

ABHINAV RANJAN  
RA1911003010003  
CSE A1 SECTION  
SRMIST , KTR

## **DBMS LAB 13 - TRIGGERS** **IN PL/SQL**

### **AIM :**

To show the implementation of triggers in PL/SQL

### **THEORY :**

#### **TRIGGERS :**

Triggers are stored programs, which are automatically executed or fired when some events occur.

Triggers are, in fact, written to be executed in response to any of the following events –

- A database manipulation (DML) statement (DELETE, INSERT, or UPDATE)
- A database definition (DDL) statement (CREATE, ALTER, or DROP).
- A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Triggers can be defined on the table, view, schema, or database with which the event is associated.

#### **BENEFITS OF TRIGGERS**

Triggers can be written for the following purposes –

- Generating some derived column values automatically
- Enforcing referential integrity
- Event logging and storing information on table access
- Auditing
- Synchronous replication of tables
- Imposing security authorizations
- Preventing invalid transactions

### **SYNTAX FOR CREATING TRIGGERS :**

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
DECLARE
    Declaration-statements
BEGIN
    Executable-statements
EXCEPTION
    Exception-handling-statements
END;
```

Where,

- CREATE [OR REPLACE] TRIGGER trigger\_name – Creates or replaces an existing trigger with the trigger\_name.
- {BEFORE | AFTER | INSTEAD OF} – This specifies when the trigger will be executed. The INSTEAD OF clause is used for creating trigger on a view.

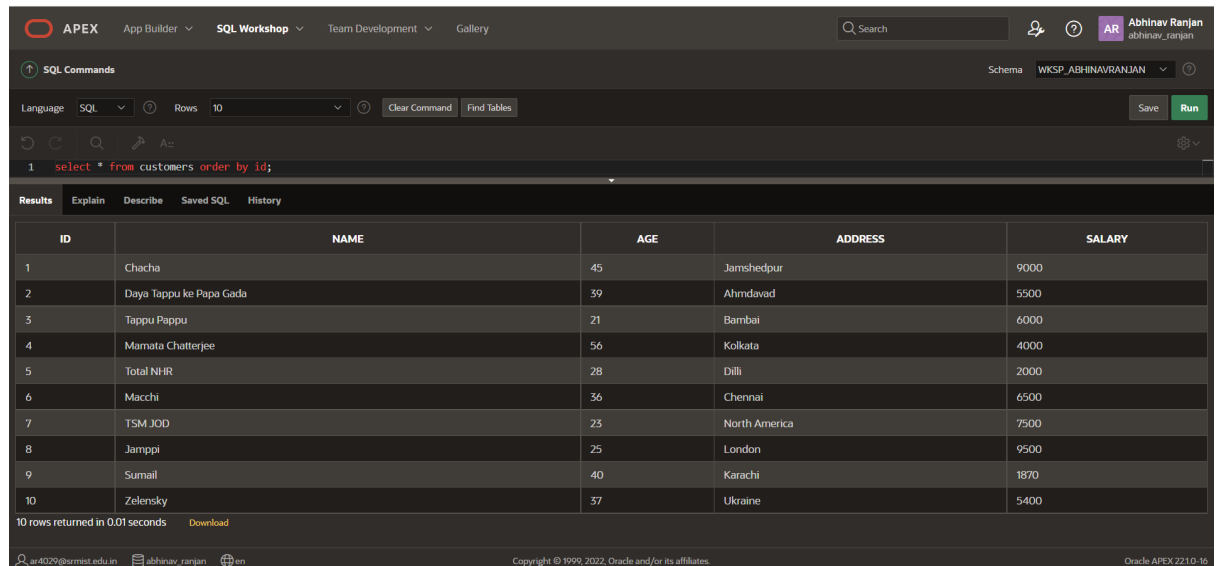
- {INSERT [OR] | UPDATE [OR] | DELETE} – This specifies the DML operation.
- [OF col\_name] – This specifies the column name that will be updated.
- [ON table\_name] – This specifies the name of the table associated with the trigger.
- [REFERENCING OLD AS o NEW AS n] – This allows you to refer new and old values for various DML statements, such as INSERT, UPDATE, and DELETE.
- [FOR EACH ROW] – This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- WHEN (condition) – This provides a condition for rows for which the trigger would fire. This clause is valid only for row-level triggers.

## **ALGORITHM :**

1. Assign a name and create trigger
2. Define the operations - before or after ,insert or update indicating when the trigger will activate
3. Declare a variable sal\_diff (salary difference)
4. The variable sal\_diff will hold the difference between old and new salary of a person
5. A message “trigger created” will pop up after we have finished defining trigger according to proper syntax as given above within the begin and end block
6. When we perform an insert or update operation next time , old and new salary is displayed as defined in the trigger

7. During insert , the old salary doesnt show anything as the record is inserted for the first time . The new salary is what we just inserted.
8. During the update operation it shows the old as well as the new salary

## SAMPLE TABLE USED :



ID	NAME	AGE	ADDRESS	SALARY
1	Chacha	45	Jamshedpur	9000
2	Daya Tappu ke Papa Gada	39	Ahmdavad	5500
3	Tappu Pappu	21	Bambaai	6000
4	Mamata Chatterjee	56	Kolkata	4000
5	Total NHR	28	Dilli	2000
6	Macchi	36	Chennai	6500
7	TSM JOD	23	North America	7500
8	Jamppi	25	London	9500
9	Sumail	40	Karachi	1870
10	Zelensky	57	Ukraine	5400

## SOURCE CODE :

### 1. CREATING THE TRIGGER

CREATE OR REPLACE TRIGGER

display\_salary\_changes

BEFORE DELETE OR INSERT OR UPDATE ON  
customers

FOR EACH ROW

WHEN (NEW.ID > 0)

DECLARE

sal\_diff number;

BEGIN

sal\_diff := :NEW.salary - :OLD.salary;

dbms\_output.put\_line('Old salary: ' || :OLD.salary);

dbms\_output.put\_line('New salary: ' || :NEW.salary);

```
        dbms_output.put_line('Salary difference: ' || sal_diff);
END;
```

## 2. USING UPDATE STATEMENT TO SHOW THE WORKING OF TRIGGER

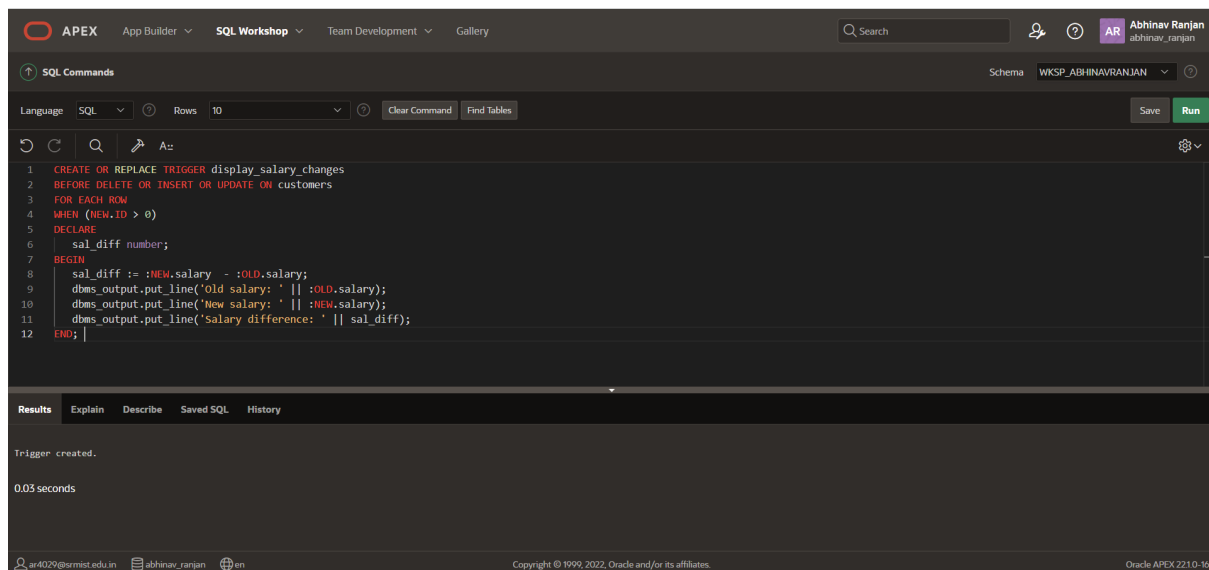
```
UPDATE customers SET salary = salary + 500
WHERE id = 2;
```

## 3. USING INSERT STATEMENT TO SHOW THE WORKING OF TRIGGER

```
insert into customers
(ID,NAME,AGE,ADDRESS,SALARY)
values (11,'LaBoiDre',22,Arizona,8970);
```

# SCREENSHOTS :

## 1. CREATING THE TRIGGER



## 2. TRIGGER WORKING AFTER UPDATE STATEMENT

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Abhinav Ranjan' are on the right. The 'SQL Commands' section is active, showing a list of commands: 1. UPDATE customers, 2. SET salary = salary + 500, 3. WHERE id = 2;. The 'Results' tab is selected, displaying the execution output: 'Old salary: 5000', 'New salary: 5500', 'Salary difference: 500', '1 row(s) updated.', and '0.00 seconds'. The bottom status bar shows the user 'ar4029@srmist.edu.in', the schema 'WKSP\_ABHINAVRANJAN', and the version 'Oracle APEX 22.1.0-16'.

```
1 UPDATE customers
2 SET salary = salary + 500
3 WHERE id = 2;
```

Results Explain Describe Saved SQL History

Old salary: 5000  
New salary: 5500  
Salary difference: 500  
1 row(s) updated.  
0.00 seconds

### 3. TRIGGER WORKING AFTER INSERT STATEMENT

The screenshot shows the Oracle APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. A search bar and user profile 'Abhinav Ranjan' are on the right. The 'SQL Commands' section is active, showing a list of commands: 1. INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY), 2. VALUES (11, 'LaBoiDre', 22, 'Arizona', 8970 );. The 'Results' tab is selected, displaying the execution output: 'Old salary:', 'New salary: 8970', 'Salary difference:', '1 row(s) inserted.', and '0.08 seconds'. The bottom status bar shows the user 'ar4029@srmist.edu.in', the schema 'WKSP\_ABHINAVRANJAN', and the version 'Oracle APEX 22.1.0-16'.

```
1 INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
2 VALUES (11, 'LaBoiDre', 22, 'Arizona', 8970 );
```

Results Explain Describe Saved SQL History

Old salary:  
New salary: 8970  
Salary difference:  
1 row(s) inserted.  
0.08 seconds

## RESULTS :

Thus we have successfully implemented triggers in SQL and shown its working after insert and update statements have been executed.