**Name:** Mimanshu Gahlaut

**UID:** 24BAI70038

**Course:** BE-CSE (AI&ML)

**Subject:** Database Management System

---

# Experiment: Library Management System Implementation

## 1. Aim of the Session

The purpose of this lab session was to design and implement a relational database model for a Library Management System. This included creating structured tables, defining relationships among different entities, and applying role-based security mechanisms within the database.

## 2. Software Requirements:

- **Database:** PostgreSQL Database (PgAdmin)

## 3. Objective of the Session

At the end of the practical, the following goals were accomplished:

- Constructed relational tables with **Primary Keys**, **Foreign Keys**, and **Check Constraints** to maintain data integrity.
- Practiced DML operations such as **INSERT**, **UPDATE**, **DELETE**, and **SELECT** for handling data.
- Applied **DCL commands** to configure role-based permissions for secure data access.
- Preserved **referential integrity** between multiple tables such as books, library_visitors, and book_issue.

## 4. Practical / Experiment Steps

The work was carried out through the following activities:

1. **Database Schema Design:** Defined tables for handling books and visitors, ensuring fields had proper constraints such as `NOT NULL`, `UNIQUE`, and `CHECK` for validating inputs (e.g., visitor minimum age requirement).

2. **Relationship Creation:** Implemented the `book_issue` table which linked book and visitor data using foreign keys to create a transaction-like model.
3. **Data Seeding:** Inserted dummy values to verify structural correctness and validate constraint enforcement.
4. **Functional Testing:** Executed update and delete operations to observe system responses and ensure operations followed referential rules.
5. **Security Configuration:** Established a database role with login privileges and controlled access using `GRANT` and `REVOKE`.

# 5. Procedure of the Practical

Execution was performed in the following order:

1. **Environment Setup:** Logged into DBMS interface and accessed the server instance.
2. **Database Setup:** Created a dedicated database for the library system.
3. **Schema Execution:** Executed CREATE TABLE commands ensuring parent tables were defined first.
4. **Data Entry Phase:** Ran insertion queries to populate tables with book data and visitor records.
5. **Verification Queries:** Used select statements to confirm correct data storage and consistency across tables.
6. **Update/Delete Checks:** Applied modification commands to test mutability and cascading behaviors.
7. **Role Creation:** Formed a librarian role and assigned relevant operations through DCL.
8. **Permission Testing:** Evaluated access controls and validated security using REVOKE and privilege-checking queries.
9. **Documentation:** Saved final SQL script and captured outputs for reporting.

# 6. I/O Analysis (Input / Output Analysis)

## Input Queries

SQL

```
-- Table Definitions
CREATE TABLE books(
        id INT PRIMARY KEY,
        name VARCHAR(50) NOT NULL,
        author_name VARCHAR(50) NOT NULL,
        count INT CHECK(count>0)
)

CREATE TABLE library_visitors(
        user_id INT PRIMARY KEY,
        user_name VARCHAR(20) NOT NULL,
        age INT CHECK(age>=18) NOT NULL,
        email VARCHAR(40) UNIQUE NOT NULL
)
```

```
CREATE TABLE book_issue(
        book_issue_id INT PRIMARY KEY,
        book_id INT NOT NULL,
        user_id INT NOT NULL,
        FOREIGN KEY (book_id) REFERENCES books(id),
        FOREIGN KEY (user_id) REFERENCES library_visitors(user_id),
        book_issue_date DATE NOT NULL
)

-- Data Operations
INSERT INTO books VALUES(1, 'Harry Potter', 'R. Snape', 1)
INSERT INTO books VALUES(2, 'Avengers', 'Stan Lee', 3)
INSERT INTO library_visitors VALUES(101, 'Robert', 20, 'abc@gmail.com')
UPDATE library_visitors SET email='Robert@gmail.com' WHERE user_id = 101
INSERT INTO book_issue VALUES(1234,1,101,'2026-01-07')
DELETE FROM books WHERE id = 2

-- Role Management
CREATE ROLE librarian WITH LOGIN PASSWORD 'PASSWORD'
GRANT SELECT, INSERT, DELETE, UPDATE ON books TO librarian
GRANT SELECT, INSERT, DELETE, UPDATE ON library_visitors TO librarian
GRANT SELECT, INSERT, DELETE, UPDATE ON book_issue TO librarian
REVOKE SELECT, INSERT, DELETE, UPDATE ON books FROM librarian
```

## Output Details

### 1. Schema Creation

- All three tables (books, library_visitors, and book_issue) were successfully created.
- The PRIMARY KEY constraints ensured unique identification of books and visitors.
- The CHECK(age>=18) constraint prevented entries of visitors below 18 years of age.
- The CHECK(count>0) constraint disallowed non-positive values for book count.
- FOREIGN KEY constraints ensured that book issue entries could only reference existing books and visitors.

✓ Result: Schema creation completed without errors.

2. **DML Outputs**

The following SQL commands executed successfully:

```
INSERT INTO library_visitors VALUES(101, 'Robert', 20, 'abc@gmail.com')
UPDATE library_visitors SET email='Robert@gmail.com' WHERE user_id = 101
SELECT * FROM library_visitors
```

| | user_id [PK] integer | user_name character varying (20) | age integer | email character varying (40) |
|---|---|---|---|---|
| 1 | 101 | Robert | 20 | Robert@gmail.com |

Next, visitor insertion and update:

Finally, book issue entry:

```
INSERT INTO book_issue VALUES(1234,1,101,'2026-01-07')
SELECT * FROM book_issue
```

| | book_issue_id [PK] integer | book_id integer | user_id integer | book_issue_date date |
|---|---|---|---|---|
| 1 | 1234 | 1 | 101 | 2026-01-07 |

**3. DELETE Operation Result**

```
INSERT INTO books VALUES(1, 'Harry Potter', 'R. Snape', 1)
INSERT INTO books VALUES(2, 'Avengers', 'Stan Lee', 3)

SELECT * FROM books
```

| | id [PK] integer | name character varying (50) | author_name character varying (50) | count integer |
|---|---|---|---|---|
| 1 | 1 | Harry Potter | R. Snape | 1 |
| 2 | 2 | Avengers | Stan Lee | 3 |

```
DELETE FROM books WHERE id = 2
SELECT * FROM books
```

| | id [PK] integer | name character varying (50) | author_name character varying (50) | count integer |
|---|---|---|---|---|
| 1 | 1 | Harry Potter | R. Snape | 1 |

**4. DCL (Security / Role-Based Access Control) Output**

```
CREATE ROLE librarian WITH LOGIN PASSWORD 'PASSWORD'
GRANT SELECT, INSERT, DELETE, UPDATE ON books TO librarian
GRANT SELECT, INSERT, DELETE, UPDATE ON library_visitors TO librarian
GRANT SELECT, INSERT, DELETE, UPDATE ON book_issue TO librarian
```

```
GRANT

Query returned successfully in 119 msec.
```

Next, revoking privileges:

```
REVOKE SELECT, INSERT, DELETE, UPDATE ON books FROM librarian
```

```
REVOKE

Query returned successfully in 96 msec.
```

✓ **Effect:**
After revocation, librarian could no longer perform operations on the books table, verifying that permission control worked correctly.

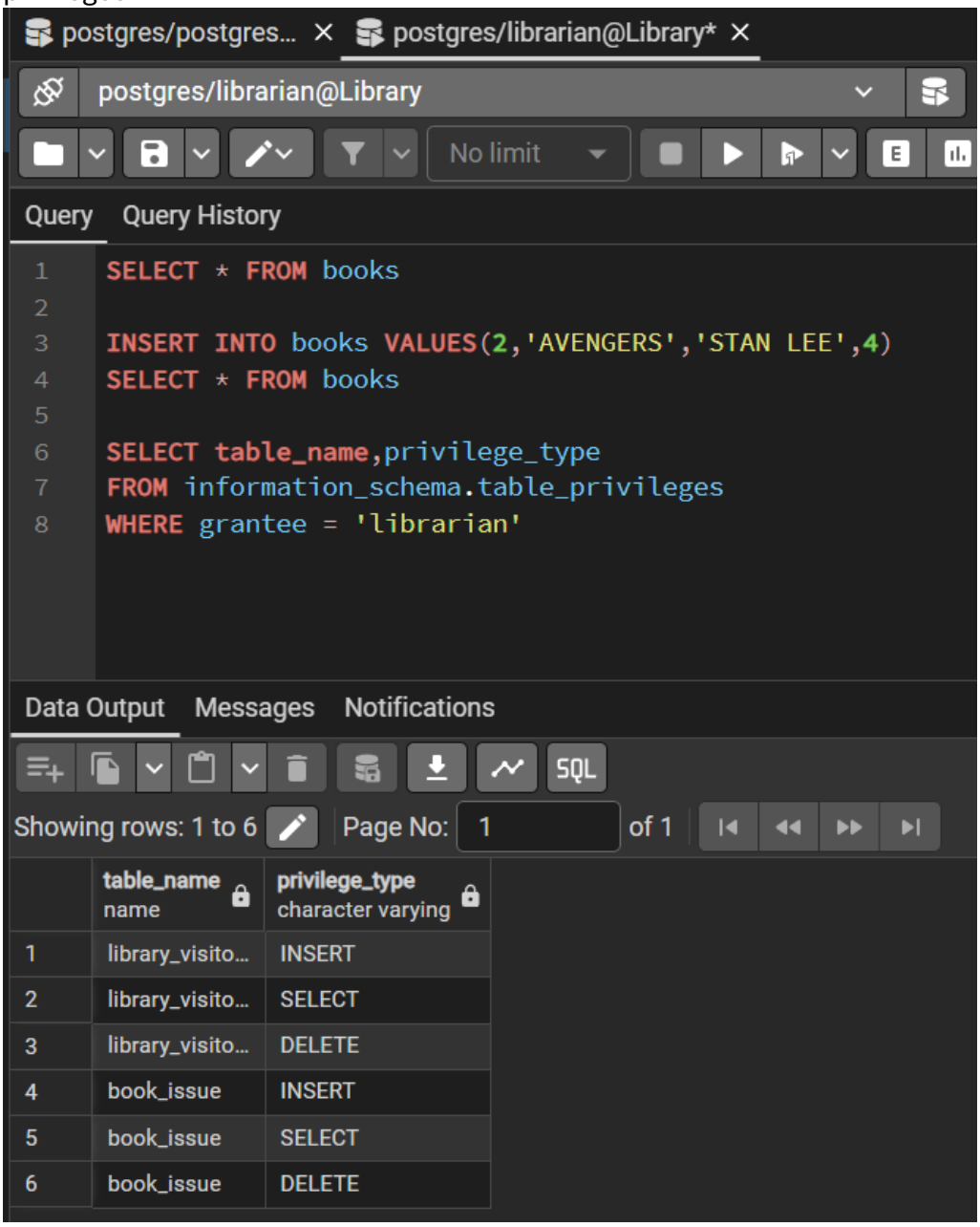
- **Validation:** Testing confirmed that after the `REVOKE` command, the `librarian` could no longer perform operations on the `books` table, ensuring the security policy is functional.

Data Output Messages Notifications

```
ERROR:  permission denied for table books

SQL state: 42501
```

- We also confirmed the permissions of the role “librarian” by checking the table privileges.



# 7. Learning Outcome

From this practical, the following knowledge and skills were gained:

- **Schema Design Insight:** Learned how relational constraints like CHECK, UNIQUE, and FOREIGN KEY contribute to logical data consistency.
- **Database Security Skills:** Understood the advantage of assigning roles instead of individual user permissions for scalable security.
- **Real-world Contextualization:** Saw how SQL is applied in real applications (such as library systems) where multiple entities interact systematically.