PL/SQL Triggers

#### **What is Trigger?**

Oracle engine invokes automatically whenever a specified event occurs.

Trigger is stored into database and invoked repeatedly, when specific condition match.

You can change trigger mode activate/deactivate but you can't explicitly run.

Trigger automatically associated with DML statement, when DML statement execute trigger implicitly execute.

You can create trigger using the CREATE TRIGGER statement. If trigger activated, implicitly fire DML statement and if trigger deactivated can't fire.

#### **Component of Trigger**

Triggering SQL statement : SQL DML (INSERT, UPDATE and DELETE) statement that execute and implicitly called trigger to execute.

Trigger Action : When the triggering SQL statement is execute, trigger automatically call and PL/SQL trigger block execute.

Trigger Restriction : We can specify the condition inside trigger to when trigger is fire.

### Type of Triggers

1. BEFORE Trigger : BEFORE trigger execute before the triggering DML statement (INSERT, UPDATE, DELETE) execute. Triggering SQL statement is may or may not execute, depending on the BEFORE trigger conditions block.
2. AFTER Trigger : AFTER trigger execute after the triggering DML statement (INSERT, UPDATE, DELETE) executed. Triggering SQL statement is execute as soon as followed by the code of trigger before performing Database operation.
3. ROW Trigger : ROW trigger fire for each and every record which are performing INSERT, UPDATE, DELETE from the database table. If row deleting is define as trigger event, when trigger file, deletes the five rows each times from the table.
4. Statement Trigger : Statement trigger fire only once for each statement. If row deleting is define as trigger event, when trigger file, deletes the five rows at once from the table.
5. Combination Trigger : Combination trigger are combination of two trigger type,
   1. Before Statement Trigger : Trigger fire only once for each statement before the triggering DML statement.
   2. Before Row Trigger : Trigger fire for each and every record before the triggering DML statement.
   3. After Statement Trigger : Trigger fire only once for each statement after the triggering DML statement executing.
   4. After Row Trigger : Trigger fire for each and every record after the triggering DML statement executing.

### PL/SQL Triggers Syntax

PL/SQL trigger define using CREATE TRIGGER statement.

CREATE [OR REPLACE] TRIGGER trigger\_name

BEFORE | AFTER

[INSERT, UPDATE, DELETE [COLUMN NAME..]

ON table\_name

Referencing [ OLD AS OLD | NEW AS NEW ]

FOR EACH ROW | FOR EACH STATEMENT [ WHEN Condition ]

DECLARE

[declaration\_section

variable declarations;

constant declarations;

]

BEGIN

[executable\_section

PL/SQL execute/subprogram body

]

EXCEPTION

[exception\_section

PL/SQL Exception block

]

END;

Syntax Description

1. CREATE [OR REPLACE] TRIGGER trigger\_name : Create a trigger with the given name. If already have overwrite the existing trigger with defined same name.

1. BEFORE | AFTER : Indicates when the trigger get fire. BEFORE trigger execute before when statement execute before. AFTER trigger execute after the statement execute.

1. [INSERT, UPDATE, DELETE [COLUMN NAME..] : Determines the performing trigger event. You can define more then one triggering event separated by OR keyword.

1. ON table\_name : Define the table name to performing trigger event.

1. Referencing [ OLD AS OLD | NEW AS NEW ] : Give referencing to a old and new values of the data. :old means use existing row to perform event and :new means use executing new row to perform event. You can set referencing names user define name from old (or new).  
   You can't referencing old values when inserting a record, or new values when deleting a record, because It's does not exist.

1. FOR EACH ROW | FOR EACH STATEMENT : Trigger must fire when each row gets Affected (ROW Trigger). and fire only once when the entire sql statement is execute (STATEMENT Trigger).

1. WHEN Condition : Optional. Use only for row level trigger. Trigger fire when specified condition is satisfy.

### PL/SQL Triggers Example

You can make your own trigger using trigger syntax referencing. Here are fewer trigger example.

#### **Inserting Trigger**

This trigger execute BEFORE to convert ename field lowercase to uppercase.

CREATE or REPLACE TRIGGER trg1

BEFORE

INSERT ON emp1

FOR EACH ROW

BEGIN

:new.ename := upper(:new.ename);

END;

/

#### **Restriction to Deleting Trigger**

This trigger is preventing to deleting row.

Delete Trigger Code:

CREATE or REPLACE TRIGGER trg1

AFTER

DELETE ON emp1

FOR EACH ROW

BEGIN

IF :old.eno = 1 THEN

raise\_application\_error(-20015, 'You can't delete this row');

END IF; END;

/

Delete Trigger Result :

SQL>delete from emp1 where eno = 1;  
Error Code: 20015  
Error Name: You can't delete this row

**DISABLE TRIGGER**

ALTER TRIGGER CHANGECASE DISABLE;

**ENABLE TRIGGER**

ALTER TRIGGER CHANGECASE ENABLE;