

## Triggers in PL/SQL

TRIGGERS are stored programs that are block of PL/SQL code which Oracle engine can execute automatically based on some action or event.

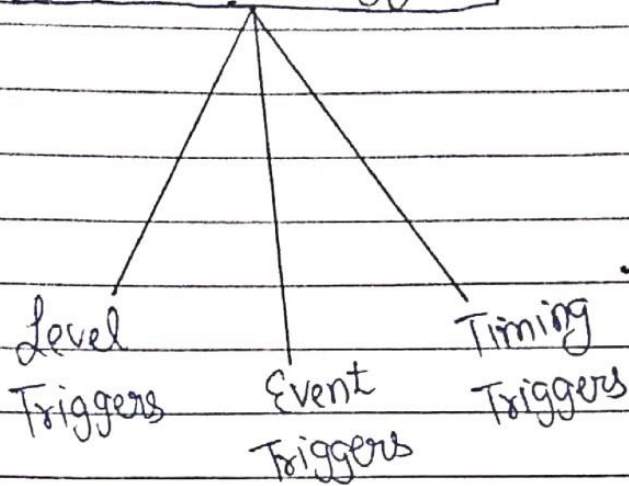
Events can be :-

- DDL statements (CREATE, ALTER, DROP, TRUNCATE).
- DML statements (INSERT, SELECT, UPDATE, DELETE).
- Database operations (LOGON, LOGOFF, SHUTDOWN).

A trigger is stored procedure in database which automatically invokes whenever a special event in the database occurs.

For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

# Types of Triggers



① Level Triggers :- There are 2 different types of level triggers, they are:-

(i) Row-Level Trigger :-

It fires for every record that got affected with the execution of DML statements like INSERT, UPDATE, DELETE etc.

It always use a FOR EACH ROW clause in a triggering statement.

(ii) Statement level Triggers :-

It fires once for each statement that is executed.

It is fired once on behalf of the triggering statement.

## (2) Event Triggers :-

There are 3 different types of event triggers, they are:

### (i) DDL EVENT TRIGGER :-

It fires with the execution of every DDL statement (CREATE, ALTER, DROP, TRUNCATE).

### (ii) DML EVENT TRIGGER :-

It fires with the execution of every DML statement (INSERT, UPDATE, DELETE).

### (iii) Database EVENT TRIGGER :-

It fires with the execution of every database operation which can be LOGON, LOGOFF, SHUTDOWN, SEVEREERROR.

## (3) Timing Triggers

(i) Before Triggers :- It fires before executing DML statement.

Triggering statement may or may not be executing depending upon the before condition block.

(ii) After Triggers :- It fires after executing DML statement.

## \* Syntax for Creating Triggers :-

`CREATE OR REPLACE TRIGGER <trigger name>`  
`BEFORE / AFTER / INSTEAD OF` ← specify the timing of trigger  
`INSERT / DELETE / UPDATE ON <tablename>`  
`REFERENCING (OLD AS O, NEW AS N)`  
`FOR EACH ROW WHEN (test condition)` ← specify condition, only for row level trigger  
`DECLARE`  
`-- Variable declaration;`  
`BEGIN`  
`-- Executable statements;`  
`EXCEPTION`  
`-- Error handling statements;`  
`END <trigger_name>;`  
`END;`

*SQL Statement* →  
*refer to old and new values* →  
*keyword* ↑  
*Specify row level trigger* →

## \* Example

Table name : emp  
 select \* from emp;

EMPID	ENAME	SALARY
1	Johan	30,000
2	Bob	31000
3	James	42000
4	Jasmine	43000
5	Garima	45000

- Create or replace trigger salary difference
- before insert or delete or update on emp
- for each row
- declare

```

salary_diff number;
begin
  salary_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: ' ||
                          salary_diff);
end ;
/

```

Output: Trigger created

→ Insert into emp values (6, 'Rahul', 34000); ← Query

old salary :  
 New salary: 34000  
 Salary difference :

Output: 1 row created.

→ delete from emp where lid = 6; ← Query

old salary : 34000  
 New salary :  
 Salary difference :

1 row deleted

\_ / \_ / \_

→ update emp set salary = 35000 where  
eid = 1

old salary : 30000

New Salary : 35000

Salary difference : 500

Output → 1 row updated

## \* Dropping a TRIGGER

In drop a Trigger, DROP TRIGGER statement is used to drop a trigger.

### Syntax

```
DROP TRIGGER <trigger_name>;
```

### Example

```
DROP TRIGGER T1;
```

## \* Storage for TRIGGERS

Upon first execution, the source code of a trigger is 'compiled and stored in the database.

→ To obtain information about any trigger, the data dictionary view 'USER\_TRIGGERS' is used.

### Example

```
Select * from USER_TRIGGERS;
```