**EXPERIMENT 9**

**AIM:** To develop procedures and function for various operations.

**FACILITIES REQUIRED**

|  |  |  |
| --- | --- | --- |
| **Serial No.** | **Facilities required** | **Quantity** |
| 1 | System | 1 |
| 2 | Operating System | Windows |
| 3 | Front End |  |
| 4 | Backend | Oracle Apex |

**PL/SQL Syntax:**

**TRIGGER**

A Trigger is a stored procedure that defines an action that the database automatically take when some database-related event such as Insert, Update or Delete occur.

**TRIGGER VS. PROCEDURE VS CURSOR**

|  |  |  |
| --- | --- | --- |
| **TRIGGER** | **PROCEDURES** | **CURSORS** |
| These are named PL/SQL blocks. | These are named PL/SQL blocks. | These are named PL/SQL blocks. |
| These are invoked automatically. | User as per need invokes these. | These can be created both explicitly and implicitly. |
| These can’t take parameters. | These can take parameters. | These can take parameters. |
| These are stored in database | These are stored in database. | These are not stored in database. |

**TYPES OF TRIGGERS**

The various types of triggers are as follows,

•**Before:** It fires the trigger before executing the trigger statement.

•**After:** It fires the trigger after executing the trigger statement.

•**For each row:** It specifies that the trigger fires once per row.

• **For each statement:** This is the default trigger that is invoked. It specifies that the trigger fires once per statement.

**VARIABLES USED IN TRIGGERS**

•:new

•:old

These two variables retain the new and old values of the column updated in the database. The values in these variables can be used in the database triggers for data manipulation

**Row Level Trigger vs. Statement Level Trigger:**

|  |  |
| --- | --- |
| **Row Level Trigger** | **Statement Level Trigger** |
| These are fired for each row affected by the DML statement. | These are fired once for the statement instead of the no of rows modified by it. |
| These are used for generating/checking the values begin inserted or updated. | These are used for generated the summary information |

**Before trigger vs. after trigger**

|  |  |
| --- | --- |
| **Before Triggers** | **After Triggers** |
| Before triggers are fired before the DML statement is actually executed. | After triggers are fired after the DML statement has finished execution. |

**Sytax:**

Create or replace trigger <trg\_name> Before /After Insert/Update/Delete

[of column\_name,

column\_name….] on <table name>

[for each row] [when condition]

begin ---statement

end;

**Queries**

**Q1: Create a trigger that insert current user into a username column of an existing table**

**Ans: SQL>** create table itstudent(name varchar2(15),username varchar2(15));

**SQL>**create or replace trigger itstudent before insert on itstudent for each row

declare

name varchar2(20);

begin

select user into name from dual;

:new.username:=name;

end;

/

**SQL>** insert into itstudent values('akbar','ranjani');

**SQL>** insert into itstudent values('suji','priya');

**SQL>** select\* from itstudent;

**Q2: Create a Simple Trigger that does not allow Insert Update and Delete Operations on the Table**

**Ans: SQL>** insert into itempls values('xxx',11,10000)

insert into itempls values('yyy',12,10500)

insert into itempls values('zzz',13,15500)

**SQL>** select \* from itempls;

**Trigger:**

**SQL>** create trigger ittrigg before insert or update or delete on itempls for each row

begin

raise\_application\_error(-20010,'You cannot do manipulation');

end;

/

**Output**:

**SQL>** insert into itempls values('aaa',14,34000);

**SQL>** delete from itempls where ename='xxx';

**SQL>** update itempls set eid=15 where ename='yyy';

**Q3: Create a Trigger that raises an User Defined Error Message and does not allow updating and Insertion**

**Table used:**

**Ans:SQL>** select \* from itempls;

**Trigger:**

**SQL>** create trigger ittriggs before insert or update of salary on itempls for each row

declare

triggsal itempls.salary%type;

begin

select salary into triggsal from itempls where eid=12;

if(:new.salary>triggsal or :new.salary<triggsal) then

raise\_application\_error(-20100,'Salary has not been changed');

end if;

end;

/

**Output:**

**SQL>** insert into itempls values ('bbb',16,45000);

**SQL>** update itempls set eid=18 where ename='zzz';

**Q4: develop a query to Drop the Created Trigger**

**Ans:** **SQL>** drop trigger ittrigg;

**Result:**

Thus the creation of triggers for various events such as insertion, updation, etc., was performed and executed successfully.