Library Borrowing Tracker System

Final Report for Database CSA

Farrel Tsaqif Anindyo [1] Akhtar Gadang Abimanyu [1] Faiq Athaya Urumsah [1]

[1]UGM
Group 7

December 20, 2025

Presentation Outline

Primary Objectives

- Reliance on manual logging or spreadsheets.
- Issues: Duplicate records, lost data, and poor inventory visibility.
- Difficulty in tracking overdue books.

- **Digitize and Automate** core library operations.
- Create a centralized relational database.
- Provide interfaces for **Borrowers** and **Librarians**.
- Ensure data integrity (Normalization, Constraints).

System Users

- **Borrower (Student/User)**
  - Register and log in.
  - View available books (catalog).
  - Request to borrow books and view active loan status.
- **Librarian (Admin)**
  - Manage inventory (add/remove books).
  - Approve borrowing requests (sets loan duration).
  - Process book returns.

Conceptual Design

- Focuses on relationships between four core entities:
  1. **Borrower**
  2. **Book**
  3. **Librarian**
  4. **Borrowing** (Transaction/Associative Entity)

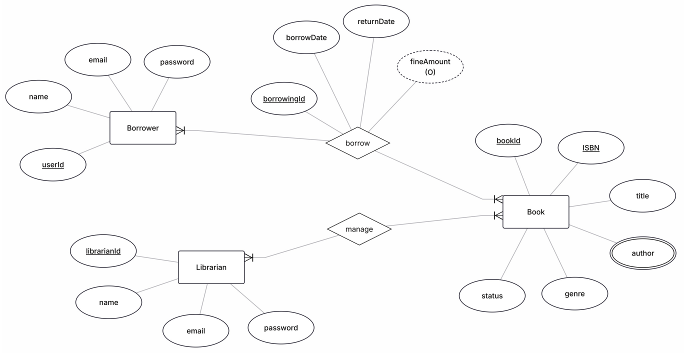

Figure: Entity Relationship Diagram (ERD)

Database Normalization (Achieved 3NF)

- **1NF:** Ensured atomic values and unique records.
- **2NF:** Separated `Borrowing History` into its own table to ensure full dependency on the Primary Key.
- **3NF:** Removed transitive dependencies (e.g., book details tied strictly to `bookID`).

Physical Implementation (MySQL)

- Uses Primary Keys (PK) for uniqueness and Foreign Keys (FK) for linkage.
- The central `Borrowing` table links `Book`, `Borrower`, and `Librarian`.

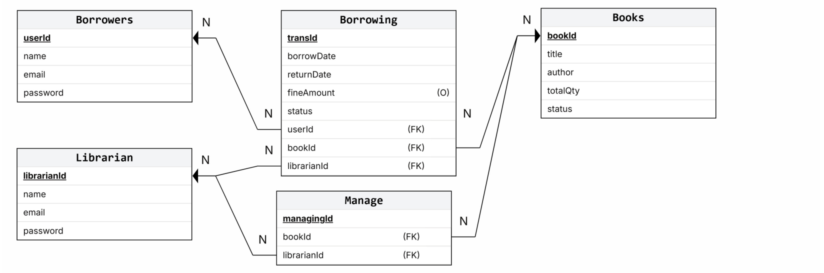

Figure: Relational Schema

SQL Creation Example

Listing 1: Borrowing Table Creation (Excerpt)

```
CREATE TABLE Borrowing (
    borrowID INT NOT NULL PRIMARY KEY,
    userID INT,
    bookID INT,
    status ENUM('pending', 'approved', 'returned'),
    FOREIGN KEY (bookID) REFERENCES Book (bookID),
    FOREIGN KEY (userID) REFERENCES Borrower (userID)
        ON DELETE CASCADE
);
```

- **Logic:** Book request → `status`='pending'. Approved → `bookStatus` updated to 'borrowed'.

Application Flow (Node.js/Express + MySQL)

- **Borrower Dashboard**:
  - Sends POST request for loan.
  - Status shows "PENDING" initially.
- **Librarian Dashboard (Approvals)**:
  - Views pending requests.
  - Approves request and sets loan duration (calculates `returnDate`).
- **Return Process**:
  - Librarian processes return by `borrowID`.
  - Updates book status to 'AVAILABLE' and calculates fine if overdue.

Backend API Verification (Postman)

Create (Registration)

- `POST /borrower/register` → Result: `200 OK`.

Read (Get Books)

- `GET /books` → Returns JSON array of inventory.

Update (Borrowing Flow)

- `POST /borrow` → "Request sent."
- `POST /borrow/approve` → "Approved."

Delete (Book History)

- Deletion of a Book also deletes associated history (due to FK constraints).

Summary and Reflection

- **Success:** Implemented a functional Library Borrowing Tracker, meeting the core objective of digitizing the library workflow.
- **Key Learning:** Valuable experience integrating MySQL with a Node.js backend and managing Foreign Key constraints.

Challenges & Future Improvements

- Synchronizing status updates between Books and Borrowing tables was a challenge.
- **Future Work:** Implement automated fine calculations and email notifications for overdue books.