Ex.No.: 13		**WORKING WITH TRIGGER**
Date:	08/10/2024	**TRIGGER**

DEFINITION

A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database. The parts of a trigger are,

- **Trigger statement**: Specifies the DML statements and fires the trigger body. It also specifies the table to which the trigger is associated.
- **Trigger body or trigger action**: It is a PL/SQL block that is executed when the triggering statement is used.
- **Trigger restriction:** Restrictions on the trigger can be achieved

The different uses of triggers are as follows,

- *To generate data automatically*
- *To enforce complex integrity constraints*
- *To customize complex securing authorizations*
- *To maintain the replicate table*
- To audit data modifications

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

SYNTAX

```
create or replace trigger triggername [before/after] {DML statements}
on [tablename] [for each row/statement]
begin
------------------------
------------------------
------------------------
exception
end;
```

USER DEFINED ERROR MESSAGE

The package "raise_application_error" is used to issue the user defined error messages

Syntax: raise_application_error(error number,'error message');

The error number can lie between -20000 and -20999.

The error message should be a character string.

TABLE CREATION:

```
create table employeebonus(empno number(5)constraint emppk primary key, empname
varchar2(25)not null, experience number(2)not null,  bonus number(7,2));
Table created.
```

TRIGGER CREATION FOR BONUS CALCULATION:

```
SQL> set serveroutput on
SQL> create or replace trigger employeebonus_tgr
 after insert on employeebonus
declare
cursor emp is select * from employeebonus;
emprec employeebonus%rowtype;
 begin
```

```
open emp;
loop
fetch emp into emprec;
exit when emp%notfound;
if(emprec.experience<5) then
emprec.bonus:=5000;
elsif(emprec.experience>=5 and emprec.experience<8) then
emprec.bonus:=8000;
else
emprec.bonus:=10000;
 end if;
update employeebonus set bonus=emprec.bonus where empno=emprec.empno;
end loop;
close emp;
 dbms_output.put_line('Bonus calculated and Updated Sucessfully');
end;
/
```

Trigger created.

TABLE DESCRIPTION:

```
SQL> desc employeebonus;
Name Null? Type
-------------------------------- -------- ----------------------------
EMPNO NOT NULL NUMBER(5)
EMPNAME NOT NULL VARCHAR2(25)
EXPERIENCE NOT NULL NUMBER(2)
BONUS NUMBER(7,2)
```

RECORD INSERTION:

```
SQL> insert into employeebonus(empno,empname,experience)
values(&empno,'&empname',&experience);
Enter value for empno: 101
Enter value for empname: murugan
Enter value for experience: 25
old 1: insert into employeebonus(empno,empname,experience)
```

```
values(&empno,'&empname',&experience)
new 1: insert into employeebonus(empno,empname,experience)
values(101,'murugan',25)
Bonus calculated and Updated Sucessfully
1 row created.
```

RECORD SELECTION:

```
SQL> select * from employeebonus;
EMPNO EMPNAME EXPERIENCE BONUS
---------- ------------------------- ---------- ----------
101 murugan 25 10000
102 suresh 3 5000
103 akash 7 8000
104 mahesh 2 5000
```

RESULT:

Thus, the above program was Created and Executed Successfully.

Program 1

Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
Create or Replace TRIGGER P_del_par Before DELETE ON dept
FOR EACH ROW
DECLARE
      C_count NUMBER;
BEGIN
      select count(*) into c_count from emp where dept_id := OLD.
                                                          dept_id;
   IF C_count > 0 THEN
         RAISE_APPLICATION_ERROR (-20001;'cannot delete dept');
   END IF;
 END del_par
```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER check_dup_name BEFORE INSERT
OR UPDATE ON users
FOR EACH ROW
DECLARE
      V_count NUMBER;
BEGIN
      Select count(*) INTO v_count from users where username=
                                                   NEW.username;

      IF v_count > 0 THEN
            RAISE_APPLICATION_ERROR (20001,'Duplicate username found');

      END IF;
END;
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE OR REPLACE TRIGUER C-tot-amt BEFORE INSERT ON
Sales FOR EACH ROW
DECLARE
      tot-amt NUMBER;
      threshold CONSTANT NUMBER:=10000;
BEGIN
      select SUM(amt) INTO tot-amt from sales;
      IF tot-am + NEW-amt > threshold THEN
          RAISE-APPLICATION-ERROR(20001, 'cannot insert');
      END IF;
END;
```

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE OR REPLACE TRIGGER trg-emp-changes
AFTER UPDATE OF sal. dept_id ON emp
FOR EACH ROW
BEGIN
    INSERT into emp-(emp-id, change-col, old-value,
        new-value)
    (SELECT :OLD.emp-id, 'salary', TOCHAR(:OLD.salary)
      TO_CHAR(NEW: salary) from dual where:old salary,
      != new salary) UNION ALL (SELECT: OLD.emp-id,
      'dept-id', TO-CHAR(:OLD.dept-id), TO-CHAR
      (:new.dept-id) FROM dual where: OLD.dept-id!
      =NEW. dept-id;
END;
```

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE or replace trigger trg audit_emp
after insert or delete or update on emp
for each row
Begin
  insert into audit log (action_type, table_name, old_val,
   new_val, changed_by) values (case when inserting the
  'insert' when updating THEN 'update' ELSE 'DELETE', END,
   'employee', case when updating or deleting THEN :OLD:
   END, case when inserting or updating then :new * END, user);
END;
```

Program 7

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
Create or replace trigger trg
after insert on sales
for each row
Declare
    total number;
Begin
   select NVL(max (running_total), 0) + :new . amount
   into total from sales
   update sales set running_total = total where id =:
   new.id;
END;
```

Program 8

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
create or replace trigger trg
Before insert on orders
for each row
Declare
    v_stock_level NUMBER;
Begin
    select stock_level INTO v_stock_level from inventory where
    item.id =:new.item_id;
    IF v_stock_level < :NEW.quantity THEN
     Raise_application_error (20001,'Insufficient stock');
  END IF;
END;
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	