Cursors with Parameters

Syntax:

```
CURSOR cursor_name
  [(parameter_name datatype, ...)]
IS
  select_statement;
```

- Pass parameter values to a cursor when the cursor is opened and the query is executed.
- Open an explicit cursor several times with a different active set each time.

```
OPEN  cursor_name(parameter_value,......) ;
```

Cursors with Parameters

You can pass parameters to a cursor. This means that you can open and close an explicit cursor several times in a block, returning a different active set on each occasion. For each execution, the previous cursor is closed and reopened with a new set of parameters.

Each formal parameter in the cursor declaration must have a corresponding actual parameter

in the OPEN statement. Parameter data types are the same as those for scalar variables, but you do not give them sizes. The parameter names are for reference in the query expression of the cursor. In the syntax:

cursor_name   Is a PL/SQL identifier for the declared cursor

parameter_name Is the name of a parameter datatype   Is the

scalar data type of the parameter select_statement   Is a SELECT

statement without the INTO clause

Oracle Database: PL/SQL Fundamentals

The parameter notation does not offer greater functionality; it simply allows you to specify input values easily and clearly. This is particularly useful when the same cursor is referenced repeatedly.

Cursors with Parameters

```
DECLARE
   CURSOR   c_emp_cursor (deptno NUMBER) IS
    SELECT  employee_id, last_name
    FROM    employees
    WHERE   department_id = deptno;
   . . .
BEGIN
   OPEN c_emp_cursor (10);
   . . .
   CLOSE c_emp_cursor;
   OPEN c_emp_cursor (20);
   . . .
```

```
anonymous block completed
200 Whalen
201 Hartstein
202 Fay
```

Cursors with Parameters (continued)

Parameter data types are the same as those for scalar variables, but you do not give them sizes. The parameter names are for reference in the cursor's query. In the following example, a cursor is declared and is defined with one parameter:

```
DECLARE
CURSOR c_emp_cursor(deptno NUMBER) IS SELECT ...
```

The following statements open the cursor and return different active sets:

```
OPEN c_emp_cursor(10);
OPEN c_emp_cursor(20);
```

You can pass parameters to the cursor that is used in a cursor FOR loop:

```
DECLARE
CURSOR c_emp_cursor(p_deptno NUMBER, p_job
VARCHAR2)IS
  SELECT ...
BEGIN
  FOR emp_record IN c_emp_cursor(10, 'Sales') LOOP ...
```