Experiment No. 11

Experiment Title: Design and implementation of fragmentation in dist

Objective: The objective for implementing triggers is to enforce data integrity, business logic, automation, auditing, security, and other actions in response to specific database events.

Student Roll No. 23101B2001 **Name: Om Sagvekar** **Division: B (INFT)**

Aim: implementing triggers is to enhance database functionality by automating responses to events, ensuring data integrity, and enforcing business logic.

Code:

```
-- create
CREATE TABLE EMPLOYEE (
 empId INTEGER PRIMARY KEY,
 name TEXT NOT NULL,
 dept TEXT NOT NULL
);
CREATE TABLE AUDIT(
  EMP_ID INT NOT NULL,
  ENTRY_DATE TEXT NOT NULL
);

CREATE OR REPLACE FUNCTION auditlogfunc() RETURNS TRIGGER AS $example_table$
  BEGIN
    INSERT INTO AUDIT(EMP_ID, ENTRY_DATE) VALUES (new.empId, current_timestamp);
    RETURN NEW;
  END;
$example_table$ LANGUAGE plpgsql;

CREATE TRIGGER example_trigger AFTER INSERT ON EMPLOYEE
FOR EACH ROW EXECUTE PROCEDURE auditlogfunc();
-- insert
INSERT INTO EMPLOYEE VALUES (0001, 'Clark', 'Sales');
INSERT INTO EMPLOYEE VALUES (0002, 'Dave', 'Accounting');
```

```
INSERT INTO EMPLOYEE VALUES (0003, 'Ava', 'Sales');


-- fetch
SELECT * FROM EMPLOYEE;
SELECT * FROM AUDIT;
```

Output:

```
CREATE TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
INSERT 0 1
INSERT 0 1
 empid | name  |    dept
-------+-------+------------
     1 | Clark | Sales
     2 | Dave  | Accounting
     3 | Ava   | Sales
(3 rows)

 emp_id |          entry_date
--------+-------------------------------
      1 | 2024-04-18 06:51:32.120558+00
      2 | 2024-04-18 06:51:32.122062+00
      3 | 2024-04-18 06:51:32.123177+00
(3 rows)
```

Code:

```
CREATE OR REPLACE FUNCTION auditlogfunc() RETURNS TRIGGER AS $example_table$
  BEGIN
    INSERT INTO AUDIT(EMP_ID, ENTRY_DATE) VALUES (new.empId, current_timestamp);
    RETURN NEW;
  END;
$example_table$ LANGUAGE plpgsql;
CREATE TRIGGER example_trigger BEFORE INSERT ON EMPLOYEE
FOR EACH ROW EXECUTE PROCEDURE auditlogfunc();
```

```
-- insert
INSERT INTO EMPLOYEE VALUES (0001, 'Clark', 'Sales');
INSERT INTO EMPLOYEE VALUES (0002, 'Dave', 'Accounting');
INSERT INTO EMPLOYEE VALUES (0003, 'Ava', 'Sales');
```

Output:

```
CREATE TABLE
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
INSERT 0 1
INSERT 0 1
INSERT 0 1
 empid | name  |    dept
-------+-------+------------
     1 | Clark | Sales
     2 | Dave  | Accounting
     3 | Ava   | Sales
(3 rows)

 emp_id |          entry_date
--------+-------------------------------
      1 | 2024-04-18 06:54:37.012729+00
      2 | 2024-04-18 06:54:37.014727+00
      3 | 2024-04-18 06:54:37.016487+00
(3 rows)
```

Code:

```
CREATE OR REPLACE FUNCTION auditlogfunc2() RETURNS TRIGGER AS $example_table$
  BEGIN
    UPDATE AUDIT SET ENTRY_DATE = current_timestamp;
    RETURN NEW;
  END;
$example_table$ LANGUAGE plpgsql;
CREATE TRIGGER example_trigger2 AFTER UPDATE ON EMPLOYEE
FOR EACH ROW EXECUTE PROCEDURE auditlogfunc();
update employee set name = 'rohit' where empid = 0001;
```

```
CREATE FUNCTION
CREATE TRIGGER
UPDATE 1
 empid | name  |    dept
-------+-------+------------
     2 | Dave  | Accounting
     3 | Ava   | Sales
     1 | rohit | Sales
(3 rows)

 emp_id |          entry_date
--------+-------------------------------
      1 | 2024-04-18 07:01:48.136441+00
      2 | 2024-04-18 07:01:48.136441+00
      3 | 2024-04-18 07:01:48.136441+00
(3 rows)
```

Conclusion: In conclusion, triggers serve as crucial mechanisms in databases, facilitating automation, ensuring data integrity, enforcing business rules, and enhancing overall system functionality.