**Library Management System**

**Project Overview**

Create a database for managing a library’s collection of books, members, and borrowing records. This system will track the books available in the library, the members who can borrow books, and the details of borrowed books.

**Key Features**

1. **Books Management**
   * Add, update, and delete book records.
   * Track book details such as title, author, genre, ISBN, and availability status.
2. **Members Management**
   * Add, update, and delete member records.
   * Store member details like name, address, phone number, and membership date.
3. **Borrowing Records**
   * Record the borrowing and returning of books.
   * Track which member borrowed which book and when it is due to be returned.
4. **Queries and Reports**
   * Generate reports of books currently borrowed and their due dates.
   * List overdue books and the members who borrowed them.
   * Search for books by title, author, or genre.
   * List all members and their borrowing history.

**Steps to Complete the Project**

1. **Database Design**
   * **Books Table**
     + book\_id (Primary Key)
     + title
     + author
     + genre
     + isbn
     + availability\_status
   * **Members Table**
     + member\_id (Primary Key)
     + name
     + address
     + phone
     + membership\_date
   * **Borrowing Table**
     + borrow\_id (Primary Key)
     + book\_id (Foreign Key referencing Books)
     + member\_id (Foreign Key referencing Members)
     + borrow\_date
     + due\_date
     + return\_date
2. **Set Up the Database**
   * **Create the database and tables using MySQL.**
   * **Define relationships between tables using foreign keys**.

*In the given schema, the Books, Members, and Borrowing tables are related to each other through foreign key constraints. Here’s how these relationships are defined and how they connect:*

***Relationships Between the Tables***

1. ***Books and Borrowing****:*
   * ***Relationship****: A book can be borrowed many times, but each borrowing record is associated with a single book.*
   * ***Foreign Key****: book\_Id in the Borrowing table is a foreign key that references book\_Id in the Books table.*

*This relationship ensures that every book\_Id in the Borrowing table must exist in the Books table. If a record in the Books table is deleted, any corresponding borrowings will be handled according to the specified ON DELETE action (which is not explicitly defined but defaults to RESTRICT).*

1. ***Members and Borrowing****:*
   * ***Relationship****: A member can borrow many books, but each borrowing record is associated with a single member.*
   * ***Foreign Key****: Member\_Id in the Borrowing table is a foreign key that references Member\_Id in the Members table.*

*This relationship ensures that every Member\_Id in the Borrowing table must exist in the Members table. Similar to the Books table, if a record in the Members table is deleted, any corresponding borrowings will be handled according to the specified ON DELETE action (which defaults to RESTRICT).*

***Schema with Foreign Key Constraints***

*Here's the schema with the relationships between the tables defined:*

**CREATE TABLE** Borrowing (

borrow\_Id **INT AUTO\_INCREMENT PRIMARY KEY**,

book\_Id **INT**,

Member\_Id **INT**,

borrow\_date **DATE**,

due\_date **DATE**,

return\_date **DATE**,

**FOREIGN KEY** (book\_Id) **REFERENCES** Books(book\_Id)

**ON DELETE RESTRICT** *-- Optionally specify actions here*

**ON UPDATE CASCADE**, -- *Optionally specify actions here*

**FOREIGN KEY** (Member\_Id) **REFERENCES** Members(Member\_Id)

**ON DELETE RESTRICT** -- *Optionally specify actions here*

**ON UPDATE CASCADE** -- *Optionally specify actions here*

);

***Explanation of Foreign Key Actions***

* ***ON DELETE RESTRICT****: If you attempt to delete a record from the Books or Members table that is referenced in the Borrowing table, the deletion will be restricted. This means you cannot delete a book or member that is currently referenced in the Borrowing table.*
* ***ON UPDATE CASCADE****: If the book\_Id or Member\_Id in the Books or Members table is updated, the corresponding values in the Borrowing table will be updated automatically to maintain referential integrity.*

*You can adjust the ON DELETE and ON UPDATE actions according to your application's requirements. For instance, ON DELETE CASCADE would automatically delete borrowings when a book or member is deleted, but this might not always be desirable depending on your use case.*

1. **Implement CRUD Operations**
   * Write SQL queries to create, read, update, and delete records in each table.
2. **Advanced Queries**
   * Write complex queries to generate the required reports and searches.
   * Handle cases such as checking for overdue books.

**Creating the Database and Tables:**

CREATE DATABASE LibraryManagement;

USE LibraryManagement;

CREATE TABLE Books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

genre VARCHAR(100),

isbn VARCHAR(20) UNIQUE,

availability\_status BOOLEAN DEFAULT TRUE

);

CREATE TABLE Members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

address VARCHAR(255),

phone VARCHAR(15),

membership\_date DATE

);

CREATE TABLE Borrowing (

borrow\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

member\_id INT,

borrow\_date DATE,

due\_date DATE,

return\_date DATE,

FOREIGN KEY (book\_id) REFERENCES Books(book\_id),

FOREIGN KEY (member\_id) REFERENCES Members(member\_id)

);

**Inserting Records:---Examples**

INSERT INTO Books (title, author, genre) VALUES

('The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', '9780743273565'),

('To Kill a Mockingbird', 'Harper Lee', 'Fiction', '9780060935467');

INSERT INTO Members (name, address, phone, membership\_date) VALUES

('John Doe', '123 Elm St', '555-1234', '2024-01-01'),

('Jane Smith', '456 Oak St', '555-5678', '2024-02-15');

**Recording a Borrowing Event:**

INSERT INTO Borrowing (book\_id, member\_id, borrow\_date, due\_date) VALUES

(1, 1, '2024-07-18', '2024-08-01');

**Generating a Report of Overdue Books:**

SELECT b.title, m.name, br.due\_date

FROM Borrowing br

JOIN Books b ON br.book\_id = b.book\_id

JOIN Members m ON br.member\_id = m.member\_id

WHERE br.return\_date IS NULL AND br.due\_date < CURDATE();

PRESENTATION: Prepare a document to briefly explain the database structure, the relationships between tables and the quires performed.

***Example of Presentation***

**Library Management System Database Structure and Queries**

**Introduction**

The Library Management System (LMS) is designed to manage a library's inventory of books, member information, and borrowing records. This document outlines the database structure, relationships between tables, and sample queries to demonstrate how the system functions.

**Database Structure**

The LMS database consists of three main tables: Books, Members, and Borrowing. These tables are related to one another to ensure data integrity and facilitate complex queries.

**Tables**

1. **Books Table**
   * **Description:** Stores information about the books available in the library.
   * **Columns:**
     + book\_id: INT, Primary Key, Auto Increment
     + title: VARCHAR(255), NOT NULL
     + author: VARCHAR(255), NOT NULL
     + genre: VARCHAR(100)
     + isbn: VARCHAR(20), UNIQUE
     + availability\_status: BOOLEAN, DEFAULT TRUE
2. **Members Table**
   * **Description:** Stores information about library members.
   * **Columns:**
     + member\_id: INT, Primary Key, Auto Increment
     + name: VARCHAR(255), NOT NULL
     + address: VARCHAR(255)
     + phone: VARCHAR(15)
     + membership\_date: DATE
3. **Borrowing Table**
   * **Description:** Records details of book borrowings.
   * **Columns:**
     + borrow\_id: INT, Primary Key, Auto Increment
     + book\_id: INT, Foreign Key referencing Books(book\_id)
     + member\_id: INT, Foreign Key referencing Members(member\_id)
     + borrow\_date: DATE
     + due\_date: DATE
     + return\_date: DATE

**Relationships**

* Each book can be borrowed by multiple members over time, creating a one-to-many relationship between the **Books** and **Borrowing** tables.
* Each member can borrow multiple books over time, creating a one-to-many relationship between the **Members** and **Borrowing** tables.

**SQL Statements**

**Creating the Database and Tables**

sql

CREATE DATABASE LibraryManagement;

USE LibraryManagement;

CREATE TABLE Books (

book\_id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

author VARCHAR(255) NOT NULL,

genre VARCHAR(100),

isbn VARCHAR(20) UNIQUE,

availability\_status BOOLEAN DEFAULT TRUE

);

CREATE TABLE Members (

member\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

address VARCHAR(255),

phone VARCHAR(15),

membership\_date DATE

);

CREATE TABLE Borrowing (

borrow\_id INT AUTO\_INCREMENT PRIMARY KEY,

book\_id INT,

member\_id INT,

borrow\_date DATE,

due\_date DATE,

return\_date DATE,

FOREIGN KEY (book\_id) REFERENCES Books(book\_id),

FOREIGN KEY (member\_id) REFERENCES Members(member\_id)

);

**Inserting Records**

**Adding Books:**

INSERT INTO Books (title, author, genre, isbn) VALUES

('The Great Gatsby', 'F. Scott Fitzgerald', 'Fiction', '9780743273565'),

('To Kill a Mockingbird', 'Harper Lee', 'Fiction', '9780060935467');

**Adding Members:**

INSERT INTO Members (name, address, phone, membership\_date) VALUES

('John Doe', '123 Elm St', '555-1234', '2024-01-01'),

('Jane Smith', '456 Oak St', '555-5678', '2024-02-15');

**Borrowing Records**

**Recording a Borrowing Event:**

INSERT INTO Borrowing (book\_id, member\_id, borrow\_date, due\_date) VALUES

(1, 1, '2024-07-18', '2024-08-01');

**Advanced Queries**

**Generating a Report of Overdue Books:**

SELECT b.title, m.name, br.due\_date

FROM Borrowing br

JOIN Books b ON br.book\_id = b.book\_id

JOIN Members m ON br.member\_id = m.member\_id

WHERE br.return\_date IS NULL AND br.due\_date < CURDATE();

**List All Members and Their Borrowing History:**

SELECT m.name, b.title, br.borrow\_date, br.due\_date, br.return\_date

FROM Members m

JOIN Borrowing br ON m.member\_id = br.member\_id

JOIN Books b ON br.book\_id = b.**book\_id;**

**Search for Books by Title:**

**SELECT \* FROM Books WHERE title LIKE '%Mockingbird%';**

**List All Books Currently Borrowed:**

**SELECT b.title, m.name, br.borrow\_date, br.due\_date**

**FROM Borrowing br**

**JOIN Books b ON br.book\_id = b.book\_id**

**JOIN Members m ON br.member\_id = m.member\_id**

**WHERE br.return\_date IS NULL;**

**Conclusion**

**The Library Management System database provides a structured way to manage books, members, and borrowing records. By creating well-defined tables and establishing relationships, the system ensures data integrity and supports complex queries for efficient library management. This project is an excellent way to practice and enhance your MySQL skills.**