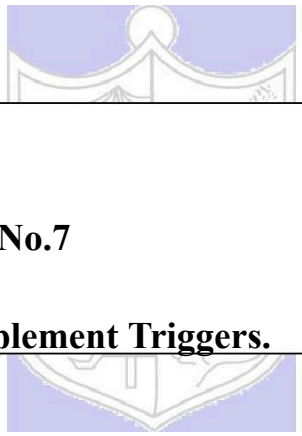

**Experiment No.7**

**Title: To Implement Triggers.**

**Batch:A3** **Roll No.:16010423099** **Experiment No.:7**

**Title:** To implement Triggers for given database.

---

**Resources needed:** PostgreSQL 9.3

---

**Theory**

**Pre Lab/ Prior Concepts:**

**Trigger:**

A trigger is a statement that the system executes automatically as a side effect of a modification to the database. Triggers are used to ensure some types of integrity.

To design a trigger mechanism you should:

1. Specify when a trigger is to be executed. This is broken up into an event that causes a trigger to be checked and a condition that must be satisfied for trigger execution to proceed
2. Specify the actions to be taken when the trigger executes.

**Generalized Model:**

Triggers are based on the Event- condition- Action (ECA) Model. A rule in the ECA model has three components.

1. The event(s) that triggers the rule.
2. The condition that determines whether the rule action should be executed. If no action is specified, the action will be executed once the event occurs.
3. The action to be taken. It could be a sequence of SQL statements, a DB transaction or an external program that will be executed automatically.

**When to use Trigger:**

In many cases, it is convenient to specify the type of action to be taken when certain event occurs and when certain conditions are satisfied.

It may be useful to specify a condition that, if violated, causes some user to be informed of the violation.

For example:-

A manager may want to be informed, if an employee's travel expenses exceed a certain limit by receiving a message whenever this occurs.

The condition is thus used to monitor the database.

CREATE TRIGGER statement is used to implement such action in SQL. Consider the triggers for following cases:-

1. Trigger for insertion:-

This trigger executes whenever condition is satisfied, during the insertion statement. If no condition is satisfied, then it executed for every insertion statement, for the relation specified.

Example:-

In DB, we have created a trigger for insertion, on the relation Employee. If salaries < 1500 then print 'Unsuccessful' else print 'Successful'.

2. Trigger for Updation:-

This trigger executes for updating of a column name(s) of a particular relation. A condition may or may not be specified.

Example:-

In our DB, we have created a trigger for updating, on the relation Employee. If salary is updated to unacceptable amount, then print unsuccessful and rollback else print successful.

3. Trigger for Deletion:-

This trigger executes during the deletion statement. If a particular condition is satisfied, the system may allow or not allow the deletion of certain tuples,

Example:-

If SSN from Employee =101, then print 'Unsuccessful' and rollback, else print 'Successful'.

In Postgresql we need to create a function to execute all action statements of trigger and call this function in create trigger statement for certain event.

Example.

In employeee table dnum is a foreign key. So the trigger writen here will increment total_emp of department table by one and total_sal of department by salary of newly inserted employee's salary.

```
CREATE or replace  FUNCTION inse_function() RETURNS trigger As $emp_update$

begin
     UPDATE dept set total_emp=total_emp + 1,
         total_sal=total_sal+ new.salary where dept.dno=new.dno;
   RETURN new;

END
$emp_update$ LANGUAGE plpgsql;


CREATE TRIGGER emp_update AFTER INSERT
      ON emp_table FOR each ROW
      EXECUTE PROCEDURE inse_function();
```

---

**Procedure / Approach /Algorithm / Activity Diagram**

1. Create new database for your application
2. Apply required integrity constraints on tables in your database

**To design a trigger.**

1. Identify events in the database.
2. Specify conditions under which the trigger is to be executed.
3. Specify actions to be taken when trigger is executed.

---

**Results: (Program printout with output / Document printout as per the format)**

**Adding course summary table:**

```
CREATE TABLE course_summary (
   course_id INT PRIMARY KEY,
   student_count INT DEFAULT 0,
   FOREIGN KEY (course_id) REFERENCES course(course_id) ON DELETE CASCADE
);
```

**Creating function for trigger:**

```
CREATE OR REPLACE FUNCTION update_student_count()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE course_summary
   SET student_count = student_count + 1
   WHERE course_id = NEW.course_id;
   RETURN NEW;
END;
```

**Creating the trigger:**

```
CREATE TRIGGER student_insert_trigger
AFTER INSERT ON student
FOR EACH ROW
EXECUTE PROCEDURE update_student_count();
```

**Initial Data Insertion:**

```
INSERT INTO course (course_id, course_name) VALUES (2, 'Physics');
INSERT INTO course_summary (course_id) VALUES (2);
INSERT INTO student (student_name, dob, course_id, city, state, pin)
VALUES ('John Doe', '2002-08-15', 2, 'San Francisco', 'CA', '94101');
```

**Verification:**

```
SELECT * FROM course_summary;
```

**Questions:**

1 Explain any one real-time application of trigger

A real-time application of triggers in PostgreSQL is an audit log for employee records. When records are modified, a trigger automatically logs the changes in an audit table, capturing details like employee ID, change type, and timestamp, ensuring continuous accountability without manual effort.