Ex. No : **CREATION OF DATABASE TRIGGERS AND FUNCTIONS**

AIM:

To create database triggers and functions.

DESCRIPTION:

TRIGGER

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 trigger name [before/after] {DML statements}

on [table name] [for each row/statement]

begin

---

exception

end;

PROCEDURE

Step1:Creates a trigger for insertion of each row.

Step2:Declare a cursor which contains the roll number field

Step3:Before insertion check of the roll number already exists in the table

Step4:If it exists raise an application error and display “roll no exists”.

Step5:Else perform insertion

PROGRAM

SQL>create table poo(rno number(5),name varchar2(10));

Table created.

SQL>insert into poo values (01.’kala’);

1 row created.

SQL>select * from poo;

RNO NAME

---

01 kala

02 priya

SQL>create or replace trigger pool before insert on poo for each row

2 declare

3 rno poo.rno%type

4 cursor c is select rno from poo;

```
5 begin
6 open c;
7  loop;
8 fetch c into rno;
9 if:new.rno=rno then
10 raise_application_error(-20005,'rno already exist');
11 end if;
12 exit when c%NOTFOUND
13 end loop;
14 close c;
15 end;
16 /
Trigger created.
```

OUTPUT

```
SQL>insert into poo values(01,'kala')
Insert into poo values (01,'kala')
*
ERROR at line1:
ORA-20005:rno already exist
ORA-06512:"SECONDCSEA.POOL",line 9
ORA-04088:error during execution at trigger "SECONDCSEA.POOL"
```

FUNCTIONS:

Functions are routines that accept parameters, perform an action such as a complex calculation and return the result of that action as a value. The return value can either be a single scalar value or a result set.

PROCEDURE

STEP 1: Start

STEP 2: Create the table with essential attributes.

STEP 3: Initialize the Function to carry out the searching procedure..

STEP 4: Frame the searching procedure for both positive and negative searching.

STEP 5: Execute the Function for both positive and negative result .

STEP 6: Stop

PROGRAM

```
SQL>create function fnfact(n number)
return number is
b number;
begin
b:=1;
for i in 1..n
loop
b:=b*i;
end loop;
return b;
end;
/
SQL>Declare
n number:=&n;
y number;
begin
y:=fnfact(n);
dbms_output.put_line(y);
end;
/
Function created.
```

INPUT

```
Enter value for n: 5
old 2: n number:=&n;
new 2: n number:=5;
```

OUTPUT

120

PL/SQL procedure successfully completed.

ALGORITHM

Step 1: Start the program

Step 2: Declare the variables f and i

Step 3: Initialize f to 1

Step 4: Start the for loop i in 1..a

Step 5: Compute f=f*i

Step 6: End the loop

Step 7: Return the factorial value

Step 8: Stop the program

PROGRAM

```
SQL> create or replace function fact(a number)return number as
2 i number;
3 f number;
4 begin
5 f:=1;
6 for i in 1..a
7 loop
8 f:=f*i;
9 end loop;
10 return f;
11 end fact;
12  /
```

Function created.

OUTPUT

```
SQL> set serveroutput on
SQL> begin
2 dbms_output.put_line('the factorial ='||fact(&a));
```

3 end;

4 /

Enter value for a:4

old 2: dbms_output.put_line('the factorial ='||fact(&a));

new 2: dbms_output.put_line('the factorial ='||fact(4));

the factorial=24

PL/SQL procedure successfully completed.

ALGORITHM

Step 1: Start the program

Step 2: Declare the variables i and f

Step 3: Initialize the values for f and i as 1

Step 4: Start the while loop 1 <= a

Step 5: Compute f=f*i

Step 6: Compute i=i+1

Step 7: Exit the loop

Step 8: Return the factorial value

Step 9: Stop the program

PROGRAM

SQL> create or replace function fact(a number)return number as

2 i number;

3 f number;

4 begin

5 f:=1;

6 i:=1;

7 while(i<=a)

8 loop

9 f:=f*i;

10 i:=i+1;

11 end loop;

12 return f;

13 end fact;

14 /

Function created.

OUTPUT

SQL> set serveroutput on

SQL> begin

2 dbms_output.put_line('the factorial ='||fact(&a));

3 end;

4 /

Enter value for a:5

old 2: dbms_output.put_line('the factorial ='||fact(&a));

new 2: dbms_output.put_line('the factorial ='||fact(5));

the factorial=120

PL/SQL procedure successfully completed.

PROGRAM

SQL> create table phonebook (phone_no number (6) primary key,username

varchar2(30),doorno varchar2(10),

street varchar2(30),place varchar2(30),pincode char(6));

Table created.

SQL> insert into phonebook values(20312,'vijay','120/5D','bharathi street','NGO

colony','629002');

1 row created.

SQL> insert into phonebook values(29467,'vasanth','39D4','RK bhavan','sarakkal vilai','629002');

1 row created.

SQL> select * from phonebook;

PHONE_NO USERNAME DOORNO STREET PLACE PINCODE

---------- ---------- ---------- ----------

20312 vijay 120/5D bharathi street NGO colony 629002

29467 vasanth 39D4 RK bhavan sarakkal vilai 629002

SQL> create or replace function findAddress(phone in number) return varchar2 as

address varchar2(100);

begin

```
select username||','||doorno ||','||street ||','||place||','||pincode into address from phonebook
where phone_no=phone;
return address;
exception
when no_data_found then return 'address not found';
end;
/
Function created.
SQL>declare
2 address varchar2(100);
3 begin
4 address:=findaddress(20312);
5 dbms_output.put_line(address);
6 end;
7 /
```

OUTPUT 1

```
Vijay,120/5D,bharathi street,NGO colony,629002
PL/SQL procedure successfully completed.
SQL> declare
2 address varchar2(100);
3 begin
4 address:=findaddress(23556);
5 dbms_output.put_line(address);
6 end;
7 /
```

OUTPUT2

```
Address not found
PL/SQL procedure successfully completed.
```

RESULT:

Thus the creation of database triggers and functions was implemented and executed successfully.