**Objective**: Learning about Database Triggers, how to create Triggers and how Triggers work.

**What is Trigger? :** Database triggers are database objects created via the SQL \* Plus tool on the client and stored on the server in the Oracle engine’s system table.

The Oracle engine allows the definition of procedures that are implicitly executed when an insert, update and delete is issued against a table. This phenomenon is known as triggers.

**Use of Database Triggers:**

1. A trigger can permit DML statements against a table only if the are issued during regular business hours or on predetermined weekdays.
2. A trigger can also be used to keep an audit trail of a table.
3. It can be used to prevent invalid transactions.
4. Enforce complex security authorizations.

**How to apply database triggers:** A trigger has three basic parts:

1. **A triggering event of statement:** It is SQL statement that causes a trigger to be fired. It can be INSERT, UPDATE or DELETE statement for specific table.
2. **A trigger restriction:** A trigger restriction specifies a Boolean expression that must be TRUE for the trigger to fire.
3. **A trigger action:** A trigger action is the PL/SQL code to be executed when a triggering statement is encountered and any trigger restriction evaluates to TRUE.

**Creating a Trigger:**

**SQL syntax:**

CREATE OR REPLACE TRIGGER [Schema .] <TriggerName>

{BEFORE, AFTER}

{DELETE, INSERT, UPDATE [OF Column,…..]}

ON [Schema .] <TableName>

[REFERENCING {OLD AS old, NEW AS nes}]

[FOR EACH ROW [WHEN Condition]]

DECLARE

<Variable declarations>;

<Constant declarations>;

BEGIN

<PL/SQL Program body>;

EXCEPTION

<Exception PL/SQL block>;

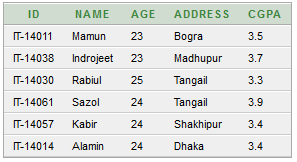
END;

**Meaning of these Keywords:**

|  |  |
| --- | --- |
| OR REPLACE | Recreates the trigger if it already exists |
| Schema | Is the schema, which contains the trigger |
| TriggerName | Is the name of the trigger to be created |
| BEFORE | Indicates that the Oracle engine fires the trigger before executing the triggering statement |
| AFTER | Indicates that the Oracle engine fires the trigger after executing the triggering statement |
| DELETE | Indicates that the Oracle engine fires the trigger whenever a DELETE statement removes a row from the table. |
| INSERT | Indicates that the Oracle engine fires the trigger whenever a INSERT statement add a row to the table. |
| UPDATE | Indicates that the Oracle engine fires the trigger when UPDATE statement changes a value in one of the columns specified in the OF clause. If OF statement is omitted, the Oracle engine fires the trigger whenever an UPDATE statement changes a value in any column of the table. |
| ON | Specifies the schema and name of the table, which the trigger is to be created. |
| REFERENCING | Specifies correlation name. |
| FOR EACH ROW | Designates the trigger to be a row trigger. |
| WHEN | Specifies the trigger restrictions. |
| PL/SQL block | Is the PL/SQL block that the Oracle engine executes when the trigger is fired |

**Example:**

I have a table named ‘student’ in my database. The table is as follows:



Now I want to create a trigger such that if a row is deleted then a variable named ‘operation’ is set to ‘UPDATED’ and if a row is deleted the variable is set to ‘DELETED’. And after that the deleted row or old value of updated row is inserted into a new table named ‘triggerInfo’.

Create the triggerInfo table:

**SQL:**

CREATE TABLE triggerInfo(id VARCHAR2(8), name VARCHAR2(25), age NUMBER, address VARCHAR2(25), cgpa NUMBER, operation VARCHAR2(10));

Now create the trigger (AUDIT):

**SQL:**

CREATE TRIGGER AUIT\_TRAIL

AFTER UPDATE OR DELETE ON student

FOR EACH ROW

DECLARE

operation VARCHAR2(10);

BEGIN

IF updating THEN

operation :='update';

END IF;

IF deleting THEN

operation :='delete';

END IF;

INSERT INTO triggerInfo values(:OLD.id, :OLD.name, :OLD.age, :OLD.address, :OLD.cgpa, operation);

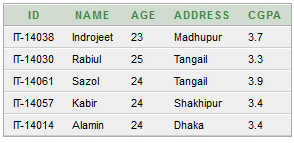
END;

Now delete a row from student table:

**SQL**: DELETE FROM STUDENT WHERE id='IT-14011';

So the student table and the triggerInfo table will be affected.

Student table:



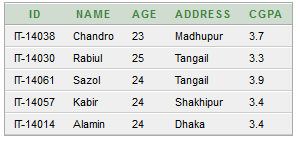
triggerInfo table:



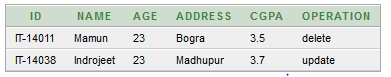
Now do update operation on a row of student table.

**SQL:** UPDATE student SET name='Chandro' WHERE name='Indrojeet';

So the student table will be:



And the triggerInfo table will be:



**Delete a trigger:**

**Syntax:**

DROP TRIGGER <TriggerName>;