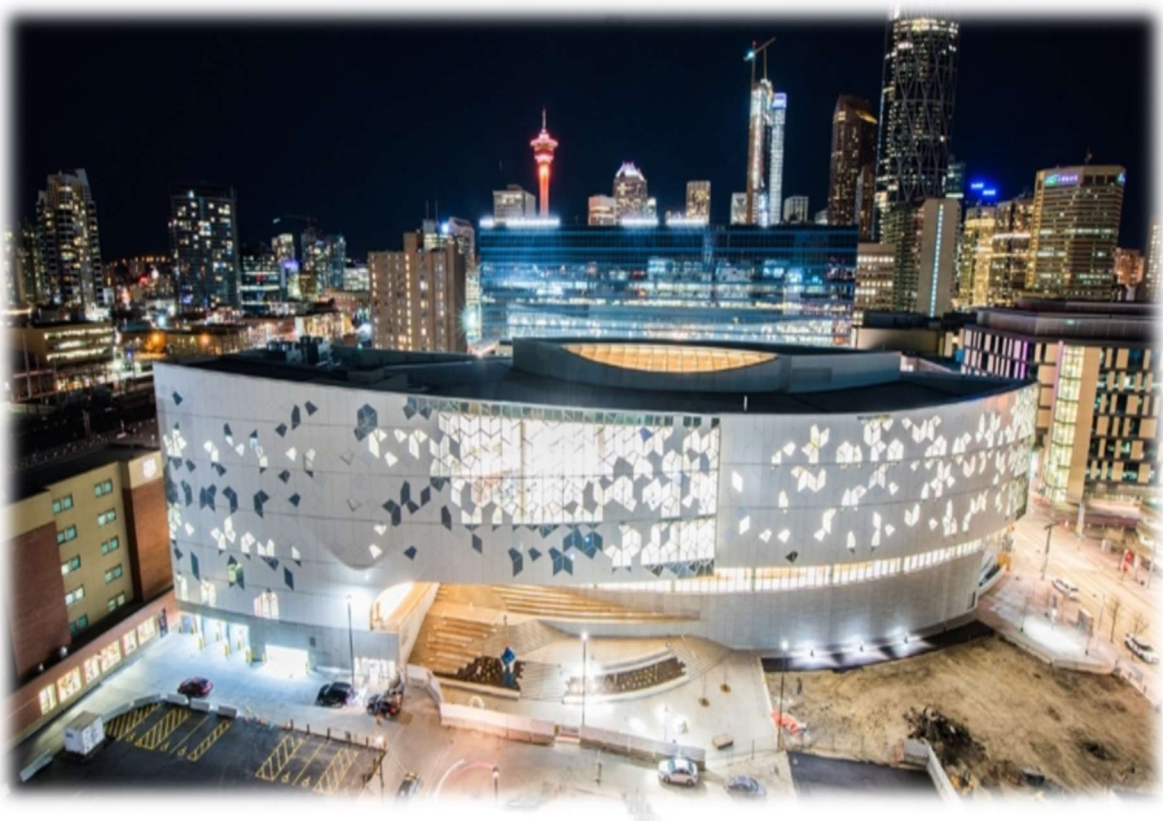





# **DATABASE DESIGN AND DEVELOPMENT FOR LIBRARY MANAGEMENT SYSTEM (LSM)**



**Submitted by: HARLEEN KAUR**

**Submitted to : Dr. Junaid Qazi**

# INDEX

- Introduction
- Mission Statement
  - Lower Expenses of Operations
  - Advance Digital Education
  - Facilitate Easy Access to Resources
- Objective
  - Data Organization
  - Efficiency
  - Scalability
  - Error-Free Transactions
- Database Design
  -  Tables and its fields
    - Books
    - Members
    - Staff
    - Borrowing Transactions
    - Book Categories
    - Reservations
    - Fines
  -  Relationships
  -  Entity Relationship Diagram
- SQL Queries
  - Retrieve All Books in a Specific Category
  - List of Books Borrowed by a Specific Member
  - Find Overdue Books
  - Count of Books in Each Category
  - Members with Active Reservations
  - Add a New Member
  - Record a Book Borrowing Transaction
  - Update a Book's Category
  - Delete a Staff Member Record
- Conclusion
- Appendix

## ❖ Introduction

The Library Management System (LMS) is a digital technology that makes daily activities at libraries run more smoothly. It tracks all the important data about library branches, staff, patrons, and transactions in addition to automating tasks like book borrowing and return. An effective LMS will handle repetitive tasks like book loan and return for the Central Library of Calgary, improve user experience, and make resource management easier. When clear connections are made between different parts of the system, such books, users, and staff, the LMS can grow and change with the library.

## ❖ Mission

### • **Lower Expenses of Operations**

Describe the financial savings that would result from automating labor-intensive tasks like transaction recording, fine management, and book categorization.

### • **Advance Digital Education**

Discuss how the system may improve user interaction with digital platforms, increasing the number of e-books, audiobooks, and online journals that library customers can access.

### • **Facilitate Easy Access to Resources**

Emphasize how the system makes it easier for members to search for, reserve, and check out books by improving their access to library materials.

## ❖ Objectives

1. **Data Organization:** We will effectively arrange data on library branches, staff, books, patrons, and events.
2. **Efficiency:** Our system will provide easy library data retrieval and storage, resulting in smooth operations and excellent customer satisfaction.
3. **Scalability:** We are creating a database that is adaptable to the library's requirements so that it can accommodate more branches and more goods.
4. **Error-Free Transactions:** By using a dependable relational database system, we want to reduce mistakes in inventory management, book loans, and returns.

## ❖ Database Design

By effectively handling data on books, members, staff, and library transactions, the Central Library of Calgary's Library Management System (LMS) database helps to ensure seamless operations. The design is built on the ideas of a relational database, where data is stored in different tables, each representing a particular entity. Primary and foreign keys are used to establish associations between these tables, protecting data integrity and cutting down on redundancy. Scalability is supported by this hierarchical design, which also makes data updates and retrieval quick and error-free.

### 1. Tables and its fields:

We have created an organized collection of database tables in order to achieve these goals:

**1. Books:** Details about each book that is available in the library are kept in the Books table. BookID (primary key), Title, Author, Publisher, Publisher Year, ISBN, and Category ID (foreign key) are among the fields that are included. To make sure that each book can be uniquely tracked in borrowing transactions or reservations, each one has a unique identification (BookID).

Field	Data Type
BookID	INT(PK)
Title	VARCHAR(50)
Author	VARCHAR(100)
Publisher	VARCHAR(15)
Publisher year	DATE
ISBN	INT
Category ID	INT(FK)

**2. Members:** Customers' personal information is kept in the Members database. In addition to further personal information like First and Last Names, Email addresses, phone numbers, addresses, and Membership dates, each member is given a Member ID (primary key). This information is useful in monitoring which member is checking out or reserving books.

Field	Data Type
Member ID	INT(PK)
First Name	VARCHAR(50)
Last Name	VARCHAR(100)
Email	VARCHAR(15)
Phone Number	VARCHAR(15)
Address	VARCHAR(100)
Membership Date	DATE

**3. Staff:** Information on library employees is kept in the Staff table. Each staff member has a unique Staff ID (primary key) and fields like First Name, Last Name, Email, Role, and Date Joined. This table is essential to ensuring that appropriate tracking is kept for the borrowing and return transactions, which are managed by staff members.

Field	Data Type
Staff ID	INT(PK)
First Name	VARCHAR(50)
Last Name	VARCHAR(30)
Email	VARCHAR(15)
Role	VARCHAR(25)
Date_joined	DATE

**4. Borrowing Transactions:** By keeping track of every book that a member loans out, the Borrowing Transactions table helps keep an eye on the movement of library resources. It includes values such as the main key Transaction ID, the borrow date, the due date, and the return date, as well as Book ID (a foreign key supporting the Books table), Member ID (a foreign key supporting the Members table), and Staff ID (a foreign key supporting the Staff database). In the context of borrowing activities, this table connects books, members, and staff, guaranteeing responsibility and accurate record keeping.



Field	Data Type
Transaction ID	INT(PK)
Book ID	INT(FK)
Member ID	INT(FK)
Staff ID	INT(FK)
Borrow_date	DATE
Due_date	DATE
Return_date	DATE

**5. Book categories:** Members can find materials more easily by searching for books by using the Book Categories table, which groups books into several genres or subjects. The primary key in the table is called Category ID, and the category name indicates which category a book falls under. The Category ID connects this table to the Books table, giving the book classification structure.

Field	Data Type
Category ID	INT(PK)
Category Name	VARCHAR(30)

**6. Reservations:** Members' book reservations are tracked in the Reservations table. This table includes fields with the following names: Member ID (foreign key referencing the Members table), Staff ID (foreign key referencing the Staff table), Reservation Date, and Book ID (primary key referencing the Books table). This table facilitates the management of book demand and keeps track of the member who is awaiting a specific book.

Field	Data Type
Reservation ID	INT(PK)
Book ID	INT(FK)
Member ID	INT(FK)
Staff ID	INT(FK)

<b>Reservation date</b>	<b>DATE</b>
-------------------------	-------------

#### 7. Fines:

<b>Field</b>	<b>Data Type</b>
<b>Fine ID</b>	INT(PK)
<b>Transaction ID</b>	INT(FK)
<b>Fine amount</b>	DECIMAL(10,2)
<b>Payment status(paid, unpaid)</b>	VARCHAR(20)

#### ❖ **Relationships:**

In order to establish connections between related entities, we utilize primary keys (PK) and foreign keys (FK) to construct the relationships between the tables in your Library Management System (LMS) database. The linkages between the tables you described operate as follows:

1. **Relationships between the Books Table and the Book Categories Table:**  
The Category ID in the Books table is referred to in the Category ID field.  
Given that numerous books might fall under a single category, this connection is many to one.  
One book can have several reservations, fines, or borrowing transactions, thus the tables for reservations, fines, and borrowing transactions all relate to the same field—the Book ID field. This creates one-to-many relationships.
2. **Members Table links:** One member may make several reservations, borrow multiple books, or accrue multiple fines. As a result, the Member ID field is referenced by the Borrowing Transactions, Reservations, and Fines tables, establishing one-to-many links.
3. **Staff Table links:** Because staff personnel are able to manage numerous reservations and borrowing transactions, one-to-many links are created between the Staff ID field and the tables that hold the borrowing transactions and reservations.
4. **Relationships Between the Borrowing Transactions Table:** The Book ID refers to the Books table. The Members table is referred to by the Member ID. The Staff table is referred to by the Staff ID.  
The Fines table references the Transaction ID column, establishing a one-to-one relationship between each borrowing transaction and the associated fine (if any). This table establishes many-to-one links between books, members, and staff.

5. **Book Categories Table Relationships:** One category may include more than one book since the Books table references the Category ID in a one-to-many relationship.
6. **Relationships Between Reservations Table:** The Book ID refers to the Books table. The Member ID refers to the Members table. The Staff ID refers to the Staff table. This table establishes many-to-one connections between members and the books they reserve, integrating staff in the reservation processing process.
7. **Relationships Between the Fines Table:** The Transaction ID refers to the table of Borrowing Transactions. This is a one-to-one relationship because there could be one fine for each transaction.

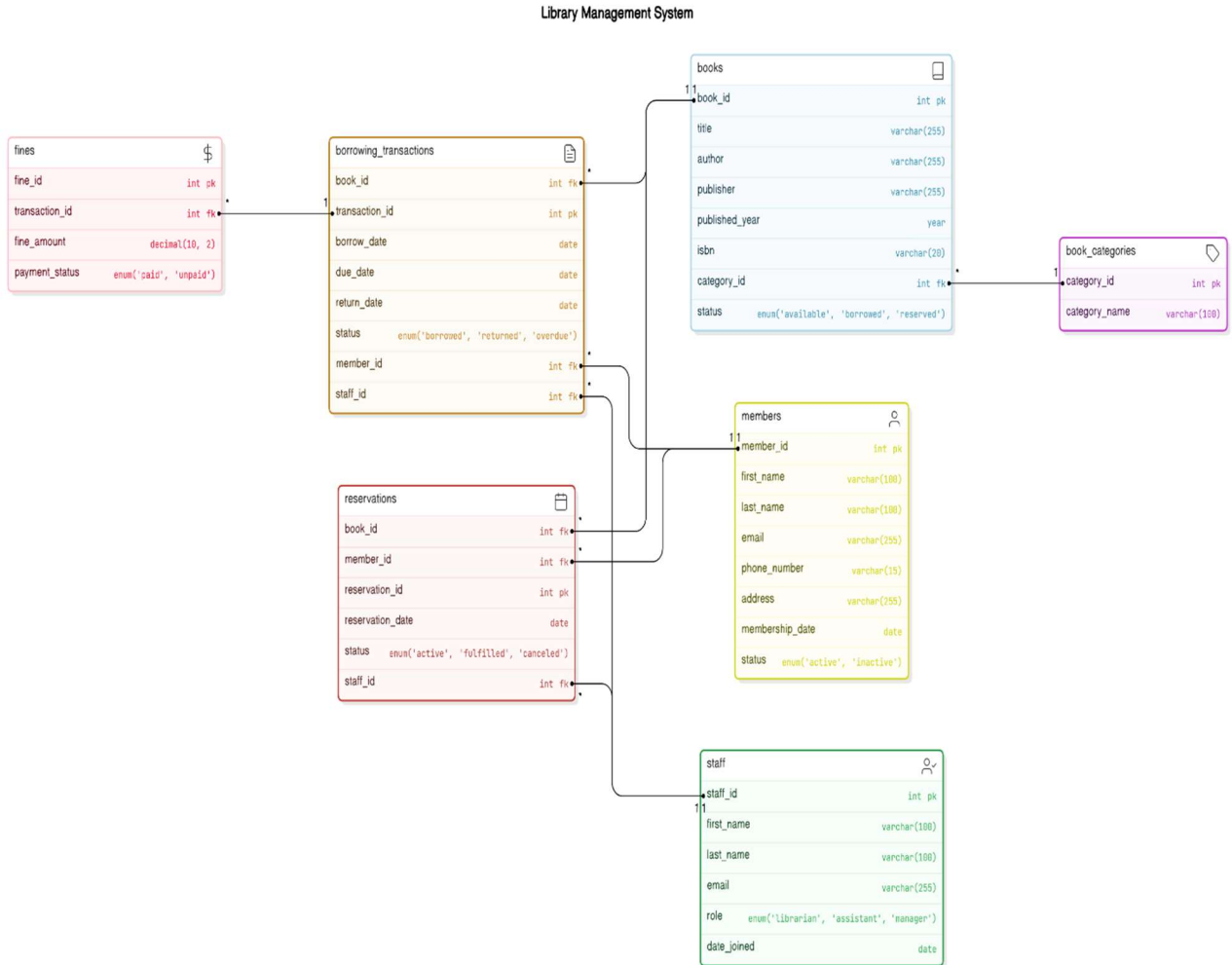
Relationship Entity (ER) Overview:

- ☐ **Books ↔ Book Categories** (Many-to-One)
- ☐ **Members ↔ Borrowing Transactions** (One-to-Many)
- ☐ **Members ↔ Reservations** (One-to-Many)
- ☐ **Staff ↔ Borrowing Transactions** (One-to-Many)
- ☐ **Staff ↔ Reservations** (One-to-Many)
- ☐ **Borrowing Transactions ↔ Fines** (One-to-One)

This design guarantees a relational database that is well-organized, reduces redundancy, scales effectively, and preserves data integrity.



## ❖ ER Diagram



## ❖ SQL Queries

Below are SQL queries that demonstrate common operations within the LMS:

### 1. Retrieve All Books in a Specific Category

```
SELECT Books.Title, Books.Author, BookCategories.CategoryName
FROM Books
JOIN BookCategories ON Books.CategoryID = BookCategories.CategoryID
WHERE BookCategories.CategoryName = 'Science';
```

### 2. List of Books Borrowed by a Specific Member

```
SELECT BorrowingTransactions.TransactionID, Books.Title, BorrowingTransactions.BorrowDate,
BorrowingTransactions.DueDate, BorrowingTransactions.ReturnDate
FROM BorrowingTransactions
JOIN Books ON BorrowingTransactions.BookID = Books.BookID
WHERE BorrowingTransactions.MemberID = 1001; -- Replace 1001 with the actual MemberID
```

### 3. Find Overdue Books

```
SELECT Members.FirstName, Members.LastName, Books.Title, BorrowingTransactions.DueDate
FROM BorrowingTransactions
JOIN Members ON BorrowingTransactions.MemberID = Members.MemberID
JOIN Books ON BorrowingTransactions.BookID = Books.BookID
WHERE BorrowingTransactions.ReturnDate IS NULL AND BorrowingTransactions.DueDate <
CURRENT_DATE;
```

### 4. Count of Books in Each Category

```
SELECT BookCategories.CategoryName, COUNT(Books.BookID) AS NumberOfBooks
FROM Books
JOIN BookCategories ON Books.CategoryID = BookCategories.CategoryID
GROUP BY BookCategories.CategoryName;
```

### 5. Members with Active Reservations

```
SELECT Members.FirstName, Members.LastName, Books.Title, Reservations.ReservationDate
FROM Reservations
JOIN Members ON Reservations.MemberID = Members.MemberID
JOIN Books ON Reservations.BookID = Books.BookID
WHERE Reservations.ReservationDate >= DATE_SUB(CURRENT_DATE, INTERVAL 30 DAY);
```

### 6. Add a New Member

```
INSERT INTO Members (MemberID, FirstName, LastName, Email, PhoneNumber, Address,
MembershipDate)
VALUES (1002, 'Jane', 'Doe', 'jane.doe@example.com', '123-456-7890', '123 Elm Street',
CURRENT_DATE);
```

7. **Record a Book Borrowing Transaction**

```
INSERT INTO BorrowingTransactions (TransactionID, BookID, MemberID, StaffID, BorrowDate,
DueDate)
VALUES (2001, 101, 1002, 501, CURRENT_DATE, DATE_ADD(CURRENT_DATE, INTERVAL 14 DAY));
```

8. **Update a Book's Category**

```
UPDATE Books
SET CategoryID = 3 -- Assuming 3 corresponds to the new category
WHERE BookID = 101;
```

9. **Delete a Staff Member Record**

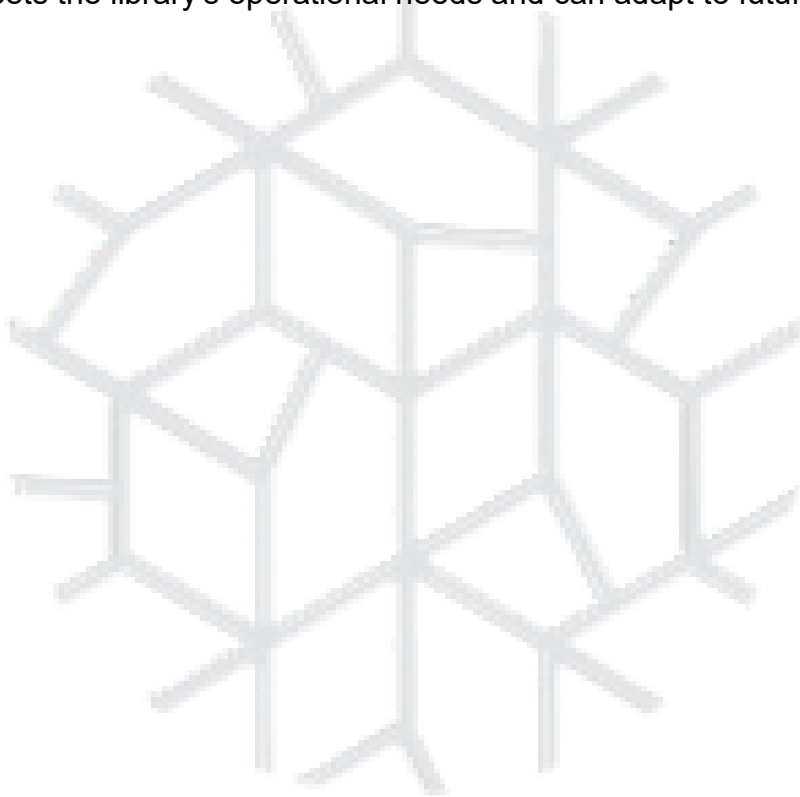
```
DELETE FROM Staff
WHERE StaffID = 501;
```

**ONE JOIN QUERY EXAMPLE FOR THIS:**

```
SELECT
    b.Title AS Book_Title,
    b.Author AS Book_Author,
    m.First_Name AS Member_First_Name,
    m.Last_Name AS Member_Last_Name,
    m.Email AS Member_Email,
    s.First_Name AS Staff_First_Name,
    s.Last_Name AS Staff_Last_Name,
    bt.Borrow_date,
    bt.Due_date,
    bt.Return_date,
    f.Fine_amount,
    f.Payment_status
FROM
    Borrowing_Transactions bt
JOIN
    Books b ON bt.Book_ID = b.BookID -- Join on Book ID
JOIN
    Members m ON bt.Member_ID = m.Member_ID -- Join on Member ID
JOIN
    Staff s ON bt.Staff_ID = s.Staff_ID -- Join on Staff ID
LEFT JOIN
    Fines f ON bt.Transaction_ID = f.Transaction_ID -- Left Join to get fines (if any)
ORDER BY
    bt.Borrow_date DESC; -- Ordering by most recent borrowing transactions
```

## ❖ Conclusion

The detailed database design ensures that the LMS is robust, scalable, and capable of handling complex operations with ease. By establishing clear relationships and enforcing data integrity through primary and foreign keys, the system minimizes errors and enhances efficiency. Thorough testing with various SQL queries validates the functionality and reliability of the database, ensuring that it meets the library's operational needs and can adapt to future expansions.



## Appendix

### ❖ Testing

Testing the database is essential to make sure that the constraints, relationships, and data integrity all function as intended after it has been designed. To make sure relationships work as intended, test cases should involve adding data, editing records, removing entries, and querying the database.

#### Test Case Examples

**1. *Insert Test:***

Place a fresh book, member, and staff into the appropriate tables, and make sure the IDs are accurately recorded and auto-incremented.

**2. *Test using Foreign Keys:***

Attempt to add a borrowing transaction using a Member ID or Book ID that is not present in the Members or Books tables, respectively. Data consistency should be ensured by the system raising an error.

**3. *Test Update:***

Modify a borrowing transaction's due date and make sure the table accurately reflects the update.

**4. *Eliminate Test:***

Make an effort to remove a book associated with a loan transaction. Unless the foreign key constraints are appropriately managed (e.g., the borrowing transaction is finished first), the system ought to prohibit this behavior.