



Student Name: Aditya Verma

Branch: AIT-CSE (AIML)

Semester: 4

Subject Name: Database Management System

UID: 24BAI70937

Section/Group: 24AIT_KRG2

Experiment

Experiment 1.1: Design and implementation of a Library Management System using PostgreSQL with DDL, DML and DCL commands.

Aim

The aim of this experiment is to design and implement a Library Management System database using PostgreSQL. The database is created using proper tables, primary keys, foreign keys and constraints. DML operations are performed and database security is implemented using roles and privileges.

Objective

The objective of this experiment is to gain practical knowledge of DDL, DML and DCL commands in PostgreSQL. It also helps in understanding how to create roles, grant permissions and revoke permissions to secure the database using role based access control.

Practical / Experiment Steps

1. Design the database structure for the Library Management System.
2. Create tables for books, members and issue records using DDL commands.
3. Apply primary keys, foreign keys and constraints to maintain data integrity.
4. Insert sample records into the tables using DML commands.
5. Update and delete records as required.
6. Create a database role named Librarian.
7. Grant required permissions like SELECT, INSERT and DELETE to the Librarian role.
8. Revoke permissions when needed to ensure database security.

Procedure of the Experiment

1. Start the system and log in to the computer.
2. Open pgAdmin and connect to PostgreSQL server.
3. Create a new database for the Library Management System.
4. Create tables such as Books, Members and Issue_Records using CREATE TABLE command.
5. Define primary keys and foreign keys while creating the tables.
6. Insert records into tables using INSERT command.
7. Update existing data using UPDATE command.
8. Delete unwanted records using DELETE command.
9. Create a role named Librarian with password using CREATE ROLE command.
10. Grant SELECT, INSERT and DELETE permissions to the Librarian role.
11. Revoke permissions using REVOKE command when required.
12. Execute all queries and verify the output.

CODE :

1. ADMIN

```
CREATE TABLE BOOKS(  
    ID INT PRIMARY KEY,  
    NAME VARCHAR(20) NOT NULL,  
    AUTHOR VARCHAR(20) NOT NULL  
)
```

```
INSERT INTO BOOKS VALUES(1,'HARRY POTTER','JK ROWLING');
```

```
ALTER TABLE BOOKS
```

```
ADD COUNT INT CHECK(COUNT>=1)
```



SELECT * FROM BOOKS

UPDATE BOOKS

SET COUNT=3

WHERE ID = 1

INSERT INTO BOOKS VALUES(2,'GAME OF THRONS','I DONT KNOW ',12);

```
CREATE TABLE LIBRARY_VISITOR_USER(  
    USER_ID INT PRIMARY KEY,  
    USER_NAME VARCHAR(20) NOT NULL,  
    AGE INT CHECK(AGE>=17) NOT NULL,  
    EMAIL_ID VARCHAR(20) UNIQUE NOT NULL  
)
```

```
INSERT INTO LIBRARY_VISITOR_USER  
VALUES(101,'ARYAN DAHIYA',18,'ARD@GMAIL.COM')
```

```
INSERT INTO LIBRARY_VISITOR_USER  
VALUES(102,'RAMESH',19,'RMD@GMAIL.COM')
```

SELECT * FROM LIBRARY_VISITOR_USER

```
INSERT INTO LIBRARY_VISITOR_USER  
VALUES(1,'RAM',19,'RAM@GMAIL.COM')
```

UPDATE LIBRARY_VISITOR_USER

SET USER_ID = 103

WHERE USER_ID =1



```
ALTER TABLE LIBRARY_VISITOR_USER
```

```
ALTER COLUMN EMAIL_ID TYPE VARCHAR(15);
```

```
CREATE TABLE BOOK_ISSUE(  
    ISSUE_ID INT PRIMARY KEY ,  
    BOOK_ID INT REFERENCES BOOKS(ID) NOT NULL,  
    USER_ID INT REFERENCES LIBRARY_VISITOR_USER(USER_ID),  
    ISSUE_DATE DATE NOT NULL DEFAULT CURRENT_DATE  
)
```

```
INSERT INTO BOOK_ISSUE  
VALUES(121,1,101,'2026-01-09')
```

```
SELECT * FROM BOOK_ISSUE
```

```
CREATE ROLE LIBRARIAN  
WITH LOGIN PASSWORD 'RAKESH101'
```

```
GRANT SELECT, INSERT, DELETE, UPDATE ON BOOKS TO LIBRARIAN
```

```
REVOKE SELECT,INSERT,DELETE,INSERT,UPDATE ON BOOKS FROM LIBRARIAN
```

2. LIBRARIAN

```
SELECT * FROM books;  
select * from book_issue;  
select * from LIBRARY_VISITOR_USER;
```



```
INSERT INTO books VALUES(110,'ABCD','CLANS',2);  
INSERT INTO books VALUES(150,'THE LORD','HRM',7);
```

```
DELETE FROM books  
WHERE ID=150;
```

```
SELECT * FROM book_issue;  
SELECT * FROM LIBRARY_VISITOR_USER;
```

Learning Outcomes:

1. Understood the basics of **relational database design** using tables, keys, and relationships.
2. Learned to apply **primary key and foreign key constraints** to maintain data integrity.
3. Gained hands-on experience with **INSERT, UPDATE, and DELETE** operations safely.
4. Understood how **roles and privileges** control access to database objects.
5. Learned to use **GRANT and REVOKE** for implementing **read-only users**.
6. Practiced **ALTER TABLE and DROP TABLE** for managing database changes.

SCREENSHOTS

Query Query History

```

1  CREATE TABLE BOOKS(
2      ID INT PRIMARY KEY,
3      NAME VARCHAR(20) NOT NULL,
4      AUTHOR VARCHAR(20) NOT NULL
5  )
6
7  INSERT INTO BOOKS VALUES(1, 'HARRY POTTER', 'JK ROWLING');
8
9  ALTER TABLE BOOKS
10 ADD COUNT INT CHECK(COUNT >= 1)
11
12 SELECT * FROM BOOKS
13

```

Data Output Messages Notifications

SQL

	id [PK] integer	name character varying (20)	author character varying (20)	count integer
1	1	HARRY POTTER	JK ROWLING	[null]

```

20
21 CREATE TABLE LIBRARY_VISITOR_USER(
22     USER_ID INT PRIMARY KEY,
23     USER_NAME VARCHAR(20) NOT NULL,
24     AGE INT CHECK(AGE >= 17) NOT NULL,
25     EMAIL_ID VARCHAR(20) UNIQUE NOT NULL
26 )
27
28 INSERT INTO LIBRARY_VISITOR_USER
29 VALUES(101, 'Dibyanshu', 18, 'qwert@GMAIL.COM')
30
31 INSERT INTO LIBRARY_VISITOR_USER
32 VALUES(102, 'RAj', 19, 'Raj@GMAIL.COM')
33
34 SELECT * FROM LIBRARY_VISITOR_USER
35

```

Data Output Messages Notifications

SQL

	user_id [PK] integer	user_name character varying (20)	age integer	email_id character varying (20)
1	101	Dibyanshu	18	qwert@GMAIL.COM
2	102	RAj	19	Raj@GMAIL.COM


```

35
36 INSERT INTO LIBRARY_VISITOR_USER
37 VALUES(1, 'RAM', 19, 'RAM@GMAIL.COM')
38
39 UPDATE LIBRARY_VISITOR_USER
40 SET USER_ID = 103
41 WHERE USER_ID = 1
42
43 ALTER TABLE LIBRARY_VISITOR_USER
44 ALTER COLUMN EMAIL_ID TYPE VARCHAR(15);
45
46 CREATE TABLE BOOK_ISSUE(
47     ISSUE_ID INT PRIMARY KEY ,
48     BOOK_ID INT REFERENCES BOOKS(ID) NOT NULL,
49     USER_ID INT REFERENCES LIBRARY_VISITOR_USER(USER_ID),
50     ISSUE_DATE DATE NOT NULL DEFAULT CURRENT_DATE
51 )
52
53 INSERT INTO BOOK_ISSUE
54 VALUES(100, 1, 101, '2026-01-09')
55
56 SELECT * FROM BOOK_ISSUE
57

```

Data Output Messages Notifications

	Issue_Id [PK] integer	book_Id integer	user_Id integer	issue_date date
1	100	1	101	2026-01-09

```

62
63 GRANT SELECT, INSERT, DELETE, UPDATE ON BOOKS TO LIBRARIAN2
64
65
66

```

Data Output Messages Notifications

GRANT

Query returned successfully in 60 msec.

67

68

REVOKE **SELECT,INSERT,DELETE,INSERT,UPDATE ON** **BOOKS FROM** **LIBRARIAN2**

Data Output Messages Notifications

REVOKE

Query returned successfully in 51 msec.

Query Query History

```

1  SELECT * FROM books;
2  select * from book_issue;
3  select * from LIBRARY_VISITOR_USER;
4
5  INSERT INTO books VALUES(120,'arya','xya',2);
6  INSERT INTO books VALUES(150,'THE LORD','HRM',7);
7
8  DELETE FROM books
9  WHERE ID=150;
10
11 SELECT * FROM book_issue;
12 SELECT * FROM LIBRARY_VISITOR_USER;

```

Data Output Messages Notifications

	user_id [PK] integer	user_name character varying (20)	age integer	email_id character varying (15)
1	101	Dibyanshu	18	qwert@GMAIL.COM
2	102	RAj	19	Raj@GMAIL.COM
3	103	RAM	19	RAM@GMAIL.COM

Query Query History

```

1  SELECT * FROM books;
2  select * from book_issue;
3  select * from LIBRARY_VISITOR_USER;
4
5  INSERT INTO books VALUES(120,'arya','xya',2);
6  INSERT INTO books VALUES(150,'THE LORD','HRM',7);
7
8  DELETE FROM books
9  WHERE ID=150;
10
11 SELECT * FROM book_issue;
12 SELECT * FROM LIBRARY_VISITOR_USER;

```

Data Output Messages Notifications








	id [PK] integer	name character varying (20)	author character varying (20)	count integer
1	1	HARRY POTTER	JK ROWLING	3
2	2	GAME OF THRONES	I DONT KNOW	12
3	120	arya	xya	2
4	150	THE LORD	HRM	7