CS306 – Project Phase II

Alper Çamlı 30858
Ege Yardımcı 31024
Emre Berk Hamarat 31188
Korhan Erdoğdu 30838

1. Trigger Implementations

Trigger: `AfterInsertOnBorrowsSetLoanDates`

```
CREATE TRIGGER AfterInsertOnBorrowsSetLoanDates AFTER INSERT ON borrows
 FOR EACH ROW BEGIN
    -- Update the Loan table with current date as LoanDate and 3 months
from now as ReturnDate
    UPDATE Loan
    SET LoanDate = CURDATE(),
        ReturnDate = DATE_ADD(CURDATE(), INTERVAL 3 MONTH)
    WHERE LoanID = NEW.LoanID;
END
```

1. AfterInsertOnBorrowsSetLoanDates

- **Purpose**:
  This trigger ensures that whenever a record is inserted into the `borrows` table, the corresponding entry in the `Loan` table gets updated with proper `LoanDate` and `ReturnDate` values. It automates the process of assigning these dates to avoid manual intervention or errors.
- **Process**:
  - When a new record is inserted into the `borrows` table, the trigger is activated.
  - The `NEW.LoanID` refers to the loan associated with the book that is being borrowed.
  - The `LoanDate` is set to the current date using `CURDATE()`.
  - The `ReturnDate` is set to three months from the current date by adding a time interval (`INTERVAL 3 MONTH`).
  - The corresponding row in the `Loan` table, identified by the `LoanID`, is updated with these calculated dates.
- **Practical Use Case**:
  Suppose a member borrows a book, and the `LoanID` for the borrowing action is `101`.
  - Before the trigger runs: The `Loan` table may not have any values for `LoanDate` or `ReturnDate`.
  - After the trigger runs: The `Loan` table will now show the `LoanDate` as the current date (e.g., `2024-12-06`) and the `ReturnDate` as three months later (e.g., `2025-03-06`).

Trigger: `BeforeInsertOnBorrowsCheckIfReserved`

```
CREATE TRIGGER BeforeInsertOnBorrowsCheckIfReserved BEFORE INSERT ON
borrows
 FOR EACH ROW BEGIN
    DECLARE v_Reserved INT;

    -- Check if the BookID exists in the IsReservedFor table
    SELECT COUNT(*) INTO v_Reserved
    FROM IsReservedFor
    WHERE BookID = NEW.BookID;

    IF v_Reserved > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: This book is reserved and cannot be
borrowed!';
    END IF;
END
```

2. BeforeInsertOnBorrowsCheckIfReserved

- **Purpose**:
  This trigger prevents a book from being borrowed if it is currently reserved for someone else. It enforces the rule that reserved books cannot be borrowed by other members.
- **Process**:
  - Before a record is inserted into the `borrows` table, the trigger checks if the `BookID` exists in the `IsReservedFor` table.
  - The `IsReservedFor` table holds records of books that are currently reserved.
  - If the count of records in the `IsReservedFor` table for the `NEW.BookID` is greater than zero, it means the book is reserved.
  - In such cases, an error is raised using `SIGNAL SQLSTATE '45000'` with the message: *"Error: This book is reserved and cannot be borrowed!"*.
  - If the book is not reserved, the trigger allows the insertion to proceed.
- **Practical Use Case**:
  Imagine a book with `BookID = 20` is reserved by another member.
  - If a new borrow record is attempted for `BookID = 20`, the trigger will block the insertion and show an error message.

Trigger: `BeforeInsertOnBorrowsCheckMemberExpirationDate`

```
CREATE TRIGGER BeforeInsertOnBorrowsCheckMemberExpirationDate BEFORE INSERT
ON borrows
 FOR EACH ROW BEGIN
    DECLARE v_ExpirationDate DATE;

    -- Get the expiration date of the member from the Member table
    SELECT ExpirationDate INTO v_ExpirationDate
    FROM Member
    WHERE MemberID = NEW.MemberID;

    -- Check if the member's membership has expired
    IF v_ExpirationDate < CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Member''s membership has expired and
cannot borrow books!';
    END IF;
END
```

3. BeforeInsertOnBorrowsCheckMemberExpirationDate

- **Purpose**:
  This trigger ensures that members with expired memberships are not allowed to borrow books. It maintains compliance with library policies.
- **Process**:
  - Before a record is inserted into the `borrows` table, the trigger retrieves the `ExpirationDate` of the member from the `Member` table using `NEW.MemberID`.
  - If the `ExpirationDate` is earlier than the current date (`CURDATE()`), it means the member's membership has expired.
  - In such cases, an error is raised using `SIGNAL SQLSTATE '45000'` with the message: *"Error: Member's membership has expired and cannot borrow books!"*.
  - If the membership is still valid, the trigger allows the insertion to proceed.
- **Practical Use Case**:
  Suppose a member with `MemberID = 50` has an expiration date of `2024-12-01`.
  - If they attempt to borrow a book on `2024-12-06`, the trigger will block the action and raise an error because their membership has expired.

Trigger: `BeforeInsertOnBorrowsDoesUserHaveFines`

```
CREATE TRIGGER BeforeInsertOnBorrowsDoesUserHaveFines BEFORE INSERT ON
borrows
 FOR EACH ROW BEGIN
    DECLARE v_HasFine INT;

    -- Check if the MemberID has an outstanding fine in the Has table
    SELECT COUNT(*) INTO v_HasFine
    FROM Has
    WHERE MemberID = NEW.MemberID;

    IF v_HasFine > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Member has an outstanding fine and
cannot borrow books!';
    END IF;
END
```

4. BeforeInsertOnBorrowsDoesUserHaveFines

- **Purpose**:
  This trigger prevents members with outstanding fines from borrowing books. It enforces financial accountability within the library system.
- **Process**:
  - Before a record is inserted into the `borrows` table, the trigger checks if the `MemberID` of the borrower has any outstanding fines recorded in the `Has` table.
  - The `Has` table tracks fines associated with members.
  - If a count of records in the `Has` table for `NEW.MemberID` is greater than zero, it means the member has outstanding fines.
  - In such cases, an error is raised using `SIGNAL SQLSTATE '45000'` with the message: *"Error: Member has an outstanding fine and cannot borrow books!"*.
  - If no fines exist for the member, the trigger allows the insertion to proceed.
- **Practical Use Case**:
  Suppose a member with `MemberID = 25` has a fine of $5 recorded in the `Has` table.
  - If this member attempts to borrow a book, the trigger will block the insertion and raise an error, asking them to clear the fine first.

Benefits of Using These Triggers

1. **Automation**: Reduces the need for manual checking and updating, ensuring accurate and consistent operations.
2. **Policy Enforcement**: Ensures that library rules (e.g., reserved books, membership expiration, outstanding fines) are strictly followed.
3. **Data Integrity**: Prevents invalid or unauthorized borrowing actions from being recorded in the system.

Screenshots

- Here we see the effect of the AfterInsertOnBorrowsSetLoanDates trigger which is the only trigger that alters the state of the database.

Before the trigger:

←T→				LoanID	LoanDate	ReturnDate
☐	Edit	Copy	Delete	1	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	2	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	3	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	4	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	5	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	7	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	8	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	9	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	10	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	31	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	51	2024-12-05	2025-03-05
☐	Edit	Copy	Delete	54	2024-12-05	2025-03-05

After the trigger:

←T→			LoanID	LoanDate	ReturnDate
Edit	Copy	Delete	1	2024-12-05	2025-03-05
Edit	Copy	Delete	2	2024-12-05	2025-03-05
Edit	Copy	Delete	3	2024-12-05	2025-03-05
Edit	Copy	Delete	4	2024-12-05	2025-03-05
Edit	Copy	Delete	5	2024-12-05	2025-03-05
Edit	Copy	Delete	7	2024-12-05	2025-03-05
Edit	Copy	Delete	8	2024-12-05	2025-03-05
Edit	Copy	Delete	9	2024-12-05	2025-03-05
Edit	Copy	Delete	10	2024-12-05	2025-03-05
Edit	Copy	Delete	31	2024-12-05	2025-03-05
Edit	Copy	Delete	51	2024-12-05	2025-03-05
Edit	Copy	Delete	54	2024-12-05	2025-03-05
Edit	Copy	Delete	62	2024-12-06	2025-03-06

2. Stored Procedure Implementation

SQL Script for the Stored Procedure

Procedure: `BorrowBook`

```
DELIMITER $$
CREATE DEFINER=root@localhost PROCEDURE BorrowBook(IN p_MemberID INT, IN
p_BookID INT, OUT p_Result VARCHAR(255))
BEGIN
    DECLARE v_LoanID INT;
    DECLARE v_BookExists INT;
    DECLARE v_BookAlreadyBorrowed INT;
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Rollback any partial changes
        ROLLBACK;
        -- Re-throw the error
        RESIGNAL;
    END;

    -- Start a transaction
    START TRANSACTION;

    -- Check if the book exists in the Book table
    SELECT COUNT(*) INTO v_BookExists
    FROM Book
    WHERE BookID = p_BookID;

    IF v_BookExists = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Book does not exist!';
    END IF;

    -- Check if the book is already borrowed
    SELECT COUNT(*) INTO v_BookAlreadyBorrowed
    FROM Borrows
    WHERE Borrows.BookID = p_BookID;

    IF v_BookAlreadyBorrowed > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: This book is already borrowed!';
```

```
    END IF;




 -- Create a new loan record
    INSERT INTO Loan (LoanDate, ReturnDate)
    VALUES (CURDATE(), NULL);

    -- Get the LoanID of the newly created loan
    SET v_LoanID = LAST_INSERT_ID();

    -- Insert into Borrows table to record the borrowing
    INSERT INTO Borrows (MemberID, BookID, LoanID)
    VALUES (p_MemberID, p_BookID, v_LoanID);

    -- If all operations succeed, commit the transaction
    COMMIT;

    -- Set the result message
    SET p_Result = CONCAT('Book borrowed successfully with LoanID: ',
v_LoanID);
END$$
DELIMITER ;
```

Functionality Explanation

- The `BorrowBook` procedure handles the borrowing of a book in a library system by validating that the book exists and is not already borrowed. It uses transactions to ensure data consistency and integrity, rolling back changes if an error occurs. The procedure first checks if the specified book exists in the `Book` table and ensures it is not currently borrowed by querying the `Borrows` table. If these checks pass, it creates a new loan record in the `Loan` table, retrieves its ID, and records the borrowing in the `Borrows` table. Finally, it commits the transaction and returns a success message with the loan ID via the output parameter. Errors during execution trigger a rollback and an appropriate error message.

Procedure Execution Test

- Here you can find execution of the stored procedure and output generated by the stored procedure. New borrow is added with LoanID: 62, Member 1 borrowed book 16.

Library Book Borrowing System

Member ID: 1 Book ID: 16 Borrow Book

Book borrowed successfully with LoanID: 62

Loan Table

LoanID	LoanDate	ReturnDate
1	2024-12-05	2025-03-05
2	2024-12-05	2025-03-05
3	2024-12-05	2025-03-05
4	2024-12-05	2025-03-05
5	2024-12-05	2025-03-05
7	2024-12-05	2025-03-05
8	2024-12-05	2025-03-05
9	2024-12-05	2025-03-05
10	2024-12-05	2025-03-05
31	2024-12-05	2025-03-05
51	2024-12-05	2025-03-05
54	2024-12-05	2025-03-05
62	2024-12-06	2025-03-06

Borrows Table

MemberID	BookID	LoanID
1	11	0
1	12	54
1	14	31
1	15	51
1	16	62
2	2	2
2	7	7
3	3	3
3	8	8
4	4	4
4	9	9
5	5	5
5	10	10

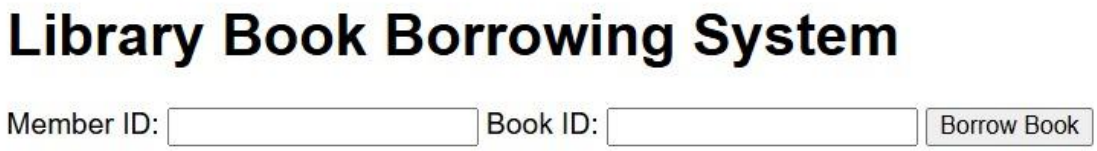
3. Web Access Module

3.1 Description of the Web Interface

This web access module allows the user to input a member id, and a book id to reserve a book from the library.

3.2 Trigger Tests

UI for user input which gets member id and book id.



Trigger: `BeforeInsertOnBorrowsCheckIfReserved`

Error 1

Since the book whose bookID is 13 is reserved, an error occurs and the book can't be borrowed.

Library Book Borrowing System

Member ID: 1 Book ID: 13 Borrow Book

Error: This book is reserved and cannot be borrowed!

Loan Table

LoanID	LoanDate	ReturnDate
1	2024-12-05	2025-03-05
2	2024-12-05	2025-03-05
3	2024-12-05	2025-03-05
4	2024-12-05	2025-03-05
5	2024-12-05	2025-03-05
7	2024-12-05	2025-03-05
8	2024-12-05	2025-03-05
9	2024-12-05	2025-03-05
10	2024-12-05	2025-03-05
31	2024-12-05	2025-03-05
51	2024-12-05	2025-03-05
54	2024-12-05	2025-03-05

Borrows Table

MemberID	BookID	LoanID
1	11	0
1	12	54
1	14	31
1	15	51
2	2	2
2	7	7
3	3	3
3	8	8
4	4	4
4	9	9
5	5	5
5	10	10

Trigger: `BeforeInsertOnBorrowsCheckMemberExpirationDate`

Error 2

Since the membership of the member whose memberID is 8 has expired, an error occurs and she can't borrow books anymore.

Library Book Borrowing System

Member ID: 8 Book ID: 16 Borrow Book

Error: Member's membership has expired and cannot borrow books!

Loan Table

LoanID	LoanDate	ReturnDate
1	2024-12-05	2025-03-05
2	2024-12-05	2025-03-05
3	2024-12-05	2025-03-05
4	2024-12-05	2025-03-05
5	2024-12-05	2025-03-05
7	2024-12-05	2025-03-05
8	2024-12-05	2025-03-05
9	2024-12-05	2025-03-05
10	2024-12-05	2025-03-05
31	2024-12-05	2025-03-05
51	2024-12-05	2025-03-05
54	2024-12-05	2025-03-05

Borrows Table

MemberID	BookID	LoanID
1	11	0
1	12	54
1	14	31
1	15	51
2	2	2
2	7	7
3	3	3
3	8	8
4	4	4
4	9	9
5	5	5
5	10	10

←T→	MemberID	MemberName	MembershipDate	ExpirationDate
Edit Copy Delete	1	Alice Smith	2023-01-10	2025-01-10
Edit Copy Delete	2	Bob Johnson	2023-02-15	2024-02-15
Edit Copy Delete	3	Charlie Brown	2023-03-20	2024-03-20
Edit Copy Delete	4	Daisy Miller	2023-04-25	2024-04-25
Edit Copy Delete	5	Ethan Hunt	2023-05-30	2024-05-30
Edit Copy Delete	6	Fiona Green	2023-06-05	2024-06-05
Edit Copy Delete	7	George White	2023-07-15	2024-07-15
Edit Copy Delete	8	Hannah Blue	2021-08-10	2022-08-10
Edit Copy Delete	9	Ian Black	2023-09-12	2024-09-12
Edit Copy Delete	10	Julia Red	2023-10-01	2024-10-01

Image: “Member” table

Trigger: `BeforeInsertOnBorrowsDoesUserHaveFines`

Error 3

Since the member whose memberID is 2 has a fine, an error occurs and she can't borrow another book.

Library Book Borrowing System

Member ID: 2 Book ID: 16 Borrow Book

Error: Member has an outstanding fine and cannot borrow books!

Loan Table

LoanID	LoanDate	ReturnDate
1	2024-12-05	2025-03-05
2	2024-12-05	2025-03-05
3	2024-12-05	2025-03-05
4	2024-12-05	2025-03-05
5	2024-12-05	2025-03-05
7	2024-12-05	2025-03-05
8	2024-12-05	2025-03-05
9	2024-12-05	2025-03-05
10	2024-12-05	2025-03-05
31	2024-12-05	2025-03-05
51	2024-12-05	2025-03-05
54	2024-12-05	2025-03-05

Borrows Table

MemberID	BookID	LoanID
1	11	0
1	12	54
1	14	31
1	15	51
2	2	2
2	7	7
3	3	3
3	8	8
4	4	4
4	9	9
5	5	5
5	10	10

←T→				FineID	MemberID
☐	Edit	Copy	Delete	2	2
☐	Edit	Copy	Delete	3	3
☐	Edit	Copy	Delete	4	4
☐	Edit	Copy	Delete	7	2
☐	Edit	Copy	Delete	8	3
☐	Edit	Copy	Delete	9	4
☐	Edit	Copy	Delete	10	5

Image: “Has” relationship table consists of the members who has fine

Trigger: `AfterInsertOnBorrowsSetLoanDates`

Success

- Here you can find execution of the stored procedure and output generated by the stored procedure. New borrow is added with LoanID: 62, Member 1 borrowed book 16.

Library Book Borrowing System

Member ID: 1 Book ID: 16 Borrow Book

Book borrowed successfully with LoanID: 62

Loan Table

LoanID	LoanDate	ReturnDate
1	2024-12-05	2025-03-05
2	2024-12-05	2025-03-05
3	2024-12-05	2025-03-05
4	2024-12-05	2025-03-05
5	2024-12-05	2025-03-05
7	2024-12-05	2025-03-05
8	2024-12-05	2025-03-05
9	2024-12-05	2025-03-05
10	2024-12-05	2025-03-05
31	2024-12-05	2025-03-05
51	2024-12-05	2025-03-05
54	2024-12-05	2025-03-05
62	2024-12-06	2025-03-06

Borrows Table

MemberID	BookID	LoanID
1	11	0
1	12	54
1	14	31
1	15	51
1	16	62
2	2	2
2	7	7
3	3	3
3	8	8
4	4	4
4	9	9
5	5	5
5	10	10