21/9/24

BVC GROUP
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

Experiment-8

Aim:-

Programs development using creation of Procedure passing parameters IN and OUT.

Sol:-

Create a procedure which generate all the prime numbers below the given number and count the no. of Prime numbers.

```
create procedure prime_proc (nIN number, tot OUT number)as
    i number;
    c number;
    j number;
Begin
    i:=1;
    tot:=0;
while (i<=n)
    loop
    j:=1;
    c:=0;
while (j<=i)
loop
    if(mod(i,j)=0)then
            c:c+1;
    end if;
        j:=j+1;
end loop;
```

output:-

SQL Procedure created

2
3
5
7

The total prime number are 4.

```
if (c=2) then
        dbms_output.Put_line(i);
tot := tot+1;
end if;
i:=i+1;
end loop;
end;
/

/* Calling Procedure */
set serveroutput on
declare
        t number;
begin
        Prime_proc(u,t);
        dbms_output.put_line('the total prime number are' ||t);
end;
```

28/9/24

BVC GROUP
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

Experiment-9

Aim:-

Program development using creation of stored functions, invoke functions in SQL statements and write complex function

```
create table dept(deptno int, dname varchar(10));
insert into dept values(1219,'CSE')
```

Program:-

```
create function getname (dno number)
return varchar2 as
fname1 varchar2(30);
begin
    select dname
    into fname1
    from dept
    where deptno=dno;
    return(fname1);
exception
    when no_data_found then
    raise_application_error(-20100, 'The dno is not present');
end;

/* calling function */
declare
    fname2 varchar2(30);
    deptno2 number(5);
begin
```

Output:

Table Created
1 row(s) inserted
function created
Statement processed
CSE

```
deptno2 := 1219;
fname2 := getname(deptno2);
dbms_output.put_line(fname2);
end;
/
```

O/P:-

```
1 Table created
1 row(s) inserted
1 row(s) inserted
1 table created
Package created

Package body created
```

BVC GROUP
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

3/10/24

Experiment-10

~~10~~ ~~Program~~

10 Aim:-

Program Development using creation of package specification, package bodies, private objects, package variables and cursors and calling stored packages.

1) create a table dept1

->create table dept1(dname varchar2(10), deptno number);

->Insert into dept values('accounting', 10);

->Insert into dept values('hr', 20);

2) Create a table dept

->create table dept(dno number, vt varchar2(10), dloc varchar2(20));

3) Creating package header

create or replace package test

is

procedure savedept

(dno in number, dloc in varchar);

end;

/

4) creating package body:-

create (or) replace package body test

is

function getdno(dno in number)

return varchar

is

Output:-

PNO	VT	DLOC
10	accanting	Vijayawada

```
dnum varchar(20);
begin
    select dname into dnum from dept
    where deptno=dno;
    return dnum;
end;
procedure savedept
(dno in number,dloc in varchar)
is
vt varchar(20)
begin
    vt:=getno(dno);
    insert into dept values(dno,vt,dloc);
exception
    when dup_value_on_index then
    raise_application_error(-2007,'duplicate');
end;
end;
/
```

5) Executing the procedure

```
exec test.savedept(10,'Vijayawada');
```

6) Display the table

```
select * from dept;
```

Sid	Sname	Rating	Age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horatio	7	35
71	Zobra	10	16
74	Horatio	9	35
85	Art	3	25.5
95	Bob	3	63.5

BVC GROUP
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

9/10/24

Page No. 36

Experiment-11

Working CURSORS

Develop programs using Features Parameters in a CURSOR. FOR UPDATE CURSOR, WHERE CURRENT of clause and CURSOR Variable

Cursors:- Oracle uses temporary work area Cursor for storing output of an SQL statement.

Cursors are defined as

CURSOR C1 is SELECT SID, SNAME, RATING, AGE FROM SAILORS;

OR

CURSOR C1 is SELECT * FROM SAILORS;

Generally while using Cursors we have to open the Cursor then extract one row (record) from the cursor using Fetch operation, do the necessary operations on the record. After completing the work Close the CURSOR. But if we want to do automatic opening & closing to cursor then we can use FOR Loop in cursor.

FOR loop in cursor

The cursor FOR loop gives easy way to handle the cursor. The FOR loop opens the cursor, fetches rows and closes the cursor after all rows are processed.

For eg:-

```
FOR Z IN C1 LOOP
------
END LOOP;
```

The cursor FOR loop declare z as record, which can hold row, returned from cursor.

Ex using cursor:-

```
DECLARE
    FOR Z IN C1 LOOP
        DBMS_OUTPUT.PUT_LINE(Z.SID||'  '||Z.SNAME
    END LOOP;
END;
/
```

Same example using while:-

```
DECLARE
    CURSOR C1 IS SELECT * FROM SAILORS;
    Z C1%ROWTYPE;
BEGIN
    OPEN C1;
        FETCH C1 INTO Z;
        WHILE (C1%FOUND) LOOP
            DBMS_OUTPUT.PUT_LINE(Z.SID||'  '||Z.SNAME);
            FETCH C1 INTO Z;
        END LOOP;
    CLOSE C1;
END;
/
```

Statement Processed
Sailors with rating 1 are

sid	name	age
29	brutus	33

Sailors with rating 2 are

sid	name	age

Passing parameters to cursor

We can pass parameters to cursor, when you open the value is passed to cursor and processing can be done in the cursor definition:

Ex:

```
DECLARE
    CURSOR C1(R NUMBER) IS
    SELECT * FROM SAILORS WHERE RATING=R;
    I INTEGER;
BEGIN
    FOR I IN 1.....10 LOOP
        DBMS_OUTPUT.PUT_LINE('SAILORS WITH RATING'|| i ||'ARE');
        DBMS_OUTPUT.PUT_LINE('SID NAME AGE');
    FOR Z IN C1(1)LOOP
/* It is not compulsory to define variable using rowtype for
simple cursor as well as for update. Cursor */
    DBMS_OUTPUT.PUT_LINE(Z.SID||' '||Z.SNAME||''||Z.AGE);
    END LOOP;
    END LOOP;
END;
```

Use of Cursor for update AND WHERE CURRENT OF
(FOR UPDATE CURSOR, WHERE CURRENT of clause and CURSOR Variabe is used).

We can use update cursors for update operation only. The following example shows change of rating using update, cursor

~~Statement Processed~~

Sid	SNAME	RATING	AGE
22	Dustin	8	45
29	Brutus	3	33
31	Lubber	9	59
32	Andy	9	26
58	Rusty	10	35
64	Horatio	8	35
71	Zobra	10	16
74	Horatio	10	35
85	Art	5	26
95	Bob	5	64

Ex:-

```
DECLARE
    CURSOR C1 IS SELECT * FROM SAILORS FOR UPDATE;
BEGIN
  FOR Z IN C1 LOOP
/* It's not compulsory to define variable using rowtype for
update cursor as well as for simple cursors */

  IF(Z.RATING<=5)THEN
      UPDATE.SAILORS.SET RATING=RATING+2
      WHERE CURRENT OF C1;
  ELSE IF(Z.RATING>5 AND Z.RATING<10)
  UPDATE SAILORS SET RATING=RATING+1
    WHERE CURRENT OF C1;
    END IF;
    END LOOP;
END;
```

O/P:-

Trigger Created

9/24

BVC GRO

ODALAREVU • AMALAPURAM • RAJA

Working wi

Develop pro

Statement -

Trigger:-

A tr

the DBMS

typically s

-> A datab

Active Da

-> A trigge

• Event:-

• Condition:-

• Action:-

and it's

-

-> An ins

regardles

statemer

executed

Trigger

1) Trigge

BEFOR

10/24

BVC GROUP
ODALAREVU • AMALAPURAM • RAJAHMUNDRY

Experiment-12

Page No. 40

Working with Tri

Develop programs using before and after triggers, Row and Statement Triggers and INSTEAD OF Trigger.

Trigger:-

A trigger is a procedure that is automatically invoked by the DBMS in response to specied changes to the database, and is typically specified by the DBA.

→ A database that has a set of associated triggers is called an Active Database.

→ A trigger description contains three parts:-

- Event:- A change to the database that activates the trigger.
- Condition:- A query (or) test that is run when the trigger is activated.
- Action:- A procedure that is executed when the trigger is activated and it's condition is true.

→ An insert, delete (or) update statement could activate a trigger, regardless of which user (or) application invoked the activating statement, users may not even be aware that a trigger was executed as a side effect of their program.

Trigger Examples:-

1) Trigger to convert SNAME field lowercase to uppercase - executes BEFORE INSERT :

Trigger Name	Trigger type	Triggering event	Table-owned	Base-object type	Table-Name	Column-name	Referencing name	When class.
Age-limit	Before each row	insert	SQL-YCWDGFUF IXGYWOMBOG	Table	sailors	-	Referencing new (or) new old as old	NEW-age >60
NO Delete	After each row	Delete	SQL-YEWDQFU NGILWOXSMCQS	Table	sailors	-	Referencing new as new old as old	-
Upper name	Before each row	Insert	SQL-YWDQEUN YGIYWOXSMCQ	Table	sailors	-	Referencing new as new old (or) old	-

status	description	ACTION-TYPE	Trigger body	Before Statement	Before-Row	After Statement	After row	instead of Row
enabled	age limit before insert on sailors for open each row	PL/SQL	Begin raise-application NO (-20998, error invalid) end;	NO	NO	NO	NO	NO
enabled	NO delete after delete on sailors for each row	PL/SQL	begin if :old.sid-58 then NO raise_application error (-2009, you can it)	NO	NO	NO	NO	NO
Enabled	Upper name before insert on sailors for each row	PL/SQL	end if; end begin :new.sname := upper (:new.name); end;	NO	NO	NO	NO	NO

First-once	Apply-server only
Yes	NO
Yes	NO
Yes	NO

```
CREATE OR REPLACE TRIGGER UPPERNAME
BEFORE INSERT ON SAILORS FOR EACH ROW
BEGIN
    NEW.SNAE := UPPER(:NEW.SNAME);
END;
/
```

⇒TO test whether UPPERNAME Trigger working properly or not.

Triggers to restrict Deleting:-
This trigger prevents deleting SID=1 row

```
CREATE OR REPLACE TRIGGER NODELETE
AFTER DELETE ON SAILORS FOR EACH
ROW BEGIN
IF : OLD.SID = 58 THEN
RAISE_APPLICATION_ERROR(-20015,'YOU CAN NOT DELETE THIS
                                ROW');

ENDIF;
END;
/
```

Program to indicate invalid age (if sailors age is more than 65) using trigger.

```
Create or Replace trigger Age limit
Before insert on Sailors for each row
```

Description	Trigger Body

age limit, before insert on sailors for each row	begin
	raise_application_error(-20998, error-invaid age);
	end;

Trigger dropped

Trigger_name	Trigger_type	Triggering event	Trigger body

Age limit	Before each row	Insert	begin
			raise_application_error
			(-20998, 'error'; invalid age);
			end;

No Delete	After each row	Delete	begin
			if: old.sid = 56 then
			raise_error(-20016; 'You can't leave');
			end if;
			end;

Upper name	Before each row	Insert	begin
			: new.sname_upper(: real.sname);
			end;

when (NEW.
Begin
RAISE
END;
/

By using
CREATE
BEFORE
BEGIN
IF C:N
RAISE_A
END IF
END;
/

->TO see
SELE

->TO Sh
SELECT
WHERE

->TO S
SELE
TRIG
FRO

```
when (New.Age >60)
Begin
    RAISE_APPLICATION_ERROR(-20998,'ERROR:INVALID AGE');
END;
/
```

By using IF Statement:-

```
CREATE OR REPLACE TRIGGER AGE LIMIT
BEFORE INSERT ON SAILORS FOR EACH ROW
BEGIN
IF C:NEW-AGE> 60 THEN
RAISE_APPLICATION_ERROR(-20998,'Error Invalid Age');
END IF
END;
/
```

->TO see all users defined triggers details.

```
SELECT * FROM USER_TRIGGERS;
```

->TO Show particular TRIGGER

```
SELECT DESCRIPTION,TRIGGER_BODY FROM USER_TRIGGERS
WHERE TRIGGER_NAME='AGE LIMIT';
```

->TO Show different triggers

```
SELECT TRIGGER_NAME,TRIGGER TYPE,
TRIGGERING_EVENT,TRIGGER_BODY
FROM USER_TRIGGERS;
```

→ TO delete a particular user defined trigger.

```
DROP TRIGGER AGELIMIT;
```