



Database Programming with PL/SQL

8-2

Using Parameters in Procedures



Objectives

This lesson covers the following objectives:

- Describe how parameters contribute to a procedure
- Define a parameter
- Create a procedure using a parameter
- Invoke a procedure that has parameters
- Differentiate between formal and actual parameters

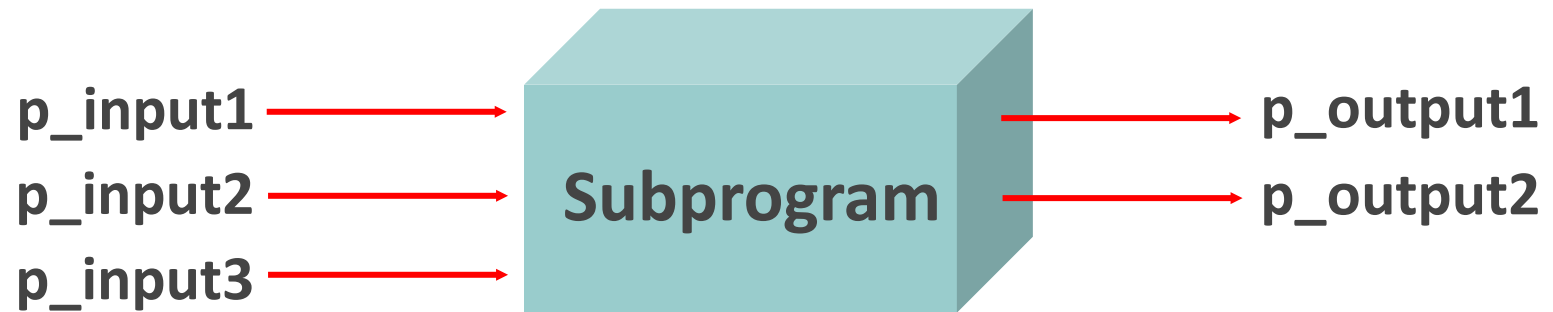
Purpose

- Much time can be spent creating a procedure.
- It is important to create the procedure in a flexible way so that it can be used, potentially, for more than one purpose or more than one piece of data.
- To make procedures more flexible, it is important that varying data is either calculated or passed into a procedure by using input parameters.
- Calculated results can be returned to the caller of a procedure by using parameters.

What are Parameters?

- Parameters pass or communicate data between the caller and the subprogram.
- You can think of parameters as a special form of a variable, whose input values are initialized by the calling environment when the subprogram is called, and whose output values are returned to the calling environment when the subprogram returns control to the caller.
- By convention, parameters are often named with a “p_” prefix.

What are Parameters?



What Are Parameters?

- Consider the following example where a math teacher needs to change a student's grade from a C to a B in the student administration system.



Student id is 1023

1023

The math class id is 543

543

The new grade is B

B

Calling
environment

- In this example, the calling system is passing values for student id, class id, and grade to a subprogram.
- Do you need to know the old (before) value for the grade?
- Why or why not?

What Are Parameters?

- The `change_grade` procedure accepts three parameters: `p_student_id`, `p_class_id`, and `p_grade`.
- These parameters act like local variables in the `change_grade` procedure.



Student id is 1023

1023

The math class id is 543

543

The new grade is B

B

Calling
environment

```
PROCEDURE change_grade (p_student_id IN NUMBER,  
p_class_id IN NUMBER, p_grade IN VARCHAR2) IS  
BEGIN  
...  
    UPDATE grade_table SET grade = p_grade  
        WHERE student_id = p_student_id AND class_id = p_class_id;  
...  
END;
```


What Are Arguments?

- Parameters are commonly referred to as arguments.
- However, arguments are more appropriately thought of as the actual values assigned to the parameter variables when the subprogram is called at runtime.

Student id is 1023

1023

The math class id is 543

543

The new grade is B

B

What Are Arguments?

- Even though parameters are a kind of variable, `IN` parameters are treated as constants within the subprogram and cannot be changed by the subprogram.
- In the previous example, `1023` is an argument passed in to the `p_student_id` parameter.

Student id is 1023

1023

The math class id is 543

543

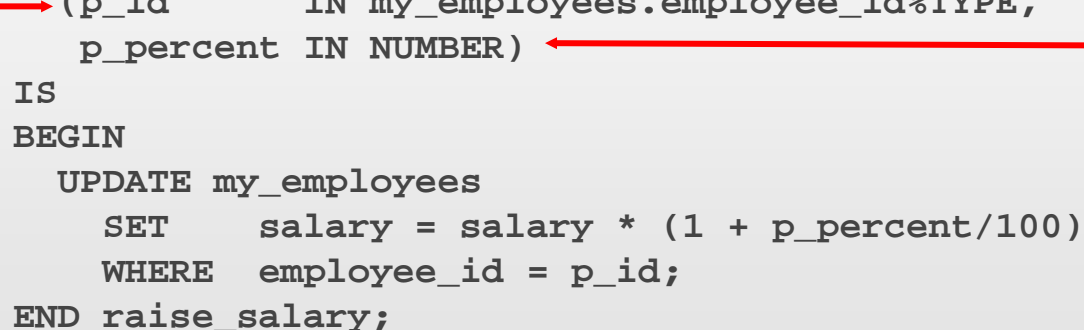
The new grade is B

B

Creating Procedures with Parameters

- The example shows a procedure with two parameters. Running this first statement creates the `raise_salary` procedure in the database.
- The second example executes the procedure, passing the arguments 176 and 10 to the two parameters.

```
CREATE OR REPLACE PROCEDURE raise_salary
(p_id      IN my_employees.employee_id%TYPE,
 p_percent IN NUMBER)
IS
BEGIN
    UPDATE my_employees
        SET    salary = salary * (1 + p_percent/100)
        WHERE employee_id = p_id;
END raise_salary;
```



```
BEGIN raise_salary(176, 10); END;
```

Invoking Procedures with Parameters

- To invoke a procedure from Oracle Application Express, create an anonymous block and use a direct call inside the executable section of the block.
- Where you want to call the new procedure, enter the procedure name and parameter values (arguments).
- For example:

```
BEGIN  
    raise_salary (176, 10);  
END;
```

- You must enter the arguments in the same order as they are declared in the procedure.

Invoking Procedures with Parameters

- To invoke a procedure from another procedure, use a direct call inside an executable section of the block.
- At the location of calling the new procedure, enter the procedure name and parameter arguments.

```
CREATE OR REPLACE PROCEDURE process_employees
IS
  CURSOR emp_cursor IS
    SELECT employee_id
      FROM my_employees;
BEGIN
  FOR v_emp_rec IN emp_cursor
  LOOP
    raise_salary(v_emp_rec.employee_id, 10);
  END LOOP;
END process_employees;
```

Types of Parameters

- There are two types of parameters: Formal and Actual.
- A parameter-name declared in the procedure heading is called a formal parameter.
- The corresponding parameter-name (or value) in the calling environment is called an actual parameter.



Types of Parameters

In the following example, can you identify which parameter is the formal parameter and which parameter is the actual parameter?

```
CREATE OR REPLACE PROCEDURE fetch_emp
  (p_emp_id IN employees.employee_id%TYPE) IS
  ...
END;

/* Now call the procedure from an anonymous block or
subprogram */

BEGIN
  ...
  fetch_emp(v_emp_id);
  ...
END;
```

Formal Parameters

- Formal parameters are variables that are declared in the parameter list of a subprogram specification.
- In the following example, in the procedure `raise_sal`, the identifiers `p_id` and `p_sal` represent formal parameters.

```
CREATE PROCEDURE raise_sal(p_id IN NUMBER, p_sal IN
    NUMBER) IS
BEGIN
    ...
END raise_sal;
```

- Notice that the formal parameter data types do not have sizes.
- For instance `p_sal` is `NUMBER`, not `NUMBER(6, 2)`.

Actual Parameters

- Actual parameters can be literal values, variables, or expressions that are sent to the parameter list of a called subprogram.
- In the following example, a call is made to `raise_sal`, where the `a_emp_id` variable is the actual parameter for the `p_id` formal parameter, and 100 is the argument (the actual passed value).

```
a_emp_id := 100;  
raise_sal(a_emp_id, 2000);
```

Actual Parameters

Actual parameters:

- Are associated with formal parameters during the subprogram call
- Can also be expressions, as in the following example:

```
raise_sal(a_emp_id, v_raise + 100);
```



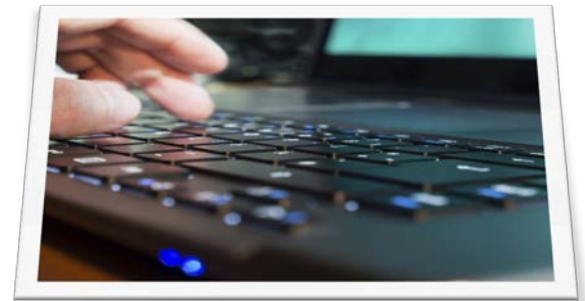
Formal and Actual Parameters

- The formal and actual parameters should be of compatible data types.
- If necessary, before assigning the value, PL/SQL converts the data type of the actual parameter value to that of the formal parameter.



Formal and Actual Parameters

- For instance, you can pass in a salary of '1000.00' in single quotes, so it is coming in as the *letter* 1 and the *letters* zero, etc., which get converted into the *number* one thousand.
- This is *slower* and should be *avoided* if possible.
- You can find out the data types that are expected by using the command `DESCRIBE proc_name`.



Terminology

Key terms used in this lesson included:

- Actual parameter
- Argument
- Formal parameter
- Parameters

Summary

In this lesson, you should have learned how to:

- Describe how parameters contribute to a procedure
- Define a parameter
- Create a procedure using a parameter
- Invoke a procedure that has parameters
- Differentiate between formal and actual parameters

