**Batch: B2 Roll No.: 110 and 109**

**Experiment / assignment / tutorial No. 6**

## Title: Queries based on Triggers

**Objective:** To be able to use trigger on table.

**Expected Outcome of Experiment:**

CO 3 : Use SQL for Relational database creation, maintenance and query processing

**Books/ Journals/ Websites referred:**

1. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
2. www.db-book.com
3. Korth, Slberchatz, Sudarshan : "Database Systems Concept", 5th Edition , McGraw Hill
4. Elmasri and Navathe,"Fundamentals of database Systems", 4th Edition,PEARSON Education.

**Resources used:** Postgresql

**Theory**

**Triggers** are database call-back functions, which are automatically performed/invoked when a specified database event occurs.

**Triggers** can be specified to fire

- Before the operation is attempted on a row (before constraints are checked and the INSERT, UPDATE or DELETE is attempted)
- After the operation has completed (after constraints are checked and the INSERT, UPDATE, or DELETE has completed)
- Instead of the operation (in the case of inserts, updates or deletes on a view)

The basic syntax of creating a trigger is as follows −

```
CREATE  TRIGGER trigger_name [BEFORE|AFTER|INSTEAD OF] event_name

ON table_name

[

 -- Trigger logic goes here....

];
```

event_name could be INSERT, DELETE, UPDATE, and TRUNCATE database operation on the mentioned table table_name. You can optionally specify FOR EACH ROW after table name.

The following is the syntax of creating a trigger on an UPDATE operation on one or more specified columns of a table as follows −

```
CREATE  TRIGGER trigger_name [BEFORE|AFTER] UPDATE OF column_name

ON table_name

[

 -- Trigger logic goes here....

];
```

## Implementation Screenshots (Problem Statement, Query and Screenshots of Results):

```
DROP DATABASE "AAtmaj";

CREATE DATABASE "AAtmaj";

drop table updates;

create table svvcred(netid varchar(15),  idno varchar(15),primary key(idno));

insert into svvcred (netid,idno) values('aatmaj.m','16010121110');

create table updates(foreign key (idno) references svvcred(idno),idno varchar(15),
updatedtime timestamp(6));

create function updatecheckk()

returns trigger as
```

```
$body$
begin
        if(old.netid<>new.netid) then

        insert into updates(idno,updatedtime) values(old.idno,now());
        end if;
        return new;
        end
        $body$
        language plpgsql;

create trigger updattion
before update
on svvcred
for each row
execute procedure updatecheckk();

update svvcred
set netid ='aa'
where netid = 'aatmaj.m';

select * from svvcred;
select * from updates ;
```

## Conclusion:

Triggers in a database management system are used to automatically execute a set of actions when certain events occur. These events can be database operations such as inserts, updates, or deletes on specific tables. Triggers can be very useful for enforcing business rules, maintaining data integrity, or auditing changes to the database.

## Post Lab Questions:

1. **Write a trigger to count number of new tuples inserted using each insert statement.**

```
CREATE OR REPLACE FUNCTION count_inserts()

RETURNS TRIGGER AS $$

DECLARE

  insert_count INTEGER;

BEGIN

  IF TG_OP = 'INSERT' THEN

    SELECT COUNT(*) INTO insert_count

    FROM inserted_table; -- Replace "inserted_table" with the actual name of the table being inserted into

    RAISE NOTICE 'Number of new tuples inserted: %', insert_count;

  END IF;

  RETURN NEW;

END;

$$ LANGUAGE plpgsql;

 CREATE TRIGGER insert_trigger

AFTER INSERT ON target_table -- Replace "target_table" with the actual name of the table being inserted into

FOR EACH STATEMENT
```

```
EXECUTE FUNCTION count_inserts();
```

2. **Trigger is special type of __________ procedure.**

a) Stored
b) Function
c) View
d) Table

3. **Triggers can be enabled or disabled with the ________ statement.**

a) ALTER TABLE statement
b) DROP TABLE statement
c) DELETE TABLE statement
d) None of the mentioned