UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

LIBRARY MANAGEMENT SYSTEM

Student Name: Manan Jain **UID: 23BCA10439**
Branch: UIC/BCA **Section/Group: BCA-2/b**
Semester: 4th **Submitted to: Mrs. Harkamal Kaur**
Subject Name: DBMS **Subject Code: 23CAP-252**

➢ Aim/Overview of the project:

The aim of this project is to create a Library Management System that efficiently organizes books, authors, and member transactions. It enables streamlined operations like borrowing and returning books, maintaining records, and performing database queries for effective library management.

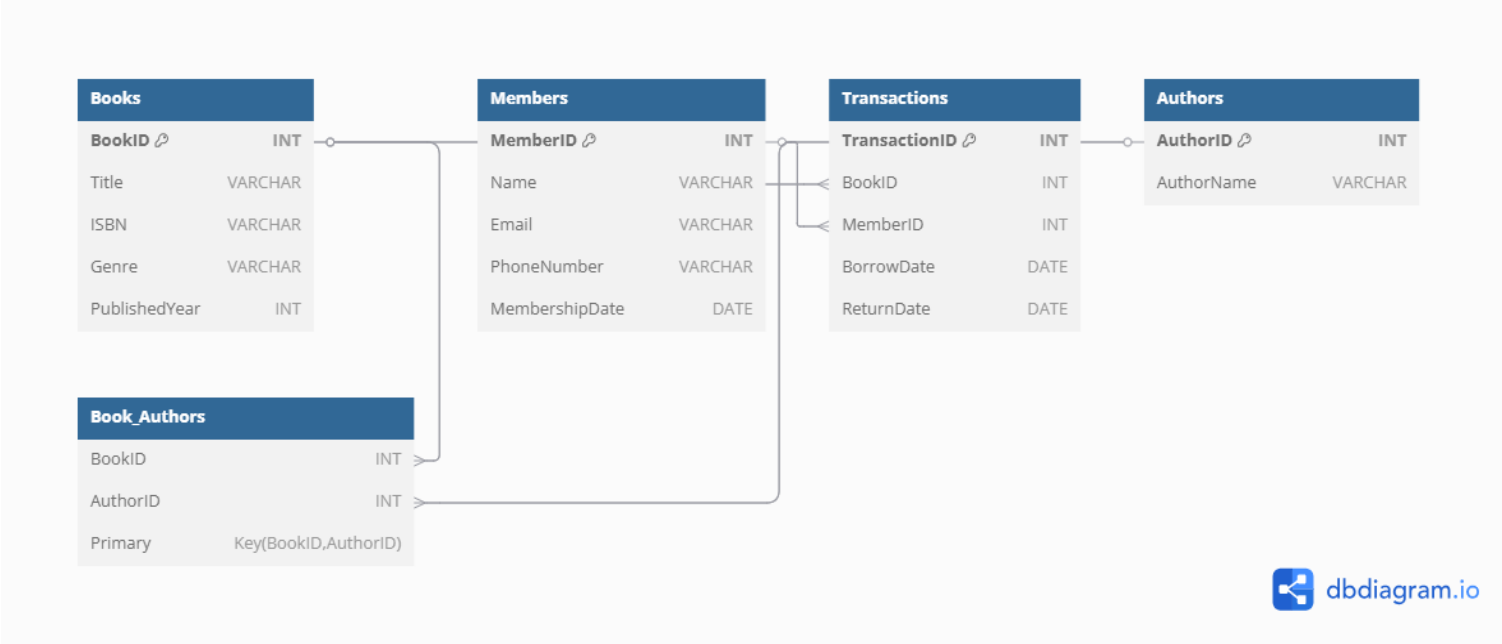
➢ ER Diagram & Schema

Database Schema

Here's a summary of the schema design:

1. **Books Table**:
   - Columns: `BookID (Primary Key)`, `Title`, `ISBN`, `Genre`, `PublishedYear`.
2. **Members Table**:
   - Columns: `MemberID (Primary Key)`, `Name`, `Email`, `PhoneNumber`, `MembershipDate`.
3. **Transactions Table**:
   - Columns: `TransactionID (Primary Key)`, `BookID (Foreign Key)`, `MemberID (Foreign Key)`, `BorrowDate`, `ReturnDate`.
4. **Authors Table**:
   - Columns: `AuthorID (Primary Key)`, `AuthorName`.
5. **Book_Authors Table**:
   - Columns: `BookID (Foreign Key)`, `AuthorID (Foreign Key)`.

The schema adheres to normalization principles, ensuring no data redundancy while maintaining data integrity.



➢ SQL Queries & Output

1) SELECT * FROM Books WHERE Genre = 'Fiction';

```
(3, 203),
(4, 204),
(5, 205);
SELECT * FROM Books WHERE Genre = 'Fiction';

INSERT INTO Members VALUES (104, 'Bob Brown', 'bobbrown@example.com', '9876512345', '
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

BookID	Title	ISBN	Genre	PublishedYear
1	The Catcher in the Rye	1234567890	Fiction	1951
2	To Kill a Mockingbird	1234567891	Fiction	1960
NULL	NULL	NULL	NULL	NULL

Books 2

2) **INSERT INTO Members VALUES (104, 'Bob Brown', 'bobbrown@example.com', '9876512345', '2025-04-05');**

15 10:34:02 INSERT INTO Members VALUES (104, 'Bob Brown', 'bobbrown@example.com', '9876512345'... 1 row(s) affected

3) **UPDATE Books SET Genre = 'Historical Fiction' WHERE BookID = 2;**

16 10:35:42 UPDATE Books SET Genre = 'Historical Fiction' WHERE BookID = 2 1 row(s) affected Rows match

Result Grid | Filter Rows: | Edit: | Export/Import:

BookID	Title	ISBN	Genre	PublishedYear
1	The Catcher in the Rye	1234567890	Fiction	1951
2	To Kill a Mockingbird	1234567891	Historical Fiction	1960
3	1984	1234567892	Dystopian	1949
4	Pride and Prejudice	1234567893		1813
5	The Great Gatsby	1234567894		1925
NULL	NULL	NULL	NULL	NULL

1234567893

Books 3

4) **DELETE FROM Transactions WHERE TransactionID = 1002;**

18 10:37:55 DELETE FROM Transactions WHERE TransactionID = 1002 1 row(s) affected

Result Grid | Filter Rows: | Edit:

TransactionID	BookID	MemberID	BorrowDate	ReturnDate
1001	1	101	2025-04-03	NULL
1003	3	103	2025-04-04	NULL
NULL	NULL	NULL	NULL	NULL

5) **SELECT Members.Name, Books.Title, Transactions.BorrowDate**
FROM Transactions
JOIN Members ON Transactions.MemberID = Members.MemberID
JOIN Books ON Transactions.BookID = Books.BookID;

Result Grid | Filter Rows: | Export:

Name	Title	BorrowDate
John Doe	The Catcher in the Rye	2025-04-03
Alice Johnson	1984	2025-04-04

Result 5

6) **SELECT Genre, COUNT(*) AS TotalBooks**
FROM Books
GROUP BY Genre;

Result Grid | Filter Rows:

Genre	TotalBooks
Fiction	1
Historical Fiction	1
Dystopian	1
Romance	1
Classic	1

Result 6

7) **SELECT SUM(PublishedYear) AS TotalPublishedYears FROM Books;**

Result Grid | Filter Rows:

	TotalPublishedYears
▶	9598

Result 7

8) **SELECT AVG(PublishedYear) AS AveragePublishedYear FROM Books;**

Result Grid | Filter Rows:

	AveragePublishedYear
▶	1919.6000

Result 8

9) **SELECT MAX(PublishedYear) AS LatestPublishedYear FROM Books;**

Result Grid | Filter Rows: | Ex

	LatestPublishedYear
▶	1960

Result 9

10) SELECT MIN(PublishedYear) AS OldestPublishedYear FROM Books;

Result Grid | Filter Rows:

OldestPublishedYear
1813

Result 10

11) SELECT Genre, COUNT(*) AS TotalBooks FROM Books GROUP BY Genre;

Result Grid | Filter Rows:

Genre	TotalBooks
Fiction	1
Historical Fiction	1
Dystopian	1
Romance	1
Classic	1

Result 11

12) SELECT Genre, AVG(PublishedYear) AS AveragePublishedYear FROM Books GROUP BY Genre;

Result Grid | Filter Rows:

Genre	AveragePublishedYear
Fiction	1951.0000
Historical Fiction	1960.0000
Dystopian	1949.0000
Romance	1813.0000
Classic	1925.0000

Result 12

13) **SELECT Genre, COUNT(*) AS TotalBooks**
FROM Books
GROUP BY Genre
HAVING COUNT(*) > 1;

Result Grid | Filter Rows:

Genre	TotalBooks

Result 13

14) **SELECT Members.Name, COUNT(Transactions.TransactionID) AS BooksBorrowed**
FROM Transactions
JOIN Members ON Transactions.MemberID = Members.MemberID
GROUP BY Members.Name;

Result Grid | Filter Rows:

Name	BooksBorrowed
John Doe	1
Alice Johnson	1

Result 14

- **Conclusion:**

Observations

1. The schema is well-structured, ensuring data consistency through relationships and foreign key constraints.
2. Queries such as selection, insertion, updates, and deletions are straightforward to execute for this schema.
3. The normalization of the database avoids data redundancy and ensures data integrity.

Limitations

1. This system doesn't handle advanced functionalities like reservation of books or overdue fines.
2. No authentication or user roles (e.g., admin, member) are implemented in this basic system.
3. Scalability might be limited for a large-scale library with millions of records unless database optimization techniques are applied.

4.The schema doesn't currently support multimedia resources like eBooks or audiobook.