An-toan-va-bao-mat-he-thong-thong-tin lab-04 pl-sql-(1) - [cuuduongthancong

Kỹ thuật hệ thống (Trường Đại học Bách khoa, Đại học Quốc gia Thành phố Hồ Chí Minh)

Bài thực hành số4

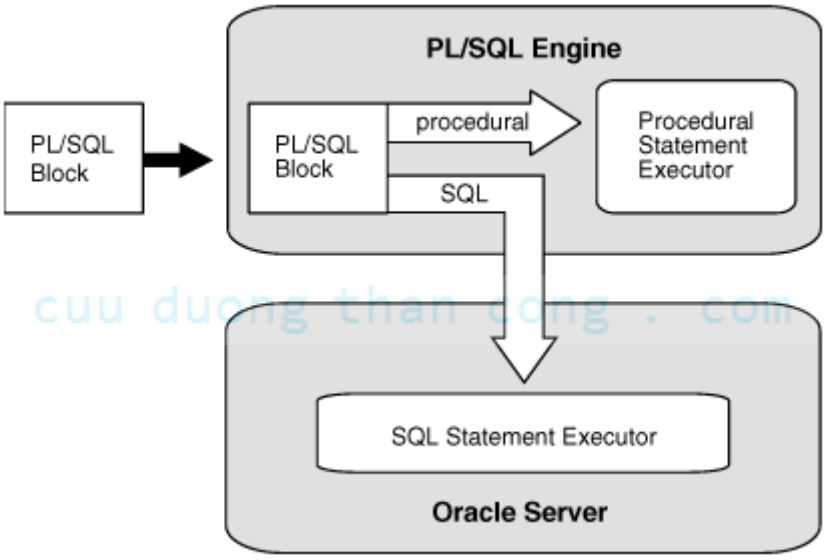
PL/SQL (1)

❖ *Tóm tắt nội dung:*

- Khái niệm PL/SQL
- Các vấn đề liên quan đến kiểu dữ liệu trong PL/SQL
- Hằng và Biến
- Cấu trúc khối PL/SQL
- Các câu lệnh điều khiển

I. PL/SQL là gì ?

PL/SQL (PL : Procedural Language – Ngôn ngữ Thủ tục) là một mở rộng của SQL, kết hợp vào trong đó rất nhiều đặc tính của các ngôn ngữ lập trình gần đây. Nó cho phép các thao tác dữ liệu và các câu lệnh query SQL bao gồm các đoạn mã có cấu trúc khối và tính thủ tục (block-structure and procedural unit of code), làm cho PL/SQL thành một ngôn ngữ xử lý giao dịch mạnh mẽ.



II. Các lệnh SQL trong PL/SQL

- PL/SQL cung cấp một số câu lệnh thủ tục cho việc thao tác và kiểm tra dữ liệu, thường không cần phải dính dáng với các lệnh SQL. Dù vậy, khi cần lấy thống tin từ CSDL hoặc thay đổi trên CSDL thì nên dùng SQL.
- PL/SQL hỗ trợ tốt cho đa số các lệnh DML và các lệnh điều khiển giao dịch trong SQL. Ngoài ra, các câu lệnh SELECT có thể dùng để gán các giá trị query từ 1 hàng trong bảng cho các biến.
- Một số điểm lưu ý:
  - ✓ Một khối PL/SQL không phải là một đơn vị giao dịch (transaction unit) – các lệnh COMMIT và ROLLBACK là độc lập với các khối nhưng có thể nằm trong nó.
  - ✓ Mỗi câu lệnh SQL cần phải kết thúc bởi dấu chấm phẩy.
  - ✓ Câu lệnh SELECT có thể dùng để gán các giá trị query từ 1 hàng trong bảng cho các biến.
  - ✓ Các câu lệnh SELECT mà không trả lại **đúng một hàng** sẽ gây ra một lỗi cần phải giải quyết (thường là phải dùng phương pháp xử lý ngoại lệ hoặc cursor).
  - ✓ Các lệnh DDL không dùng được trong PL/SQL. Ví dụ:
    - Tất cả các lệnh bắt đầu bằngALTER, CREATE, DROP, FLASHBACK
    - Các lệnh quản lý quyền: GRANT, REVOKE
    - Các lệnh audit: AUDIT, NOAUDIT

      (và còn nhiều lệnh khác)
  - ✓ Các lệnh DML có thể xử lý nhiều hàng (multiple rows).

III. Kiểu dữ liệu

- PL/SQL hỗ trợ rất nhiều kiểu dữ liệu để có thể khai báo các biến và các hằng. Có thể gán một giá trị ban đầu cho các biến khi khai báo biến và có thể thay đổi các giá trị của chúng thông qua các phát biểu gán về sau trong khối. Các hằng là các danh hiệu (identifier) lưu giữ một giá trị cố định và giá trị này phải được gán cho hằng khi hằng được khai báo.
- Các kiểu dữ liệu:
  - ✓ Dữ liệu số: **NUMBER**

    Ví dụ: NUMBER(7,2)

    Nghĩa là có 7 ký số trong đó có 2 ký số sau dấu thập phân.Nếu ta không khai báo độ

chính xác là 2 như câu lệnh trên thì độ chính xác mặc định là 38 ký số.

- ✓ Dữ liệu luận lí: **BOOLEAN**
- ✓ Dữ liệu ngày tháng: **DATE**
- ✓ Dữ liệu chuỗi:

VARCHAR2	Lưu trữ các dữ liệu ký tự có chiều dài thay đổi. Chiều dài mặc định là 1 ký tự. Chiều dài tối đa là 32767. Ví dụ: `VARCHAR2(30)`
CHAR	• PL/SQL Version 1: giống như VARCHAR2 nhưng chiều dài tối đa là 255. • PL/SQL Version 2: chuỗi các ký tự chiều dài cố định dài tối đa là 32767 byte. Khi so sánh hai chuỗi với nhau thì các ký tự trống sẽ được thêm vào. • *Chú ý:*Khi so sánh 2 chuỗi CHAR trong PL/SQL Version 1 thì hai chuỗi này không được thêm vào các ký tự trống, ví dụ một biến kiểu CHAR chứa 'FRED' thì khác với một biến kiểu CHAR chứa 'FRED '.

IV. Khai báo biến và hằng

1. Khai báo các biến

- Các biến PL/SQL có thể được khai báo và có thể được gán một giá trị ban đầu trong phần DECLARE của khối. Các biến khác được tham khảo đến trong phần khai báo thì chúng phải được khai báo ở trong một phát biểu trước đó.
- *Cú pháp:*

  identifierdatatype [(precision, scale)] [NOT NULL] [:= expression];

  trong đó

  identifier - tên biến

  datatype - kiểu dữ liệu của biến

  precision - chiều dài của biến (số ký số của phần nguyên và phần thập phân)

  scale - số số lẻ (số ký số của phần thập phân)

- Nếu không gán giá trị ban đầu cho biến thì biến sẽ chứa giá trị NULL cho đến khi gán giá trị mới. Ràng buộc NOT NULL không được dùng trong trường hợp này.

Ví dụ:

```
v_count         NUMBER NOT NULL := 0;
v_saraly        NUMBER(7,2);
v_annsal        NUMBER(9,2) := month_sal * 12;
-- month_sal phải tồn tại trước
postcost        CHAR(7);
surname         VARCHAR2(25) := 'Skywalker';
v_message       VARCHAR2(80) := 'Data is wrong !';
married         BOOLEAN := FALSE;
today           DATE := SYSDATE;
```

- Không nên đặt tên của biến trùng tên với các tên cột của bảng được dùng trong khối. Nếu các biến trong các phát biểu SQL có cùng tên với tên cột, thì Oracle xem tên này là tên cột (mà không phải là tên biến).

  Ví dụ:

```
DECLARE
bonus  NUMBER(8,2);
emp_id NUMBER(6) := 100;
BEGIN
  SELECT salary * 0.10 INTO bonus FROM employees
WHERE employee_id = emp_id;
END;
```


2. Khai báo hằng

- *Cú pháp:*

  identifier CONSTANT datatype [(precision,scale)] :=expression;

  Ví dụ:

```
pi    CONSTANT   NUMBER(9,5) := 3.14159;
vat   CONSTANT   NUMBER(4,2) := 17.5;
```


- Chú ý:Có thể dung từ khóa %TYPE để dùng cùng kiểu với cột được chỉ định trong 1 table. *Ví dụ:* biến product_type sẽ có cùng kiểu với cột price của bảng products:

```
product_price  products.price%TYPE;
```

V. Các biến kết hợp của SQL*Plus

- SQL*Plus hỗ trợ biến kết hợp (bind variable). Đây là các biến dùng để gửi các giá trị vào trong hay ra ngoài một khối PL/SQL.
- *Cú pháp:*

VARIABLE variable_name [NUMBER | CHAR | CHAR(n) | VARCHAR2 | VARCHAR2(n)]

Ví dụ :

```
VARIABLE deptnum NUMBER;
/*Chúng có thể dùng trong các khối PL/SQL với dấu 2 chấm phía
trước */
BEGIN
 SELECT DEPTNO
 INTO  :deptnum
 FROM  DEPT
 WHERE DNAME = 'ACCOUNTING';
 INSERT  INTO RESULTS VALUES( :deptnum);
END;
```

Trong ví dụ trên, giá trị DEPTNO được lấy ra và gán cho biến deptnum.Sau đó được ghi vào bảng RESULTS. Sau khi chạy xong khối PL/SQL, bạn có thể hiển thị giá trị của một biến kết hợp bằng lệnh PRINT :

```
SQL> PRINT deptnum
DEPTNUM
---------------------
10
```


VI. Hàm chuyển đổi kiểu

- TO_CHAR
- TO_DATE
- TO_NUMBER

Ví dụ :

```
v_message    VARCHAR2(80) := 'SCOTT earns '
                             || TO_CHAR (month_sal * 12);
```

VII. Độ ưu tiên của toán tử

	Toán tử	Tác vụ
Đầu tiên	**, NOT	Toán tử mũ, phủ định luận lý
	+, -	Đồng nhất, dấu âm
	*, /	Nhân, chia
	+, -, ||	Cộng, trừ, nối chuỗi
	=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	So sánh
	AND	Giao
Cuối cùng	OR	Hội

VIII. Cấu trúc khối

- *Cú pháp:*

[DECLARE

declaration_statements **]**

BEGIN

executable_statements

[EXCEPTION

exception_handling_statements **]**

END;

- DECLARE và EXCEPTION là phần tự chọn, có vài khối không có 2 phần này.

Ví dụ:(ví dụ trong command line)

```
SQL>  DECLARE
   2  x NUMBER(7,2);
   3  BEGIN
   4     SELECT sal INTO x FROM emp WHERE empno=&&n;
   5        IF x<300 THEN
   6              UPDATE emp SET sal=3000
   7              WHERE empno=&&n;
   8        END IF;
   9  END;
  10  .
```

- Đóng buffer với dấu chấm (.)
- Để chạy PL/SQL trong buffer, gõ lệnh RUN hoặc dấu gạch chéo (/) tại dấu nhắc. Nếu khối được thi hành xong, không có một lỗi không được kiểm soát nào thì chỉ một thông báo được xuất ra :

  'PL/SQL procedure successfully completed'

- Nội dung của buffer có thể soạn thảo theo cách thông thường hay lưu xuống file bằng lệnh SAVE của SQL*Plus.

IX. Lệnh rẽ nhánh

- *Cú pháp:*

  IF condition THEN actions

  [ELSIF condition THEN actions]

  [ELSE actions]

  END IF;

  trong đó '**actions**' là một hay nhiều câu lệnh PL/SQL hay SQL, mỗi câu kết thúc bởi dấu chấm phẩy. Các 'action' này có thể chứa các câu lệnh IF khác lồng nhau.

Ví dụ:

```
IF count > 0 THEN
message := 'count is positive';
  IF area > 0 THEN
message := 'count and area are positive';
  END IF;
ELSIF count = 0 THEN
message := 'count is zero';
ELSE
message := 'count is negative';
END IF;
```

X. Vòng lặp

1. Vòng lặp cơ bản

- *Cú pháp:*

  LOOP

  statements

  END LOOP;

- Mỗi lần dòng chương trình gặp phải END LOOP thì quyền điều khiển trả về tại LOOP. Vòng lặp không điều khiển này sẽ lặp mãi mãi nếu trong thân của nó không có các lệnh nhảy ra khỏi nó.
- Một vòng lặp có thể kết thúc từ bên trong nếu dùng câu lệnh EXIT. EXIT cho phép điều khiển chuyển cho câu lệnh kế tiếp ngay sau END LOOP và kết thúc vòng lặp ngay lập tức.

  Cú pháp :

  EXIT [loop-label] [WHEN condition];

- EXIT có thể là một tác vụ nằm trong câu lệnh IF hoặc đứng một mình trong vòng lặp. Khi đứng một mình thì mệnh đề WHEN có thể dùng để kết thúc có điều kiện

Ví dụ 1:

```
LOOP
      counter := counter + 1;
      INSERT INTO numbered_rows VALUES (counter);
      …
      IF counter = 10 THEN
            COMMIT;
            EXIT;
      END IF;
END LOOP;
```

Ví dụ 2 :

```
LOOP
      …
      EXIT WHEN total_sals = 60000;
      …
END LOOP;
```


- Cách ngắt vòng lặp khác là rẽ nhánh đến một nhãn ra ngoài vòng lặp, đó là dùng lệnh GOTO. Nhưng đây không phải cách viết có cấu trúc.

2. Vòng lặp WHILE

- *Cú pháp :*

  WHILE *condition* LOOP

  statements

  END LOOP;

- Điều kiện này (condition) được tính toán tại điểm bắt đầu của vòng lặp và vòng lặp sẽ kết thúc nếu điều kiện này là FALSE. Nếu điều kiện này FALSE ngay tại lúc bắt đầu vào đến vòng lặp thì vòng lặp không xảy ra.

  Ví dụ:

```
counter := 0;
WHILE counter < 6 LOOP
      counter := counter + 1;
END LOOP;
```

3. Vòng lặp FOR

- *Cú pháp:*

  FOR loop_variable IN [REVERSE] lower_bound..upper_bound

  LOOP

  statements

  END LOOP;

Ví dụ:

```
FOR count2 IN 1..5 LOOP
      DBMS_OUTPUT.PUT_LINE(count2);
END LOOP;
```

4. Điều khiển các vòng lặp lồng nhau

- Thông thường, vòng lặp trong kết thúc thì không kết thúc vòng lặp ngoài (ngoại trừ có lỗi). Dù vậy, các vòng lặp có thể gán nhãn và có thể kết thúc vòng lặp ngoài bằng lệnh EXIT.
- Các nhãn trong PL/SQL được định nghĩa như sau :

  <<label-name>>

Ví dụ :

```
<<main>> LOOP
            …
            LOOP
            …
            --thoát cả 2 vòng lặp
            EXIT main WHEN total_done='YES';
            --thoát khỏi vòng lặp trong
            EXIT WHEN innder_done='YES';
            …
            END LOOP;
END LOOP main;
```

- Ngoài ra nhãn còn dùng để định danh vòng lặp khi chúng có cấu trúc lồng nhau.

Ví dụ:

```
<<block1>> DECLARE
var1 NUMBER;
BEGIN
      <<block2>> DECLARE
      var1 NUMBER := 400;
      BEGIN
            -- biến var1 của khối block1 được tăng lên 1
            block1.var1 := block1.var1 + 1;
      END block2;
END block1;
```

XI. Bài tập

- Trong mỗi bài tập dưới đây, bạn có thể tạo các khối PL/SQL trong buffer của SQL*Plus và sau đó lưu chúng xuống file hoặc tạo ra file riêng bằng các trình soạn thảo khác.
- Trong nhiều bài tập, bạn sẽ cần phải lưu trữ các kết quả trong bảng, giả thiết là bảng chung MESSAGES được dùng. Nó được định nghĩa như sau :

Table MESSAGES

Column	Description
NUMCOL1	NUMBER (9,2)
NUMCOL2	NUMBER (9,2)
CHARCOL1	VARCHAR2 (60)
CHARCOL2	VARCHAR2 (60)
DATECOL1	DATE
DATECOL2	DATE

1. Tạo một khối dùng các biểu thức PL/SQL đơn giản, trong đó khai báo 4 biến :

V_BOOL1	Boolean
V_BOOL2	Boolean
V_CHAR	Character (chiều dài thay đổi)
V_NUM	Number

Sau đó gán cho nó các giá trị sau :

Variable	Value
V_CHAR	Câu '42 is the answer'
V_NUM	Hai ký tự đầu từ V_CHAR
V_BOOL1	TRUE hoặc FALSE (tùy vào V_NUM có nhỏ lơn 100 hay không)
V_BOOL2	Ngược lại với V_BOOL1

2. Tạo và chạy một khối PL/SQL nhận 2 biến của SQL*Plus. Biến đầu tiên cần phải lũy thừa lên với số mũ là số thứ hai và kết quả được gán cho biến PL/SQL. Lưu kết quả này trong bảng MESSAGES hoặc trong biến kết hợp (bind) của SQL*Plus.

3. Viết một khối PL/SQL để chèn một hàng vào trong bảng MESSAGES với NUMCOL1 có giá trị 1 nếu hàng đầu tiên được chèn vào, 2 nếu là hàng thứ 2, … Không chèn hàng vào nếu đếm đến 6 hoặc 8 và thoát khỏi vòng lặp sau khi sau khi chèn đến 10. COMMIT khi kết thúc vòng lặp.