Les09b PLSQL More on IN OUT INOUT

(from Donald Burleson's notes)

There are 3 modes called IN, OUT and INOUT which is a variable that can be used in a procedure

IN

Can be read and used but cannot be changed in the procedure or function. They then can be considered constants.

The following is an example, however there is an error in the coding. Run it to see the error.

```
create or replace procedure example_defaults
    (n_1 in number := 5,
     n_2 in number := 6,
     n_3 in number := 7)
  as
  begin
    n_1 := n_2 + n_3;
  end;
  /
```

You cannot change the value of n_1 as it was called IN. Assigning it values (n_1 := n_2 + n_3;) is not allowed.

OUT

A variable that is used in the OUT mode is passed back to the calling procedure. It has no value until the block assigns a value to it. At the end of the procedure the value is copied back to the calling procedures variable. An uncaught exception results in nothing sent/copied back. The following generates a compiler error

```
create or replace procedure example_defaults
    (n_1 in number := 5,
     n_2 in number := 6,
     n_3 OUT number := 7)
  as
  begin
    null;
  end;
  /
```

INOUT

This mode has both IN and OUT characteristics. The value passed in can be read within the procedure, the value can be changed by the procedure and the value can be copied to the calling procedure when this procedure finishes.

EXAMPLE:

```
CREATE OR REPLACE PROCEDURE GET_EMP_NAME (
        i_empno IN employees.employee_id%TYPE,  -- input variable
        o_ename OUT employees.last_name%TYPE) – output variable
IS
CURSOR c_ename (p_empno employees.employee_id%TYPE)
IS
        SELECT last_name
        FROM employees
        WHERE employee_id = p_empno;

BEGIN
        OPEN c_ename (i_empno);

        FETCH c_ename INTO o_ename;
        CLOSE c_ename;


END get_emp_name;
/
set serveroutput on

DECLARE
var_name employees.last_name%TYPE;
BEGIN
        get_emp_name (40, var_name);
        DBMS_OUTPUT.PUT_LINE (var_name);
END;
```

RESULT:
Whiteduck

PL/SQL procedure successfully completed.

ANOTHER EXAMPLE:

```
DECLARE
        hello varchar2(40) := 'From 1960s C Programming -- hello world';
        procedure myproc (p_val in out varchar2) is
BEGIN
    dbms_output.put_line('p_val was ' || p_val);
    p_val := 'something else';
END;

BEGIN
   myproc(hello);
   dbms_output.put_line('l_val is now ' || hello);
END;
/
```

RESULT:

```
p_val was From 1960s C Programming -- hello world
l_val is now something else


PL/SQL procedure successfully completed.
```

OTHER SOURCES TO READ:
PL/SQL Tutorial - PL/SQL - passing parameters in procedures and functions. (plsql-tutorial.com)

http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96624/08_subs.htm#895