Library Management System Design Document

Contents

1 Conceptual & Logical Design

1.1 Functional Requirements

- **Book Catalog Management:** Staff shall be able to add, update, and delete book records (title, ISBN, publication year, category) through a web interface.
- **Search & Discovery:** Patrons shall search the catalog by title, author, ISBN, or category with autocomplete suggestions.
- **Loan/Return Processing:** The system shall record each loan (LoanDate, DueDate) and return event, automatically calculating overdue status and fines.
- **Member Management:** Staff shall register new members, edit member profiles (name, contact, membership type), and deactivate accounts as needed.
- **Author & Publisher Records:** Maintain separate AUTHOR and PUBLISHER entities to avoid redundancy when multiple books share the same creators or publishers.
- **Reporting & Analytics:** Authorized users shall generate reports on overdue loans, most-borrowed titles, and member activity over custom date ranges.

1.2 Non-Functional Requirements

- **Availability:** System must achieve $\geq 99.9\%$ uptime for 24×7 access.
- **Performance:** Search and loan transactions should complete within 200 ms under normal load (up to 100 req/s).
- **Scalability:** Handle at least 50,000 active members and 200,000 catalog entries, with horizontal scaling for peak periods.
- **Data Integrity & Durability:** All write operations shall use ACID-compliant transactions; daily backups with off-site replication.
- **Security:** Role-based access control (staff vs. patron), encrypted connections (TLS), and audit logging of critical operations.
- **Usability & Accessibility:** Web UI must be responsive and WCAG-compliant, supporting desktop and mobile browsers.

1.3 Entity–Relationship Diagram

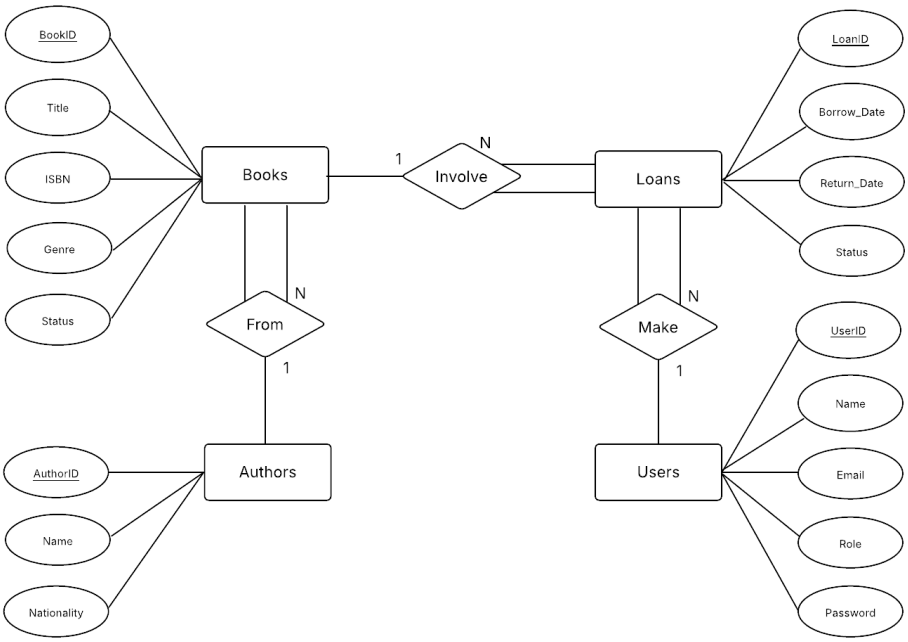

Figure 1: Entity–Relationship Diagram for Library Management System

1.4 Normalization Proof (up to 3NF)

1NF:

Raw relation:

```
RawLib(MemberID, Name, Email, BookID, Title, ISBN, Category, PublisherID,
       PublisherName, AuthorID, AuthorName, LoanID, LoanDate, DueDate, ReturnDate)
```

2NF: Remove partial dependencies on composite keys.

- MEMBER(MemberID → Name, Email)
- BOOK(BookID → Title, ISBN, Category, PublisherID)
- PUBLISHER(PublisherID → PublisherName)
- AUTHOR(AuthorID → AuthorName)
- LOAN(LoanID, MemberID, BookID, LoanDate, DueDate, ReturnDate)

3NF: No transitive dependencies.

- All relations: attributes depend only on the primary key.
- Final relations are free of partial and transitive dependencies.

2 Physical Schema Definition

2.1 DDL Scripts

```
-- AUTHOR Table
CREATE TABLE Author (
    Author_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Nationality VARCHAR(50)
);

-- PUBLISHER Table
CREATE TABLE Publisher (
    Publisher_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Country VARCHAR(50)
);

-- USER Table (Borrower)
CREATE TABLE User (
    User_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Role ENUM('Admin', 'Librarian', 'User') NOT NULL,
    Password VARBINARY(255) NOT NULL
);

-- BOOK Table
CREATE TABLE Book (
    Book_ID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    Author_ID INT NOT NULL,
    Publisher_ID INT NOT NULL,
    Genre VARCHAR(50),
    ISBN VARCHAR(20) UNIQUE,
    Status ENUM('Available', 'Borrowed', 'Reserved') DEFAULT '
        Available',
    FOREIGN KEY (Author_ID) REFERENCES Author(Author_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (Publisher_ID) REFERENCES Publisher(
        Publisher_ID)
        ON DELETE CASCADE ON UPDATE CASCADE
```

```
);

-- LOAN Table
CREATE TABLE Loan (
    Loan_ID INT AUTO_INCREMENT PRIMARY KEY,
    Book_ID INT NOT NULL,
    User_ID INT NOT NULL,
    Borrow_Date DATE NOT NULL,
    Return_Date DATE,
    Status ENUM('Borrowed', 'Returned', 'Overdue') DEFAULT '
        Borrowed',
    FOREIGN KEY (Book_ID) REFERENCES Book(Book_ID)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

2.2 Views, Indexes, and Partitioning

```
-- View: Active Loans
CREATE VIEW ActiveLoans AS
SELECT
    l.Loan_ID,
    b.Title,
    u.Name AS Borrower,
    l.Borrow_Date,
    l.Return_Date,
    l.Status
FROM
    Loan l
JOIN Book b ON l.Book_ID = b.Book_ID
JOIN User u ON l.User_ID = u.User_ID
WHERE
    l.Status = 'Borrowed';

-- Indexes
CREATE INDEX idx_book_author ON Book(Author_ID);
CREATE INDEX idx_book_publisher ON Book(Publisher_ID);
CREATE INDEX idx_loan_user ON Loan(User_ID);
CREATE INDEX idx_loan_book ON Loan(Book_ID);

-- Partitioning: by year of loan
ALTER TABLE Loan
PARTITION BY RANGE (YEAR(Borrow_Date)) (
    PARTITION p2023 VALUES LESS THAN (2024),
    PARTITION p2024 VALUES LESS THAN (2025),
    PARTITION pMax  VALUES LESS THAN MAXVALUE
);
```

3 Task Division & Project Plan

3.1 Responsibilities

- Can Ha An: ER modeling, normalization, DDL implementation
- Le Gia Duc: Stored procedures, triggers, data loading scripts
- Nguyen Nhat Minh: Website development (CRUD, reporting, authentication)

3.2 Timeline Overview

- Activity 1 – Conceptual design and ERD (May 10–13)
- Activity 2 – Physical implementation and sample data (May 14–16)
- Activity 3 – Testing and verification (May 17–18)
- Activity 4 – Performance tuning (May 19–21)
- Activity 5 – Final demo and presentation prep (May 22–26)

4 Supporting Documentation

4.1 Design Justification

- Separate AUTHOR and PUBLISHER tables prevent redundancy across books.
- Use of surrogate key `Loan_ID` ensures each loan transaction is uniquely tracked, even with repeated borrowings.
- Indexes improve performance for reporting and frequent loan lookups.

4.2 Sample Data Loading Notes

- Sample data to be imported using CSV and MySQL's `LOAD DATA INFILE`.
- Foreign key dependencies respected by import order: Publisher → Author → Book → User → Loan.