**PL SQL**

\*Procedure language /structural query language.

1.Anonymous block

2.Stored procedure

3.triggers

4.functions.

Anonymous Block:

Declare

Declaration stmts;

Begin

Stmts;

[Exception]

End;

/

\*To see the outputs generated by pl/sql blocks in the SQL plus window execute the command

Set serveroutput on;

\*The statement used to print the data is

Dbms\_output.put\_line(‘hello’);

Ex: Declare

Begin

Dbms\_output.put\_line(‘hello’);

End;

/

\*concatenation

Dbms\_output.put\_line(‘hello ‘ ||250);

\*variable declaration in pl/sql:

Variable datatype;

\*Assignment operator in pl/sql:

:=

Ex:declare

Var1 varchar(20);

Var2 varchar(10)

Begin

Var1:=’leela’;

Var2:=100;

Dbms\_output.put\_line(var1 ||’ ‘ ||var2);

End;

/

Decision Making Statements:

If condition

Then

Stmts;

End if;

If condition

Then

Stmts;

Else

Stmts;

End if;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*(Elsif)

If condition

Then

Stmts;

Elsif condition

Stmts;

[else]

End if;

\*\*\*\*\*\*\*\*\*(Nested if)

If condition

Then

If condition

Then

Stmts;

End if;

End if;

/

\*\*Maximum of two numbers

Ex:declare

Num1 number(5);

Num2 number(5);

Begin

Num1:=500;

Num2:=100;

If num1>num2

then

Dbms\_output.put\_line(num1);

End if;

If num2>num1

then

Dbms\_output.put\_line(num2);

End if;

End;

/

Ex:declare

Num1 number(5);

Num2 number(5);

Begin

Num1:=500;

Num2:=100;

If num1>num2

then

Dbms\_output.put\_line(num1);

Else

then

Dbms\_output.put\_line(num2);

End if;

End;

/

\*\*Maximum of three numbers

Ex:declare

Num1 number(5);

Num2 number(5);

Num3 number(5)

Begin

Num1:=500;

Num2:=100;

Num3:=20;

If num1>num2 and num1>num3

then

Dbms\_output.put\_line(num1);

Elsif num2>num3 and num2>num3

then

Dbms\_output.put\_line(num2);

Else

Dbms\_output.put\_line(num3);

End if;

End;

/

Loops statements:

syntax:

loop

loop stmts;

exit when loop\_condition\_toStop;

end loop;

Ex:

declare

num number(2);

begin

num:=1;

loop

dbms\_output.put\_line(num);

num:=num+1;

exit when num=11;

end loop;

end;

/

Ex:program to print even numbers from 20 to 10

declare

num number(2);

begin

num:=20;

loop

dbms\_output.put\_line(num);

num:=num-2;

exit when num<10;

end loop;

end;

/

Ex: declare

num number(2);

begin

num:=20;

loop

if mod(num,2)=0 // crct way to do // % operator not works in plSQL

then

dbms\_output.put\_line(num);

end if;

num:=num-2;

exit when num<10;

end loop;

end;

/

INTO CLAUSE:

\*into clause is used to read the output of the sql query from the cursor into the variables of PLSQL block.

\*rules to be followed:

1.the output of the sql query should have only one row.

2.the corresponding arguments of select clause and into clause must have same datatypes.

ex:anonymous block to obtain employee name and dept name of an employee whose empno=7782.

declare

name varchar(20);

deptname varchar(20);

begin

select ename,dept.dname into name,deptname

from emp,dept

where emp.deptno=dept.deptno and empno=7782;

dbms\_output.put\_line(name||' '||deptname);

end;

/

% type:-

\*\*if datatype is smaller than the returned type it gives error.

\*% type identify the data type of the column specified and assigns the datatype to the variable.

syntax:

identifier username.table\_name.cloumn\_name%type;

Ex: name scott.emp.ename%type;

%rowtype:-

Identifier username.table\_name%rowtype;

Ex: r scott.emp%rowtype;

\*row type is an object which consists of all the column types. here r is reference variable.

Ex:

declare

r scott.emp%rowtype;

begin

select \* into r from emp

where sal=800;

dbms\_output.put\_line(r.ename);

dbms\_output.put\_line(r.hiredate);

dbms\_output.put\_line(r.sal);

end;

/

STORED PROCEDURE:-

syntax:

create [or replace] procedure procedue\_name[(variable IN/OUT/INOUT,..)]

as

[declaration]

begin

stmts;

end [procedure\_name];

Ex:create or replace procedure to\_read(eid in number)

as

r scott.emp%rowtype;

begin

select \* into r from emp

where empno=eid;

dbms\_output.put\_line('name : '||r.ename);

end;

/

assignment:

1 wa stored procedure which can accept one argument as input ,it should display the nth max salary.

2.wa stored procedure to display details of the department such as

1.no of employees working in it

2.avg salary of the department

3.minimum salary of the department

which can accept deptno as argument

1A) create or replace procedure p1(n in number)

as

salary scott.emp.sal%type;

begin

select sal into salary from emp e1

where (select count(distinct sal) from emp e2 where e1.sal<e2.sal)=n-1;

dbms\_output.put\_line(n||’ th max salary is ‘||salary);

end;

/

\*execute p1(1); //to execute the stored procedure

Or

\* Begin //executing stored procedure using anonymous block

P1(1);

End;

\*show errors; //to show errors in stored procedure

2A) create or replace procedure p2(dno in number)

as

N\_emp number;

minsal scott.emp.sal%type;

avgsal scott.emp.sal%type;

begin

select count(\*),min(sal) ,avg(sal) into N\_emp,minsal,avgsal from emp

where deptno=dno;

dbms\_output.put\_line(‘details of dept no : ‘||dno);

dbms\_output.put\_line(‘no \_of\_ emp : ‘||N\_emp);

dbms\_output.put\_line(‘min of salary : ‘||minsal);

dbms\_output.put\_line(‘avg of salary : ‘||avgsal);

end;

/

To modify the plsql procedure:

\*\*Desc ALL\_SOURCE

\*\*Select text from All\_source

where type='PROCEDURE' and name='p1';

3)wasp to take input as emp number and display his reporting managers name.

create or replace procedure p3(eid in number)

as

name scott.emp.ename%type;

begin

select ename into name from emp

where empno IN (select mgr from emp where empno=eid);

dbms\_output.put\_line(‘the name of the mgr is :’||name);

end;

/

\*\*if the emp doesn’t have mgr then it gives run time error which can be handled by exception.

CURSOR:

\*Cursor is a block of memory where an sql query gets executed.

\*we have implicit cursor and explicit cursor.

\*implicit cursors are the default cursors created by the plsql and we can access only one row from it.

\*explicit cursors are the cursors created by the programmers explicitly(it is similar to declaring a variable).

To declare a cursor:

We declare the cursor in declaration section.

\*Cursor cursor\_name

IS

Sql stmts;

\*\*For an explicit cursor the programmer should follow three steps.

--open the cursor

--fetch the rows from cursor

--close the cursor

Open the cursor:

Open cursor\_name;

Ex: open A;

Fetch the rows:

Read row from the cursor.

Fetch cursor\_name into variable;

Close the cursor:

Close cursor\_name;

Ex:create or replace procedure p4(dno in number)

as

name scott.emp.ename%type;

Cursor A

IS

select ename into name from emp

where deptno=dno;

begin

open A;

Fetch A into name;

Close A;

dbms\_output.put\_line(name);

end;

/

Ex: create or replace procedure p4(dno in number)

As

Name emp.ename%type;

I number;

N number;

Cursor A

Is

Select ename from emp

Where deptno=dno;

Begin

I:=0;

Select count (\*) into n from emp

Where deptno=dno;

Open A;

Loop

Fetch A into name;

Dbms\_output.put\_line(name);

If n=0

Then

Dbms\_output.put\_line(‘NO employees exist in the dept ’||dno);

Exit;

End if;

I:=I+1;

Exit when i=n;

End loop;

Close A;

End;

/

%NOTFOUND:

\*%notfound returns true if there are no rows found while fetching from the cursor.

Ex:create or replace procedure p4(dno in number)

As

Name emp.ename%type;

Cursor A

Is

Select ename from emp

Where deptno=dno;

Begin

Open A;

Loop

Fetch A into name;

If A%NOTFOUND

Then

Exit;

End if;

Dbms\_output.put\_line(name);

End loop;

Close A;

End;

/

Fetching the records from the cursor using cursor for loop:

\*\*For iterable\_variable in cursor\_name

Loop

Stmts;

End loop;

Ex:

create or replace procedure p4(dno in number)

As

Cursor A

Is

Select \* from emp

Where deptno=dno;

Begin

For R in A

Loop

Dbms\_output.put\_line(R.ename);

End loop;

End;

/

1).wasp to obtain empname and deptname of all the employees who are working in a dept which

has atleast three employees in it.

create or replace procedure p5(dno in number)

as

cursor A

is

select ename, dname from emp e,dept d

where e.deptno=d.deptno

and d.deptno in(select dno from emp

group by deptno

having count(\*)>=3 and deptno=dno);

begin

for r in a

loop

dbms\_output.put\_line('NAME :' ||r.ename);

dbms\_output.put\_line('DNAME :' ||r.dname);

end loop;

end;

TRIGGERS:

\*Trigger is an procedure which implicitly called by the software when an event is generated.

Note: A trigger is created for a particular table.

Syntax:

Create [or replace] trigger trigger\_name

Before/After insert/update/delete

on table\_name [for\_each\_row]

declare

[declaration]

Begin

End;

/

\*Triggers can be classified into two

1.table level triggers

2.row level triggers

Table level trigger:

\*table level trigger gets executed only once irrespective to no.of.rows affected by the event.

Note: for table level triggers we should not use for each row.

Row level trigger:

\*row level trigger gets executed for each and every record being affected by the event.

Note:

1.for row level trigger we use foreach row in the trigger.

2.in row level triggers we can use bind variables such as

1.:NEW

2.:OLD

\*bind variable old,new is used to access the values from the rows which is taking part in the event.

RAISE\_APPLICATION\_ERROR:

\*this method is used to raise an error and stop the event which is in the process.

\*it can accept two arguments.

Arg1 is an error number.(it can be anything between -20001 to 21999).

Arg2 is an error message.

Ex:A trigger to generate an error while inserting a record if the salary is lessthan or equal to 1000.

Create or replace trigger t1

Before insert on e1

For each row

Declare

Begin

If :new.sal>1000

Then

Dbms\_output.put\_line(:new.empno ||’ successfully inserted..!!’);

Else

Raise\_application\_error(-20001, ‘ cannot insert..!!’);

End if;

End;

/

Ex:create or replace trigger t2

After update on e1

For each row

Declare

Begin

Insert into e1\_log values(:old.empno,’update’,sysdate,to\_char(sysdate,’hh24’));

End;

/

Assignment :

1.wasp which can accept deptname as an input and provide names of all the employees working in it only if the salary of the employee is greater than 2000.display a suitable message if no data is found in the particular deptno.

2.create a log table for emp table,log table should have an entry for insert update as well as deleted records from the employee table,the attributes of the log tables are

1.the empno,2.the event,3.date of event,4.time(hrs) of event,5.mins of event generated.

3.write a trigger to check for an emp salary before its been modified.if the sal is 500 the new modified salary should not be lessthan 500,if it is lessthan 500 generate an error.

FUNCTIONS:-

Syntax:-

Create [or replace] Function fun\_name(args,........)

Return datatype

As

[declaration]

Begin

Return value/expr/variable;

End;

/

\*Steps to execute a Function:-

-A plsql function can be executed only with the help of sql statements.

1)write a function to count the no of employees in a dept. The input to the function should be dept number.

Create or replace Function f1(dno number)

Return number

As

Count\_emp number;

Begin

Select count(\*)

Into count\_emp

From emp

Where deptno=dno;

Return count\_emp;

End;

/

To execute:

\*Select f1(10) from dual;

\*Select deptno,f1(deptno) from dept;

2)write a function to display names of all the employees working in a dept .the function should accept deptno as input .if the dept doesn’t have any employees display a suitable message.

Create or replace Function f2(dno in number)

Return number

As

C number;

Cursor S1

Is

Select ename from emp

Where deptno=dno;

Begin

Select count(\*) into c

from emp

where deptno=dno;

for R1 in S1

loop

dbms\_output.put\_line(R1.ename);

end loop;

if c=0

then

dbms\_output.put\_line(‘ no employees ‘);

end if;

return dno;

end;

/

3)write a function to obtain manager name of an employee. The function should accept empid .

If the employee doesnot have a manager display suitable message.

Create or replace function f3(eid number)

Return varchar

As

name varchar(20);

Begin

Select ename into name from emp

Where empno in(select mgr from emp where empno=eid);

Return name;

End;

/

EXCEPTION:-

\*In plsql we have named exceptions as well as user defined exceptions.

\*named exceptions are the one which are already defined.

Ex:No\_Data\_Found

\*User defined exceptions:-

-We can create an exception with the following syntax

Exception\_name Exception;

-We can throw the exception with the help of raise keyword.

-The exception which is been thrown can be handled with the help of exception block.

Exception

When exception1 then stmts;

When exception2 then stmts;

When others then stmts;

End;

Ex:

Create or replace function f4(eid number)

Return varchar

As

Name varchar(20);

Cnt number(2);

Excep1 EXCEPTION;

Begin

Select ename,count(\*) into name ,cnt

From emp where empno in (select mgr from emp where empno=eid)

Group by ename;

If cnt=0

Then

Raise excep1;

End if;

Return name;

Exception

When excep1 then

Dbms\_output.put\_line(‘no manager..’);

When no\_data\_found then

Dbms\_output.put\_line(‘doesnt report to a manager..’);

When others then

Dbms\_output.put\_line(‘some prob....!!’);

Return name;

End;

/

Ex: (for no\_data\_found exception)

Create or replace function f5(eid number)

Return varchar

As

Name varchar(20);

Begin

Select ename into name from emp

Where empno in(select mgr from emp

Where empno=eid);

Return name;

Exception

When no\_data\_found then

Dbms\_output.put\_line(‘no reporting manager’);

Return name;

End;

/

To Execute:-select f5(7839) from dual;

Because 7839 doesnot have mgr.

\*no\_data\_found arises only when data is not present to be returned by sql query to store it in plsql variables.i.e.INTO clause