Library Database: Triggers & Stored Procedures Documentation

Database: `library_db` **Purpose:** Manage library operations including user management, book borrowing, returns, reservations, and invoicing using triggers and stored procedures.

---

1. TRIGGERS

1.1 `trg_after_borrow_insert`

- **Type:** `AFTER INSERT`
- **Table:** `Borrowing`
- **Purpose:** Automatically decrement the number of available copies of a book when it is borrowed.
- **Behavior:** When a new borrowing record is added, `Available_Copies` of the borrowed book is decreased by 1.
- **Example:**

```
INSERT INTO Borrowing (BookID, CusID, BorrowDate, DueDate, Status)
VALUES (1, 101, CURDATE(), DATE_ADD(CURDATE(), INTERVAL 14 DAY),
'Borrowed');
```

→ `Books.Available_Copies` for `BookID = 1` decreases by 1.

1.2 `trg_after_borrow_update`

- **Type:** `AFTER UPDATE`
- **Table:** `Borrowing`
- **Purpose:** Automatically increase the number of available copies when a book is returned.
- **Behavior:** Checks if `ReturnDate` is set (was previously `NULL`). If yes, increases `Available_Copies` by 1 for that book.
- **Example:**

```
UPDATE Borrowing SET ReturnDate = CURDATE() WHERE BorrowID = 5;
```

→ `Books.Available_Copies` for the borrowed book increases by 1.

1.3 `trg_before_borrow_insert`

- **Type:** `BEFORE INSERT`
- **Table:** `Borrowing`
- **Purpose:** Prevent borrowing if no copies are available.
- **Behavior:** Checks `Books.Available_Copies` before inserting a borrow record. Throws an error if `Available_Copies <= 0`.
- **Example:**

```
INSERT INTO Borrowing (BookID, CusID, BorrowDate, DueDate, Status)
VALUES (2, 101, CURDATE(), DATE_ADD(CURDATE(), INTERVAL 14 DAY),
'Borrowed');
```

→ If `Available_Copies` for `BookID=2` is 0, the insert is rejected.

1.4 `trg_reservation_expire`

- **Type:** `BEFORE UPDATE`
- **Table:** `Reservation`
- **Purpose:** Automatically mark a reservation as expired if its expiry date has passed.
- **Behavior:** If `ReservationExpiryDate < CURDATE()`, sets `Status = 'Expired'`.

---

2. STORED PROCEDURES

2.1 `RegisterUser`

- **Purpose:** Add a new user to the system.
- **Parameters:** | Name | Type | Description | |------|------|-------------| | `p_FirstName` | VARCHAR | User’s first name | | `p_LastName` | VARCHAR | User’s last name | | `p_BirthDate` | DATE | User’s birth date | | `p_UserName` | VARCHAR | Unique username | | `p_Password` | VARCHAR | User password | | `p_Role` | ENUM('admin','customer') | Role of the user |
- **Usage:**

```
CALL RegisterUser('John', 'Doe', '1990-01-01', 'johndoe', 'password123',
'customer');
```

2.2 `BorrowBook`

- **Purpose:** Record a new book borrowing.
- **Parameters:** | Name | Type | Description | |------|------|-------------| | `p_BookID` | INT | ID of the book to borrow | | `p_UserID` | INT | ID of the borrowing user |
- **Behavior:** Inserts a new record into `Borrowing` with a 14-day due date. Triggers handle updating available copies.
- **Usage:**

```
CALL BorrowBook(1, 101);
```

2.3 `ReturnBook`

- **Purpose:** Process the return of a borrowed book and create an invoice with fines if applicable.
- **Parameters:** | Name | Type | Description | |------|------|-------------| | `p_BorrowID` | INT | ID of the borrow record being returned |

- **Behavior:** Sets `ReturnDate` to today, updates `Status = 'Returned'`, calculates fine: $5 per late day, inserts an invoice record with fine.
- **Usage:**

```
CALL ReturnBook(5);
```

2.4 `ReserveBook`

- **Purpose:** Create a reservation for a book.
- **Parameters:** | Name | Type | Description | |------|------|-------------| | `p_BookID` | INT | ID of the book to reserve | | `p_UserID` | INT | ID of the user reserving the book |
- **Behavior:** Sets `ReservationDate` to today, `ReservationExpiryDate` to 3 days later, `Status = 'Active'`.
- **Usage:**

```
CALL ReserveBook(2, 101);
```

2.5 `PayInvoice`

- **Purpose:** Mark an invoice as paid.
- **Parameters:** | Name | Type | Description | |------|------|-------------| | `p_InvoiceID` | INT | ID of the invoice to pay |
- **Behavior:** Updates `Status = 'Paid'` and sets `PaymentDate = CURDATE()`.
- **Usage:**

```
CALL PayInvoice(10);
```

---

3. GENERAL NOTES

1. **Triggers:** Stored internally in `library_db` and automatically execute on insert, update, or delete events. Enforce data integrity like book availability and reservation expiry.
2. **Stored Procedures:** Allow centralized business logic for the library system. Can be called from applications or manually in SQL.
3. **Business Logic Covered:**
4. Borrowing & returning books
5. Reservation expiration
6. Available copy management
7. Fine calculation & invoicing
8. User registration