

PL/SQL

Các tính năng chính của PL/SQL

- Khởi lệnh PL/SQL
- PL/SQL Input và Output
- Biến và hằng số trong PL/SQL
- Cấu trúc điều khiển trong PL/SQL
- Quản lý lỗi trong PL/SQL
- Trừu tượng dữ liệu PL/SQL (data abstraction)
- **Chương trình con PL/SQL (Subprogram)**
- PL/SQL Packages

Tổng quan về chương trình con

- Một chương trình con (PL/SQL subprogram) chính là một khối lệnh PL/SQL được đặt tên và được gọi với một tập các đối số.
- Một chương trình con có thể là một thủ tục (procedure) hoặc là một hàm (function).
- Thông thường, procedure được sử dụng để thực hiện một tác vụ nào đó còn function được sử dụng để tính toán và trả về kết quả.
- Chương trình con có thể được tạo ở mức schema, trong một package, hay trong một khối lệnh PL/SQL.

Chương trình con trong một khối lệnh PL/SQL

```
DECLARE
```

```
    in_string VARCHAR2(100) := 'Test string';
```

```
    PROCEDURE double ( original VARCHAR2) AS
```

```
    BEGIN
```

```
        DBMS_OUTPUT.PUT_LINE (original || original);
```

```
    END;
```

```
BEGIN
```

```
    double (in_string);
```

```
END;
```

Chương trình con ở mức schema

(Standalone subprogram)

```
CREATE OR REPLACE PROCEDURE remove_emp (employee_id NUMBER) AS
    tot_emps NUMBER;
BEGIN
    DELETE FROM employees
    WHERE employees.employee_id = employee_id;
    --tot_emps := tot_emps - 1;
END;
```

```
CREATE OR REPLACE FUNCTION get_bal(acc_no IN NUMBER)
RETURN NUMBER IS
    acc_bal NUMBER(11,2); --declare acc_bal
BEGIN
    SELECT order_total INTO acc_bal
    FROM orders
    WHERE customer_id = acc_no;
    DBMS_OUTPUT.PUT_LINE (TO_CHAR(acc_no));
    RETURN(acc_bal);
END;
```

Các phần trong một chương trình con

```
PROCEDURE double (original IN VARCHAR2, new_string OUT VARCHAR2)
IS
    -- Declarative part of procedure (optional) goes here

BEGIN
    -- Executable part of procedure begins
    new_string := original || ' + ' || original;
    -- Executable part of procedure ends

    -- Exception-handling part of procedure (optional) begins
    EXCEPTION
    WHEN VALUE_ERROR THEN
        DBMS_OUTPUT.PUT_LINE('Output buffer not long enough.');
```

END;

Các phần trong một chương trình con

- Một thủ tục (procedure) và một hàm (function) có cùng cấu trúc, ngoại trừ:
 - Phần đầu của function phải chứa mệnh đề RETURN (return clause) xác định kiểu dữ liệu trả về. Còn procedure không chứa mệnh đề RETURN này.
 - Một function phải chứa ít nhất một câu lệnh RETURN (return statement) trong phần thực thi. Trong procedure, câu lệnh RETURN không bắt buộc.

Chương trình con

Ví dụ

```
PROCEDURE raise_salary ( emp_id NUMBER, amount NUMBER)
IS
BEGIN
    IF emp_id IS NULL THEN
        RETURN; ← return statement
    END IF;
    UPDATE employees SET salary = salary + amount WHERE employee_id = emp_id;
END raise_salary;
```

```
FUNCTION compute_bonus (emp_id NUMBER, bonus NUMBER)
IS
    emp_sal NUMBER;
BEGIN
    SELECT salary INTO emp_sal FROM employees WHERE employee_id = emp_id;
    RETURN emp_sal + bonus; ← return statement
END compute_bonus;
```

RETURN NUMBER
↑
return clause

Xác định các loại đối số trong chương trình con

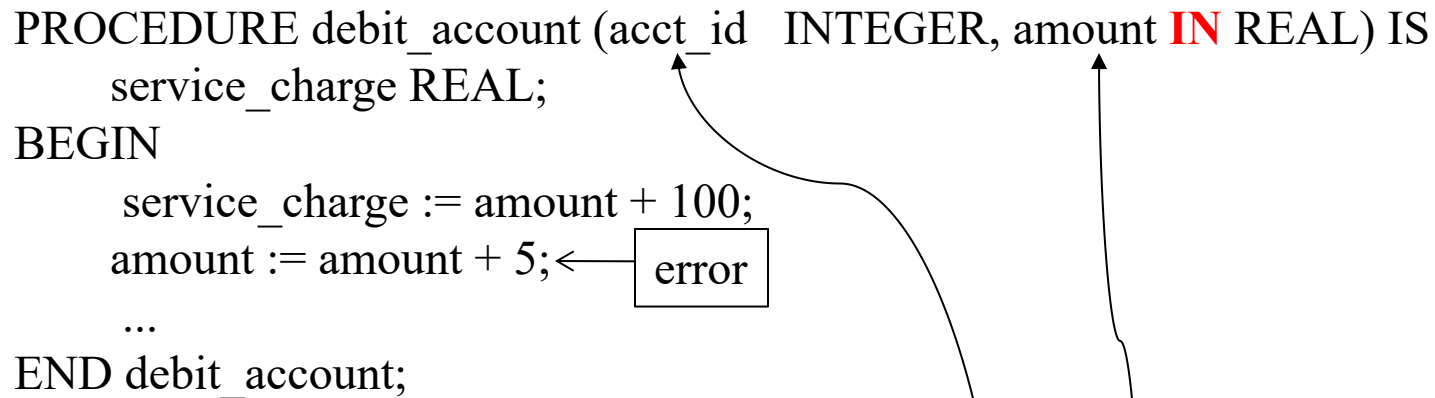
- Có 3 phương thức truyền đối số là IN (mặc định), OUT, và IN OUT.
- Tránh việc sử dụng OUT và IN OUT với function.

Đôi số trong chương trình con sử dụng IN

- Đôi số IN cho phép truyền giá trị tới chương trình con.
- Trong một chương trình con, đôi số IN hoạt động giống như một hằng số. Nó không được gán giá trị.
- Đôi số IN có thể được khởi tạo một giá trị mặc định.

Đổi số trong chương trình con sử dụng IN

```
PROCEDURE debit_account (acct_id INTEGER, amount IN REAL) IS
    service_charge REAL;
BEGIN
    service_charge := amount + 100;
    amount := amount + 5; ← error
    ...
END debit_account;
```



```
BEGIN
    debit_account (100034, 20);
END;
```

Đổi số trong chương trình con sử dụng IN-truyền đổi số mặc định

PROCEDURE **Get_emp_names** (Dept_num IN NUMBER **DEFAULT** 20) IS ...

Gọi chương trình con với đổi số mặc định:

Get_emp_names (); -- truyền đổi số mặc định là 20

Get_emp_names(47); -- truyền đổi số là 47

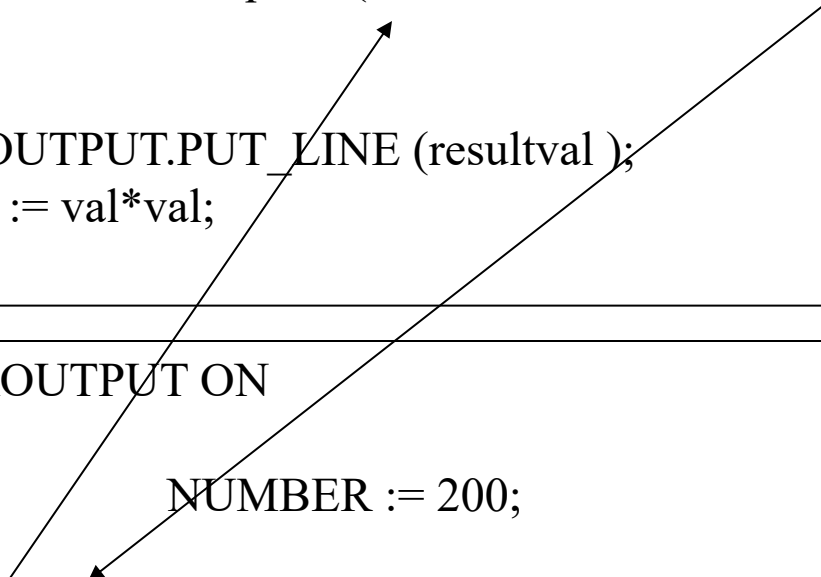
Đổi số trong chương trình con sử dụng OUT

- Đổi số OUT trả về giá trị tới môi trường gọi nó.
- Trong chương trình con, đổi số OUT hoạt động như là một biến (variable).
- Có thể thay đổi giá trị của nó và sử dụng giá trị này sau khi gán.

Đổi số trong chương trình con sử dụng OUT

```
CREATE PROCEDURE square (val IN NUMBER, resultval OUT NUMBER)
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE (resultval);
    resultval := val*val;
END;
```

```
SET SERVEROUTPUT ON
DECLARE
    v_kq NUMBER := 200;
BEGIN
    square (5, v_kq);    --gọi chương trình con, đổi số là OUT trả kết quả về
                        -- môi trường gọi nó thông qua biến v_kq
    DBMS_OUTPUT.PUT_LINE (v_kq);
END;
```



Đổi số trong chương trình con sử dụng OUT

```
SET SERVEROUTPUT ON  
DECLARE
```

```
    emp_num NUMBER(6) := 102;
```

```
    emp_last_name VARCHAR2(25);
```

```
    PROCEDURE find_emp_name (emp_id IN NUMBER, emp_name OUT VARCHAR2)  
    IS
```

```
    BEGIN
```

```
        SELECT last_name INTO emp_name
```

```
        FROM employees
```

```
        WHERE employee_id = emp_id;
```

```
    END;
```

```
BEGIN
```

```
    find_emp_name(emp_num, emp_last_name); --trả kết quả về môi trường gọi ctrình con
```

```
    DBMS_OUTPUT.PUT_LINE ('EMP_ID OF ' || emp_last_name);
```

```
END;
```

Đổi số trong chương trình con sử dụng OUT

DECLARE

emp_num NUMBER(6) := 120;

bonus NUMBER(6) := 50;

emp_last_name VARCHAR2(25);

PROCEDURE raise_salary (emp_id **IN** NUMBER, amount **IN** NUMBER,
emp_name **OUT** VARCHAR2) IS

BEGIN

UPDATE employees SET salary = salary + amount

WHERE employee_id = emp_id;

SELECT last_name INTO **emp_name**

FROM employees

WHERE employee_id = emp_id;

END raise_salary;

BEGIN

raise_salary(emp_num, bonus, emp_last_name);

DBMS_OUTPUT.PUT_LINE ('Salary was updated for: ' || emp_last_name);

END;


```
CREATE OR REPLACE PROCEDURE
```

```
    Get_emp_rec (Emp_number IN Emp_tab.Empno%TYPE,  
                Emp_ret    OUT Emp_tab%ROWTYPE) IS
```

```
BEGIN
```

```
    SELECT Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno  
    INTO Emp_ret  
    FROM Emp_tab  
    WHERE Empno = Emp_number;
```

```
END;
```

```
DECLARE
```

```
    Emp_row    Emp_tab%ROWTYPE;    -- declare a record matching a  
                                   -- row in the Emp_tab table
```

```
BEGIN
```

```
    Get_emp_rec(197, Emp_row); -- call for Emp_tab# 197  
    DBMS_OUTPUT.PUT(Emp_row.Ename || ' ' || Emp_row.Empno);  
    DBMS_OUTPUT.PUT(' ' || Emp_row.Job || ' ' || Emp_row.Mgr);  
    DBMS_OUTPUT.PUT(' ' || Emp_row.Hiredate || ' ' || Emp_row.Sal);  
    DBMS_OUTPUT.PUT(' ' || Emp_row.Comm || ' ' || Emp_row.Deptno);  
    DBMS_OUTPUT.NEW_LINE;
```

```
END;
```

```
CREATE OR REPLACE PROCEDURE
```

```
    Get_emp_rec (Emp_number IN Emp_tab.Empno%TYPE,  
                Emp_ret    OUT Emp_tab%ROWTYPE) IS
```

```
BEGIN
```

```
    SELECT Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno  
        INTO Emp_ret  
        FROM Emp_tab  
        WHERE Empno = Emp_number;
```

```
END;
```

```
CREATE OR REPLACE PROCEDURE
```

```
    Display_emp_rec (Emp_row  Emp_tab%ROWTYPE) IS
```

```
BEGIN
```

```
    DBMS_OUTPUT.PUT(Emp_row.Ename || ' ' || Emp_row.Empno);  
    DBMS_OUTPUT.PUT(' ' || Emp_row.Job || ' ' || Emp_row.Mgr);  
    DBMS_OUTPUT.PUT(' ' || Emp_row.Hiredate || ' ' || Emp_row.Sal);  
    DBMS_OUTPUT.PUT(' ' || Emp_row.Comm || ' ' || Emp_row.Deptno);  
    DBMS_OUTPUT.NEW_LINE;
```

```
END;
```

```
DECLARE
```

```
    Emp_row  Emp_tab%ROWTYPE;  -- declare a record matching a  
                                -- row in the Emp_tab table
```

```
BEGIN
```

```
    Get_emp_rec(197, Emp_row); -- call for Emp_tab# 197
```

```
    Display_emp_rec (Emp_row);
```

```
END;
```

Sử dụng function thay vì sử dụng OUT trong procedure

```
CREATE OR REPLACE FUNCTION Get_emp_rec1 (Emp_number IN Emp_tab.Empno%TYPE)
  RETURN Emp_tab%ROWTYPE IS

  Emp_ret  Emp_tab%ROWTYPE;

BEGIN
  SELECT Empno, Ename, Job, Mgr, Hiredate, Sal, Comm, Deptno
    INTO Emp_ret
   FROM Emp_tab
  WHERE Empno = Emp_number;
  RETURN Emp_ret;
END;
```

```
DECLARE
  Emp_row  Emp_tab%ROWTYPE;  -- declare a record matching a
                             -- row in the Emp_tab table

BEGIN
  Emp_row := Get_emp_rec1(197); -- call for Emp_tab# 197
  Display_emp_rec (Emp_row);
END;
```

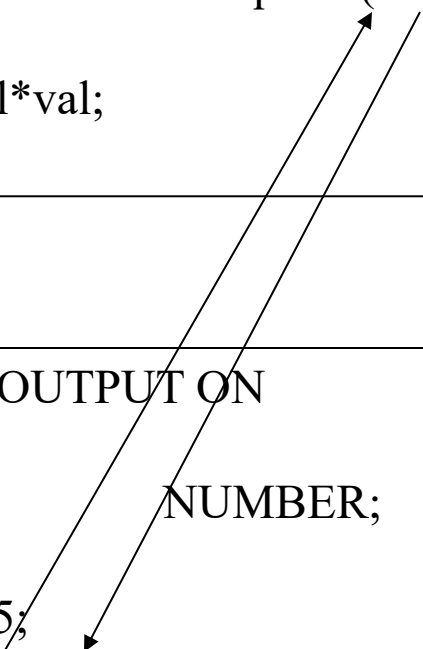
Đổi số trong chương trình con sử dụng IN OUT

- Đổi số IN OUT truyền giá trị khởi tạo tới chương trình con và trả về một giá trị mới cho môi trường gọi nó.
- Đổi số IN OUT phải là một biến, không được là hằng số hay một biểu thức.

Đổi số trong chương trình con sử dụng IN OUT

```
CREATE PROCEDURE square (val IN OUT NUMBER) IS  
BEGIN  
    val := val*val;  
END;
```

```
SET SERVEROUTPUT ON  
DECLARE  
    v_init NUMBER;  
BEGIN  
    v_init := 5;  
    square (v_init);  
    DBMS_OUTPUT.PUT_LINE (v_init);  
END;
```



Đổi số trong chương trình con sử dụng IN OUT

DECLARE

x NUMBER;

PROCEDURE fact (a **IN OUT** NUMBER) is

b NUMBER :=1;

BEGIN

FOR i IN 1..a LOOP

 b := b*i;

END LOOP;

a := b;

END;

BEGIN

--x is the input variable

x:=&values;

fact(**x**);

DBMS_OUTPUT.PUT_LINE(**x**);

END;

Giá trị mặc định trong đối số của chương trình con

DECLARE

```
emp_num NUMBER(6) := 120;
```

```
bonus NUMBER(6) := 20;
```

```
merit NUMBER(4);
```

```
PROCEDURE raise_salary (emp_id IN NUMBER,  
                        amount IN NUMBER DEFAULT 100,  
                        extra IN NUMBER DEFAULT 50) IS
```

BEGIN

```
    UPDATE employees SET salary = salary + amount + extra  
    WHERE employee_id = emp_id;
```

```
END raise_salary;
```

BEGIN

```
-- Same as raise_salary(120, 100, 50)
```

```
raise_salary(120);
```

```
raise_salary(121, 30);
```

END;

Gọi chương trình con

sử dụng ký hiệu vị trí (positional), đặt tên (named), và hỗn hợp (mixed)

DECLARE

```
emp_num NUMBER(6) := 120;
```

```
bonus NUMBER(6) := 50;
```

```
PROCEDURE raise_salary (emp_id NUMBER, amount NUMBER) IS  
BEGIN
```

```
    UPDATE employees SET salary =  
        salary + amount WHERE employee_id = emp_id;
```

```
END raise_salary;
```

BEGIN

```
-- Positional notation:
```

```
raise_salary(emp_num, bonus);
```

```
-- Named notation (parameter order is insignificant):
```

```
raise_salary(amount => bonus, emp_id => emp_num);
```

```
raise_salary(emp_id => emp_num, amount => bonus);
```

```
-- Mixed notation:
```

```
raise_salary(emp_num, amount => bonus);
```

```
END;
```


Summary

IN	OUT	IN OUT
The default.	Must be specified.	Must be specified.
Passes values to a subprogram.	Returns values to the caller.	Passes initial values to a subprogram; returns updated values to the caller.
Formal parameter acts like a constant.	Formal parameter acts like an uninitialized variable.	Formal parameter acts like an initialized variable.
Formal parameter cannot be assigned a value.	Formal parameter cannot be used in an expression; must be assigned a value.	Formal parameter should be assigned a value.
Actual parameter can be a constant, initialized variable, literal, or expression.	Actual parameter must be a variable.	Actual parameter must be a variable.