Subject: Database management system & RDBMS
Topic: Triggers
Name of the teacher: Lisna Thomas
Academic year:2020-21

## PL/SQL TRIGGERS

## PL/SQL – 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 could 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

## Creating Triggers

- The syntax for creating a trigger is:

```
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;
```

## Trigger Example:

- The following program creates a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new values:

```
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;
/
```

- When the above code is executed at SQL prompt, it produces the following result:

```
Trigger created.
```

# Triggering a Trigger

- Let us perform some DML operations on the CUSTOMERS table. Here is one INSERT statement, which will create a new record in the table:

  INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)VALUES (7, 'Kriti', 22, 'HP', 7500.00 );

  When a record is created in CUSTOMERS table, above create trigger **display_salary_changes** will be fired and it will display the following result:

  ```
  Old salary:
  New salary: 7500
  Salary difference:
  ```

- Because this is a new record so old salary is not available and above result is coming as null. Now, let us perform one more DML operation on the CUSTOMERS table. Here is one UPDATE statement, which will update an existing record in the table:
- UPDATE customersSET salary = salary + 500WHERE id = 2;
- When a record is updated in CUSTOMERS table, above create trigger **display_salary_changes** will be fired and it will display the following result:

  ```
  Old salary: 1500
  New salary: 2000
  Salary difference: 500
  ```

# TRIGGERS IN PLSQL

1. **What is a Trigger?**

Ans:- Trigger is an database Event or Action (or it is also called any DML statement or programs which will be fire after or before any DML statement).

2. **When it will be fired or worked?**

Ans:- Trigger is fired after or before any DML (Data Manipulation Language[insert,update and delete] not select statement).

3. **Is it will be fired based on Table or Any database object?**

Ans:- It will be fired on Table only.

4. **What is instead of Trigger?**

Ans:- It will be fired on Complex View. In complex view update are not possible. But if you used instead of trigger "It perform operation on base table instead of view"

That means Trigger is database event which will be fired before any DML or after any DML operation. And it will work on table. If you want to use view in trigger then it need base table to perform DML operation. That time use the instead of keyword, Which perform operation on Table instead of view.

**Types of Trigger**

1. Before insert
2. After insert
3. Before Update
4. After Update
5. Before Delete
6. After Delete
7. Instead of

How to references value in Trigger

**We are used two types of references in Trigger**

1. :new.columnname

2. :old.columnname

Suppose Here I am using two tables in trigger. My first table name is: Table1 and second table name is Table2. Table1 data already contains value. But Table2 value is NULL(Blank Table). I will create a trigger which will delete the row from Table1 and, same row inserted into Table2. That time we use "old" keyword to Table1 because values are old. Already stored. We referred as :old.colname. And new keyword for Table2 because new value to be inserted into Table2. We use :new.colname for Table2. We understand that already stored old value we refered as :old.columnname and new value to be inserted to Table2 is :new.columnname.

## Benefits of Trigger

1. Enforcing referential integrity. [ We can add constraints by trigger]
2. Preventing Invalid Transactions. [ prevents to delete important records ]

## Following parameter needs to create a trigger

1. Timing
2. Event
3. Classification based Level

## Timing [ before / after ]

When your statement will be fire. Means in which time trigger statement will execute? Before any DML command or after any DML command.

## Event [insert / delete / update ]

It specifies the event. After that event your trigger will fired. That events are insert, update or delete.

## Classification based Level [ statement / for each row ]

1. Statement level trigger-> It directly affects more than one row at same time for one statement.
2. Row level trigger-> It affects each individual rows for one statement or execute one rows for one statement. Use keyword "for each row" for row level trigger. If you will not write this then by default it takes

statement level trigger. In row level trigger we use new and old keyword to reference them.

**Example of Statement Level And Row level trigger.**

Update student set mark=50 [ This is one command but it will set mark 50 for all rows. This command affects more than one row is called statement level]

**Examples of Row level trigger**

Update student set mark=50 where roll=1 [This command will affect only one row whose roll=1. It is called row level]

**Syntax of crating trigger.**

```
Create or replace trigger trigger_name
[ After|before ]        [insert|update|delete]
On
[ Source table ]
[for each row]  or [don't write automatically it is statement level]
Declare
Variable declarations
Begin
Trigger Code [ we can use Target Table also ]
Exception
When.........Then
End;
```

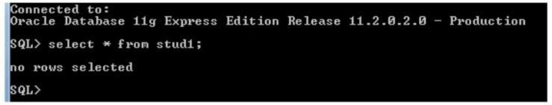
Or

If you want to create simple trigger then you can write small trigger code

```
Create or replace trigger trigger_name
[after|before] [insert|delete|update]
On
[table name]
For each row
Begin
Trigger Code
```

End;

Suppose now i will create a Trigger, when you will be insert record into stud table, it will first convert your name from any case to Capital letter then insert it.



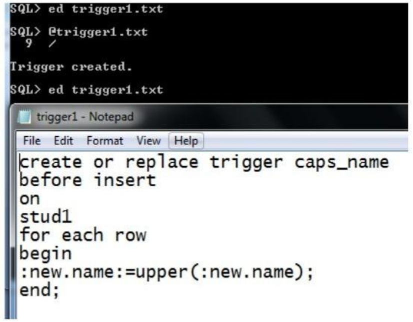
```
Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - Production

SQL> select * from stud1;

no rows selected

SQL>
```

Inside stud1 table there is no data. Now I will create trigger that will fired before inserted data into stud1 table. Trigger task is first converted any case to upper case then inserted.



```
SQL> ed trigger1.txt

SQL> @trigger1.txt
  9  /

Trigger created.

SQL> ed trigger1.txt
```

```
create or replace trigger caps_name
before insert
on
stud1
for each row
begin
:new.name:=upper(:new.name);
end;
```

I want insert new value to table so i used here "new" references.

## Description of Trigger details

1. Trigger Name:= caps_name
2. Trigger Timing:=before
3. Trigger Event:=insert
4. Category Level:=It is row level trigger [ for each row ]
5. Trigger body:= between [begin to end ]