Technical Report on Mario Library Database System

A Detailed Analysis of Database Design and Implementation

Prepared by: [Mario Mina]
Date: May 14, 2025

Table of Contents

---

1. Introduction to Database Systems

This report details the design and implementation of the "Mario" library database system, which manages books, members, librarians, and loans. A database system is an organized collection of data, typically stored and accessed electronically via a database management system (DBMS). In this project, MySQL is used as the DBMS to handle library operations efficiently, ensuring data integrity, security, and fast retrieval.

The Mario database was created with the following command:

SQL Command:

```
CREATE DATABASE Library;
```

The system was populated using a script sourced from a file:

SQL Command:

```
SOURCE /Users/mariomenafaragalla/Desktop/database project/mario.sql;
```

The database includes tables for books, members, librarians, authors, publishers, categories, and loans, all interconnected to support library functions like tracking loans and managing inventory. This report will cover the SQL queries, entity relationships, normalization, and practical examples based on the provided data.

2. SQL Fundamentals

SQL (Structured Query Language) is the standard language for managing relational databases. In the Mario library system, SQL is used to create tables, insert data, and query the database. The database was switched to "Mario" with the following command:

SQL Command:

```
USE Mario;
```

A simple SQL query example retrieves book titles containing the word "Nights":

SQL Query:

```
SELECT Title

FROM BOOK

WHERE Title LIKE '%Nights%';
```

Query Output Description (Insert Image: query1.png): The output shows two books: "The Cheapest Nights" and "Beirut Nightmares". The query took 0.00 seconds and returned 2 rows.

SQL commands in this project include creating tables with primary and foreign keys, inserting data, and querying for specific information. These operations allow the library to manage its resources effectively, such as finding books by title or sorting member information.

---

3. Entities and Users

The Mario library system includes the following entities: BOOK, MEMBER, LIBRARIAN, AUTHOR, PUBLISHER, CATEGORY, and LOAN. Each entity has specific attributes:

- **BOOK**: Book_ID, Title, Author_ID, Publisher_ID, ISBN, Category_ID, Number_of_Copies.
- **MEMBER**: Member_ID, Name, Address, Phone, Email, Membership_Date.
- **LIBRARIAN**: Librarian_ID, Name, Email, Phone.
- **LOAN**: Loan_ID, Book_ID, Member_ID, Librarian_ID, Loan_Date, Return_Date.

Users include members who borrow books and librarians who manage the system. For example, a query to list all members in alphabetical order is:

SQL Query:

```
SELECT Name

FROM MEMBER
```

```
ORDER BY Name ASC;
```

Query Output Description (Insert Image: query2.png): The output lists 10 members: Ahmed Samir, Fatima Ahmed, Khaled Nour, Laila Hassan, Mohamed Ali, Mona Zaki, Nourhan Adel, Omar Khaled, Sara Mostafa, and Youssef Amr. The query took 0.00 seconds and returned 10 rows.

This organization of entities and users ensures that the library can track loans, manage member records, and maintain accurate book inventories.

---

4. Mapping with ER and Normalization

The Entity-Relationship (ER) diagram defines the structure of the Mario database. It shows how entities are related:

- A BOOK is written by one AUTHOR, published by one PUBLISHER, and belongs to one CATEGORY.
- A LOAN connects a BOOK, a MEMBER, and a LIBRARIAN, with attributes like Loan_Date and Return_Date.

ER Diagram Description (Insert Image: erdiagram.png): The diagram includes entities like BOOK (attributes: Book_ID, Title, Number_of_Copies), MEMBER (attributes: Member_ID, Name, Email), and LOAN (attributes: Loan_ID, Book_ID, Member_ID). Relationships are marked, such as a many-to-one relationship between BOOK and AUTHOR, and a many-to-many relationship between BOOK and MEMBER through LOAN.

Normalization ensures the database is efficient and free of redundancy. The tables are normalized to the Third Normal Form (3NF):

- **1NF**: All attributes are atomic (e.g., no multi-valued attributes in the BOOK table).
- **2NF**: No partial dependencies (e.g., Title depends fully on Book_ID, not a subset of a composite key).
- **3NF**: No transitive dependencies (e.g., Author_Name is stored in the AUTHOR table, not BOOK, to avoid redundancy).

Normalization reduces data duplication and ensures consistency across the database.

---

6. Aggregate Functions

Aggregate functions in SQL summarize data. The Mario database uses functions like COUNT and AVG. For example, to find the average number of copies per book:

SQL Query:

```
SELECT AVG(Number_of_Copies) AS AverageCopies
```

```
FROM BOOK;
```

Query Output Description (Insert Image: query3.png): The output shows an average of 4.7 copies per book. The query took 0.01 seconds and returned 1 row.

Another example counts the total number of books:

SQL Query:

```
SELECT COUNT(*) AS TotalBooks

FROM BOOK;
```

Query Output Description (Insert Image: query4.png): The output shows there are 10 books in the library. The query took 0.01 seconds and returned 1 row.

These functions help librarians analyze data, such as determining the average availability of books or the total inventory, which aids in decision-making for restocking or reporting.

---

7. Example SQL Query

A practical SQL query in the Mario system lists all books and their number of copies, ordered by the number of copies in descending order:

SQL Query:

```
SELECT Title, Number_of_Copies

FROM BOOK

ORDER BY Number_of_Copies DESC;
```

Query Output Description (Insert Image: query5.png): The output lists books in descending order: Utopia (8 copies), Beirut Nightmares (7 copies), The Cheapest Nights (6 copies), Palace Walk (5 copies), The Story of Zahra (5 copies), The Moon Does Not Tell (4 copies), Season of Migration to the North (4 copies), The Days (3 copies), Destiny (3 copies), and For Bread Alone (2 copies). The query took 0.01 seconds and returned 10 rows.

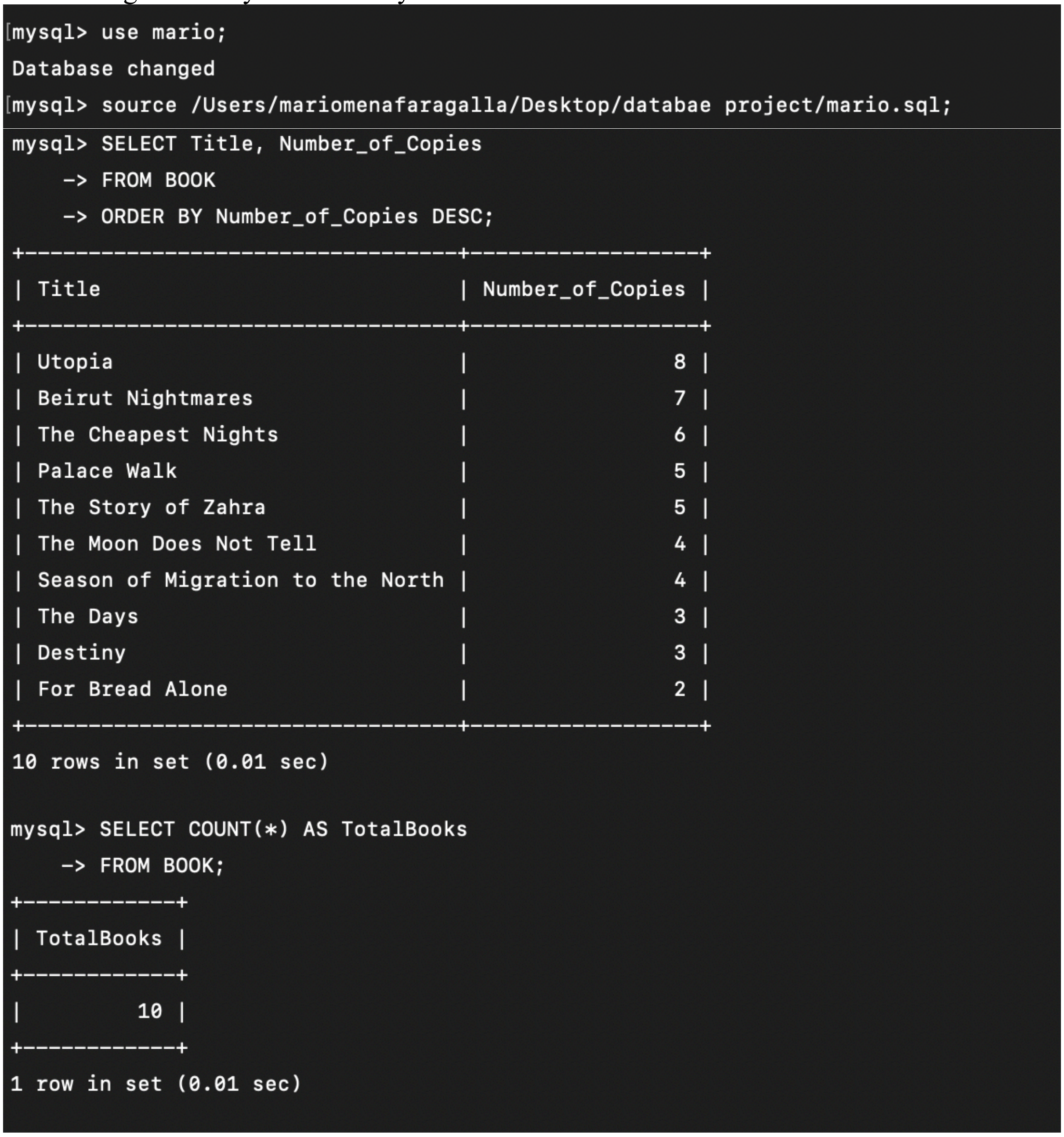
This query helps librarians identify which books are most available and which might need restocking, improving inventory management.

---

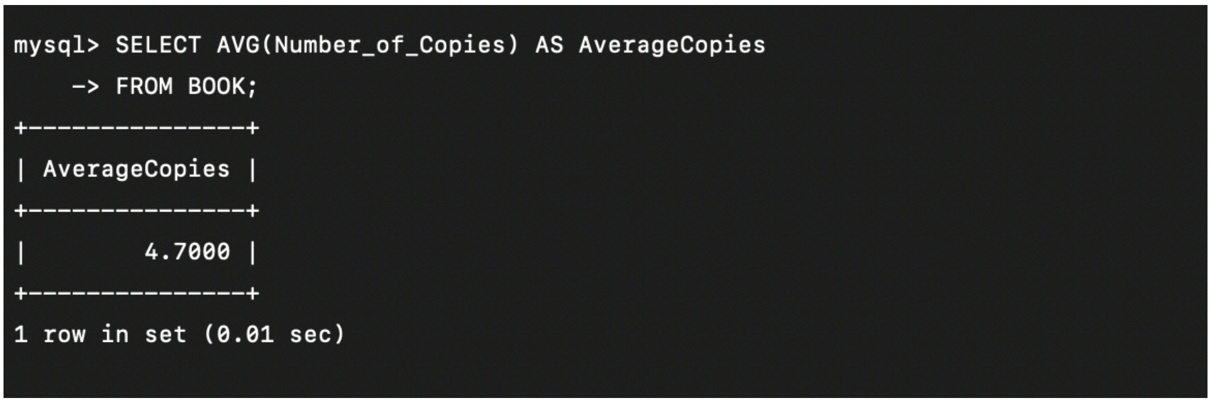
8. Conclusion

This report has provided a comprehensive overview of the Mario library database system. It covered the introduction to database systems, SQL fundamentals, entity and user management, ER mapping and normalization, database design with normal forms, aggregate functions, and a practical SQL query example. The system uses MySQL to manage library operations efficiently, with normalized tables to ensure data integrity and SQL queries to support daily tasks.

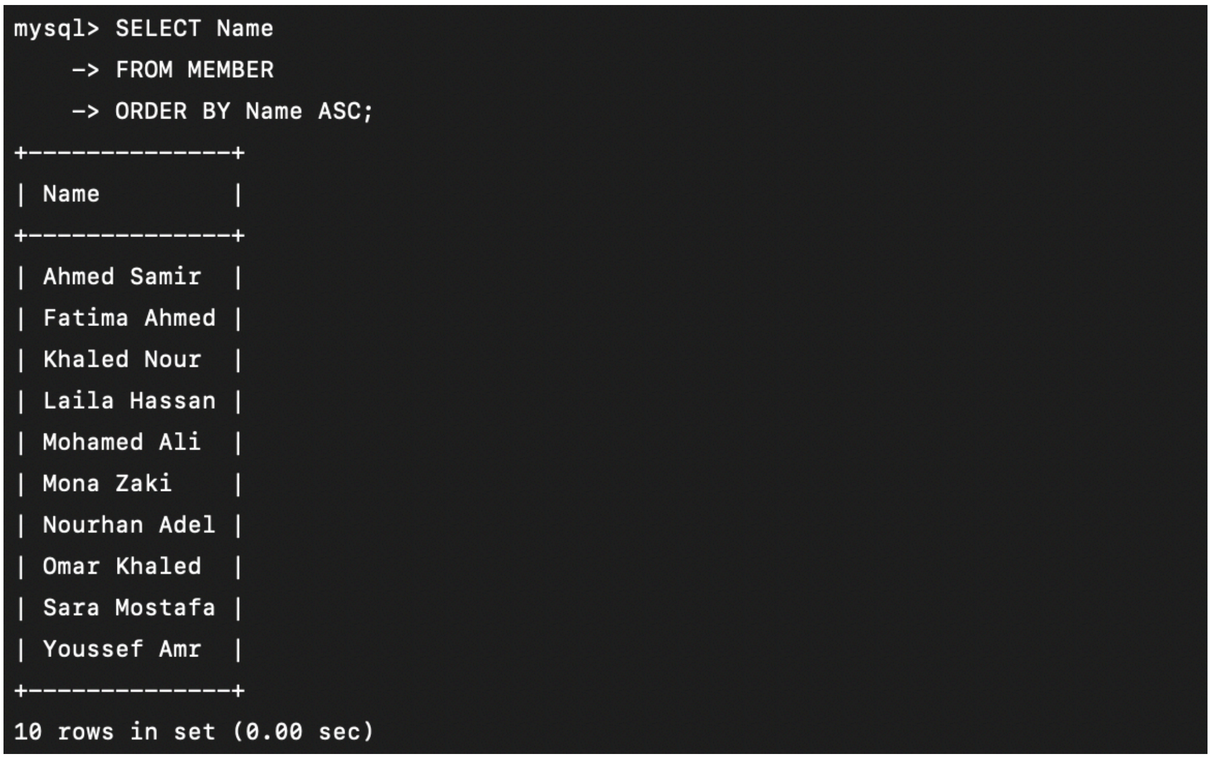
The provided queries and ER diagram demonstrate the system's functionality, from retrieving specific book titles to analyzing inventory with aggregate functions. The Mario database is a robust solution for library management, capable of scaling to meet future needs while maintaining efficiency and accuracy.



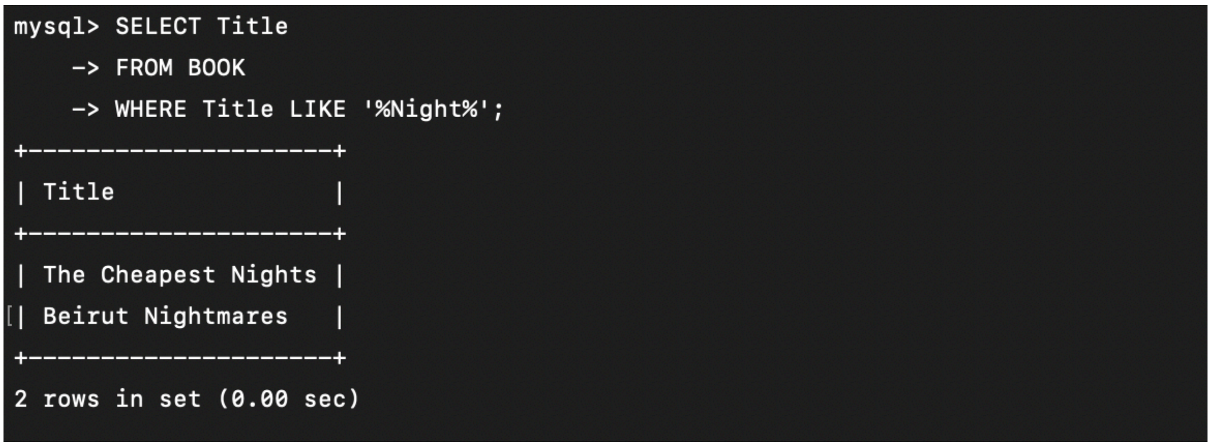
```
[mysql> use mario;
Database changed
[mysql> source /Users/mariomenafaragalla/Desktop/databae project/mario.sql;
mysql> SELECT Title, Number_of_Copies
    -> FROM BOOK
    -> ORDER BY Number_of_Copies DESC;
+-----------------------------------+------------------+
| Title                             | Number_of_Copies |
+-----------------------------------+------------------+
| Utopia                            |                8 |
| Beirut Nightmares                 |                7 |
| The Cheapest Nights               |                6 |
| Palace Walk                       |                5 |
| The Story of Zahra                |                5 |
| The Moon Does Not Tell            |                4 |
| Season of Migration to the North  |                4 |
| The Days                          |                3 |
| Destiny                           |                3 |
| For Bread Alone                   |                2 |
+-----------------------------------+------------------+
10 rows in set (0.01 sec)

mysql> SELECT COUNT(*) AS TotalBooks
    -> FROM BOOK;
+------------+
| TotalBooks |
+------------+
|         10 |
+------------+
1 row in set (0.01 sec)
```



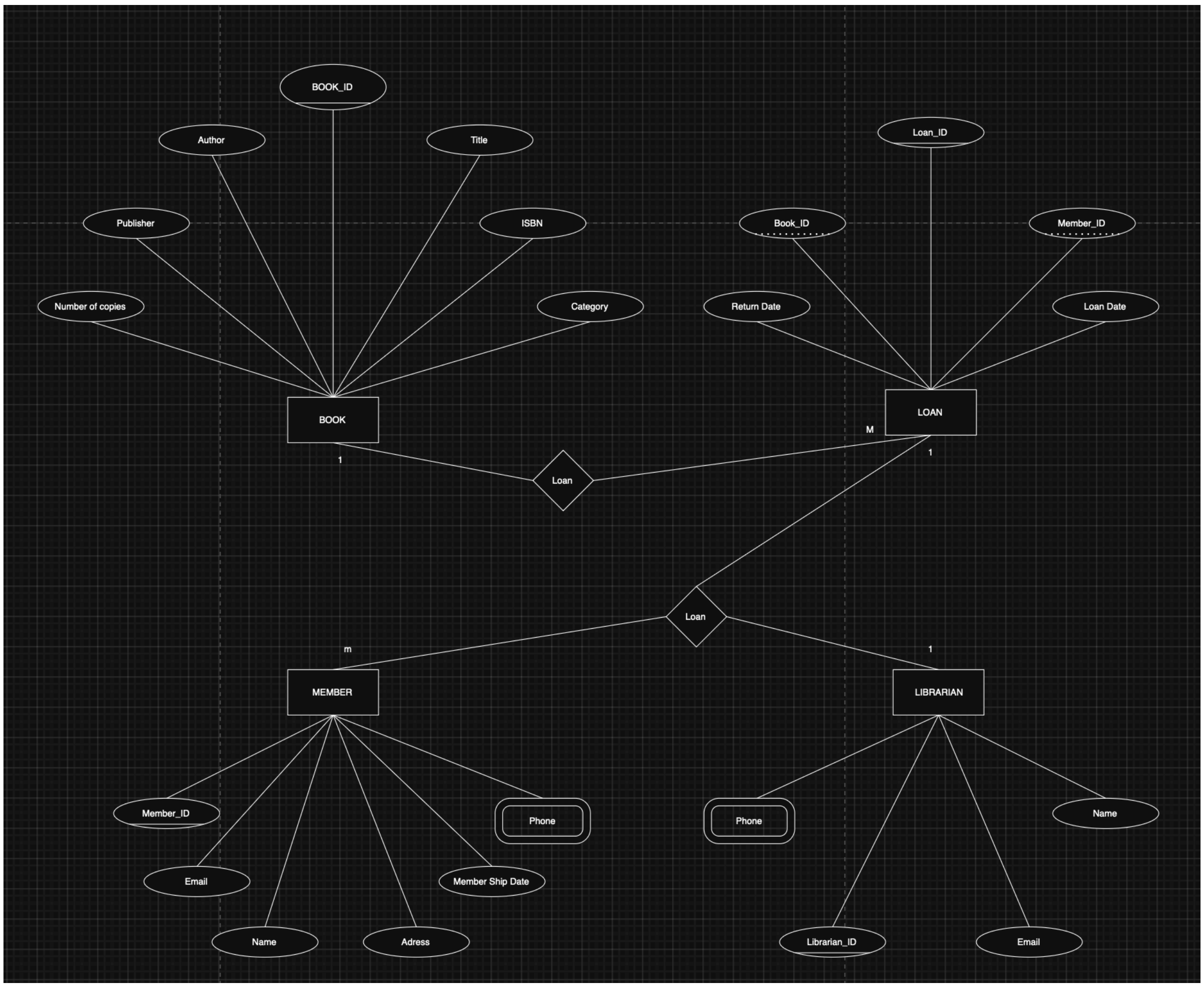
```
mysql> SELECT AVG(Number_of_Copies) AS AverageCopies
    -> FROM BOOK;
+---------------+
| AverageCopies |
+---------------+
|        4.7000 |
+---------------+
1 row in set (0.01 sec)
```



```
mysql> SELECT Name
    -> FROM MEMBER
    -> ORDER BY Name ASC;
+--------------+
| Name         |
+--------------+
| Ahmed Samir  |
| Fatima Ahmed |
| Khaled Nour  |
| Laila Hassan |
| Mohamed Ali  |
| Mona Zaki    |
| Nourhan Adel |
| Omar Khaled  |
| Sara Mostafa |
| Youssef Amr  |
+--------------+
10 rows in set (0.00 sec)
```



```
mysql> SELECT Title
    -> FROM BOOK
    -> WHERE Title LIKE '%Night%';
+---------------------+
| Title               |
+---------------------+
| The Cheapest Nights |
| Beirut Nightmares   |
+---------------------+
2 rows in set (0.00 sec)
```


BOOK_ID
Author
Title
Publisher
ISBN
Number of copies
Category
BOOK
1
Loan
Loan_ID
Book_ID
Member_ID
Return Date
Loan Date
LOAN
M
1
Loan
m
1
MEMBER
LIBRARIAN
Member_ID
Phone
Email
Member Ship Date
Name
Adress
Phone
Name
Librarian_ID
Email

BOOK

Book_ID (PK)	Title	Author_ID (FK)	Publisher_ID (FK)	Category_ID (FK)	ISBN	Number of copies

AUTHOR

Author_ID (PK)	Author Name

MEMBER

Member_ID (PK)	Name	Adress	Phone	Email	Member Ship Date

PUBLISHER

Publisher_ID (PK)	Publisher Name

CATEGORY

Category_ID (PK)	Category Name

LIBRARIAN

Librarian_ID (PK)	Name	Email	Phone

LOAN

Loan_ID (PK)	Book_ID (FK)	Member_ID (FK)	Librarian_ID(FK)	Loan Date	Return Date