Labsheet-9

(Prof.R Gururaj)

PL-SQL Stored Procedures (continued)

We started working on table BOOK (already created in the prev session)

With structure (bid int pk, title vc(10), price int)

Inserted following tuples

<101, ‘OPERATIONS’, 300>;

<107, ‘DATABASES’ , 370>;

<128, ‘NETWORKS’, 175>

// Concept of IN / OUT Parameters in procedures

IN mode:

- Default mode
- Passes a value to the subprogram.
- Formal parameter acts like a constant: When the subprogram begins, its value is that of either its actual parameter or default value, and the subprogram cannot change this value.
- Actual parameter can be a constant, initialized variable, literal, or expression.

`OUT` mode:

- Must be specified.
- Returns a value to the invoker.
- Formal parameter is initialized to the default value of its type. The default value of the type is `NULL` except for a record type with a non-`NULL` default value.
- When the subprogram begins, the formal parameter has its initial value regardless of the value of its actual parameter. Oracle recommends that the subprogram assign a value to the formal parameter.

`IN OUT` mode:

- Must be specified.
- Passes an initial value to the subprogram and returns an updated value to the invoker.

// demonstrating IN and OUT parameters

Example-1:

```
SQL> create or replace procedure proc1(arg1 in number, arg2 out number) as
  2  n number;
  3  begin
  4  n:=arg1+10;
  5  arg2:=60;
  6  end;
  7  /

Procedure created.

SQL> declare
  2  m number;
  3  p number;
  4  begin
  5  m:=20;
  6  p:=5;
  7  dbms_output.put_line('    :'||m ||'  :'||p);
  8  proc1(m,p);
  9  dbms_output.put_line('    :'||m ||'  :'||p);
 10  end;
 11  /
:20  :5
:20  :60
```

PL/SQL procedure successfully completed.

:20 :5

:20 :60

Example:2

// Procedure to insert a new book record into Book table.

```
SQL> create or replace procedure proc6(id in number, name in varchar2, pr in nu
mber) as
 begin
insert into book values(id, name,pr);
dbms_output.put_line(' Book : '||id||  ':  name : '||name||' :  added to table');
 end;
/

Procedure created.

SQL> exec proc6(144,'Economics',570);
```

Example 3

// Procedure to insert BOOK record into BOOK table, if the total number of records after the new insertion is even it is ok, or else print an error message.

```
SQL> create or replace procedure proc7(id in number, name in varchar2, pr in
number) as
  2  n number:=0;
  3  begin
  4  insert into book values(id, name,pr);
  5  select count(*) into n from book;
  6  select MOD(n,2) into n from dual;
  7  if n<>0 then
  8  dbms_output.put_line(' ODD Number of tuples in Book table  ');
  9  end if;
 10  end;
 11  /Procedure created.
```

See what is the effect of executing this proc7 to insert
few more tuples as above

Exercise: write a procedure that takes bookid as argument and prints the book id : title : is   costly/cheap costly if price is >300 else cheap
Example if pass on book id 128
The book 128 with title NETWORKS is Cheap

IF THEN ELSE ladder in PLSQL

```
IF condition1 THEN

   {...statements to execute when condition1 is TRUE...}


ELSIF condition2 THEN

   {...statements to execute when condition1 is FALSE and condition2 is
TRUE...}


ELSE

   {...statements to execute when both condition1 and condition2 are
FALSE...}


END IF;
```

Example 4:
Assume that we have Two tables EMP (eid, ename, sal, dno) and DEPT(dnum, dname, total_emps)

// Procedure to insert new employee record into EMP table, and update DEPT table's field- *total_emps accordingly*

```
SQL> create or replace procedure proc8(id in number, name in varchar2, sal in nu
mber, dep in number) as
  2  n number;
  3  begin
  4  insert into emp values(id, name,sal,dep);
  5  select count(*) into n from emp where dno=dep;
  6  update dept set total_emps=n where dnum=dep;
  7  dbms_output.put_line(' Insert and update done for eid: '|| id ||'  ');
```

```
end;
/

Procedure created.
```

// take this example 4 as homework.

Example 5

Write a procedure to get price and title of a book if bid is given as in argument.

```
SQL> create or replace procedure proc22(id in number) as
   pr number;
   name  varchar(10);
   begin
   select title, price into pr,name from book where bid=id;
   dbms_output.put_line(' For the book with bid: '|| id ||' Title is:  '||name||'
price is : '||pr);
   end;
   /
```

Looping in PLSQL

Example:6

```
SQL> create or replace procedure proc8 as
n number:=10;
begin
loop
dbms_output.put_line(' Value of n is: '||n||'  ');
n:=n+10;
exit when n>100;
end loop;
end;
/

Procedure created.
SQL> exec proc8;
Value of n is: 10
Value of n is: 20
Value of n is: 30
Value of n is: 40
Value of n is: 50
Value of n is: 60
Value of n is: 70
Value of n is: 80
Value of n is: 90
Value of n is: 100

PL/SQL procedure successfully completed.
```

Pl-SQL Functions

How Functions are different from Stored Procedures

- **Functions**: these subprograms return a single value, mainly used to compute and return a value.
- **Procedures**: these subprograms do not return a value directly, mainly used to perform an action. Or executing a set of data manipulation operations in one go at DB server.

A PL/SQL function is same as a procedure except that it returns a value. Therefore, all the discussions of the previous chapter are true for functions too.

Example:7

```
SQL> create function Func1(arg in number) return number is
  2  var number;
  3  begin
  4  select price into var from book where bid=arg;
  5  return var;
```

```
end;
/
```

Function created.

// write anonymous code calling the function.

```
SQL> declare
n number;
begin
n:=Func1(107);
dbms_output.put_line('  '||' price is   : '||n);
end;
/
```

Example 8: write a function to take two numbers and return the sum

```
create or replace function Func2(a in number, b in number) return number is
n number;
begin
n:=a+b;
return n;
end;
/
```

//anonymous code to call the function Func2

```
declare
s number;
begin
s:=Func2(20,10);
dbms_output.put_line('  '||' the sum  is   : '||s);
end;
/
the sum  is   : 30
```