



Experiment 7

Student Name: Jobanjot Singh Grewal

UID: 23BIA50005

Branch: BE-AIT-CSE

Section/Group: 23AML_KRG-1

Semester: 5th

Date of Performance: 22 Nov 2025

Subject Name: ADBMS

Subject Code: 23CSP-333

MEDIUM - LEVEL

1. Problem Title: Triggers Medium

2. Problem Description: WHENEVER THERE IS A INSERTION ON STUDENT TABLE THEN, THE CURRENTLY INSERTED OR DELETED ROW SHOULD BE PRINTED AS IT AS ON THE OUTPUT CONSOLE WINDOW.

SQL Commands:

```
CREATE TABLE student (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(50),  
    age INT,  
    class VARCHAR(10)  
);
```

```
CREATE OR REPLACE FUNCTION fn_student_audit()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
$$  
BEGIN
```

```
    IF TG_OP = 'INSERT' THEN  
        RAISE NOTICE 'Inserted Row -> ID: %, Name: %, Age: %, Class: %',  
            NEW.id, NEW.name, NEW.age, NEW.class;  
        RETURN NEW;
```

```
    ELSIF TG_OP = 'DELETE' THEN  
        RAISE NOTICE 'Deleted Row -> ID: %, Name: %, Age: %, Class: %',  
            OLD.id, OLD.name, OLD.age, OLD.class;
```

```

        RETURN OLD;
    END IF;

    RETURN NULL;
END;
$$;

CREATE TRIGGER trg_student_audit
AFTER INSERT OR DELETE
ON student
FOR EACH ROW
EXECUTE FUNCTION fn_student_audit();

INSERT INTO student (name, age, class)
VALUES ('Manoj', 21, 'CS111');

DELETE FROM student WHERE id = 1;

```

Output:

Output:

```

DROP TRIGGER
DROP FUNCTION
DROP TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
INSERT 0 1
DELETE 1

psql:commands.sql:6: NOTICE:  relation "student" does not exist, skipping
psql:commands.sql:7: NOTICE:  function fn_student_audit() does not exist, skipping
psql:commands.sql:8: NOTICE:  table "student" does not exist, skipping
psql:commands.sql:48: NOTICE:  Inserted Row -> ID: 1, Name: Ravi, Age: 20, Class: CS101
psql:commands.sql:49: NOTICE:  Inserted Row -> ID: 2, Name: Priya, Age: 21, Class: CS102
psql:commands.sql:51: NOTICE:  Deleted Row -> ID: 1, Name: Ravi, Age: 20, Class: CS101

```

Fig1: View OUTPUT

Learning Outcomes:

- I learned how to create new triggers.
- I learned how to perform different types of triggers.
- I learned how to create triggers with specific types.

HARD - LEVEL

- 1. Problem Title:** Triggers Hard level
- 2. Problem Description:** Whenever a new employee is inserted in tbl_employee, a record should be added to tbl_employee_audit like:
"Employee name <emp_name> has been added at <current_time>"
Whenever an employee is deleted from tbl_employee, a record should be added to tbl_employee_audit like:
"Employee name <emp_name> has been deleted at <current_time>"

The solution must use PostgreSQL triggers.

SQL Commands:

```
CREATE TABLE tbl_employee (  
    emp_id SERIAL PRIMARY KEY,  
    emp_name VARCHAR(100) NOT NULL,  
    emp_salary NUMERIC  
);
```

```
CREATE TABLE tbl_employee_audit (  
    sno SERIAL PRIMARY KEY,  
    message TEXT  
);
```

```
CREATE OR REPLACE FUNCTION audit_employee_changes()  
RETURNS TRIGGER  
LANGUAGE plpgsql  
AS  
$$  
BEGIN  
    IF TG_OP = 'INSERT' THEN  
        INSERT INTO tbl_employee_audit(message)  
        VALUES ('Employee name ' || NEW.emp_name || ' has been added at ' || NOW());  
        RETURN NEW;  
  
    ELSIF TG_OP = 'DELETE' THEN  
        INSERT INTO tbl_employee_audit(message)  
        VALUES ('Employee name ' || OLD.emp_name || ' has been deleted at ' || NOW());  
        RETURN OLD;  
    END IF;  
  
    RETURN NULL;  
END;  
$$
```

```
CREATE TRIGGER trg_employee_audit
AFTER INSERT OR DELETE
ON
tbl_employee
FOR EACH ROW
EXECUTE FUNCTION audit_employee_changes();
```

```
INSERT INTO tbl_employee(emp_name, emp_salary) VALUES ('Aman', 50000);
```

```
DELETE FROM tbl_employee WHERE emp_name = 'Aman';
```

```
SELECT * FROM tbl_employee_audit;
```

Output:

Output:

```
DROP TRIGGER
DROP FUNCTION
DROP TABLE
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
DELETE 1
sno |
-----+-----
  1 | Employee name Aman has been added at 2025-11-06 03:57:17.175172+00
  2 | Employee name Aman has been deleted at 2025-11-06 03:57:17.177479+00
(2 rows)
```

```
psql:commands.sql:6: NOTICE:  relation "tbl_employee" does not exist, skipping
psql:commands.sql:7: NOTICE:  function audit_employee_changes() does not exist, skipping
psql:commands.sql:8: NOTICE:  table "tbl_employee_audit" does not exist, skipping
psql:commands.sql:9: NOTICE:  table "tbl_employee" does not exist, skipping
```

Fig1: View OUTPUT

9. Learning Outcomes:

- I learned how to create triggers.
- I learned how to perform types of triggers.