

DBS 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.

// we work on problems demonstrating IN and OUT parameters

Example:2

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

```
SQL> create or replace procedure addbook(id in number, name in varchar2, pr in
number) 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.

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

Example 5

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

Looping in PLSQL

Example:6

SQL> create or replace procedure proc8 as

```

2  n number:=10;
3  begin
4  loop
5  dbms_output.put_line(' Value of n is: ' || n || ' ');
6  n:=n+10;
7  exit when n>100;
8  end loop;
9  end;
10 /

```

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(bookid in number) return number is
  2  prc number;
  3  begin
  4  select price into prc from book where bid=bookid;
```

```
5 return prc;  
6 end;  
7 /
```

Function created.

// write anonymous code calling the function.

```
SQL> declare  
2 n number;  
3 begin  
4 n:=Func1(107);  
5 dbms_output.put_line(' || ' price is : ' || n);  
6 end;  
7 /
```

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

Some more PL-SQL Built in Functions

11. initcap(char): *initcap('hello') → Hello*
12. lower(string): *lower('HELLO') → hello*
13. upper(string): *upper('hello') → HELLO*

14. `translate(char, from string, to string):` ✓

15. `abs(n):` ✓ $\text{abs}(-13.46) = 13.46$

16. `ceil(n):` $\text{ceil}(-13.46) = 13$ $\text{ceil}(13.46) = 14$

17. `cos(n):`

18. `exp(n):`

19. `floor(n):`

20. `mod(m, n):` — $22/12 = 10$

21. `power(x, y):` $\text{power}(2, 5) = 32$

22. `round(x [, y]):` $\text{round}(2.7686766, 1) = 2.8$

23. `sign(n):` — $\text{sig}(122) = 1$
 $\text{sign}(-122) = -1$

select time (124.378) for deal;
will give 124