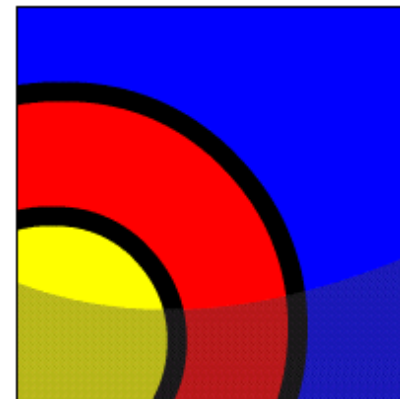


Cursors with Parameters

What Will I Learn?

In this lesson, you will learn to:

- List the benefits of using parameters with cursors
- Create PL/SQL code to declare and use manipulate a cursor with a parameter





Why Learn It?

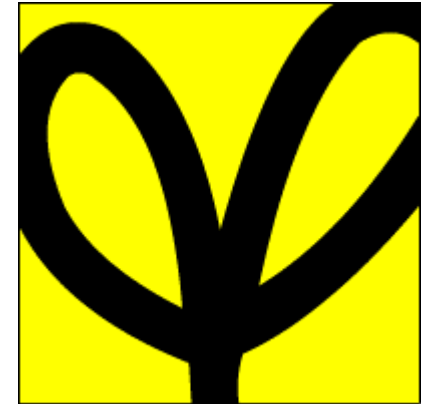
Imagine a program which declares a cursor to fetch and process all the employees in a given department. The department is chosen by the user at runtime.

How would we declare our cursor? We could try:

```
DECLARE  
    cursor country_cursor IS  
        SELECT * FROM wf_countries  
        WHERE region_id = ??? ;
```

Hmm... obviously not.

There are several regions. Do we need to declare several cursors, one for each region, each with a different value in the WHERE clause? No. We can declare just one cursor to handle all regions by using parameters.



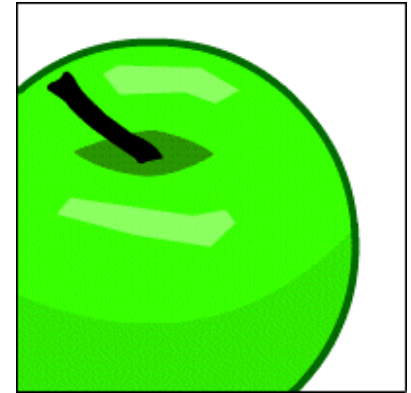
Tell Me/Show Me

Cursors with Parameters

A parameter is a variable whose name is used in a cursor declaration. When the cursor is opened, the parameter value is passed to the Oracle server, which uses it to decide which rows to retrieve into the active set of the cursor.

This means that you can open and close an explicit cursor several times in a block, or in different executions of the same block, returning a different active set on each occasion.

Consider the example where you pass any `region_id` to a cursor and it returns the names of countries in that region. The next slide shows how.





Tell Me/Show Me

Cursors with Parameters: Example

```
DECLARE
  CURSOR c_country (p_region_id NUMBER) IS
    SELECT country_id, country_name
      FROM wf_countries
     WHERE region_id = p_region_id;
  v_country_record    c_country%ROWTYPE;
BEGIN
  OPEN c_country (5);
  LOOP
    FETCH c_country INTO v_country_record;
    EXIT WHEN c_country%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_country_record.country_id
                        || ' ' || v_country_record.country_name);
  END LOOP;
  CLOSE c_country;
END;
```

Change to whichever region is required.

Tell Me/Show Me

Defining Cursors with Parameters

Each parameter named in the cursor declaration must have a corresponding value 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 used in the `WHERE` clause of the cursor `SELECT` statement.

Syntax:

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



Tell Me/Show Me

Defining Cursors with Parameters (continued)

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

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



Tell Me/Show Me

Opening Cursors with Parameters

The following is the syntax for opening a cursor with parameters:

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


Tell Me/Show Me

Cursors with Parameters

You pass parameter values to a cursor when the cursor is opened. Therefore you can open a single explicit cursor several times and fetch a different active set each time. In the following example, a cursor is opened several times.

```
DECLARE
  CURSOR c_country (p_region_id NUMBER) IS
    SELECT country_id, country_name
      FROM wf_countries
     WHERE region_id = p_region_id;
  v_country_record  c_country%ROWTYPE;
BEGIN
  OPEN c_country (5);

  ...
  CLOSE c_country;
  OPEN c_country (145);
  ...
```

Open the cursor and
return different active sets.



Tell Me/Show Me



Tell Me/Show Me

Another Example of a Cursor with a Parameter

```
DECLARE
  v_deptid      employees.department_id%TYPE;
  CURSOR empcur (p_deptid NUMBER) IS
    SELECT employee_id, salary
      FROM employees
     WHERE department_id = p_deptid;
  v_emp_rec empcur%ROWTYPE;
BEGIN
  SELECT MAX(department_id) INTO v_deptid
    FROM employees;
  OPEN empcur(v_deptid);
  LOOP
    FETCH empcur INTO v_emp_rec;
    EXIT WHEN empcur%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_emp_rec.employee_id
                        || ' ' || v_emp_rec.salary);
  END LOOP;
  CLOSE empcur;
END;
```



Tell Me/Show Me

Cursor FOR

We can use a

```
DECLARE
  CURSOR
  SEL
  F
  W
BEGIN
  FOR v
  ...
  END L
END;
```





Tell Me/Show Me

Cursors with Multiple Parameters

In the following example, a cursor is declared and is called with two parameters:

```
DECLARE
    CURSOR    countrycursor2 (p_region_id NUMBER,
                               p_population NUMBER) IS
        SELECT  country_id, country_name, population
        FROM    wf_countries
        WHERE    region_id = p_region_id
        OR       population > p_population;
BEGIN
    FOR v_country_record IN countrycursor2(145,10000000)
    LOOP
        DBMS_OUTPUT.PUT_LINE(v_country_record.country_id || ' '
                               || v_country_record.
country_name || ' ' || v_country_record.population);
    END LOOP;
END;
```



Tell Me/Show Me

Cursors with Multiple Parameters: Another Example

This cursor fetches all IT Programmers who earn more than \$10000.

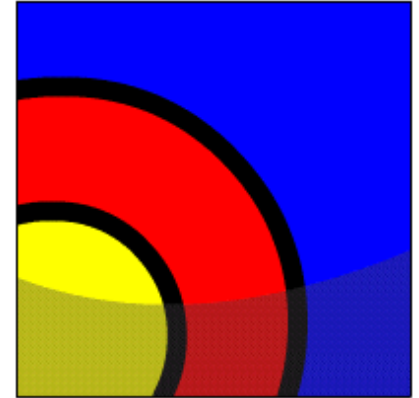
```
DECLARE
    CURSOR    emp_cursor3 (p_job VARCHAR2,
                           p_salary NUMBER) IS
        SELECT  employee_id, last_name
          FROM    employees
         WHERE   job_id   = p_job
         AND     salary > p_salary;
BEGIN
    FOR v_emp_record IN emp_cursor3('IT_PROG', 10000)
    LOOP
        DBMS_OUTPUT.PUT_LINE(v_emp_record.employee_id || ' '
                               || v_emp_record.last_name);
    END LOOP;
END;
```



Summary

In this lesson, you learned to:

- List the benefits of using parameters with cursors
- Create PL/SQL code to declare and use manipulate a cursor with a parameter





Try It/Solve It

The exercises in this lesson cover the following topic:

- Declaring and using cursors with parameters

