CHƯƠNG VIII:

PL/SQL

GV: PHẠM VĂN ĐĂNG

Tháng: 02/2009

1. Tổng quan về PL/SQL

- Là ngôn ngữ thủ tục của Oracle được dùng để giúp xây dựng các ứng dụng.
- Kết hợp với các lệnh SQL để truy xuất dữ liệu.
- PL/SQL(**P**rocedure **L**anguage / **SEQUEL - S**tructured **E**nglish **Que**ry **L**angguage) là sự mở rộng của SQL, bao gồm các đặc điểm của ngôn ngữ lập trình, cho phép thao tác dữ liệu và phát biểu truy vấn (Query) bên trong khối (Block).

2. Thuận lợi của khối phát biểu PL/SQL(Ưu điểm)

- Là ngôn ngữ có cấu trúc khối(block-structure language), PL/SQL có thể chia chương trình thành nhiều khối, với cấu trúc hợp lý với các biến có thể được khai báo nội tại (local) bên trong khối và việc xử lý ngoại lệ(exception) có thể được thực hiện bên trong từng khối.
- Tập trung quy tắc tại hệ quản trị cơ sở dữ liệu.
- Cải thiện tốc độ ứng dụng.
- Dùng các phát biểu điều kiện, lặp.
- Có thể chia chương trình thành các khối, mỗi khối là một block.
- PL/SQL cung cấp các lệnh điều khiển, rẽ nhánh hay vòng lặp để điều khiển luồng xử lý của các thủ tục, quyết định điều kiện và thời điểm thực hiện các lệnh SQL hay các hành động khác lên cơ sở dữ liệu.

3. Cấu trúc PL/SQL

- Mỗi đơn vị của PL/SQL có thể tổ hợp một hay nhiều khối (block), các block có thể rời rạc hay lồng nhau.
- Thông thường, một block có thể là một "anonymous block – block vô danh" (một block không đặt tên) hay là một đoạn chương trình con (sub-program).

3.1. Khai báo một block vô danh "anonymous block"

- *Một block vô danh có thể là:*
  - Cấu trúc khối vô danh (PL/SQL Block)
  - Chương trình con : Thủ tục (Procedure) hay hàm (Function)
- Cấu trúc block vô danh có dạng:

```
Declare        --Khai báo biến
Begin          --Thực hiện các công việc sau begin
               …
Exception      --Nắm bắt các ngoại lệ
               …
End;
```

- Cấu trúc chương trình con có 2 dạng: *(Ta sẽ tìm hiểu chi tiết ở chương VI)*
  - ✓ Thủ tục (Procedure) không trả về giá trị, nhưng xuất kết quả ngay trong nội tại thủ tục.

Cú pháp:

```
Create [or Replace] Procedure <Tên thủ tục>
        [(      DS tham số [in | out | in out] <Kiểu dữ liệu>)]
        IS/AS  Begin
                        . . .
                End;
```

- ✓ Hàm(Function) được dùng để tính toán và trả về một giá trị kết quả.

Cú pháp:

```
Create [or Replace] Function <Tên hàm>
[ (DS tham số in KiểuDL)]    Return <kiểu DL>
Is ds biến cục bộ;
Begin
        . . .
End;
```

Ví dụ: (Về cấu trúc Block vô danh)

- Giảm số lượng(SL) của mặt hàng có MSMH='101' và MSHD=1 xuống 1.

```
DECLARE   Soluong  Number (5);
Begin
   SELECT  SL  INTO  Soluong   FROM  CT_HOADON
   WHERE  MSMH = '101' AND  MSHD = 1;
   If   Soluong > 0  Then
         Update   CT_HOADON  Set SL= SL-1
         Where   MSHD=1  And MSMH='101';
   End if;
   Commit;
         EXCEPTION
         WHEN  NO_DATA_FOUND  THEN
                  Dbms_output.Put_line ('Không tìm thấy data') ;
End;
```

Lấy SL thông qua mã hàng và mã hóa đơn

Thực hiện cập nhật lại SL

Nắm bắt ngoại lệ

Chú ý:

- Có thể ghi chú trong PL/SQL bằng ký hiệu: /*. . . . . .*/
- Các biến phải bắt đầu bằng ký tự chuỗi.
- Commit dấu hiệu kết thúc một transaction.
- Không dùng từ khóa mà Oracle đã dùng để khai báo biến

Ví dụ: Declare **Commit** Number(5) : **Sai**

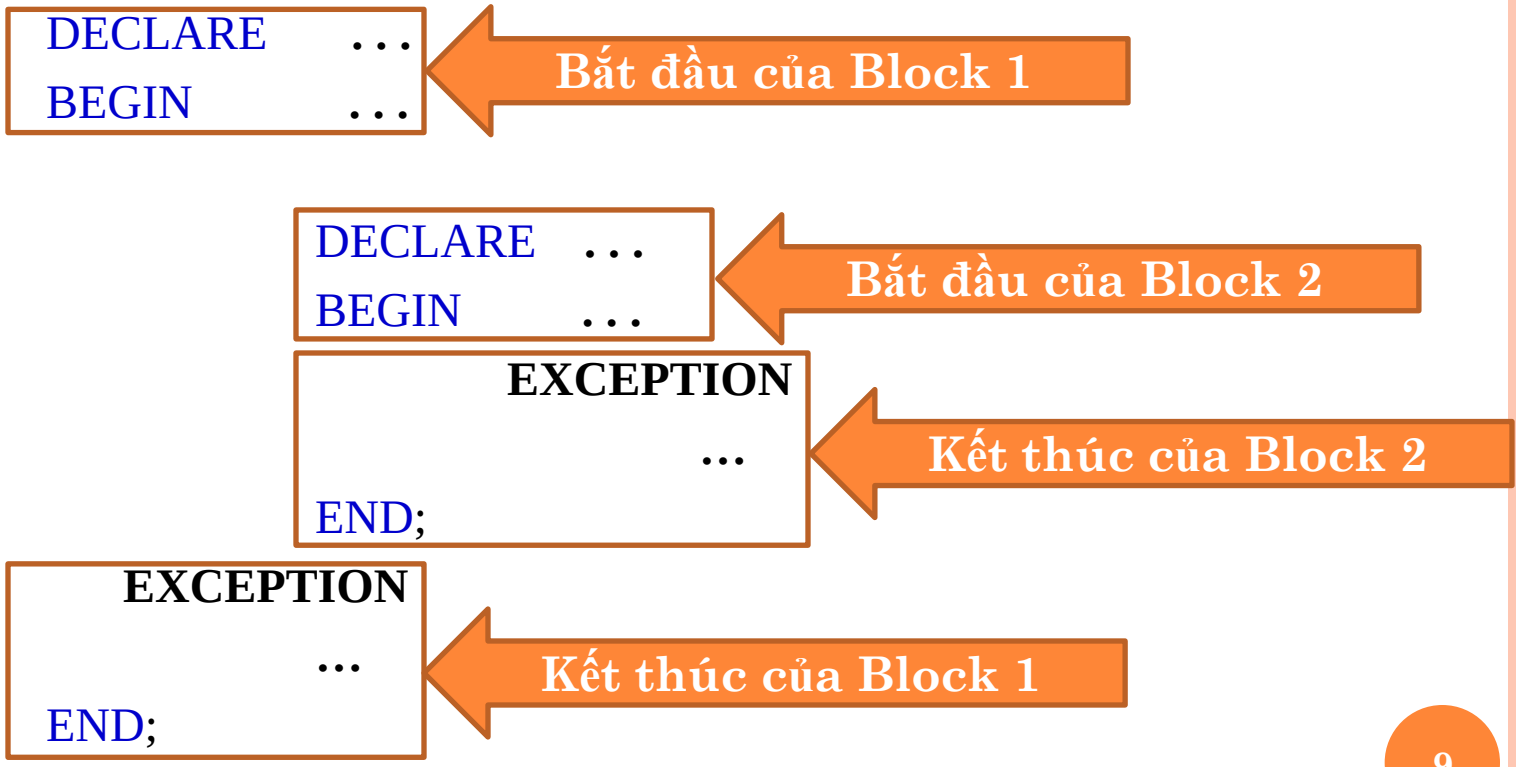
Lưu ý:

Các từ khóa **Declare**, **Begin**, **Exception** **không** được theo sau bởi dấu chấm phẩy (**;**), nhưng theo sau là từ khóa **END** và các lệnh khác của PL/SQL phải có dấu chấm phẩy(**;**) cuối câu.

3.2. Các block có thể lồng nhau

- Mỗi đơn vị của PL/SQL phải tạo thành một block. Yêu cầu nhỏ nhất sẽ được giới hạn bởi từ khóa **BEGIN** và **END** bao quanh các hành động cần thực thi.
- PL/SQL cho phép các block lồng nhau. Có thể đặt một block PL/SQL lồng nhau ở bất kỳ nơi nào mà một phát biểu thực thi được phép.

Ví dụ: Hai block lồng nhau



3.3. Phạm vi của các đối tượng

- Phạm vi của một đối tượng là toàn bộ vùng chương trình mà đối tượng đó có thể được nhìn thấy và được sử dụng. Đó là toàn bộ block mà trong đó đối tượng được khai báo và bao gồm bất kỳ block con nào lồng trong đó.

4. Cú pháp cơ bản của PL/SQL

4.1. Khai báo biến và hằng

Các biến PL/SQL có thể được khai báo và xác định giá trị ban đầu trong đoạn DECLARE của một block.

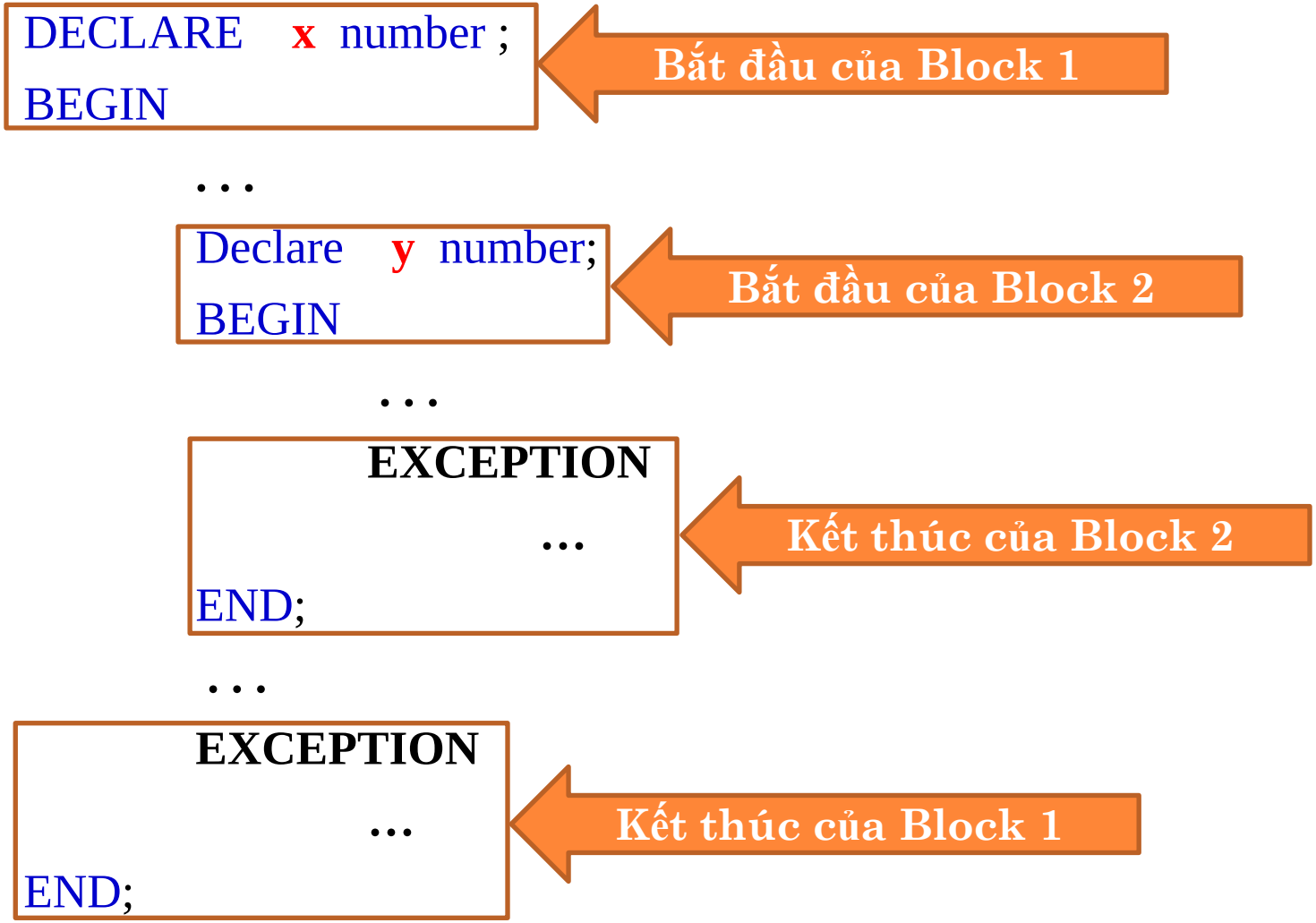
Ví dụ:
```
DECLARE   Name   char(50);
          Age    number(2);
          Pi     Constant  Number(9,5) := 3.14159;
...
```

4.2. Gán biến và biểu thức

Cú pháp: <Định danh> **:=** Biểu thức;

Ví dụ: Name **:=**UPPER('Good');

4.3. Tầm nhìn (Scope) của hai block vô danh lồng nhau



- *Khi đó:* **x** có thể sử dụng hai block ngoài và trong.
  y chỉ có thể sử dụng một block trong.

Ví dụ 1: **Cho biết kết qủa in ra màn hình? Sau khi sử dụng 2 khối vô danh và 1 nhãn hiệu (Two Block – One Lable)**

```
<<outer_block>>
DECLARE
    scope_Num number :=12;
BEGIN
    DECLARE
        scope_Num   number := 24;
        Num_A       number := outer_block.scope_Num;
    BEGIN
        dbms_output.put_line('Block trong');
        dbms_output.put_line(scope_Num);
        dbms_output.put_line(Num_A);
    END;
    dbms_output.put_line('Block ngoài');
    dbms_output.put_line(scope_Num);
END;
```

Bắt đầu của Block 1

Bắt đầu của Block 2

Kết thúc của Block 2

Kết thúc của Block 1

Ví dụ 2: Kiểm tra ngày sinh nhật có giống nhau không, sử dụng 2 khối vô danh và 2 nhãn hiệu (Two Block - Two Lables)

```
<<Block_Ngoai>>
DECLARE
    Birthdate Date:='12-NOV-1976';
BEGIN
    <<Block_Trong>>
    DECLARE
        Birthdate  Date:='11-DEC-1976';
    BEGIN
      If Block_Ngoai.Birthdate=Block_Trong.Birthdate Then
          dbms_output.put_line('Co cung ngay sinh nhat');
      Else
          dbms_output.put_line('Khong cung ngay sinh nhat');
      End if;
    END;
END;
```

Block ngoài

Block trong

Ví dụ 3: Cho biết kết quả in ra tính phân số, sử dụng 2 khối vô danh và 4 nhãn hiệu (Two Block-Four Lables)

```
<<Block_Nhanhieu1>>
<<Ngoai_Tinh_PS1>>
Declare
       Tuso     number:=22;
       Mauso    number:=4;
       Ketqua   number;
       Message_Err_Out boolean:=True;
       Message_Err_In  boolean:=True;
Begin
       <<Block_Nhanhieu2>>
       <<Trong_Tinh_PS2>>
       Declare
           Mauso    number:=6;
       Begin
           --Su dung Mauso cua Block NGOAI
           Message_Err_Out:=False;
           Ketqua:=Tuso/Ngoai_Tinh_PS1.Mauso;
           dbms_output.put_line('Ket qua block ngoai la:' || round(Ketqua,2));
           Message_Err_Out:=True;
           --Su dung Mauso cua Block TRONG
           Message_Err_In:=False;
           Ketqua:=Tuso/Trong_Tinh_PS2.Mauso;
           dbms_output.put_line('Ket qua block trong la:' || round(Ketqua,2));
           Message_Err_In:=True;
           --Su dung Mauso cua Block TRONG
           Ketqua:=Tuso/Block_Nhanhieu2.Mauso;
           dbms_output.put_line('Ket qua block trong la:' || round(Ketqua,2));
       End Trong_Tinh_PS2;
       EXCEPTION
           WHEN ZERO_DIVIDE THEN
               If Message_Err_Out=False Then
                   dbms_output.put_line('Ket qua block ngoai khong hop le');
               Elsif Message_Err_In=False Then
                   dbms_output.put_line('Ket qua block trong khong hop le');
               End if;
           WHEN OTHERS THEN
               dbms_output.put_line('Loi khac');
End Ngoai_Tinh_PS1;
```

Ví dụ 4: Đếm số ngày nghỉ lễ

```
DECLARE     SoNgay          Number;
            NgayHienTai     Date;
BEGIN
        --Lấy ngày hiện tại của hệ thống
        Select SYSDATE into NgayHienTai  From  DUAL;
        --Đếm số ngày nghỉ lễ
        Select Count(*)  Into  SoNgay  From  NGAYNGHILE
        Where   TO_DATE(NGAYNGHI, 'DD-MM')=
                TO_DATE(NgayHienTai, 'DD-MM');
        --Xuất số ngày nghỉ
        dbms_output.put_line(to_char(SoNgay));
END;
```

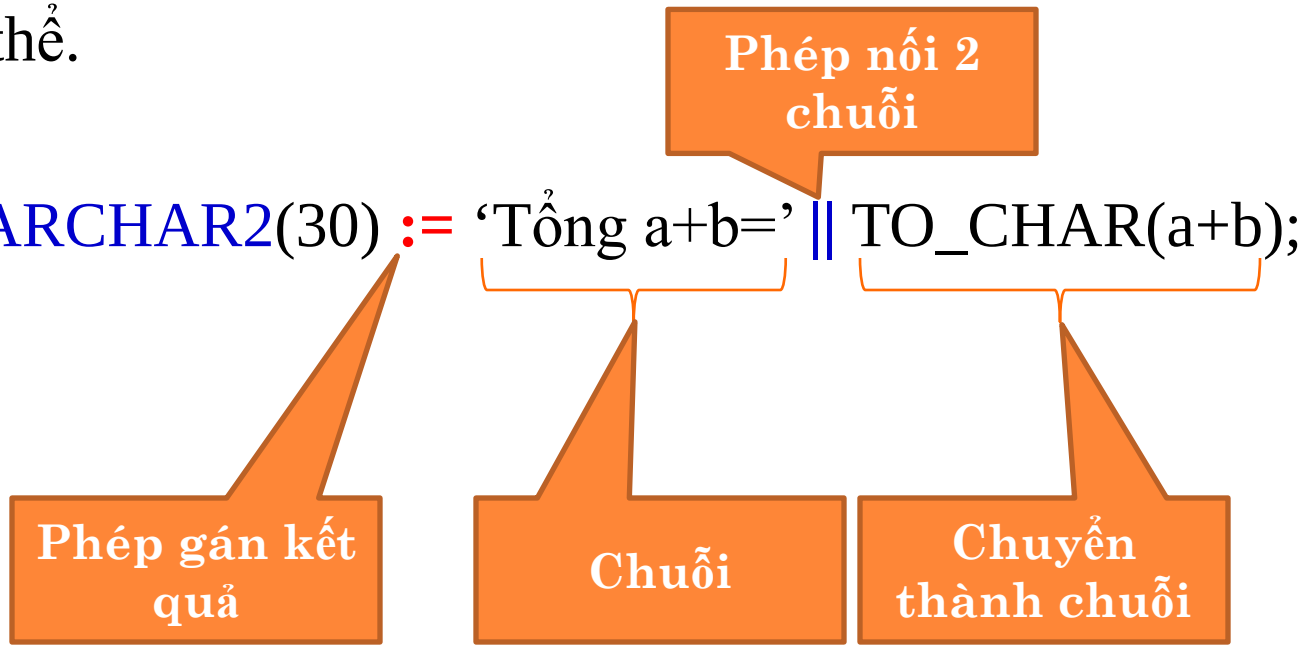
Trong đó: Ta có Table nghỉ lễ như sau:

NGAYNGHILE(MS, NGAYNGHI, LYDO)

4.4. Chuyển đổi kiểu dữ liệu

- Nếu kiểu dữ liệu hỗn hợp xuất hiện trong cùng một biểu thức thì dùng hàm chuyển đổi dữ liệu tương ứng như TO_CHAR, TO_DATE, TO_NUMBER. Khi đó, PL/SQL sẽ cố gắng chuyển đổi nếu có thể.

Ví dụ:



Lưu ý:

Ta phải chuyển a+b thành chuỗi thì mới nối chuỗi được

Ký hiệu nối chuỗi là : ||

4.5. Các kiểu dữ liệu cho khai báo biến

- ✓ Char, Varchar2
- ✓ Nchar, Nvarchar2 (sử dụng cho mã Unicode)
- ✓ Boolean, Date, Number

❖ **Phép gán***(Nhắc lại)* Ký hiệu **:=** ‘’;

Ví dụ:

Tên **:=** ‘Minh’;

❖ **Khai báo hằng***(Nhắc lại)*

Cú pháp:

<Tên hằng> Constant <kiểu DL> **:=** giá trị;

Ví dụ:

Pi Constant number (9,4) **:=**3.1415;

- Các chú ý về câu lệnh trong PL/SQL:
  - ✓ Mỗi phát biểu phải kết thúc bằng dấu chấm phẩy(;).
  - ✓ Mỗi block trong PL/SQL không phải là một transaction.
  - ✓ Phát biểu dòng dữ liệu cho phép trong PL/SQL.
  - ✓ Các SELECT không trả về dòng nào hay nhiều dòng sẽ gây ra các ngoại lệ có mã số sau:

    - ORA-01403-NO DATA FOUND

    -ORA-RETURN MORE THAN REQUEST NUMBER OF NOW
- PL/SQL cung cấp các thuộc tính sau để đánh giá kết quả:
  - ✓ SQL%ROWCOUNT : Số dòng được xử lý
  - ✓ SQL%FOUND : Tìm thấy hay không
  - ✓ SQL%NOFOUND : Tìm không thấy

Ví dụ:

```
DECLARE   Sodong  Number;
BEGIN
        Delete  From  PHONGBAN
        Where MSPB = 5;
        sodong = SQL%ROWCOUNT;
         if (Sodong>0) Then
                Dbms_output.Put_Line('Số dòng được xóa
                khỏi bảng Phòng Ban là: ' ||
                to_char(Sodong) || ' dòng');
        End if;
 End;
```

Kết quả: Giả sử ta thực thi Block trên như sau:

Số dòng được xóa khỏi bảng Phòng Ban là: 1 dòng

5. Cấu trúc điều khiển trong PL/SQL

PL/SQL cung cấp các lệnh để điều khiển dòng phát biểu như sau

5.1. Phát biểu điều kiện

Cú pháp 1:

```
If (điều kiện) Then
        . . . . . .
        . . . . . .
End if;
```

Ví dụ:

```
If a > b Then
        Dbms_output.put_line('a lớn hơn b');
End if;
```

Cú pháp 2:

```
If (điều kiện ) Then
        . . .
        . . .
Else
        . . .
        . . .
End if;
```

Ví dụ:

```
If a > b Then
        Dbms_output.put_line('a lớn hơn b');
Else
        Dbms_output.put_line('b lớn hơn a');
End if;
```

Cú pháp 3:

If (điều kiện) Then

. . . . . .

. . . . . .

[Elsif (điều kiện) Then

. . . .

. . . . ;]

[Else

. . . .;]

End if;

End if;

Ví dụ: Hãy xếp loại học sinh qua tiêu chuẩn sau:

Xếp Loại	Giỏi	: Nếu DTB>=8
	Khá	: Nếu DTB>=7
	TB	: Nếu DTB>=5
	Kém	: còn lại

Trong đó: DTB là điểm trung bình.

Hướng dẫn:

```
DECLARE      DTB      Number:=7;
BEGIN
        If  DTB>=8 Then
                        dbms_output.put_line('Loại giỏi');
        ElsIf  DTB>=7 Then
                        dbms_output.put_line('Loại khá');
        ElsIf DTB>=5 Then
                        dbms_output.put_line('Loại TB');
        Else
                        dbms_output.put_line('Loại kém');
        End if;
END;
```

5.2. Phát biểu lặp

Dạng 1(Cú pháp):

```
LOOP
    . . . ;
    EXIT or EXIT WHEN Đk_kết thúc;
END LOOP;
```

Dạng 2(Cú pháp):

```
<<Tên_loop>>
LOOP
    ……;
    EXIT Tên_loop [WHEN Đk_kết thúc];
    ……;
END LOOP;
```

Vòng lặp kết thúc khi gặp phát biểu Exit, lệnh Exit có thể nhận hai dạng sau: Exit when (điều kiện); **Hoặc** Exit;

Ví dụ 1: Hiển thị các số từ 1 đến 10

```
DECLARE   n_Num number:=1;
BEGIN
        <<Loop_n_Num>>
        LOOP
                dbms_output.put_line(n_Num);
                Exit Loop_n_Num
                WHEN (n_Num>=10);
                        n_Num := n_Num+1;
        END LOOP;
END;
```

Ví dụ 2: Hiển thị các số từ 1 đến 10.

```
DECLARE
        Ncount          Number(6);
BEGIN
        --Khởi tạo cho biến Ncount
        Ncount:=1;
        Loop
                Dbms_output.put_line(N‘Số thứ: ’|| NCount);
                Ncount :=Ncount  + 1;
                --Nếu điều kiện đúng thì dừng vòng lặp
                --Loop…End Loop.
                Exit When Ncount > 10;
        End Loop;
End;
```

Ví dụ 3: Sử dụng GOTO (GOTO statement)

Tạo table có tên:

```
CREATE TABLE MyTable
(
    num_col    NUMBER,
    char_col   VARCHAR2(60)
);
```

Kết quả thực hiện:

```
Select * from MyTable;

   NUM_COL CHAR_COL
---------- --------------
         1 Loop count
         2 Loop count
         3 Loop count
         4 Loop count
           Done!
```

```
DECLARE
        v_Counter       BINARY_INTEGER := 1;
BEGIN
        LOOP
                INSERT INTO MyTable
                        VALUES (v_Counter, 'Loop count');
                v_Counter := v_Counter + 1;
                IF v_Counter >= 5 THEN
                        GOTO  l_EndOfLoop;
                END IF;
        END LOOP;
         <<l_EndOfLoop>>
                INSERT INTO MyTable (char_col)
                                VALUES ('Done!');
END;
```

Ví dụ 4:

```
Declare      l_loops  number := 0;
Begin
        dbms_output.put_line(Trước khi lặp!!!');

        Loop
                If  l_loops > 4  Then  --Lặp 4 lần rồi thoát
                        Exit;
                End If;
                dbms_output.put_line('Looped ' || l_loops || ' times');
                 l_loops := l_loops + 1; --Tăng lên một đơn vị
         End Loop;

      dbms_output.put_line('Sau khi lặp xong!!!');
End;
```

Kết quả:

```
Truoc khi lap!!!
Looped 0 times
Looped 1 times
Looped 2 times
Looped 3 times
Looped 4 times
Sau khi lap xong!!!

PL/SQL procedure successfully completed.
```

Ví dụ 5: **Exit to a label**

```
DECLARE
        just_a_num NUMBER := 1;
BEGIN
        <<just_a_loop>>
        LOOP
                dbms_output.put_line(just_a_num);
                EXIT just_a_loop
                WHEN (just_a_num >= 10);
                just_a_num := just_a_num + 1;
        END LOOP;
END;
```

Xem kết quả:

```
1
2
3
4
5
6
7
8
9
10

PL/SQL procedure successfully completed.
```

Cú pháp 2:

- Vòng for có giá trị tăng dần.

FOR <giá trị ban đầu> IN <giá trị nhỏ nhất> .. <giá trị lớn nhất>

LOOP

. . .

END LOOP;

- Vòng for có giá trị giảm dần (Reverse).

FOR <giá trị ban đầu> IN Reverse <giá trị nhỏ nhất> ..

<giá trị lớn nhất>

LOOP

. . .

END LOOP;

Ví dụ 1: Hiển thị các số từ 1 đến 10.

```
DECLARE     i       Number;
            m_min   Number;
            m_max   Number;
BEGIN
        --Khởi tạo 3 biến: i, m_min, m_max
        i:=1;
        m_min:=1;
        m_max:=10;
        --Khai báo vòng lặp For
        For  i  in  m_min..m_max
        Loop
                Dbms_output.put_line(N'Số thứ: '|| i);
        End loop;
End;
```

Ví dụ 2: Hiển thị các số từ 10 đến 1.

```
Declare         i       Number;
                m_min   Number;
                m_max   Number;
Begin
   --Khởi tạo 3 biến: i, m_min, m_max
   i:=1;
   m_min:=1;
   m_max:=10;
   --Khai báo vòng lặp For giảm dần(Reverse)
   For  i  in Reverse m_min..m_max
   Loop
         Dbms_output.put_line(N'Số thứ: '|| i);
   End loop;
End;
```

Ví dụ 3: Hiển thị các số từ 1 đến 5.

```
BEGIN
    FOR  i  IN  1..5
   LOOP
        DBMS_OUTPUT.PUT_LINE('Loop counter is ' || i);
    END  LOOP;
 END;
```

Kết quả thực hiện:

Loop counter is **1**

Loop counter is **2**

Loop counter is **3**

Loop counter is **4**

Loop counter is **5**

Ví dụ 5: Hiển thị các số từ 1 đến 5.(**For loop: counter IN 1..5)**

```
DECLARE
    counter   INTEGER := 3;
BEGIN
    FOR counter IN 1..5  LOOP
        DBMS_OUTPUT.PUT_LINE(counter);
    END LOOP;
END;
```

Khởi tạo cho biến counter =3

Khi vào for thì counter =1

Kết quả thực hiện:

1

2

3

4

5

Ví dụ 6: Sử dụng 2 vòng lặp for lồng nhau(**Nesting FOR loops)**

```
BEGIN
    FOR  i  IN 1..2  LOOP
        FOR  j IN 1..4 LOOP
            DBMS_OUTPUT.PUT_LINE('Outer Loop counter is ' ||
                i || ' Inner Loop counter is ' || j);
        END LOOP;
    END LOOP;
END;
```

Vòng for ngoài

Vòng for trong

Kết quả thực hiện:

Outer Loop counter is 1 Inner Loop counter is 1
Outer Loop counter is 1 Inner Loop counter is 2
Outer Loop counter is 1 Inner Loop counter is 3
Outer Loop counter is 1 Inner Loop counter is 4
Outer Loop counter is 2 Inner Loop counter is 1
Outer Loop counter is 2 Inner Loop counter is 2
Outer Loop counter is 2 Inner Loop counter is 3
Outer Loop counter is 2 Inner Loop counter is 4

Ví dụ 7: Sử dụng vòng lặp for đi ngược(REVERSE: Reversing the loop)

```
DECLARE
        loop_start  Integer := 1;
BEGIN
        FOR  i  IN  REVERSE  loop_start..5  LOOP
            DBMS_OUTPUT.PUT_LINE('Loop counter is ' || i);
        END LOOP;
END;
```

Khởi tạo cho biến cận dưới

Vòng for đi ngược

Kết quả thực hiện:

Loop counter is 5
Loop counter is 4
Loop counter is 3
Loop counter is 2
Loop counter is 1

Ví dụ 8: Sử dụng vòng lặp for để Changing the loop increment.

```
BEGIN
    FOR i IN 1..6 LOOP
        IF MOD(i,2) = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Loop counter is ' || i);
        END IF;
    END LOOP;
END;
```

Lấy các số chẵn

Kết quả thực hiện:

Loop counter is 2

Loop counter is 4

Loop counter is 6

Ví dụ 9: **Use variable as an upper bound of for loop**

```
DECLARE
    upper  INTEGER := 5;
BEGIN
    FOR  i  IN  1..upper  LOOP
     DBMS_OUTPUT.PUT_LINE('This has executed ' ||TO_CHAR(i)||' time(s)');
    END LOOP;
END;
```

Khởi tạo cho biến cận trên

Kết quả thực hiện:

This has executed 1 time(s)
This has executed 2 time(s)
This has executed 3 time(s)
This has executed 4 time(s)
This has executed 5 time(s)

Ví dụ 10: **Exit(break) a for loop**

```
DECLARE
    myValue   INTEGER := 5;
BEGIN
    FOR  i  IN  1..5  LOOP
        myValue := myValue +5;
        DBMS_OUTPUT.PUT_LINE(myValue);
        EXIT WHEN  myValue > 100;
    END LOOP;
END;
```

Khởi tạo cho biến

Kết quả thực hiện:

10
15
20
25
30

Ví dụ 11: **Call EXIT to exit a for loop**

```
BEGIN
    FOR  v_loopcounter  IN  1..20  LOOP
        IF  MOD(v_loopcounter,2) = 0  THEN
            dbms_output.put_line('The AREA of the circle is '
                                    ||v_loopcounter*v_loopcounter );
        END IF;
        IF v_loopcounter = 10 THEN
            EXIT;
        END IF;
    END LOOP;
END;
```

Kết quả thực hiện:

The AREA of the circle is 4
The AREA of the circle is 16
The AREA of the circle is 36
The AREA of the circle is 64
The AREA of the circle is 100

Ví dụ 12: **forall from 1 to 10**

Tạo table có tên:

```
CREATE TABLE MyTable
(
    num_col    NUMBER,
    char_col   VARCHAR2(60)
);
```

Kết quả thực hiện: `Select * from MyTable;`

NUM_COL	CHAR_COL
1	Row number 1
2	Row number 2
3	Row number 3
4	Row number 4
5	Row number 5
6	Row number 6
7	Row number 7
8	Row number 8
9	Row number 9
10	Row number 10

```
DECLARE
    TYPE  t_Numbers        IS TABLE OF  MyTable.num_col%TYPE
                                        INDEX BY  BINARY_INTEGER;
    TYPE  t_Chars          IS TABLE OF  MyTable.char_col%TYPE
                                        INDEX BY  BINARY_INTEGER;
    v_Numbers      t_Numbers;
    v_Chars        t_Chars;
BEGIN
    -- Fill up the arrays with 10 rows.
   FOR  v_Count  IN  1..10  LOOP
          v_Numbers(v_Count)          :=  v_Count;
          v_Chars(v_Count)            :=  'Row number ' || v_Count;
   END LOOP;
    -- And insert them into the database using bulk binds.
   FORALL v_Count  IN  1..10
                  INSERT INTO MyTable (num_col, char_col)
                         VALUES(v_Numbers(v_Count), v_Chars(v_Count));
END;
```

Ví dụ 13: Exit outer loop with 'EXIT LabelName When' statement

```
BEGIN
        <<outerloop>>
        FOR  v_outerloopcounter  IN  1..2  LOOP
            <<innerloop>>
            FOR  v_innerloopcounter  IN  1..4  LOOP
                Dbms_output.Put_line('Outer Loop counter is '
                    || v_outerloopcounter ||
                    ' Inner Loop counter is ' || v_innerloopcounter);
                    EXIT  outerloop  WHEN  v_innerloopcounter = 3;
          END  LOOP  innerloop;
        END  LOOP  outerloop;
END;
```

Nhãn ngoài

Vòng lặp ngoài chạy đúng 1 lần.

Nhãn trong

Vòng lặp trong chạy đến 3 là dừng

Xem kết quả:

```
Outer Loop counter is 1 Inner Loop counter is 1
Outer Loop counter is 1 Inner Loop counter is 2
Outer Loop counter is 1 Inner Loop counter is 3

PL/SQL procedure successfully completed.
```

Cú pháp 3: Còn lặp khi điều kiện gắn với mệnh đề WHILE còn đúng

```
While (điều kiện)
Loop
        . . .
End loop;
```

Ví dụ 1: Hiển thị các số từ 1 đến 10.

```
Declare        Ncount          Number;
Begin
        --Khởi tạo biến Ncount
        Ncount:=1;
        --Khai báo vòng lặp While
        While (Ncount <= 10)
        Loop
                Dbms_output.put_line(N'Số thứ: '|| Ncount);
                Ncount := Ncount + 1;
        End loop;
End;
```

Ví dụ 2: Tính diện tích hình tròn (S= r^2*PI).

```
Declare
        r       Number          := 2;
        mypi    Number(1,2)     :=3.14; --Khởi tạo trị PI
Begin
        While True
        Loop
            Dbms_output.Put_line('The Area is ' || mypi * r * r);
            If r = 10 Then --Nếu bán kính tới 10 thì dừng
                Exit;
            End If;
            r := r + 2 ;
        End Loop;
End;
```

Kết quả:

```
The Area is 12.56
The Area is 50.24
The Area is 113.04
The Area is 200.96
The Area is 314
```

Ví dụ 3: Example of a WHILE loop that never executes.

```
Declare
        n  Number := 0;
Begin
      Dbms_output.Put_line('n: ' || n);
      While n >= 10 Loop
            n := n + 1;
            Dbms_output.Put_line('n: ' || n);
      End Loop;
End;
/
```

Ví dụ 4: Change while loop counter.

```
Declare
        loops Number := 0;
Begin
        dbms_output.put_line('Before my loop');

        While loops < 5  Loop
            dbms_output.put_line('Looped ' || loops || ' times');
            loops := loops + 1;
        End Loop;

       dbms_output.put_line('After my loop');
End;
```

Ví dụ 5: While loop with complex conditions.

```
Declare
      counter1  Number(2):= 1;
      counter2  Number(2):= 1;
Begin
        While counter1 <= 12  And  counter2 <= 14
        Loop
                counter1 :=  counter1 + 1;
                counter2 :=  counter2 + 1;
        End Loop;
        Dbms_output.Put_line('counter2:' || counter2);
        Dbms_output.Put_line('counter1:' || counter1);
End;
/
```

Ví dụ 6: WHILE..LOOP, Cursor Loop.

```
DECLARE     CURSOR  myCursor  IS
            SELECT MANV, HONV, TENLOT, TENNV, TENPB
            FROM PHONGBAN pb, NHANVIEN nv
            WHERE nv.MAPHG = pb.MAPHG AND pb.MAPHG=1;
            cs  myCursor%ROWTYPE;
BEGIN
    OPEN myCursor;
     FETCH  myCursor  INTO  cs;
     WHILE  myCursor%FOUND  LOOP
          INSERT INTO NHANVIEN_PHONGBAN (MS,  HOTENNV, TENPB)
          VALUES (cs.MANV, cs.HONV|| ' ' ||cs.TENLOT|| ' ' ||cs.TENNV, cs.TENPB);
     FETCH  myCursor  INTO cs;
     END LOOP;
     CLOSE myCursor;
     COMMIT;
END;
```

Ví dụ 7: Use EXIT WHEN to exit a while loop.

```
DECLARE
        v_X NUMBER := 2;
BEGIN
        WHILE TRUE LOOP
            Dbms_output.Put_line('X^X: ' ||v_X * v_X);
            EXIT WHEN v_X = 10;
            v_X := v_X + 2 ;
        END LOOP;
END;
```

5.3. Phát biểu chọn theo nhiều chọn lựa

A. Dạng 1:

Xem cú pháp sau:

<Biến nhận Kq>:=

CASE <Giá trị ĐK>

When	<Biểu thức ĐK 1>	**Then**	Kq1
When	<Biểu thức ĐK 2>	**Then**	Kq2
When	<Biểu thức ĐK 3>	**Then**	Kq3
…			
Else	Kq khác		

END;

- **Xem các ví dụ case:**

Ví dụ 1: Cho giá trị n nhận 1,2 hay 3

```
Case n
        When    1    Then  'One'
        When    2    Then  'Two'
        When    3    Then  'Three'
        Else               'Other'
End;
```

- Xem lại cú pháp If:

```
if      n = 1    Then  'One';
Elsif   n = 2    Then  'Two';
Elsif   n = 3    Then  'Three';
Else               'Other';
End If;
```

- Nhận kết quả của CASE

```
KETQUA :=      Case  n
                  When   1     Then  'one'
                  When   2     Then  'two'
                  When   3     Then  'three'
                  Else                'other '
               End;
```

- Xem lại cú pháp If:

```
if      n = 1  Then      KETQUA := 'one';
Elsif   n = 2  Then      KETQUA := 'two';
Elsif   n = 3  Then      KETQUA := 'three';
Else                     KETQUA := 'other';
End If;
```

Ví dụ 2: Xét phái (Nam hoặc Nữ)

```
DECLARE
      PHAI      BOOLEAN := TRUE;
      KQ        CHAR(5);
BEGIN
      KQ:= CASE   PHAI
                  WHEN        TRUE          THEN  'NAM'
                  WHEN        FALSE         THEN  'NỮ'
                  ELSE                            'KHAC'
            END;
      DBMS_OUTPUT.PUT_LINE(KQ);
END;
```

Phái = True -> Nam
Còn lại là: Nữ

B. Dạng 2:

<Biến nhận Kq>:=

```
Case
    When    <Biểu thức ĐK 1>   Then  Kq1
    When    <Biểu thức ĐK 2>   Then  Kq2
    When    <Biểu thức ĐK 3>   Then  Kq3
    …
    Else    Kq khác;
End;
```

Ví dụ 1: Cho giá trị n có thể nhận 1, 2, 3 hay >3 và <8

```
Case
    When  n = 1                    Then  'One'
    When  n = 2                    Then  'Two'
    When  n = 3                    Then  'Three'
    When ( n > 3 And n < 8 )       Then  'Từ 4->7'
    Else                                 'Other'
End;
```

- Nhận kết quả như sau:

```
KETQUA:= Case
        When  n = 1     Then     'One'
        When  n = 2     Then     'Two'
        When  n = 3     Then     'Three'
        When ( n > 3 And n < 8 ) Then  'Từ 4->7'
        Else                           'Other'
End;
```

Ví dụ 2: Xét ví dụ sử dụng case trong câu truy vấn

Tạo bảng nhân viên với các thuộc tính sau:

```
CREATE TABLE NHANVIEN
(
    MANV        CHAR(10)     NOT NULL,
    HONV        NCHAR(10)    NOT NULL,
    TENLOT      NCHAR(10)    NOT NULL,
    TENNV       NCHAR(15)    NOT NULL,
    NGAYSINH    DATE         NOT NULL,
    PHAI        INT          NOT NULL,
    LUONG       FLOAT        NULL,
    PHG         INT          NOT NULL,
    CONSTRAINT  PK_NHANVIEN PRIMARY KEY(MANV)
);
```

Xét ví dụ sử dụng case trong câu truy vấn

Nhập liệu:

```
INSERT INTO NHANVIEN
VALUES('NV01','NGUYEN‘,'VAN','AN', '10-OCT-1980', 1, 500, 1);
INSERT INTO NHANVIEN
VALUES('NV02','NGUYEN','THI','BINH','05-SEP-1985',0, 400, 1);
INSERT INTO NHANVIEN
VALUES('NV03', 'TRAN', 'THI', 'SAU', '02-MAR-1988', 0, 200, 2);
```

Xét ví dụ sử dụng case trong câu truy vấn

Sử dụng CASE:

```
SELECT    MANV, RTRIM(HONV) ||' ' || RTRIM(TENLOT) || ' '
                  || RTRIM(TENNV) AS  "HO VA TEN",
                --CASE xét phái: 1 là NAM, 0 là NỮ
                CASE
                        WHEN  PHAI=1  THEN  'NAM'
                        WHEN  PHAI=0  THEN  'NỮ'
                END  AS  "PHAI" ,
                --CASE tăng lương phòng 1 lên 10$, phòng 2 lên 5$
                CASE  PHG
                        WHEN 1  THEN   LUONG +10
                        WHEN 2  THEN   LUONG +5
                END  AS  "LUONG + THUONG"
FROM NHANVIEN;
```

Kết quả của ví dụ 2 như sau:

Chưa khi sử dụng **CASE**:

MANV	HONV	TENLOT	TENNV	NGAYSINH	PHAI	LUONG	PHG
NV01	NGUYEN	VAN	AN	10-OCT-80	1	500	1
NV02	NGUYEN	THI	BINH	05-SEP-85	0	400	1
NV03	TRAN	THI	SAU	02-MAR-88	0	200	2

3 rows selected.

Chưa sử dụng CASE

Chưa sử dụng CASE

Sau khi sử dụng **CASE**

MANV	HO VA TEN	PHAI	LUONG + THUONG
NV01	NGUYEN VAN AN	NAM	510
NV02	NGUYEN THI BINH	NU	410
NV03	TRAN THI SAU	NU	205

3 rows selected.

Sử dụng CASE

Sử dụng CASE

C. Mở rộng của dạng 2 với nhiều biến làm điều kiện chọn

```
...
p:=1; q:=2; r:=2;
CASE
        When p = 1  Then  'Action1'
        When r = 2  Then  'Action2'
        When q > 1  Then  'Action3'
        Else              'Action Others'
End;
    EXCEPTION –Xảy ra lỗi
    WHEN  case_not_found   THEN
    Dbms_Output.Put_Line ( 'Bẫy lỗi xảy ra:
                        các trường hợp không hợp lệ');
  ...
```

- Một số lưu ý khi sử dụng **CASE**
  - ✓ Nếu trong case có nhiều điều kiện đúng thì kết quả nhận được sẽ ưu tiên điều kiện đầu thỏa.
  - ✓ Case dùng cho trường hợp, dùng biến kết quả thì ta không được dùng từ khóa **CASE** sau từ khóa **END;** là dấu hiệu kết thúc của **CASE**.
  - ✓ Còn Case dùng cho trường hợp không dùng biến nhận kết quả thì ta phải có từ khóa **CASE** đứng sau từ khóa **END;** là dấu hiệu kết thúc của **CASE**.
  - ✓ Trong câu truy vấn có sử dụng CASE thì sau từ khóa END không có dấu chấm phẩy (;) và không có từ khóa CASE.
  - ✓ Nếu trong case có nhiều điều kiện đúng thì kết nhận được phụ thuộc vào:
    - Ta sử dụng hàm xuất kết quả.
    - Hay dùng biến nhận kết quả.
    - Nhưng vẫn ưu tiên điều kiện nào thỏa trước thì nhận hay xuất kết quả.

Ví dụ 1:Cấu trúc CASE có sử dụng nhãn *(This CASE statement is labeled) , không có biến nhận kết quả, nên sau từ khóa END phải có từ khóa CASE*

```
DECLARE
    v_TestVar NUMBER := 1;
BEGIN
    <<MyCase>>
    CASE  v_TestVar
       WHEN 1 THEN DBMS_OUTPUT.PUT_LINE('One!');
       WHEN 2 THEN DBMS_OUTPUT.PUT_LINE('Two!');
       WHEN 3 THEN DBMS_OUTPUT.PUT_LINE('Three!');
       WHEN 4 THEN DBMS_OUTPUT.PUT_LINE('Four!');
    END CASE MyCase;
END;
```

Sau từ khóa END phải có từ khóa CASE

Ví dụ 2: Có biến nhận kết quả, nên sau từ khóa END không được có từ khóa **CASE**.

```
DECLARE
    P       INT:=1;
    Q       INT:=2;
    R       INT:=2;
    KETQUA  CHAR(20);
BEGIN
        KETQUA:= CASE
                    WHEN P=1  THEN  'ACTION1'
                    WHEN R=2  THEN  'ACTION2'
                    WHEN Q>1  THEN  'ACTION3'
                    ELSE            'ACTION OTHERS'
                 END;
        DBMS_OUTPUT.PUT_LINE(KETQUA);
END;
```

Sau từ khóa END không được có từ khóa CASE

BÀI TẬP

1. Xuất thông báo trên màn hình Output của SQL*Plus là “Xin chao ban den voi Oracle 9I”
2. Giải phương trình bậc 1
3. Giải phương trình bậc 2
4. Tìm USCLN và BSCNN của 2 số nguyên dương A và B (Với A và B là 2 số nguyên cho trước)
5. Liệt kê các số nguyên tố có giá trị nhỏ hơn hay bằng N (Với N là số nguyên dương cho trước)
6. Liệt kê các số hoàn thiện có giá trị nhỏ hơn N (Với N là số nguyên dương cho trước)
7. Tìm số ngày trong tháng của năm (Với tháng và năm là 2 số nguyên cho trước)

Chúc Bạn Thành Công