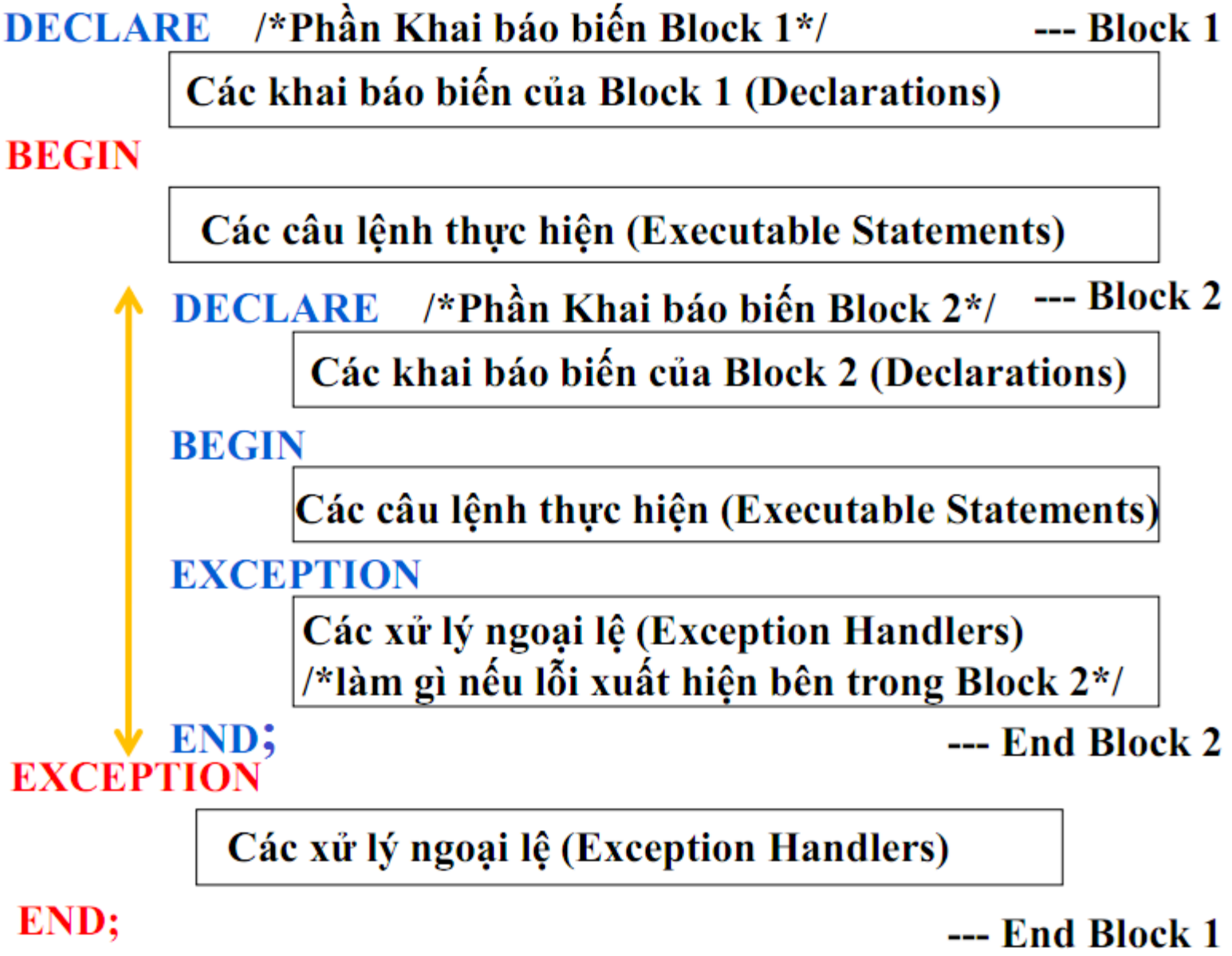
NGÔN NGỮ THỦ TỤC PL/SQL

Biên soạn: Nguyễn Việt Hưng
Bộ môn: Khoa Học Máy Tính - Khoa Công Nghệ Thông Tin
Trường Đại Học Giao Thông Vân Tải
Website: https://sites.google.com/site/viethung92gtvt/oracle
Email: viethung92gtvt@gmail.com

1/ GIỚI THIỆU PL/SQL

- PL/SQL là sự kết hợp giữa SQL và các cấu trúc điều khiển, các thủ tục (function), thao tác con trỏ (cursor), xử lý ngoại lệ (exception) và các lệnh giao tác.
- Ngôn ngữ thủ tục PL/SQL (Procedural Language/SQL) của Oracle được dùng để xây dựng các ứng dụng.
- PL/SQL cho phép sử dụng tất cả lệnh thao tác dữ liệu gồm INSERT, DELETE, UPDATE và SELECT, COMMIT, ROLLBACK, SAVEPOINT, cấu trúc điều khiển như vòng lặp (for, while, loop), rẽ nhánh (if),…mà với SQL chúng ta không làm được.

2/ CẤU TRÚC PL/SQL



2/ CẤU TRÚC PL/SQL

Ví dụ: In ra màn hình dòng chữ “Hello, World!”

```
DECLARE
 message varchar2(20):= 'Hello,
World!';
BEGIN
 dbms_output.put_line(message);
END;
```

3/ KHAI BÁO BIẾN VÀ HẰNG

❖ **Khai báo biến:**

mucluong NUMBER(5);

❖ **Khai báo hằng:**

heso CONSTANT NUMBER(3,2) := 1.86;

Ghi chú: Ký hiệu **:=** được sử dụng như là toán tử gán.

❖ **Gán biến và biểu thức:**

biến := biểu thức;

Ví dụ:

x:=UPPER('Nguyen'); y:=100; mucluong:= mucluong + mucluong*10/100;

Ví dụ: kq BOOLEAN; kq:= mucluong>3500000;

Độ ưu tiên của toán tử: ** (phép lũy thừa), NOT, *, /, +, -, || (phép nối chuỗi), =, !=, <>, <=, >=, IS NULL, LIKE, BETWEEN, IN, AND, OR.

❖ XUẤT/NHẬP TRONG PL/SQL

❖ LỆNH XUẤT MỘT NỘI DUNG LÊN MÀN HÌNH:

Cú pháp: DBMS_OUTPUT.PUT_LINE ('Nội dung');

Ví dụ:

```
declare
x number(6) := 25;
begin
dbms_output.put_line('Gia tri x la: ' || x);
end;
```

Kết quả khi chạy : **Gia tri x la: 25**

❖ XUẤT/NHẬP TRONG PL/SQL

❖ LỆNH NHẬP MỘT GIÁ TRỊ CHO 1 BIẾN

Biến thay thế &: dấu & đặt trước tên biến. Biến được nhập giá trị lúc thực thi câu SQL.

Lưu ý: biến kiểu chuỗi, kiểu ngày đặt trong cặp dấu nháy đơn ‘ ’

Ví dụ:

```
DECLARE
  x number;
BEGIN
  x := &x;
  dbms_output.put_line('Gia tri x = ' || x);
END;
```

3/ KHAI BÁO BIẾN VÀ HẰNG

VD:

```
DECLARE
 a integer := 10;
 b integer := 20;
 c integer;
 f real;
BEGIN
 c := a + b;
 dbms_output.put_line('Value of c: ' || c);
 f := 70.0/3.0;
 dbms_output.put_line('Value of f: ' || f);
END;
```

3/ KHAI BÁO BIẾN VÀ HẰNG

❖ **(Các thuộc tính %TYPE và %ROWTYPE)**

- **Thuộc tính %TYPE**

Dùng để khai báo một biến mà nó tham chiếu đến một cột trong cơ sở dữ liệu. (có cấu trúc như một cột trong Table).

Ví dụ: khai báo biến v_Manv có cùng kiểu dữ liệu với cột Manv trong bảng NHANVIEN

v_Manv NHANVIEN.Manv%TYPE;

Khai báo có điểm thuận lợi là: kiểu dữ liệu chính xác của biến v_Manv không cần được biết, nếu định nghĩa của cột Manv trong bảng NHANVIEN bị thay đổi thì kiểu dữ liệu của biến v_Manv thay đổi tương ứng.

3/ KHAI BÁO BIẾN VÀ HẰNG

❖ **(Các thuộc tính %TYPE và %ROWTYPE)**

- **Thuộc tính %ROWTYPE**

Dùng để khai báo một biến mà nó tham chiếu đến một dòng trong cơ sở dữ liệu (Có cấu trúc như một dòng trong Table).

Ví dụ: khai báo biến **v_nv** có kiểu dữ liệu là một dòng trong bảng **NHANVIEN**

v_nv NHANVIEN%ROWTYPE;

Khi truy xuất đến từng cột ta sử dụng giống như một bảng dữ liệu (trong trường hợp này chỉ gồm 1 record) tham chiếu đến một cột.

Cú pháp: Tên-biến.Tên-cột

Ví dụ: v_nv.HoTen

Gán giá trị trả về của câu lệnh truy vấn cho các biến:

Sử dụng mệnh đề SELECT INTO của SQL để gán giá trị cho các biến. Với mỗi trường giá trị trả về trong SELECT phải có một biến cùng kiểu dữ liệu với trường đó trong INTO.

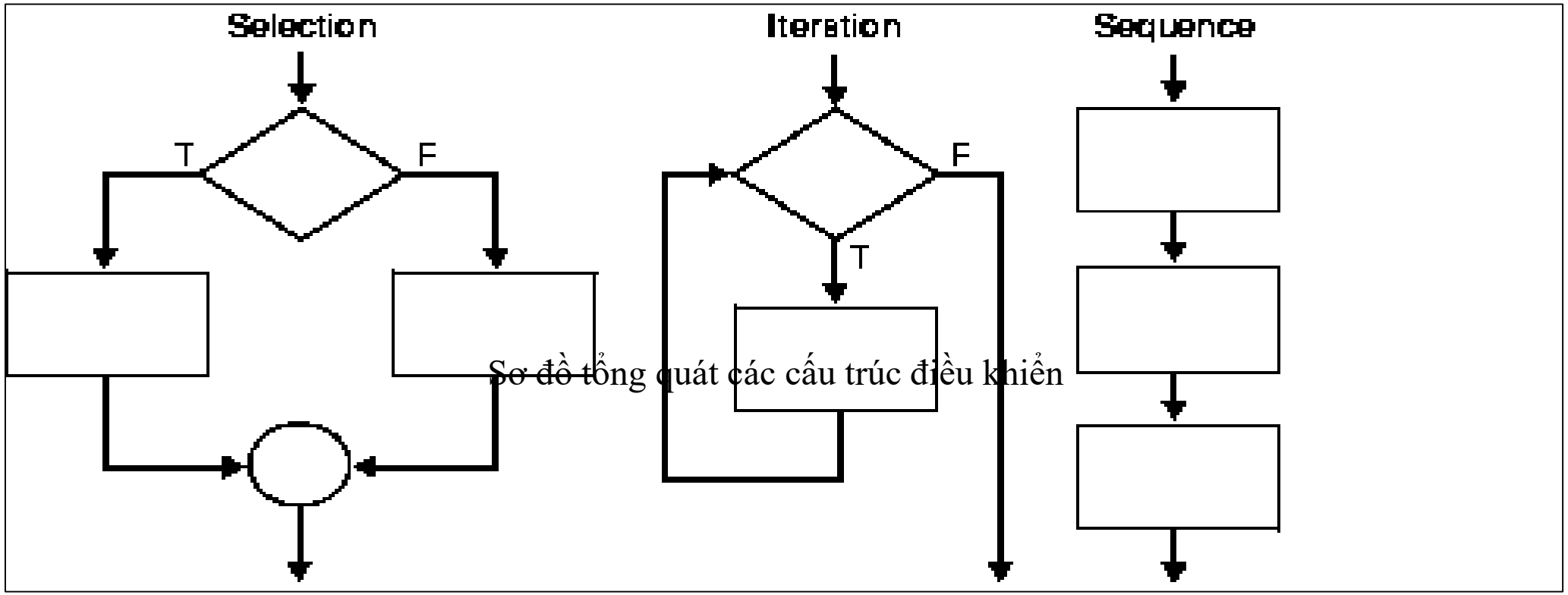
VD:

```
DECLARE
 v_empno integer := 7788;
 v_ename emp.ename%type;/*Khai báo biến v_ename có kiểu dữ liệu giống của cột ename
trong bảng emp.*/
 v_hiredate emp.hiredate%type;
 v_sal emp.sal%type;
BEGIN
 SELECT ename,hiredate,sal INTO v_ename, v_hiredate, v_sal
 FROM emp WHERE empno = v_empno;

 dbms_output.put_line
 ('Employer ' ||v_ename || ' - Hiredate: ' || v_hiredate || ' - Sal: ' || v_sal);
END;
```

Kết quả: `Employer SCOTT - Hiredate: 19-APR-87 - Sal: 3000`

5/ CÁC CẤU TRÚC ĐIỀU KHIỂN PL/SQL



Sơ đồ tổng quát các cấu trúc điều khiển

Sơ đồ tổng quát các cấu trúc điều khiển

5/ CÁC CẤU TRÚC ĐIỀU KHIỂN PL/SQL

5.1/ Cấu trúc lệnh rẽ nhánh – IF .. THEN.. END IF

Cú pháp 1:

```
IF <điều kiện 1> THEN
  khối lệnh 1;
ELSE
  IF <điều kiện 2> THEN
    khối lệnh 2;
  END IF;
ELSE
    …..;
END IF;
```

Ví dụ cú pháp 1:

```
DECLARE
 NO INTEGER(2) := 14;
BEGIN
 IF ( NO MOD 2 = 0 ) THEN
 DBMS_OUTPUT.PUT_LINE(NO || ' IS EVEN');
 ELSE
 DBMS_OUTPUT.PUT_LINE(NO || ' IS ODD');
 END IF;
END;
----
```

5/ CÁC CẤU TRÚC ĐIỀU KHIỂN PL/SQL

5.1/ Cấu trúc lệnh rẽ nhánh – IF .. THEN.. END IF

Cú pháp 2:

```
IF <điều kiện 1> THEN
  khối lệnh 1;
ELSIF <điều kiện2>
THEN
  khối lệnh 2;
ELSIF <điều kiện 3>
THEN
  khối lệnh 3;
ELSIF <điều kiện n>
THEN
  khối lệnh n;
END IF;
```

Ví dụ cú pháp 2:

```
DECLARE
 grade char(1) := 'A';
BEGIN
 IF grade = 'A' THEN
 DBMS_OUTPUT.PUT_LINE('Excellent');
 ELSIF grade = 'B' THEN
 DBMS_OUTPUT.PUT_LINE('Very Good');
 ELSIF grade = 'C' THEN
 DBMS_OUTPUT.PUT_LINE('Good');
 ELSE
 DBMS_OUTPUT.PUT_LINE('no such grade');
 END IF;
END;
```

5/ CÁC CẤU TRÚC ĐIỀU KHIỂN PL/SQL

5.2/ Cấu trúc lặp LOOP không định trước

Trong lệnh lặp này, số lần lặp tuỳ thuộc vào điều kiện kết thúc vòng lặp và không xác định được ngay tại thời điểm bắt đầu vòng lặp.

Cú pháp:

```
LOOP
Công việc;
EXIT WHEN điều kiện;
END LOOP;
```

Hoặc:

```
LOOP
Công việc;
IF <Điều kiện thoát>
THEN EXIT;
END LOOP;
```

Ví dụ:

```
DECLARE
  z  number :=1; /*khởi tạo biến z*/
BEGIN
    LOOP
        z :=z+3; /*tính biểu thức lặp*/
        dbms_output.put_line(z);
        IF  (z>=100) THEN   /*nếu  thỏa điều kiện thoát
khỏi vòng lặp*/
       exit;
         END IF;
 END LOOP;
END;
```

5/ CÁC CẤU TRÚC ĐIỀU KHIỂN PL/SQL

5.3/ Cấu trúc lặp – WHILE … LOOP

Cú pháp:

```
WHILE <điều kiện>
LOOP
  <khối lệnh>
END LOOP;
```

Ví dụ:

```
DECLARE
  z  number:=1; /*khởi tạo biến z*/
  i   number:=1; /*khởi tạo biến i*/
BEGIN
     WHILE (i<=10)  LOOP
      i:=i+1;
      z :=z+3; /*tính biểu thức lặp*/
      dbms_output.put_line(z);
          END LOOP;
END;
```

5/ CÁC CẤU TRÚC ĐIỀU KHIỂN PL/SQL

5.3/ Cấu trúc lặp có định trước - FOR … LOOP

Cú pháp:

```
FOR <biến_chạy> IN <giá_trị_khởi_tạo .. Giá_trị_kết_thúc> LOOP
   <khối lệnh>
 END LOOP;
```

Ví dụ:

```
DECLARE
  z  number:=1; /*khởi tạo biến z*/
  i   number;
BEGIN
   FOR i IN 1 .. 10 LOOP
      z :=z+3; /*tính biểu thức lặp*/
       END LOOP;
END;
```

6/ XỬ LÝ CÁC NGOẠI LỆ (EXCEPTION)

Khi một lỗi phát sinh, một ngoại lệ được đưa ra, việc thực hiện chương trình bình thường được dừng lại và điều khiển được chuyển tới ***khối PL/SQL chứa phần xử lý ngoại lệ.***

❖ **Có 2 dạng ngoại lệ (exception) :**

- **Ngoại lệ không tường minh (implicit):** là những ngoại lệ bên trong được sinh ra một cách tiềm ẩn VD: Nếu chia một số cho zero, một ngoại lệ do Oracle định nghĩa trước (ví dụ: ZERO_DIVIDE) sẽ tự động sinh ra.
- **Ngoại lệ tường minh (explicit):** là ngoại lệ do người dùng định nghĩa bằng cách sử dụng câu lệnh **RAISE**

6/ XỬ LÝ CÁC NGOẠI LỆ (EXCEPTION)

6.2/ Ngoại lệ không tường minh (IMPLICIT) – Exception của Oracle

CURSOR_ALREADY_OPEN	Mở một cursor, mà cursor đó đã ở trạng thái đang mở.
DUP_VAL_ON_INDEX	Khi có thao tác INSERT UPDATE vi phạm ràng buộc UNIQUE .
INVALID_CURSOR	Mở cursor chưa tạo,đóng một cursor mà nó chưa được mở.
INVALID_NUMBER	Lỗi chuyển kiểu dữ liệu từ string sang kiểu number.
LOGIN_DENIED	Đăng nhập sai username/password.
NO_DATA_FOUND	Câu lệnh SELECT INTO không trả về dòng nào.
NOT_LOGGED_ON	Một chương trình PL/SQL cần thao tác đến Cơ sở dữ liệu Oracle nhưng lại chưa đăng nhập vào Cơ sở dữ liệu.
PROGRAM_ERROR	Một số lỗi chương trình, ví dụ một hàm (function) không chứa mệnh đề RETURN để trả về giá trị.
STORAGE_ERROR	Lỗi bộ nhớ
TIMEOUT_ON_RESOURCE	Lỗi timeout xảy ra khi Oracle đang chờ tài nguyên.
TOO_MANY_ROWS	Câu lệnh SELECT INTO trả về nhiều hơn một dòng.
VALUE_ERROR	Lỗi chuyển kiểu dữ liệu hoặc thao tác vi phạm RBTV.
ZERO_DIVIDE	Lỗi chia một số cho zero.

6/ XỬ LÝ CÁC NGOẠI LỆ (EXCEPTION)

Ví dụ 1: Nhập 2 số từ bàn phím, tính tổng, hiệu, tích, thương 2 số đó.

```
declare
 a number;
 b number;
begin
 a := &number1;
 b := &number2;
 dbms_output.put_line(a || '+' || b || '=' || (a + b));
 dbms_output.put_line(a || '-' || b || '=' || (a - b));
 dbms_output.put_line(a || '*' || b || '=' || (a * b));
 dbms_output.put_line(a || '/' || b || '=' || (a / b));
exception
 when zero_divide then
 dbms_output.put_line('Error divide by 0');
end;
```

7/ CON TRỎ (CURSOR)

7.1/ Giới thiệu Cursor

❖ Con trỏ (cursor) là một đối tượng liên kết với một tập dữ liệu và cho phép người lập trình làm việc với từng dòng của tập dữ liệu đó.

❖ Để xử lý một câu SQL, PL/SQL mở một vùng làm việc có tên là vùng ngữ cảnh (context area). PL/SQL sử dụng vùng này để thi hành câu SQL và chứa kết quả trả về. Vùng ngữ cảnh đó là phạm vi hoạt động của con trỏ.

❖ **Có hai loại con trỏ:**

- con trỏ được khai báo tường minh (explicit cursor)
- con trỏ không được khai báo tường minh (implicit cursor) (hay còn gọi là con trỏ tiềm ẩn).

7/ CON TRỎ (CURSOR)

- **Con trỏ tiềm ẩn:** một lệnh SQL được xử lý bởi Oracle và không được đặt tên bởi người sử dụng. Các lệnh SQL được thực hiện trong một con trỏ tiềm ẩn bao gồm SELECT .. INTO, UPDATE, INSERT, DELETE.

- **Có bốn thuộc tính:**

- SQL%NOTFOUND: kết quả trả về là TRUE nếu câu lệnh SQL không tìm thấy dữ liệu
- SQL%FOUND: kết quả trả về là TRUE nếu câu lệnh SQL tìm thấy dữ liệu
- SQL%ROWCOUNT: kết quả trả về là số dòng dữ liệu mà câu lệnh SQL tìm thấy
- SQL%ISOPEN: kết quả trả về là TRUE nếu con trỏ đang ở trạng thái mở

Trước khi thi hành câu SQL, các thuộc tính của con trỏ tiềm ẩn có giá trị NULL.

7/ CON TRỎ (CURSOR)

Ví dụ 1: ***thuộc tính %NOTFOUND***

```
DELETE FROM emp WHERE empno='222';
IF SQL%NOTFOUND THEN
   DBMS_OUTPUT.PUT_LINE ('Ko co nhan vien 222');
END IF;
```

Ví dụ 2: ***thuộc tính %FOUND***

```
SELECT empno into v_eno  FROM EMP WHERE empno=7788;
IF SQL%FOUND THEN
  DELETE FROM EMP WHERE empno=7788;
END IF;
```

Ví dụ 3: ***thuộc tính %R*** UPDATE EMP SET SAL=5000 WHERE empno=7788;

```
  IF SQL%ROWCOUNT >0 THEN
    DBMS_OUTPUT.PUT_LINE (‘Luong moi’);
  END IF; OWCOUNT
```

7/ CON TRỎ (CURSOR)

Ví dụ 4: Thủ tục tăng lương một nhân viên có mã truyền vào từ tham số.

```
CREATE OR REPLACE Procedure Tang_Luong(manv number) As
  old_luong number;
  new_luong number;
Begin
  select sal into old_luong from emp where empno=manv;
  if SQL%FOUND then
       new_luong:=old_luong+old_luong*10/100;
       update emp set sal=new_luong where empno=manv;
       if SQL%ROWCOUNT<>0 then
    DBMS_OUTPUT.PUT_LINE('Luong  NV ' || manv ||' duoc tang 10%');
       end if;
  end if;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Khong  tim thay nhan vien ' || manv);
END;
```

7/ CON TRỎ (CURSOR)

❑ Con trỏ tường minh

Con trỏ tường minh: là con trỏ được đặt tên bởi người sử dụng (câu SELECT được đặt tên).

Cú pháp: CURSOR tên-cursor IS câu-lệnh-SELECT;

- Trong đó, câu lệnh SELECT phải chỉ ra các cột cụ thể cần lấy cho con trỏ này.
- Phần khai báo này phải được đặt trong vùng khai báo biến (trước BEGIN của khối (Block)).
- - Trong ngôn ngữ thủ tục PLSQL, để xử lý dữ liệu lưu trong cơ sở dữ liệu, đầu tiên dữ liệu cần được ghi vào các biến. Giá trị trong biến có thể được thao tác. Dữ liệu các bảng không thể được tham khảo trực tiếp.

Ví dụ: *cursor c_nv is select empno,sal from emp;*

7/ CON TRỎ (CURSOR)

❑ **Con trỏ tường minh**

➢ Thao tác trên con trỏ: **OPEN, FETCH, CLOSE**

➢ **Cú pháp:**

- OPEN tên-cursor; /*Mở con trỏ thi hành câu truy vấn*/
- FETCH tên-cursor INTO biến1, biến2, …, biếnn;

hoặc

- FETCH tên-cursor INTO biến_có_kiểu_record;
- /*Lệnh FETCH dùng để gọi một dòng trong tập dữ liệu của con trỏ, có thể được lặp để gọi tất cả các dòng của con trỏ*/.
- CLOSE tên-cursor /*đóng con trỏ, giải phóng khỏi bộ nhớ*/

➢ Mọi con trỏ khai báo tường minh đều có bốn thuộc tính:
%NOTFOUND, %FOUND, %ROWCOUNT, %ISOPEN

➢ Các thuộc tính này được thêm vào sau phần tên của con trỏ.

7/ CON TRỎ (CURSOR)

❑ Con trỏ tường minh

a) Thuộc tính %NOTFOUND (đi kèm lệnh Fetch)

Mang giá trị TRUE hoặc FALSE. %NOTFOUND bằng TRUE khi đã fetch đến dòng cuối cùng của con trỏ, ngược lại, bằng FALSE khi lệnh fetch trả về ít nhất một dòng hoặc chưa fetch đến dòng cuối cùng.

Ví dụ:

```
OPEN cur_first;
    LOOP
      FETCH cur_first INTO v_empno,v_sal;
      EXIT WHEN cur_first%NOTFOUND;
    END LOOP;
```

7/ CON TRỎ (CURSOR)

- **Con trỏ tường minh**

b) Thuộc tính %FOUND (đi kèm lệnh Fetch)

Ngược với thuộc tính %NOTFOUND.

Ví dụ:

```
OPEN cur_first;
LOOP
   FETCH cur_first INTO v_empno,v_sal;
  IF cur_first%FOUND THEN
    ……….
  ELSE
    CLOSE cur_first;
    EXIT;
  END IF;
END LOOP;
```

7/ CON TRỎ (CURSOR)

❑ Con trỏ tường minh

c) Thuộc tính %ROWCOUNT (đi kèm lệnh Fetch)

Trả về số dòng con trỏ đã được FETCH.

Ví dụ:

```
OPEN cur_first;
LOOP
   FETCH cur_first INTO v_empno,v_sal;
  IF cur_first%ROWCOUNT=1000 THEN
    EXIT;
  END IF;
END LOOP;
```

7/ CON TRỎ (CURSOR)

❖ **Duyệt con trỏ tường minh sử dụng câu lệnh FOR .. LOOP**

Ví dụ:

```
declare
cursor c_emp is select * from emp;/*Khai báo 1 con trỏ trả về tất cả các bản ghi của bảng emp*/
v_emp c_emp%rowtype; /*Khai báo 1 biến có kiểu dữ liệu là từng record của con trỏ c_emp*/
begin
for v_emp in c_emp loop/*Duyệt từng bản ghi trong con trỏ c_emp và lưu vào biến v_emp*/
dbms_output.put_line('Ma NV: ' || v_emp.empno || ' Ten NV: ' || v_emp.ename);
/*In mã và tên của từng nhân viên duyệt được*/
end loop;
end;
```

8/ HÀM, THỦ TỤC

8.1/ HÀM (Function)

Hàm là một chương trình con có trả về giá trị. Hàm và thủ tục giống nhau, chỉ khác nhau ở chỗ hàm thì có mệnh đề RETURN.

Cú pháp:

```
CREATE [OR REPLACE] FUNCTION  tên-hàm [(argument1
[, argument2,…])] RETURN datatype
IS
  [khai báo biến]
BEGIN
  <khối lệnh>
[EXCEPTION <xử lý ngoại lệ>]
END; /*kết thúc hàm*/
```

8/ HÀM, THỦ TỤC

8.1/ HÀM (Function)

- **Datatype** có thể là Number, Char hoặc Varchar2,….
- Từ khóa **OR REPLACE** để tự động xóa và tạo mới hàm nếu tên hàm đó đã tồn tại.

Ví dụ:

CREATE OR REPLACE Hien_Thi_Ngay (m number)
RETURN VARCHAR IS ….

- **Argument được thay bởi:**

tên-đối-số-truyền-vào [IN | OUT | IN OUT] kiểu-dữ-liệu [{ := | DEFAULT value}]

8/ HÀM, THỦ TỤC

8.1/ HÀM (Function)

Ví dụ:

```
CREATE FUNCTION Hien_Thi_Ngay
(n NUMBER) RETURN CHAR
IS
ngay CHAR(15);
BEGIN
  IF n =1 THEN
    ngay :='Sunday';
  ELSIF n =2 THEN
    ngay :='Monday';
  ELSIF n =3 THEN
    ngay :='Tuesday';
  ELSIF n =4 THEN
    ngay :='Wednesday';
  ELSIF n =5 THEN
    ngay :='Thursday';
  ELSIF n =6 THEN
    ngay :='Friday';
  ELSIF n =7 THEN
    ngay :='Saturday';
  END IF;
  RETURN ngay;
END;
```

8/ HÀM, THỦ TỤC

8.1/ HÀM (Function)

- **Gọi hàm trong PL/SQL:**

Đầu tiên khai báo biến có kiểu dữ liệu trùng với kiểu dữ liệu trị trả về của một hàm.

Thực hiện lệnh sau:

Ví dụ:

```
Declare
  x   CHAR(20);
BEGIN
  x:=Hien_Thi_Ngay(3);
  /*Tổng quát:    biến:=Tên-hàm(danh sách đối số);*/
    ….
END;
```

- **Lệnh xóa hàm:** DROP FUNCTION tên-hàm;

8/ HÀM, THỦ TỤC

8.2/ Thủ tục (PROCEDURE)

Thủ tục là một chương trình con để thực hiện một hành động cụ thể nào đó. Hàm và thủ tục giống nhau, khác nhau ở chỗ hàm thì có mệnh đề RETURN.

Cú pháp:

```
  CREATE [OR REPLACE] PROCEDURE  tên-thủ tục
[(parameter1 [, parameter2,…])]  IS
    [khai báo biến]
  BEGIN
    <khối lệnh>a
  [EXCEPTION <xử lý ngoại lệ>]
  END; /*kết thúc thủ tục*/
```

8/ HÀM, THỦ TỤC

8.2/ Thủ tục (PROCEDURE)

Ví dụ: Thủ tục thêm 1 bản ghi vào bảng EMP

```
CREATE OR REPLACE Procedure Insert_EMP   (v_EMPNO in varchar2, v_ENAME in varchar2,
v_HIREDATE in date, v_MGR in varchar2, v_SAL in varchar2)
As
        emp_cnt int;
Begin
        select count(*) into emp_cnt from EMP where EMPNO = v_EMPNO;
        if ( emp_cnt=1) then
                DBMS_Output.Put_line('Trung khoa chinh');/*tru`ng khoa chinh */
        else
                savepoint Point_1;
                insert into EMP (EMPNO, ENAME,HIREDATE, MGR, SAL) values (v_EMPNO,
v_ENAME,v_HIREDATE,v_MGR, v_SAL) ;
                if SQL%ROWCOUNT = 0 then
                    DBMS_Output.Put_line('Xay ra loi giao tac');  /*loi khac*/
                    ROLLBACK to savepoint Point_1;
                end if;
                DBMS_Output.Put_line('Them nhan vien thanh cong') ;
                COMMIT ;
        end if; End;
```

BÀI TẬP

1. Viết thủ tục giải phương trình bậc 2.
2. Viết thủ tục liệt kê các nhân viên vào làm việc tính từ ngày truyền vào từ tham số.
3. Liệt kê các cột ENAME, HIREDATE, SAL Với điều kiện EMPNO bằng giá trị biến

&EMPLOYEE_NO được đưa vào, sau đó kiểm tra:

a) Có phải mức lương lớn hơn 1200

b) Tên nhân viên có phải có chứa chữ T

c) ngày gia nhập cơ quan có phải là tháng 10 (DEC)

Hiển thị các kết quả lên màn hình.