

Name: Vishesh Kansugia

UID: 24BAI70184

Course: BE-CSE (AI&ML)

Subject: Database Management Systems

Experiment: Library Management System Implementation

1. Aim of the Session

The aim of this practical is to design and implement a relational database schema for a Library Management System. This involves defining tables with specific constraints, establishing relationships between entities, and managing database security through role-based access control.

2. Objective of the Session

Upon completing this session, the following objectives were achieved:

- Developed table structures using **Primary Keys**, **Foreign Keys**, and **Check Constraints** for data validation.
- Gained proficiency in **DML (Data Manipulation Language)** operations, specifically INSERT, SELECT, UPDATE, and DELETE.
- Implemented **DCL (Data Control Language)** to manage user roles and granular permissions.
- Maintained referential integrity across multiple related tables (`books`, `library_visitors`, and `book_issue`).

3. Practical / Experiment Steps

The implementation was carried out through the following tasks:

1. **Schema Definition:** Created the base tables for `books` and `library_visitors` with specific constraints such as NOT NULL, UNIQUE, and CHECK (e.g., ensuring visitor age is 18+).
2. **Relational Setup:** Created the `book_issue` table to act as a transaction bridge, linking books and visitors via Foreign Keys.
3. **Data Population:** Populated the tables with initial records to test the schema's validity.

4. **Operational Testing:** Performed updates on user information and attempted deletion of records to observe constraint behavior.
5. **Security Administration:** Created a `librarian` role with login credentials and configured its access levels using `GRANT` and `REVOKE` commands.

4. Procedure of the Practical

The following steps were followed during the execution:

1. **System Initialization:** Logged into the database environment and established a connection to the server.
2. **Database Creation:** Initialized a new database to house the library management system.
3. **Executing Table Scripts:** Ran the `CREATE TABLE` commands in a specific sequence (creating parent tables before dependent transaction tables).
4. **Data Entry:** Executed `INSERT` statements to add sample books and visitor profiles.
5. **Query Verification:** Used `SELECT` queries to verify that the data was correctly stored and consistent across tables.
6. **Data Modification:** Tested the `UPDATE` and `DELETE` commands to ensure the system handles changes as intended.
7. **Role Configuration:** Defined the `librarian` role and assigned specific table privileges.
8. **Security Verification:** Tested and then revoked permissions to confirm the effectiveness of the security policy.
9. **Record Maintenance:** Saved the SQL script and took screenshots of the execution results.

5. I/O Analysis (Input / Output Analysis)

Input Queries

SQL

```
-- Table Creation
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 Manipulation
INSERT INTO books VALUES(1, 'Hairy Poppter', 'R. Snap', 1);
INSERT INTO library_visitors VALUES(101, 'Robert', 20, 'abc@il.com');
UPDATE library_visitors SET email='Robel.com' WHERE user_id = 101;

-- Role Management
CREATE ROLE librarian WITH LOGIN PASSWORD 'WHIPWHIP';
GRANT SELECT, INSERT, DELETE, UPDATE ON books TO librarian;

```

Output Details

- **Schema Success:** All tables were created successfully. The system correctly enforced the CHECK (age>=18) constraint, rejecting invalid entries.

The screenshot shows a SQL IDE interface. The top pane displays a series of SQL queries executed, including foreign key constraints, data insertion, and updates. The bottom pane shows the 'Data Output' tab with a table of results from a SELECT query on the 'books' table.

Query Log:

```

21 FOREIGN KEY (book_id) REFERENCES books(id),
22 FOREIGN KEY (user_id) REFERENCES library_visitors(user_id),
23 book_issue_date DATE NOT NULL
24 )
25
26 INSERT INTO books VALUES(1, 'Hairy Poppter', 'R. Snap', 1);
27 INSERT INTO books VALUES(2, 'Revengers', 'Stan Man', 3);
28
29 SELECT * FROM books
30
31 INSERT INTO library_visitors VALUES(101, 'Robert', 20, 'abc@il.com')
32
33 UPDATE library_visitors SET email='Robel.com' WHERE user_id = 101
34
35 SELECT * FROM library_visitors

```

Data Output Table:

	id [PK] integer	name character varying (50)	author_name character varying (50)	count integer
1	1	Hairy Poppter	R. Snap	1
2	2	Revengers	Stan Man	3

- **DML Results:** The UPDATE query correctly modified the email field for user 101, and SELECT queries displayed the current state of all tables accurately.

Query Query History

```

25
26 INSERT INTO books VALUES(1, 'Hairy Popter', 'R. Snap', 1);
27 INSERT INTO books VALUES(2, 'Revengers', 'Stan Man', 3);
28
29 SELECT * FROM books
30
31 INSERT INTO library_visitors VALUES(101, 'Robert', 20, 'abc@il.com')
32
33 UPDATE library_visitors SET email='Robel.com' WHERE user_id = 101
34
35 SELECT * FROM library_visitors
36
37 INSERT INTO book_issue VALUES(1234,1,101,'2026-01-07')
38
39 SELECT * FROM book_issue

```

Data Output Messages Notifications

Showing rows: 1 to 1 Page 1

	user_id [PK] integer	user_name character varying (20)	age integer	email character varying (40)
1	101	Robert	20	Robel.com

- **DCL Verification:** The librarian role was successfully created and assigned the necessary privileges for library management tasks.

Query Query History Scratch Pad

```

37 INSERT INTO book_issue VALUES(1234,1,101,'2026-01-07')
38
39 SELECT * FROM book_issue
40
41 DELETE FROM books WHERE id = 3
42
43 SELECT * FROM books
44
45 CREATE ROLE librarian WITH LOGIN PASSWORD 'WHIPWHIP'
46
47 GRANT SELECT, INSERT, DELETE, UPDATE ON books TO librarian;
48 GRANT SELECT, INSERT, DELETE, UPDATE ON library_visitors TO librarian;
49 GRANT SELECT, INSERT, DELETE, UPDATE ON book_issue TO librarian;
50
51 REVOKE SELECT, INSERT, DELETE, UPDATE ON books FROM librarian

```

Data Output Messages Notifications

GRANT

Query returned successfully in 38 msec.

- **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.

Query	Query History
1	<code>SELECT * FROM books;</code>
2	<code>INSERT INTO books (id, name, author_name, count)</code>
3	<code>VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 5);</code>

Data Output	Messages	Notifications
<code>ERROR: permission denied for table books</code>		
<code>SQL state: 42501</code>		

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

The screenshot shows a database management tool interface. At the top, the connection is set to 'AIT-DBMS/librarian@librarian'. Below the connection bar is a toolbar with various icons for file operations, filters, and execution. The main area is divided into two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying a SQL script with line numbers 1 through 11. The script includes a SELECT statement, an INSERT INTO statement, and another SELECT statement with a WHERE clause. Below the query editor is a section for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with two columns: 'table_name' and 'privilege_type'. The table contains 8 rows of data, showing privileges for 'library_visito...' and 'book_issue' tables. The interface also includes a toolbar for the data output section and a status bar indicating 'Showing rows: 1'.

```

1  SELECT * FROM books;
2  INSERT INTO books (id, name, author_name, count)
3  VALUES (3, 'The Great Gatsby', 'F. Scott Fitzgerald', 5);
4
5  SELECT
6      table_name,
7      privilege_type
8  FROM
9      information_schema.table_privileges
10 WHERE
11     grantee = 'librarian';

```

	table_name name	privilege_type character varying
1	library_visito...	INSERT
2	library_visito...	SELECT
3	library_visito...	UPDATE
4	library_visito...	DELETE
5	book_issue	INSERT
6	book_issue	SELECT
7	book_issue	UPDATE
8	book_issue	DELETE

6. Learning Outcome

This practical session provided significant insights into:

- **Structural Logic:** Understanding how Foreign Keys and Check Constraints maintain high data quality and prevent logical errors.

- **Security Implementation:** Learning to manage database security through roles rather than individual user permissions.
- **Practical Application:** Applying SQL fundamentals to a real-world scenario (Library Management), demonstrating how relational databases handle complex interactions between entities.