

## DBMS Lab Week 9-10

Name : Abhishek Aditya BS

SRN : PES1UG19CS019

Section : A

### SQL – Creating Triggers and Functions

Write the SQL Triggers and functions for the following using Postgresql.

**<1>** Create an employee table which contains employee details and the department he works for. Create another table department consisting of dname and number of employees. Write triggers to increment or decrement the number of employees in a department table when the record in the employee table is inserted or deleted respectively.

#### Commands :

```
CREATE TABLE employee_details as SELECT e.fname,e.minit,e.lname,e.dno
from employee e;
```

```
Select * from employee_details;
```

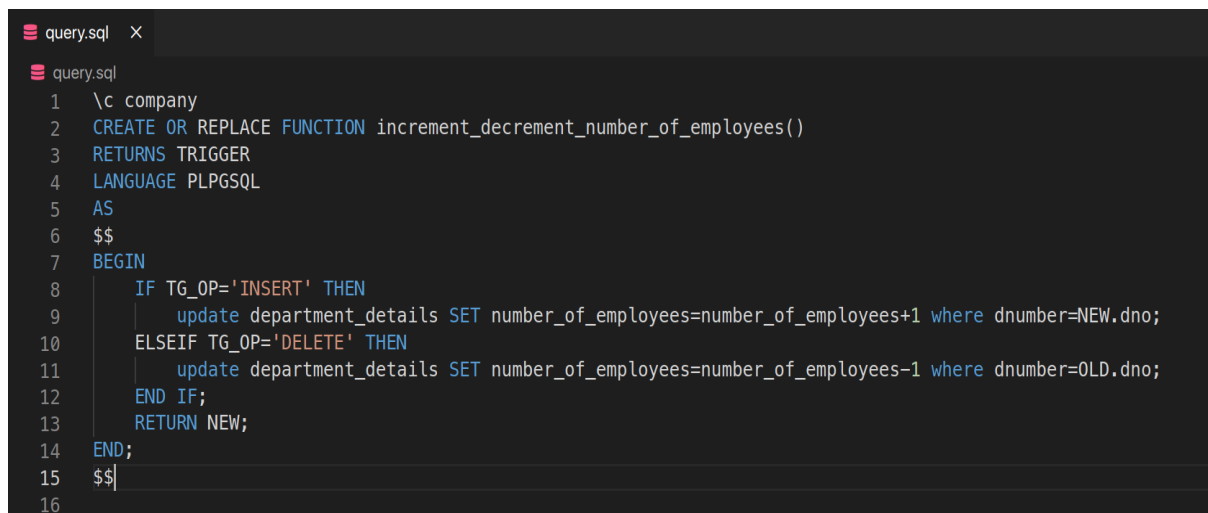
```
company=# CREATE TABLE employee_details as SELECT e.fname,e.minit,e.lname,e.dno from employee e;
SELECT 8
company=# select * from employee_details;
  fname | minit | lname | dno
-----+-----+-----+----
James   | E     | Borg  | 1
John    | B     | Smith | 5
Franklin| T     | Wong  | 5
Alicia  | J     | Zelaya| 4
Jennifer| S     | Wallace| 4
Ramesh  | K     | Narayan| 5
Joyce   | A     | English| 5
Ahmed   | V     | Jabbar | 4
(8 rows)
```

```
CREATE TABLE department_details AS select d.dname,
d.dnumber,COUNT(e.dno) AS number_of_employees from department d,
employee e where d.dnumber=e.dno GROUP BY d.dnumber;
Select * from department_details;
```

```
company=# CREATE TABLE department_details AS select d.dname, d.dnumber,COUNT(e.dno) AS number_of_employees from department d,
employee e where d.dnumber=e.dno GROUP BY d.dnumber;
SELECT 3
company=# select * from department_details;
  dname      | dnumber | number_of_employees
-----+-----+-----
Research    | 5       | 4
Administration | 4       | 3
Headquarters | 1       | 1
(3 rows)
```

## Creating a Trigger Function

```
\c company
CREATE OR REPLACE FUNCTION increment_decrement_number_of_employees()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN
    IF TG_OP='INSERT' THEN
        update department_details SET
number_of_employees=number_of_employees+1 where dnumber=NEW.dno;
    ELSEIF TG_OP='DELETE' THEN
        update department_details SET
number_of_employees=number_of_employees-1 where dnumber=OLD.dno;
    END IF;
    RETURN NEW;
END;
$$;
```

A screenshot of a SQL query editor window titled 'query.sql'. The editor shows the same SQL code as the previous block, with line numbers 1 through 16 on the left margin. The code is: 1 \c company, 2 CREATE OR REPLACE FUNCTION increment\_decrement\_number\_of\_employees(), 3 RETURNS TRIGGER, 4 LANGUAGE PLPGSQL, 5 AS, 6 \$\$, 7 BEGIN, 8 IF TG\_OP='INSERT' THEN, 9 update department\_details SET number\_of\_employees=number\_of\_employees+1 where dnumber=NEW.dno;, 10 ELSEIF TG\_OP='DELETE' THEN, 11 update department\_details SET number\_of\_employees=number\_of\_employees-1 where dnumber=OLD.dno;, 12 END IF;, 13 RETURN NEW;, 14 END;, 15 \$\$, 16 ;. The code is syntax-highlighted with blue for keywords and black for identifiers and literals.

## Executing the Trigger by executing the above procedure

```
CREATE TRIGGER increment_decrement_employee_trigger AFTER INSERT OR
DELETE ON employee_details FOR EACH ROW EXECUTE PROCEDURE
increment_decrement_number_of_employees();
```

```
company=# create TRIGGER increment_decrement_employee_trigger AFTER INSERT OR DELETE ON em
employee_details FOR EACH ROW EXECUTE PROCEDURE increment_decrement_number_of_employees();
CREATE TRIGGER
```

## Check for the trigger by inserting and deleting record

```
INSERT INTO employee_details VALUES('Adithya','M','S',5);
```

```
select * from employee_details;
```

```
select * from department_details;
```

```
company=# INSERT INTO employee_details VALUES('Adithya','M','S',5);
```

```
INSERT 0 1
```

```
company=# select * from employee_details;
```

fname	minit	lname	dno
James	E	Borg	1
John	B	Smith	5
Franklin	T	Wong	5
Alicia	J	Zelaya	4
Jennifer	S	Wallace	4
Ramesh	K	Narayan	5
Joyce	A	English	5
Ahmed	V	Jabbar	4
Adithya	M	S	5

(9 rows)

```
company=# select * from department_details;
```

dname	dnumber	number_of_employees
Administration	4	3
Headquarters	1	1
Research	5	5

(3 rows)

```
DELETE FROM employee_details where fname='Adithya';
```

```
select * from employee_details;
```

```
select * from department_details;
```

```
company=# DELETE FROM employee_details where fname='Adithya';
```

```
DELETE 1
```

```
company=# select * from employee_details;
```

fname	minit	lname	dno
James	E	Borg	1
John	B	Smith	5
Franklin	T	Wong	5
Alicia	J	Zelaya	4
Jennifer	S	Wallace	4
Ramesh	K	Narayan	5
Joyce	A	English	5
Ahmed	V	Jabbar	4

(8 rows)

```
company=# select * from department_details;
```

dname	dnumber	number_of_employees
Administration	4	3
Headquarters	1	1
Research	5	4

(3 rows)

<2> Create an order\_item table which contains details like name, quantity and unit price of every item purchased. Create an order summary table that contains the number of items and total price. Create triggers to update entry in order summary whenever an item is inserted or deleted in the order item table.

### Commands :

```
create table order_item(name VARCHAR(10),quantity INT,unit_price
DECIMAL(10,2));
```

```
company=# create table order_item(
name VARCHAR(10),
quantity INT,
unit_price DECIMAL(10,2)
);
CREATE TABLE
```

```
INSERT INTO order_item
VALUES('Apples',1,70.5),('Oranges',3,56),('Grapes',5,20);
select * from order_item;
```

```
company=# INSERT INTO order_item VALUES('Apples',1,70.5),('Oranges',3,56),('Grapes',5,20);
INSERT 0 3
company=# select * from order_item;
  name  | quantity | unit_price
-----+-----+-----
 Apples |         1 |      70.50
 Oranges |         3 |      56.00
  Grapes |         5 |      20.00
(3 rows)
```

```
CREATE TABLE order_summary as select COUNT(o.name) AS
number_of_items, SUM(o.quantity * o.unit_price) as Total_price
from order_item o;
select * from order_summary;
```

```
company=# CREATE TABLE order_summary as select COUNT(o.name) AS number_of_items, SUM(o.quantity
* o.unit_price) as Total_price from order_item o;
SELECT 1
company=# select * from order_summary;
 number_of_items | total_price
-----+-----
                3 |      338.50
(1 row)
```

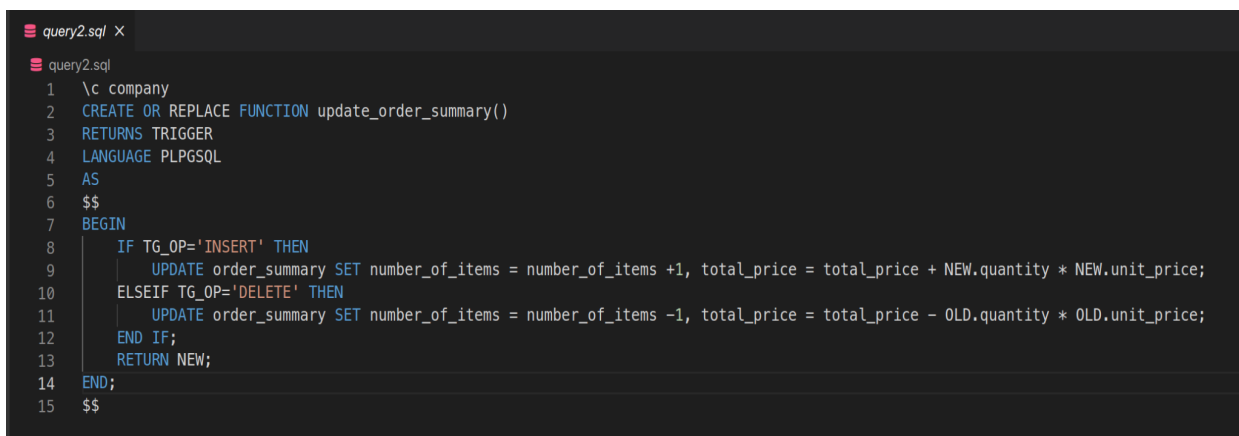
## Creating a Trigger Function

```
\c company
CREATE OR REPLACE FUNCTION update_order_summary()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN
```

```

        IF TG_OP='INSERT' THEN
            UPDATE order_summary SET number_of_items =
number_of_items +1, total_price = total_price + NEW.quantity *
NEW.unit_price;
        ELSEIF TG_OP='DELETE' THEN
            UPDATE order_summary SET number_of_items =
number_of_items -1, total_price = total_price - OLD.quantity *
OLD.unit_price;
        END IF;
        RETURN NEW;
    END;
$$

```



The screenshot shows a SQL editor window titled 'query2.sql'. The code defines a function 'update\_order\_summary()' that takes no arguments and returns a trigger. The function body contains an IF-ELSEIF statement that updates the 'order\_summary' table based on the trigger operation ('INSERT' or 'DELETE'). The function is created using 'CREATE OR REPLACE FUNCTION' and is terminated with 'END;' and '\$\$'.

```

1  \c company
2  CREATE OR REPLACE FUNCTION update_order_summary()
3  RETURNS TRIGGER
4  LANGUAGE PLPGSQL
5  AS
6  $$
7  BEGIN
8      IF TG_OP='INSERT' THEN
9          UPDATE order_summary SET number_of_items = number_of_items +1, total_price = total_price + NEW.quantity * NEW.unit_price;
10     ELSEIF TG_OP='DELETE' THEN
11         UPDATE order_summary SET number_of_items = number_of_items -1, total_price = total_price - OLD.quantity * OLD.unit_price;
12     END IF;
13     RETURN NEW;
14 END;
15 $$

```

## Executing the Trigger by executing the above procedure

```

create TRIGGER update_order_summary_trigger AFTER INSERT OR
DELETE ON order_item FOR EACH ROW EXECUTE PROCEDURE
update_order_summary();

```

```

company=# create TRIGGER update_order_summary_trigger AFTER INSERT OR DELETE ON order_item FOR
EACH ROW EXECUTE PROCEDURE update_order_summary();
CREATE TRIGGER

```

## Check for the trigger by inserting and deleting record

```

INSERT INTO order_item VALUES('Banana',1,10);
select * from order_item;
select * from order_summary;
DELETE FROM order_item where name='Banana';
select * from order_item;
select * from order_summary;

```

```
company=# INSERT INTO order_item VALUES('Banana',1,10);
INSERT 0 1
```

```
company=# select * from order_item;
   name   | quantity | unit_price
```

```
-----+-----+-----
Apples   |         1 |      70.50
Oranges  |         3 |      56.00
Grapes   |         5 |      20.00
Banana   |         1 |      10.00
```

(4 rows)

```
company=# select * from order_summary;
 number_of_items | total_price
```

```
-----+-----
                4 |      348.50
```

(1 row)

```
company=# DELETE FROM order_item where name='Banana';
DELETE 1
```

```
company=# select * from order_item;
   name   | quantity | unit_price
```

```
-----+-----+-----
Apples   |         1 |      70.50
Oranges  |         3 |      56.00
Grapes   |         5 |      20.00
```

(3 rows)

```
company=# select * from order_summary;
 number_of_items | total_price
```

```
-----+-----
                3 |      338.50
```

(1 row)