**Final project for the course of Database Systems [DT0347]**  
**a.a. 2023/2024**

ESTHER ABIODUN OSIKOYA – 287290

## 

## Requirements analysis

**R1:** Users must register for the service by providing their full name, address, and email.

**R2:** Users can search for and add books to their virtual bookshelves.

**R3:** Users can write reviews and rate books they have read.

**R4:** The platform must suggest books to users based on their preferences.

**R5:** Users can follow other users to see their reviews and ratings.

**R6:** Users can join or create book clubs to discuss their favorite books.

**R7:** The platform must keep track of the books users add to their shelves, ratings, reviews, and the dates these actions occurred.

**R8:** Users can tag books with genres and other descriptive tags to help categorize them.

## Use Cases analysis

**U1:** Users can list books of a specific genre they have rated at least a certain score.

**U2:** Users can view new book releases in a given genre or by a specific author.

**U3:** Administrators can list the email addresses of users who have not logged in for a certain period, along with their most recent bookshelf activity.

**U4:** The platform can recommend books to users based on their reading history and preferences.

## Conceptual design

A diagram of a network

Description automatically generated

**Entity types:**

* User [R1, R2, R3, R5, R6,R7, U1, U3, U4]
* Virtual\_bookshelf [R2,R7, U3]
* Review[R3, R7, U1, U4]
* Bookclub[R6, ]
* Book[R2, R3,R7, R8, U1, U2, U4]
* Tag[R8, U4]

**Relationship types:**

* **User**(n) < Add\_book  > **VirtualBookshelf**(m) [R2, R7, U3]
* **VirtualBookshelf**(m) <Contains> **Books** m)
* **User**(m)  <Write**>  Review** (m) [R3, R4, R7, U1, U4]
* Review(m) <on **>**Book(m)
* **User**(n) <join**>** Bookclub (m) [R6, ]
* Book(n) <has**>Tag**(m) [R4, R8, U4]
* **User**(n) <Follow**>** User(m) [R5, U4]

## Logical design

**Step1: Mapping of Regular Entity Types**

**USER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UserID | Fullname | Address | Email | Lastlogindate |

**Virtual\_bookshelf**

|  |  |
| --- | --- |
| ShelfID | Name |

**Review**

|  |  |  |  |
| --- | --- | --- | --- |
| ReviewID | Ratings | ReviewText | Date |

**BookClub**

|  |  |  |
| --- | --- | --- |
| ClubID | Clubname | Description |

**Book**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BookID | Author | Title | ReleaseDate | Genre | Publisher |

**Tag**

|  |  |
| --- | --- |
| Tagname | TagID |

• **Step 2: Mapping of Weak Entity Types**I do not have any weak entity type.

**• Step 3: Mapping of Binary 1:1 Relation Types**I do not have any one-to-one binary relationship type.

**• Step 4: Mapping of Binary 1: N Relationship Types.**

I do not have any one-to-N relationship type.

**• Step 5: Mapping of Binary M: N Relationship Types.**

For each regular binary M: N relationship type Add\_book, Contains, Write, Follow, Join, and Has, I created new relations UserBookshelves, BookshelfBooks, UserReviews, UserFollows, UserBookClubs, and BookTags to represent the relationships. These are relationship relations. I included as foreign key attributes in UserBookshelves, BookshelfBooks, UserReviews, UserFollows, UserBookClubs, and BookTags the primary keys of the relations that represent the participating entity types which are User, Book, VirtualBookshelf, Review, BookClub, and Tag. Their combination will form the primary keys of UserBookshelves, BookshelfBooks, UserReviews, UserFollows, UserBookClubs, and BookTags. Additionally, I included any simple attributes of the original M

relationship type as attributes of UserBookshelves, BookshelfBooks, UserReviews, UserFollows, UserBookClubs, and BookTags.

**UserBookshelves**

|  |  |  |  |
| --- | --- | --- | --- |
| UserID | ShelfID | BookID | DateAdded |

**BookshelfBooks**

|  |  |
| --- | --- |
| ShelfID | BookID |

**UserReviews**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| UserID | ReviewID | BookID | Rating | ReviewText | Date |

**UserFollows**

|  |  |
| --- | --- |
| FollowerID | FolloweeID |

**UserBookClubs**

|  |  |  |
| --- | --- | --- |
| UserID | ClubID | DateJoined |

**BookTags**

|  |  |
| --- | --- |
| BookID | TagID |

## SQL schema implementation

Below is the SQL schema I created after completing the relational model.

-- Create tables

CREATE TABLE User (

UserID INT AUTO\_INCREMENT PRIMARY KEY,

Fullname VARCHAR(100),

Address VARCHAR(255),

Email VARCHAR(100),

LastLoginDate DATE

);

CREATE TABLE VirtualBookshelf (

ShelfID INT AUTO\_INCREMENT PRIMARY KEY,

Name VARCHAR(50)

);

CREATE TABLE Book (

BookID INT AUTO\_INCREMENT PRIMARY KEY,

Title VARCHAR(255),

Author VARCHAR(100),

Genre VARCHAR(50),

ReleaseDate DATE,

Publisher VARCHAR(100)

);

CREATE TABLE Review (

ReviewID INT AUTO\_INCREMENT PRIMARY KEY,

Rating INT CHECK (Rating BETWEEN 1 AND 5),

ReviewText TEXT,

Date DATE,

UserID INT,

BookID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE BookClub (

ClubID INT AUTO\_INCREMENT PRIMARY KEY,

ClubName VARCHAR(100),

Description TEXT

);

CREATE TABLE Tag (

TagID INT AUTO\_INCREMENT PRIMARY KEY,

TagName VARCHAR(50)

);

-- Step 2: Mapping of M:N Relationships

CREATE TABLE UserBookshelves (

UserID INT,

ShelfID INT,

BookID INT,

DateAdded DATE,

Status ENUM('read', 'currently reading', 'want to read'),

PRIMARY KEY (UserID, ShelfID, BookID),

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ShelfID) REFERENCES VirtualBookshelf(ShelfID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE BookshelfBooks (

ShelfID INT,

BookID INT,

PRIMARY KEY (ShelfID, BookID),

FOREIGN KEY (ShelfID) REFERENCES VirtualBookshelf(ShelfID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE UserReviews (

UserID INT,

ReviewID INT,

BookID INT,

PRIMARY KEY (UserID, ReviewID),

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ReviewID) REFERENCES Review(ReviewID),

FOREIGN KEY (BookID) REFERENCES Book(BookID)

);

CREATE TABLE UserFollows (

FollowerID INT,

FolloweeID INT,

PRIMARY KEY (FollowerID, FolloweeID),

FOREIGN KEY (FollowerID) REFERENCES User(UserID),

FOREIGN KEY (FolloweeID) REFERENCES User(UserID)

);

CREATE TABLE UserBookClubs (

UserID INT,

ClubID INT,

DateJoined DATE,

PRIMARY KEY (UserID, ClubID),

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (ClubID) REFERENCES BookClub(ClubID)

);

CREATE TABLE BookTags (

BookID INT,

TagID INT,

PRIMARY KEY (BookID, TagID),

FOREIGN KEY (BookID) REFERENCES Book(BookID),

FOREIGN KEY (TagID) REFERENCES Tag(TagID)

);

## Use cases implementation

**U1:** Users can list books of a specific genre they have rated at least a certain score.  
 SELECT b.Title, b.Author, r.Rating

FROM Book b

JOIN Review r ON b.BookID = r.BookID

WHERE b.Genre = 'magni' AND r.Rating = 2;

**U2:** Users can view new book releases in a given genre or by a specific author.

SELECT Title, Author, ReleaseDate

FROM Book

WHERE Genre = 'rem' AND ReleaseDate >= '1977-08-27'

OR Author = 'est' AND ReleaseDate >= '1977-08-27';

**U3:** Administrators can list the email addresses of users who have not logged in for a certain period, along with their most recent bookshelf activity.

SELECT u.Email, MAX(ub.DateAdded) AS LastActivity, ub.Status

FROM User u

LEFT JOIN UserBookshelves ub ON u.UserID = ub.UserID

WHERE u.LastLoginDate < DATE\_SUB(CURDATE(), INTERVAL 364 DAY)

GROUP BY u.UserID, u.Email, ub.Status;

**U4:** The platform can recommend books to users based on their reading history and preferences.

SELECT b.Title, b.Author

FROM Book b

JOIN Review r ON b.BookID = r.BookID

JOIN UserFollows uf ON uf.FolloweeID = r.UserID

WHERE uf.FollowerID = 6

GROUP BY b.BookID

ORDER BY AVG(r.Rating) DESC

LIMIT 10;

**REPORT**I started the project by identifying the entities and relationships, using requirement analysis and use cases as guides. This process required several iterations, including renaming entities to ensure they appropriately represented the data and relationships. After thorough consideration, I finalized the entities and relationships, forming the subsequent steps' foundation.

The next phase involved developing the schemas, which proved to be demanding. Ensuring strict alignment with the predefined entities and relationships was crucial. This step required multiple revisions before finalizing the schemas. Once the schemas were established, I populated the tables with datasets.

Populating the tables was a challenging task. It involved repeatedly checking constraints to ensure data integrity. During this phase, I encountered an issue with a null column that hindered the output display. To address this, I consulted online resources to find solutions for filling null columns using code. Ultimately, I regenerated another set of data on FillDB to avoid redundancy. This meticulous approach ensured the datasets were accurate and aligned with the project requirements.

Applying the schemas to the use cases posed some challenges. However, the experience gained from resolving previous issues with the data and constraints significantly reduced the difficulties at this stage. By leveraging the well-defined schemas, I was able to address the use cases effectively.

This project was a worthwhile endeavor, requiring substantial time and effort. Throughout the process, I navigated various challenges, from defining entities and relationships to ensuring data integrity and applying schemas to use cases. This experience provided valuable insights and clarified initially ambiguous terms, enhancing my understanding and skills in database management.