Database Programming

5.Stored Procedures

Sagara Samarawickrama

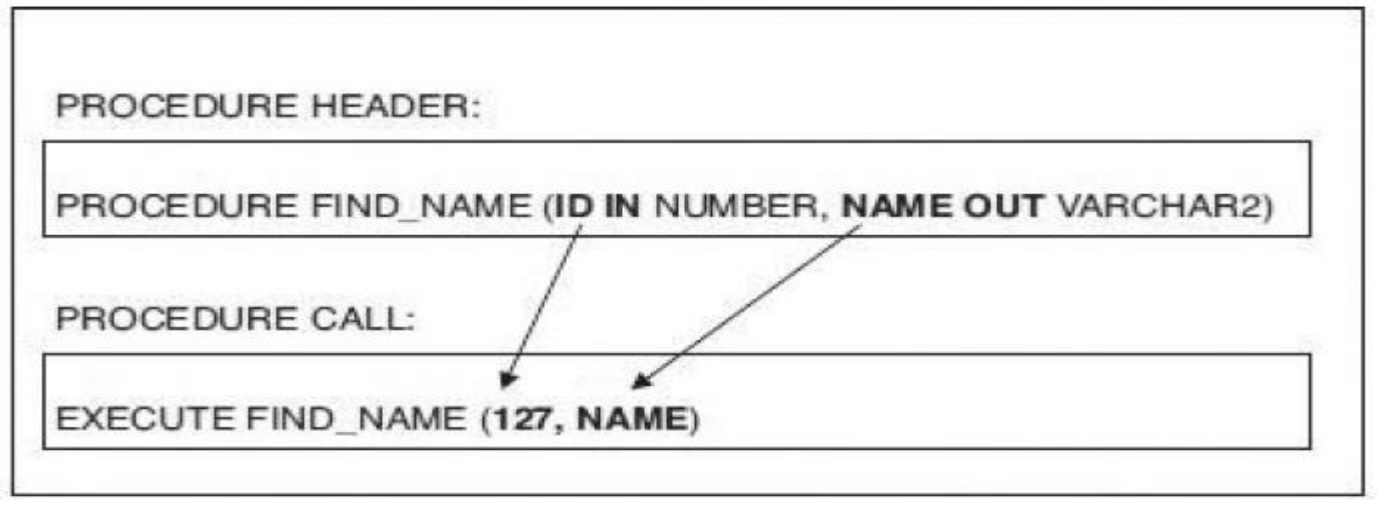
CSD 4204 - CPCM GP 1,2 & 3

5.Stored Procedures - ii

Passing Parameters IN and OUT of Procedures

All the Parameters are the means to pass values from the calling environment to the server, and vice versa. These values are processed or returned via the execution of the procedure. There are three parameter modes: IN, OUT, and IN OUT.

Following figure illustrates the relationship between the parameters when they are in the procedure header versus when the procedure is executed.



5.Stored Procedures - ii

Formal and Actual Parameters

Formal parameters are the names specified within parentheses as part of the header of a module. Actual parameters are the value expressions specified within parentheses as a parameter list when a call is made to the module. Following table explains the three types of parameters.

Mode	Description	Usage
IN	Passes a value into the program Constants, literals, expressions Cannot be changed within program Default mode	Read-only value
OUT	Passes a value back from the program Cannot assign default values Must be a variable A value is assigned only if the program is successful	Write-only value
IN OUT	Both passes values in and sends values back	Must be a variable

5.Stored Procedures - ii

Passing of Constraints (Data Types) with Parameter Values

Formal parameters do not require constraints on the data type. For example, instead of specifying a constraint such as VARCHAR2(60), you can just issue VARCHAR2 against the parameter name in the formal parameter list. The constraint is passed with the value when a call is made

Matching Actual and Formal Parameters

Two methods can be used to match actual and formal parameters: positional notation and named notation. Positional notation is simply association by position; that is, the order of the parameters used when executing the procedure matches the order in the procedure's header exactly. Named notation is explicit association using the symbol =>. It has the following syntax:

formal_parameter_name => argument_value

5.Stored Procedures - ii

In named notation, the order does not matter. If you mix notation, however, you should list the positional notation before the named notation.

```
CREATE OR REPLACE PROCEDURE find_sname
  (i_student_id IN NUMBER,
   o_first_name OUT VARCHAR2,
   o_last_name OUT VARCHAR2
   )
AS
BEGIN
  SELECT first_name, last_name
    INTO o_first_name, o_last_name
    FROM student
   WHERE student_id = i_student_id;
EXCEPTION
  WHEN OTHERS
  THEN
   DBMS_OUTPUT.PUT_LINE('Error in finding student_id:
    '||i_student_id);
END find_sname;
```

5.Stored Procedures - ii

This procedure takes in a student_id via the parameter named student_id. It passes out the parameters o_first_name and o_last_name. The procedure is a simple SELECT statement that retrieves the first_name and last_name from the STUDENT table when the student_id matches the value of i_student_id, which is the only IN parameter that exists in the procedure. To call the procedure, a value must be passed in for the i_student_id parameter.

```
DECLARE
  v_local_first_name student.first_name%TYPE;
  v_local_last_name student.last_name%TYPE;
BEGIN
  find_sname
    (145, v_local_first_name, v_local_last_name);
  DBMS_OUTPUT.PUT_LINE
    ('Student 145 is: '||v_local_first_name||
     ' '|| v_local_last_name||'.'
    );
END;
```

5.Stored Procedures - ii

When calling the procedure find_sname, a valid student_id should be passed in for the i_student_id. If it is not a valid student_id, an exception will be raised.Two variables must also be listed when calling the procedure. These variables,v_local_first_name and v_local_last_name, are used to hold the values of the parameters that are being passed out. After the procedure has been executed, the local variables will have values and can then be displayed with a DBMS_OUTPUT.PUT_LINE statement.