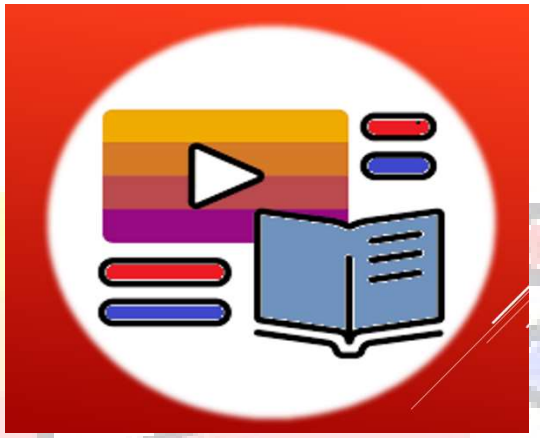
ORACLE

CHAPTER – 4



TRIGGERS

Let's make coding fun!

TRIGGERS

Triggers are stored programs, which are automatically executed or fired when some events occur. **"Oracle engine allows definition of procedures that are implicitly executed when an insert, update or delete is issued against a table from an application."**

- Enforce Complex Security
- Scheduling execution
- Prevent invalid transaction
- Keep auditing of database
- Row Triggers

Row Trigger is fired each time a row in the table is affected by triggering statement

For example UPDATE statement update multiple rows

- **Statement Triggers**

Fired once on behalf of triggering statement, independent of number of rows affected

- **Before v/s After Triggers**

Before triggers execute trigger action before triggering statement to deprive specific column in INSERT or UPDATE

After Triggers execute trigger action after triggering statement

- Cascading Triggers

When a trigger is fired, SQL statement inside the trigger's PL/SQL code block can also fire the same or other trigger.

- Combination Trigger

After / Before Statement

Row / Statement

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

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; 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.
- OLD and NEW references are not available for table-level triggers, rather you can use them for record-level triggers. If you want to query the table in the same trigger, then you should use the AFTER keyword, because triggers can query the table or change it again only after the initial changes are applied and the table is back in a consistent state.
- The above trigger has been written in such a way that it will fire before any DELETE or INSERT or UPDATE operation on the table, but you can write your trigger on a single or multiple operations, for example BEFORE DELETE, which will fire whenever a record will be deleted using the DELETE operation on the table.

Triggers Declaration Syntax

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSTEAD [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;
```

{BEFORE | AFTER | INSTEAD OF} – This specifies when the trigger will be executed.
The INSTEAD OF clause is used for creating trigger on a view.
{INSTEAD [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.

Trigger Demonstration

```
create or replace TRIGGER Trigger_1
BEFORE INSERT OR UPDATE OR DELETE ON tablea
FOR EACH ROW
WHEN (NEW.colb > 10)
DECLARE
   col_b_diff number;
BEGIN
   col_b_diff := :NEW.colb  - :OLD.colb;
   dbms_output.put_line('Old values: ' || :OLD.colb);
   dbms_output.put_line('New values: ' || :NEW.colb);
   dbms_output.put_line('COlB difference: ' || col_b_diff);
END;
```

Table A

COLA	COLB
ROW11	20
ROW21	12

CONCLUSION

In this chapter, we explained all about Triggers and their uses in database to program all operations in the tables.