

# LIBRARY MANAGEMENT SYSTEM

## INTRODUCTION

This project focuses on creating a simple Library Management System using a relational database. The system manages Books, Members, and Transactions. It allows for efficient management of books, member records, and book transactions (borrow and return). This system also ensures that the available copies of each book are updated in real time based on borrowing and returning actions, using triggers to automate these updates.

## OBJECTIVE

The objective of this project is to:

- Implement a relational database schema to manage books, members, and transactions.
- Automate the update of available copies of books when borrowed and returned.
- Provide SQL queries for common library operations like adding books, updating book details, deleting books, and handling transactions.

## KEY FEATURES

### ○ Books Table:

Stores information about books like BookID, Title, Author, Genre, PublishedYear, TotalCopies, and AvailableCopies.

### ○ Members Table:

Stores details of library members including MemberID, Name, Email, Phone number, and JoinDate.

- Transactions Table:

Tracks borrowing and returning of books, with details such as TransactionID, MemberID, BookID, BorrowDate, and ReturnDate.

- Triggers:

Decrease\_AvailableCopies: Automatically decreases the available copies when a new book is borrowed (when ReturnDate is NULL).

Increase\_AvailableCopies: Automatically increases the available copies when a book is returned (when ReturnDate is not NULL).

## DATABASE SCHEMA

- Books Table:

create table Books(

BookID integer primary key,

Title varchar(100) not null,

Author varchar(100) not null,

Genre varchar(50),

PublishedYear integer,

TotalCopies integer not null,

AvailableCopies integer not null);

- Members Table:

create table Members(

MemberID integer primary key,

Name varchar(100) not null,

Email varchar(100) unique not null,

Phone number unique not null,

JoinDate date default current\_date);

- Transactions Table:

create table Transactions(

TransactionID integer primary key,

MemberID integer not null,

BookID integer not null,

BorrowDate date default current\_date,

ReturnDate date,

foreign key (MemberID) references Members (MemberID),

foreign key (BookID) references Books (BookID));

## **LIMITATIONS**

- Basic Functionality:

The system only manages book borrowing and returning. It does not include features like reservations, overdue fees, or book suggestions.

Error Handling:

The system does not implement comprehensive error handling for cases such as trying to borrow a book with no available copies.

- Scalability:

The system is designed for small to medium-sized libraries and might need optimizations for large-scale operations.

# FUTURE IMPROVEMENTS

- Overdue Book Handling:

Implement a feature to track overdue books and impose fines.

- Book Reservation:

Add functionality to allow members to reserve books that are currently unavailable.

- Enhanced User Interface:

Develop a web or mobile interface for easier interaction with the library system.

- Reporting:

Add reporting capabilities, such as the most borrowed books, active members, etc.

# SYSTEM STUDY

- System Overview:

This Library Management System (LMS) aims to streamline the management of books, members, and transactions in a library environment. The system uses a relational database to store data related to books, library members, and the borrowing/returning of books. SQL queries and triggers are used to handle various operations like adding, updating, deleting books, and automating the process of managing available copies when books are borrowed or returned.

- System Objectives:

- To provide an efficient way of managing library data.
- To automate book borrowing and returning operations using triggers.
- To ensure the integrity and consistency of available book copies using database triggers.

- System Requirements:

Hardware Requirements:

- A computer system capable of running SQL-based database management systems like Oracle or MySQL.
- Sufficient storage for the database and backup files.

### Software Requirements:

- Database Management System (DBMS) such as Oracle or MySQL.
- SQL client tools (e.g., SQL\*Plus, MySQL Workbench).

## SYSTEM DESIGN

### ○ Database Design:

The database is designed with three main tables: Books, Members, and Transactions.

### ○ Books Table:

Holds the details of books such as book ID, title, author, genre, and available copies.

The AvailableCopies field is automatically updated through triggers when a book is borrowed or returned.

### ○ Members Table:

Stores information about library members, including member ID, name, email, phone number, and the join date.

### ○ Transactions Table:

Tracks the borrowing and returning of books.

It references both the Members and Books tables to establish relationships.

### ○ Data Flow:

A member borrows a book, creating a record in the Transactions table.

The Decrease\_AvailableCopies trigger is fired, updating the AvailableCopies of the book in the Books table.

When a book is returned, the Increase\_AvailableCopies trigger is fired, incrementing the AvailableCopies field.

## MODULES

- Books Module:

Create: Allows the addition of new books to the library.

Update: Enables updates to existing books (e.g., change the number of total and available copies).

Delete: Deletes a book from the library when it is no longer needed.

- Members Module:

Create: Adds new members to the library system.

Update: Modifies member details (e.g., name, phone number, or email).

Delete: Removes a member from the system when they are no longer a part of the library.

- Transactions Module:

Borrow Book: A member borrows a book, which is recorded in the Transactions table and decreases the AvailableCopies.

Return Book: A member returns a book, updating the Transactions table and increasing the AvailableCopies.

- Triggers Module:

Decrease\_AvailableCopies: Decreases the available copies of a book when a transaction is inserted (book borrowed).

Increase\_AvailableCopies: Increases the available copies when a transaction is updated (book returned).

## CONCLUSION

The Library Management System developed here is a simple yet effective solution for managing a small to medium-sized library. It leverages the power of relational databases and SQL triggers to automate the update of book availability as transactions (borrow and return) occur.

- **Key Takeaways:**
  - The system ensures data consistency and reduces manual updates by automating the process of tracking available copies of books.
  - The modular design allows for easy future enhancements, such as implementing overdue book tracking or adding book reservations.
  - The design is scalable for small to medium-sized libraries, and with future improvements, it could support more complex features for larger libraries.

## CODING

SQL\*Plus: Release 21.0.0.0.0 - Production on Tue Jan 14 19:42:13 2025

Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Enter user-name: system

Enter password:

Last Successful login time: Tue Jan 14 2025 19:40:46 +05:30

Connected to:

Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

Version 21.3.0.0.0

## #CREATING TABLES

(1)BOOKS:

SQL> create table Books(

- 2 BookID integer primary key,
- 3 Title varchar(100) not null,
- 4 Author varchar(100) not null,
- 5 Genre varchar(50),
- 6 PublishedYear integer,
- 7 TotalCopies integer not null,
- 8 AvailableCopies integer not null);

Table created.

(2)MEMBERS:

SQL> create table Members(

- 2 MemberID integer primary key,
- 3 Name varchar(100) not null,
- 4 Email varchar(100) unique not null,
- 5 Phone number unique not null,
- 6 JoinDate date default current\_date);

Table created.



(3)TRANSACTIONS:

SQL> create table Transactions(

2 TransactionID integer primary key,

3 MemberID integer not null,

4 BookID integer not null,

5 BorrowDate date default current\_date,

6 ReturnDate date,

7 foreign key (MemberID) references Members (MemberID),

8 foreign key (BookID) references Books (BookID));

Table created.

## #DESCRIBING TABLES

(1)BOOKS:

SQL> desc Books;

Name	Null?	Type
-----		
BOOKID	NOT NULL	NUMBER(38)
TITLE	NOT NULL	VARCHAR2(100)
AUTHOR	NOT NULL	VARCHAR2(100)
GENRE		VARCHAR2(50)
PUBLISHEDYEAR		NUMBER(38)
TOTALCOPIES	NOT NULL	NUMBER(38)

AVAILABLECOPIES                      NOT NULL NUMBER(38)

(2)MEMBERS:

SQL> desc Members;

Name	Null?	Type
-----		
MEMBERID		NOT NULL NUMBER(38)
NAME		NOT NULL VARCHAR2(100)
EMAIL		NOT NULL VARCHAR2(100)
PHONE		NOT NULL NUMBER
JOINDATE		DATE

(3)TRANSACTIONS:

SQL> desc Transactions;

Name	Null?	Type
-----		
TRANSACTIONID		NOT NULL NUMBER(38)
MEMBERID		NOT NULL NUMBER(38)
BOOKID		NOT NULL NUMBER(38)
BORROWDATE		DATE
RETURNDATE		DATE

#INSERTING VALUES

(1)BOOKS:

```
SQL> insert into Books Values(1,'The Great Gatsby','F.Scott Fitzgerald','Fiction',1925,5,5);
```

1 row created.

```
SQL> insert into Books Values(2,'1984','George Orwell','Dystopian',1949,3,3);
```

1 row created.

```
SQL> insert into Books Values(3,'To Kill a Mockingbird','Harper Lee','Fiction',1960,4,4);
```

1 row created.

(2)MEMBERS:

```
SQL> insert into Members values(10,'Alice Johnson','alice@gmail.com',8921829654,to_date('2024-06-24','yy-mm-dd'));
```

1 row created.

```
SQL> insert into Members values(20,'Bob Smith','bob@gmail.com',9495696542,to_date('2024-08-12','yy-mm-dd'));
```

1 row created.

(3)TRANSACTIONS:

```
SQL> insert into Transactions values(11,10,1,to_date('2024-06-28','yy-mm-dd'),to_date('2024-07-28','yy-mm-dd'));
```

1 row created.

```
SQL> insert into Transactions values(22,20,3,to_date('2024-09-05','yy-mm-dd'),to_date('2024-10-05','yy-mm-dd'));
```

1 row created.

## #SQL QUERIES

(1)INSERT A NEW BOOK:

```
SQL> insert into Books values(4,'Pride and Prejudice','Jane Austen','Classic',1813,2,2);
```

1 row created.

(2)UPDATE BOOK DETAILS:

```
SQL> update Books set TotalCopies=6,AvailableCopies=6 where BookID=1;
```

1 row updated.

(3)DELETE A BOOK:

```
delete from Books where Title='1948';
```

1 rows deleted.

(4)RETRIEVING COLUMNS FROM TABLE:

(.)SQL> select Title,Genre from Books;

TITLE

-----

GENRE

-----

The Great Gatsby

Fiction

1984

Dystopian

To Kill a Mockingbird

Fiction

Pride and Prejudice

Classic

(.)SQL> select \* from Members;

MEMBERID

-----

NAME

-----

EMAIL

-----

PHONE

-----

JOINDATE

-----

10

Alice Johnson

alice@gmail.com

8921829654

24-JUN-24

MEMBERID

-----

NAME

-----

EMAIL

-----

PHONE

-----

JOINDATE

-----

20

Bob Smith

bob@gmail.com

9495696542

12-AUG-24

(.)SQL> select BorrowDate,ReturnDate from Transactions;

BORROWDAT RETURNDATE

-----

28-JUN-24 28-JUL-24

05-SEP-24 05-OCT-24

## #ADVANCED FEATURES

(1)ADD A TRIGGER TO UPDATE AVAILABLE COPIES ON BORROW

SQL> CREATE OR REPLACE TRIGGER Decrease\_AvailableCopies

2 AFTER INSERT ON Transactions

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.ReturnDate IS NULL THEN

6 UPDATE Books

7 SET AvailableCopies = AvailableCopies - 1

8 WHERE BookID = :NEW.BookID;

9 END IF;

10 END;

11 /

Trigger created.

(2)ADD A TRIGGER TO UPDATE AVAILABLE COPIES ON RETURN

SQL> CREATE OR REPLACE TRIGGER Increase\_AvailableCopies

2 AFTER UPDATE OF ReturnDate ON Transactions

3 FOR EACH ROW

4 BEGIN

5 IF :NEW.ReturnDate IS NOT NULL THEN

6 UPDATE Books

7 SET AvailableCopies = AvailableCopies + 1

8 WHERE BookID = :NEW.BookID;

9 END IF;

10 END;

11 /

Trigger created.

## OUTPUT SCREENSHOT



```
SQL Plus
Copyright (c) 1982, 2021, Oracle. All rights reserved.

Enter user-name: system
Enter password:
Last Successful login time: Tue Jan 14 2025 19:40:46 +05:30

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> create table Books(
  2 BookID integer primary key,
  3 Title varchar(100) not null,
  4 Author varchar(100) not null,
  5 Genre varchar(50),
  6 PublishedYear integer,
  7 TotalCopies integer not null,
  8 AvailableCopies integer not null);

Table created.

SQL> create table Members(
  2 MemberID integer primary key,
  3 Name varchar(100) not null,
  4 Email varchar(100) unique not null,
  5 Phone number unique not null,
  6 JoinDate date default current_date);

Table created.

SQL> create table Transactions(
  2 TransactionID integer primary key,
  3 MemberID integer not null,
  4 BookID integer not null,
  5 BorrowDate date default current_date,
  6 ReturnDate date,
  7 foreign key (MemberID) references Members (MemberID),
  8 foreign key (BookID) references Books (BookID));

Table created.
```

```
SQL> desc Books;
```

Name	Null?	Type
BOOKID	NOT NULL	NUMBER(38)
TITLE	NOT NULL	VARCHAR2(100)
AUTHOR	NOT NULL	VARCHAR2(100)
GENRE		VARCHAR2(50)
PUBLISHEDYEAR		NUMBER(38)
TOTALCOPIES	NOT NULL	NUMBER(38)
AVAILABLECOPIES	NOT NULL	NUMBER(38)

```
SQL> desc Members;
```

Name	Null?	Type
MEMBERID	NOT NULL	NUMBER(38)
NAME	NOT NULL	VARCHAR2(100)
EMAIL	NOT NULL	VARCHAR2(100)
PHONE	NOT NULL	NUMBER
JOINDATE		DATE

```
SQL> desc Transactions;
```

Name	Null?	Type
TRANSACTIONID	NOT NULL	NUMBER(38)
MEMBERID	NOT NULL	NUMBER(38)
BOOKID	NOT NULL	NUMBER(38)
BORROWDATE		DATE
RETURNDATE		DATE

```
SQL> insert into Books Values(1,'The Great Gatsby','F.Scott Fitzgerald','Fiction',1925,5,5);
1 row created.

SQL> insert into Books Values(2,'1984','George Orwell','Dystopian',1949,3,3);
1 row created.

SQL> insert into Books Values(3,'To Kill a Mockingbird','Harper Lee','Fiction',1960,4,4);
1 row created.
```

```
SQL> insert into Members values(10,'Alice Johnson','alice@gmail.com',8921829654,to_date('2024-06-24','yy-mm-dd'));
1 row created.

SQL> insert into Members values(20,'Bob Smith','bob@gmail.com',9495696542,to_date('2024-08-12','yy-mm-dd'));
1 row created.
```

```
SQL> insert into Transactions values(11,10,1,to_date('2024-06-28','yy-mm-dd'),to_date('2024-07-28','yy-mm-dd'));
1 row created.

SQL> insert into Transactions values(22,20,3,to_date('2024-09-05','yy-mm-dd'),to_date('2024-10-05','yy-mm-dd'));
1 row created.
```

```
SQL> CREATE OR REPLACE TRIGGER Decrease_AvailableCopies
  2  AFTER INSERT ON Transactions
  3  FOR EACH ROW
  4  BEGIN
  5  IF :NEW.ReturnDate IS NULL THEN
  6  UPDATE Books
  7  SET AvailableCopies = AvailableCopies - 1
  8  WHERE BookID = :NEW.BookID;
  9  END IF;
 10  END;
 11  /
```

Trigger created.

```
SQL> CREATE OR REPLACE TRIGGER Increase_AvailableCopies
  2  AFTER UPDATE OF ReturnDate ON Transactions
  3  FOR EACH ROW
  4  BEGIN
  5  IF :NEW.ReturnDate IS NOT NULL THEN
  6  UPDATE Books
  7  SET AvailableCopies = AvailableCopies + 1
  8  WHERE BookID = :NEW.BookID;
  9  END IF;
 10  END;
 11  /
```

Trigger created.

\*\*\*\*\*