific member

```

133 4. Count the number of books borrowed by each member:
134
135 SELECT MemberName, COUNT(Loans.LoanID) AS NumberOfLoans
136 FROM Members
137 JOIN Loans ON Members.MemberID = Loans.MemberID
138 GROUP BY Members.MemberID;
139

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
MemberName	NumberOfLoans		
Johnny Doe	2		
Wane Smith	2		
Slice Johnson	1		

Result 5 x

Figure 7 – Solution for query to count no. of books borrowed by each member.

```

140 5. List members who have borrowed more than one book in a single loan (subquery):
141
142 SELECT MemberID, COUNT(BookID) AS NumberOfBooksBorrowed
143 FROM Loans
144 GROUP BY MemberID
145 HAVING COUNT(BookID) > 1;
146

```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
MemberID	NumberOfBooksBorrowed		
901	2		
902	2		

Result 6 x

Figure 8 – Solution for query to list members who borrowed more than one book in single loan.

```

147 6. Find the average amount of fine members have to pay:
148
149 -- 6. Find the average amount of fine members have to pay:
150
151 SELECT AVG(Amount) AS AverageFineAmount FROM Fines;
152

```

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

	AverageFineAmount
▶	375.0000

Result 7 x

Figure 9 – Solution for query to find the avg. amount of fine members have to pay.

```

153 7. Find the oldest book in the books table and give author name and published year:
154
155 SELECT Books.Title, Authors.AuthorName, Books.YearPublished
156 FROM Books
157 JOIN Books_Authors ON Books.BookID = Books_Authors.BookID
158 JOIN Authors ON Books_Authors.AuthorID = Authors.AuthorID
159 WHERE Books.YearPublished = (SELECT MIN(YearPublished) FROM Books);
160

```

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

	Title	AuthorName	YearPublished
▶	1984	Secker & Warburg	1949

Result 8 x

Figure 10 – Solution for the query to find the oldest book in the books table and give author name and published year

CHAPTER THREE – CAP Theorem Discussion (LO1, LO3)

The CAP theorem, a fundamental concept in distributed systems, presents a trade-off between three crucial properties: Consistency, Availability, and Partition Tolerance. In the context of a library database system, understanding these properties is vital for ensuring efficient information access and data integrity. The CAP theorem was invented by computer scientist Eric Brewer early in the 2000s and is thus sometimes referred to as Brewer's theorem. [3]

The three components of the CAP theorem are:

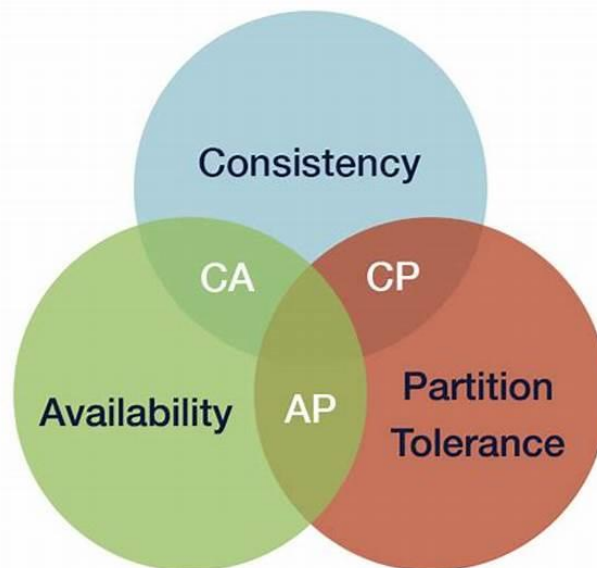


Figure 11 - The three components of the CAP theorem

Consistency:

In distributed systems, strong consistency ensures all nodes see the same data at all times. This ideal can be difficult to achieve due to network delays and coordination needs. Alternatively, some systems prioritize high availability, ensuring service even during network issues, potentially sacrificing some consistency. Here, availability implies the system remains operational and responsive to user requests despite node failures or network partitions. Techniques like data replication across multiple nodes help achieve high availability by providing alternate paths to access data.

Availability: This guarantees that the system remains accessible to users and can process requests, even

during network failures or server outages. In a library context, availability ensures that patrons can access the catalog, and renew loans, and librarians can manage resources even if there are temporary network issues.

Partition Tolerance: Partition tolerance explains the ability of a system to work efficiently and handle failures in communication due to network partitions. A network partition occurs when the nodes of a distributed network cannot communicate with each other due to different reasons or faults in the network.[4]

Understanding CAP Theorem through an Example

Consider the case that a member borrowing a book from branch A will update the center database and inform other branches that the book is loaned out to ensure consistency. But this would mean others have to wait for the update, hence latency exists.

In case of a network partition between Branch A and Branch B, the latter does not need to stop queried or processing loans against its local data, which represents availability and partition tolerance. However, Branch B may indicate that the book can be loaned out for some while until such a time when the network comes back up and all data is properly synchronized; hence, potential inconsistency.

It means that quite often in a library management system, CAP theorem constraints are tipped in favour of availability and partition tolerance with eventual consistency. This approach will at least ensure the usability of the system during network issues and secondarily that data will synchronize when normal operations resume. It is generally far more practical for ensuring a good user experience in real-world scenarios.

CONCLUDING REMARKS


In conclusion, the design and implementation of a library database system. We explored the scope and objectives, crafting an Entity-Relationship diagram to model the data structure.

Here we can see in Figure 4 the Entity-Relationship (ER) diagram which serves as visually depicts the database schema and relationships between entities. Following the visual representation of the database structure (often through an ER diagram), the next step involves translating it into a logical model using a relational database approach. This model can then be optimized and implemented before undergoing rigorous testing. Here Figure 5 shows the solution for the query of all books available in the library, Figure 6 is a solution for the query to find the loan history for a specific member, Figure 7 is the solution for the query to count no. of books borrowed by each member, Figure 8 represents the solution for the query to list members who borrowed more than one book in a single loan, Figure 9 shows the solution for the query to find the avg. amount of fine members have to pay, and Figure 10 shows the solution for the query to find the oldest book in the books table and give the author's name and published year. MySQL implementation detailed table creation, relationships, constraints, and sample data, showcasing practical application.

Lastly, the CAP theorem helped present a tradeoff between the data consistency that was crucial for storing borrowing records and the system availability that would be useful during the peak use of the catalog. By knowing these trade-offs libraries can focus on functionalities and work with such models like eventual consistency. This project has shown the eventuality of proper design for a database and the CAP theorem for any resourceful library.

Through ongoing optimization of its technical underpinnings, the library database will stay a potent force for knowledge exchange, resource leverage, and cultivating a thriving community learning space.

REFERENCE

- 
- [1] [Staatsbibliothek Unter den Linden - Wikipedia, the free encyclopedia](#)
- [2] https://lucid.app/lucidchart/ac38dc66-e6a7-40a6-963a-a56d1dafa19f/edit?invitationId=inv_49f6a6f2-3022-47ba-9352-213aca6dc71c&page=0_0#
- [3] [Understanding CAP Theorem: Balancing Consistency, Availability, and Partition Tolerance \(knowledgehut.com\)](#)
- [4] [Relate the CAP theorem discussing its impact on decisions concerning data consistency and system availability - Search Images \(bing.com\)](#)
- [5] https://www.researchgate.net/publication/378435639_Library_management_database_design_and_application