**Lab Exercise 14- Implementing Row-Level Security (RLS)**

**Objective**

* Understand how to enable and configure RLS
* Create policies to control access to specific rows in a table
* Practice RLS using simple user-role assignments

**Prerequisites**

* PostgreSQL installed on Windows
* Access to SQL Shell or pgAdmin
* A test database and superuser access

**Step 1: Create a Test Database and Table**

Connect to PostgreSQL as a superuser and run:

CREATE DATABASE rls\_test;

\c rls\_test

CREATE TABLE employee\_data (

id SERIAL PRIMARY KEY,

name TEXT,

department TEXT

);

Insert some sample data:

INSERT INTO employee\_data (name, department)

VALUES

('Alice', 'HR'),

('Bob', 'Finance'),

('Charlie', 'IT');

**Step 2: Create Roles for Different Departments**

Create login roles for users from different departments:

CREATE ROLE hr\_user LOGIN PASSWORD 'hrpass';

CREATE ROLE it\_user LOGIN PASSWORD 'itpass';

**Step 3: Enable RLS on the Table**

ALTER TABLE employee\_data ENABLE ROW LEVEL SECURITY;

**Step 4: Create a Column-Based Access Policy**

This policy allows each user to access only their department's rows:

First, add a new column for storing the role name:

ALTER TABLE employee\_data ADD COLUMN username TEXT;

Update the data to assign usernames:

UPDATE employee\_data

SET username = CASE

WHEN name = 'Alice' THEN 'hr\_user'

WHEN name = 'Bob' THEN 'hr\_user'

WHEN name = 'Charlie' THEN 'it\_user'

END;

Create a policy to allow users to view only rows assigned to their role:

CREATE POLICY user\_filter\_policy

ON employee\_data

FOR SELECT

USING (username = current\_user);

**Step 5: Grant Permissions**

Give select access on the table to the users:

GRANT SELECT ON employee\_data TO hr\_user;

GRANT SELECT ON employee\_data TO it\_user;

**Step 6: Test Row-Level Security**

Connect as hr\_user and run:

\c rls\_test hr\_user

SELECT \* FROM employee\_data;

Expected: You should see only the rows where username is hr\_user

Connect as it\_user and run:

\c rls\_test it\_user

SELECT \* FROM employee\_data;

Expected: You should see only the row where username is it\_user

**Step 7: Add Insert Policy (Optional)**

Allow users to insert only their own rows:

CREATE POLICY insert\_policy

ON employee\_data

FOR INSERT

WITH CHECK (username = current\_user);

GRANT INSERT ON employee\_data TO hr\_user;

GRANT INSERT ON employee\_data TO it\_user;

Now connect as hr\_user and try this:

INSERT INTO employee\_data (name, department, username)

VALUES ('Diane', 'HR', 'hr\_user');

It should succeed. If you change the username value to something else, it will fail.

**Summary**

| **Task** | **Command Example** |
| --- | --- |
| Enable RLS | ALTER TABLE table\_name ENABLE ROW LEVEL SECURITY |
| Create policy for SELECT | CREATE POLICY name ON table FOR SELECT USING clause |
| Create policy for INSERT | CREATE POLICY name ON table FOR INSERT WITH CHECK |
| Grant permission | GRANT SELECT ON table TO user |
| Check current user | SELECT current\_user |

To **see all Row-Level Security (RLS) policies** defined on a specific table in PostgreSQL, use the following SQL command:

SELECT \* FROM pg\_policies WHERE tablename = 'your\_table\_name';

Replace your\_table\_name with the actual name of your table.

**Example:**

SELECT \* FROM pg\_policies WHERE tablename = 'employee\_data';

This will show:

* Policy name
* Table name
* Role the policy applies to
* Command type (SELECT, INSERT, etc.)
* Expressions used in USING or WITH CHECK clauses

You can also use the \dp command in **psql** to see privilege and RLS policy info:

\dp your\_table\_name