



An-toan-va-bao-mat-he-thong-thong-tin lab-05 pl-sql-(2) - [cuuduongthancong]

Kỹ thuật xử lý SO₂ (Trường Đại học Bách khoa, Đại học Quốc gia Thành phố Hồ Chí Minh)

Bài thực hành số 5

PL/SQL (2)

❖ Tóm tắt nội dung:

- Xử lý ngoại lệ
- Procedure và Function
- Cursor
- Trigger

I. Xử lý ngoại lệ

1. Giới thiệu về Exception

- Các Exception là các danh định trong PL/SQL mà có thể gặp phải trong khi thực thi một khối dẫn đến thân chính của các tác vụ sẽ bị kết thúc. Một khối luôn luôn kết thúc khi gặp một exception, nhưng có thể chỉ ra một exception handler để thi hành tác vụ cuối cùng trước khi khối bị kết thúc. Nếu exception được kiểm soát (handled) thì exception sẽ không truyền ra ngoài khối hay ra môi trường. Hai nhóm chính của exception là :
 - ✓ Predefined: đã được định nghĩa trước bởi PL/SQL và đính với các mã lỗi xác định
 - ✓ User-defined: khai báo trong khối, chỉ thường dùng khi có nhu cầu cụ thể với chúng, ngoài ra có thể gắn chúng với các mã lỗi cần thiết.
- Trong bài này, chúng ta sẽ tập trung vào các exception đã định nghĩa trước:

Tên Exception	Lỗi Oracle
-----	-----
DUP_VAL_ON_INDEX	-1
INVALID_CURSOR	-1001
INVALID_NUMBER	-1722
LOGIN_DENIED	-1017
NO_DATA_FOUND	-1403 (ANSI +100)
NOT_LOGGED_ON	-1012

PROGRAM_ERROR	-6501
STORAGE_ERROR	-6500
TIMEOUT_ON_RESOURCE	-51
TOO_MANY_ROWS	-1422
VALUES_ERROR	-6502
ZERO_DIVIDE	-1476
CURSOR_ALREADY_OPEN	-6511
TRANSACTION_BACKED_OUT	-61

2. Bộ kiểm soát lỗi

- Nếu một exception xảy ra, quyền điều khiển sẽ chuyển cho phần EXCEPTION trong khối mà nó xảy ra. Nếu exception đó không kiểm soát được trong phần này hoặc là không có phần này thì khối sẽ kết thúc với exception unhandled và có thể tác động đến môi trường ngoài.

Ví dụ:

```
BEGIN
    INSERT INTO dept (deptno, dname)
        VALUES (50, 'CLEANING');
    INSERT INTO dept (deptno, dname)
        VALUES (50, 'TRANING');
    -- Exception DUP_VAL_ON_INDEX xảy ra tại đây
END;
-- Khối sẽ kết thúc với exception unhandled ORA-00001
```

- Để bắt các sự kiện này và chặn các exception, có thể định nghĩa các exception handler trong phần EXCEPTION.

Cú pháp:

WHEN exceptionn-identifier THEN actions;

Ví dụ :

```
DECLARE
    v_ename    emp.ename%TYPE;
    v_job emp.job%TYPE;
BEGIN
    SELECT      ename, job
    INTO  v_name, v_job
    FROM  emp
    WHERE hiredate BETWEEN '01/01/92' AND '31/12/92';
EXCEPTION
    WHEN no_data_found THEN
        INSERT INTO error_tab VALUES ('Nobody in 92');
    WHEN too_many_rows THEN
        INSERT INTO error_tab VALUES ('More than one
        person in 92');
END;
```

- Bộ kiểm soát lỗi 'WHEN OTHERS': có thể dùng định nghĩa này để chặn tất cả các exception còn lại ngoài các exception đã định nghĩa trong phần EXCEPTION. Phần này được đặt cuối cùng trong phần EXCEPTION.

Ví dụ:

```
BEGIN
    SAVEPOINT  so_far_so_good;
    INSERT INTO statistics_tab VALUES (18, 25, 91);
EXCEPTION
    WHEN dup_val_on_index THEN
        ROLLBACK TO so_far_so_good;
    WHEN OTHERS THEN
        INSERT INTO error_tab
        VALUES ('Error during block');
END;
```

3. Các hàm dùng trong bẫy lỗi

- Khi một exception xảy ra, ta có thể xác định mã lỗi và câu chú của nó. PL/SQL cung

cấp 2 hàm:

SQLCODE	Trả về mã lỗi của exception đó. Nếu dùng nó ngoài phần EXCEPTION thì mã trả ra là 0
SQLERRM	Trả về toàn bộ câu chú lỗi (error message) và có cả mã lỗi

Ví dụ:

```

DECLARE
    error_message    CHAR (100);
    error_code       NUMBER;

BEGIN
    ...

EXCEPTION
    WHEN OTHERS THEN
        error_message := SUBSTR (SQLERRM, 1, 100);
        error_code := SQLCODE;
        INSERT INTO error
        VALUES (error_message, error_code);
END;
```

II. Procedure

- *Cú pháp:*

```

CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] datatype )]
{IS | AS}
BEGIN
    procedure_body
END procedure_name;
```

- **Chú ý:** Datatype là kiểu của tham số, ở đây chỉ khai báo kiểu chứ ko khai báo chiều dài của tham số. Ví dụ không được khai báo tham số là VARCHAR2(10) mà phải khai báo là VARCHAR2.

Ví dụ:

```
CREATE OR REPLACE PROCEDURE update_product_price(  
  p_product_id IN products.product_id%TYPE,  
  p_factor IN NUMBER)  
AS  
  v_product_count INTEGER;  
BEGIN  
  SELECT COUNT(*)  
  INTO v_product_count  
  FROM products  
  WHERE product_id = p_product_id;  
  IF v_product_count = 1 THEN  
    UPDATE products  
    SET price = price * p_factor  
    WHERE product_id = p_product_id;  
    COMMIT;  
  END IF;  
EXCEPTION  
  WHEN OTHERS THEN ROLLBACK;  
END update_product_price;
```

- Vì procedure cần phải gọi trong khối PL/SQL, nên nếu muốn chạy nó từ dấu nhắc SQL*Plus ta dùng lệnh EXECUTE hoặc lồng nó trong cặp BEGIN-END.

Ví dụ :

```
SQL> EXECUTE update_product_price(1, 1.5);
```

Hay có thể

```
SQL> BEGIN
```

```
2   update_product_price(1, 1.5);
```

```
3   END;
```

III. Function

- *Cú pháp:*

```
CREATE [OR REPLACE] PROCEDURE procedure_name  
[(parameter_name [IN | OUT | IN OUT] datatype )]  
RETURN datatype  
{IS | AS}  
BEGIN  
    procedure_body  
END procedure_name;
```

Ví dụ:

```
create or replace function get_dname( y number)  
return varchar2  
is  
    m char(14);  
begin  
    select dname  
    into m  
    from dept  
    where deptno=y;  
    if SQL%notfound then  
        m:='Khong thay';  
    end if;  
    return (rtrim(m));  
end;
```

- Để gọi function ta gọi trực tiếp hoặc thông qua các phép gán.

Ví dụ:

```
SQL> select * from dept where dname=get_dname(10);
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK

```
SQL> select get_dname (20) from dual;
```

```
GET_DNAME (20)
```

```
-----
```

```
RESEARCH
```

IV. Cursor

1. Định nghĩa

- Oracle dùng các vùng làm việc gọi là ‘các vùng SQL dùng riêng’ (private SQL areas) để thi hành các câu lệnh SQL và lưu trữ thông tin của quá trình. Một cursor là một cấu trúc PL/SQL cho phép định danh các vùng này và truy cập đến các thông tin lưu trong nó. Có 2 kiểu cursor :

Implicit Cursors	Được mô tả bởi PL/SQL là ẩn dành cho tất cả các câu lệnh DML và cho các query trả ra đơn hàng (ví dụ lệnh SELECT dùng trực tiếp trong khối).
Explicit Cursors	Mô tả rõ ràng với các danh định trong khối và được thao tác bằng các câu lệnh đặc trưng trong các tác vụ của khối. Các cursor hiện chỉ dành cho các query và cho phép nhiều hàng được xử lý từ query.

2. Explicit cursor có thể điều khiển qua 4 kiểu tác vụ riêng lẻ sau :

DECLARE	Định tên của cursor và cấu trúc của query thực thi trong nó. Tại thời điểm này, query sẽ được phân tích (các cột, bảng, ...) nhưng chưa thi hành
OPEN	Thi hành query ràng buộc các biến có tham khảo đến. Các hàng trả về bởi query gọi là ‘active set’ và sẵn sàng cho việc lấy dữ liệu.
FETCH	Lấy dữ liệu từ hàng hiện tại vào các biến. Hàng hiện tại là hàng mà cursor đang chỉ đến. Mỗi một lần FETCH, cursor di chuyển con trỏ đến hàng kế tiếp trên active set, như vậy mỗi một lệnh FETCH sẽ truy cập đến các hàng khác nhau trong query.

CLOSE	Hủy bỏ tập các hàng đang làm việc được sinh ra bởi lệnh OPEN cuối cùng của cursor. Có thể OPEN lại được và như vậy sẽ có tập hàng làm việc mới hơn.
-------	---

3. Khai báo

- *Cú pháp:*

CURSOR identifier [(parameter details)] IS query-expression;

Ví dụ:

```
DECLARE
    CURSOR c1 IS
        SELECT last_name, salary, hire_date, job_id
        FROM employees
        WHERE employee_id = 120;
/*khai báo biến record để đại diện một hàng được fetch từ
bảng employees */
    employee_rec c1%ROWTYPE;
BEGIN
    -- mở cursor một cách tường minh
    -- sử dụng cursor này để fetch dữ liệu đổ vào employee_rec
    OPEN c1;
    FETCH c1 INTO employee_rec;
    DBMS_OUTPUT.PUT_LINE('Employee name: '
                          || employee_rec.last_name);
END;
```

4. Các thuộc tính của explicit cursor (Explicit Cursor Attributes)

- Giống như các implicit cursor, có 4 thuộc tính để biết các thông tin về cursor. Khi dùng, thì phải để tên cursor trước các thuộc tính này.

%FOUND	Có giá trị TRUE nếu lệnh FETCH gần nhất từ cursor lấy được 1 hàng từ active set, ngược lại sẽ là FALSE
%NOTFOUND	Ngược với %FOUND
%ROWCOUNT	Trả về số hàng đã FETCH được từ active set tính đến hiện tại
%ISOPEN	TRUE nếu cursor đang mở, FALSE nếu cursor đã đóng hoặc chưa được mở trong khối

Ví dụ :

```
IF c1%ISOPEN THEN
    FETCH c1 INTO v_ename, v_sal, v_hiredate;
ELSE
    OPEN c1;
END IF;
```

Ví dụ :

```
LOOP
    FETCH c1 INTO v_ename, v_sal, v_hiredate;
    EXIT WHEN c1%ROWCOUNT > 10;
END LOOP;
```

5. Điều khiển các việc lấy nhiều dữ liệu từ các explicit cursor

- Thường thì khi muốn xử lý nhiều hàng từ explicit cursor thì dùng một vòng lặp với lệnh FETCH tại mỗi bước lặp. Nếu quá trình tiếp tục thì tất cả các hàng trong active set sẽ được xử lý. Khi một lệnh FETCH không thành công xảy ra, thuộc tính %NOTFOUND sẽ là TRUE. Mặc dù vậy, nếu dùng lệnh FETCH kế tiếp thì sẽ xảy ra lỗi :

```
ORA-1002: Fetch out of sequence
```

- Lỗi này sẽ kết thúc khối thường là một *unhandled exception*. Vì thế cần thiết phải kiểm tra sự thành công của mỗi lần FETCH trước khi tiếp tục tham khảo cursor.

Ví dụ :

```
OPEN cursor_1;
LOOP
    FETCH cursor_1 INTO a, b, c, d;
    EXIT WHEN cursor_1%NOTFOUND;
    -- xử lý hàng hiện tại ở đây
END LOOP;
```

6. Mệnh đề FOR UPDATE OF

Ví dụ :

```
DECLARE
    CURSOR c1 IS
        SELECT empno, sal, hiredate, rowid
        FROM emp WHERE depno=20 AND job='ANALYST'
        FOR UPDATE OF sal;
    emp_record c1%ROWTYPE;
BEGIN
    OPEN c1;
    ...
    FETCH c1 INTO emp_record;
    ...
    IF emp_record.sal < 2000 THEN ...
    ...
END;
```

- Ví dụ trên dùng FOR UPDATE trong query của cursor. Nghĩa là các hàng trả về bởi query sẽ được khóa không cho ai khác truy xuất vào khi OPEN được dùng. Khi bỏ khóa tại cuối giao dịch, chúng ta không cần COMMIT.

7. Mệnh đề WHERE CURRENT OF

- Khi tham khảo 'current row' từ một explicit cursor, các lệnh SQL có thể dùng mệnh đề WHERE CURRENT OF. Nó cho phép cập nhật hay xóa bỏ tại hàng hiện tại.

Ví dụ :

```
FETCH c1 INTO emp_record;  
IF emp_record.ename = 'KING' THEN  
    DELETE FROM emp WHERE CURRENT OF c1;
```

V. Triggers

- Một Database Trigger được tạo và lưu trữ trong PL/SQL block tương ứng với table. Nó được tự động gọi đến khi có sự truy nhập đến table tương ứng với các hành động định nghĩa.

- *Cú pháp:*

```
CREATE [OR REPLACE] TRIGGER trigger_name  
BEFORE | AFTER  
UPDATE | DELETE | INSERT (OF column) ON TABLE  
(FOR EACH ROW (WHEN condition))  
BEGIN  
    PL/SQL block  
END trigger_name;
```

Ví dụ:

- ✓ Tạo bảng:

```
CREATE TABLE product_price_audit  
    (product_id INTEGER  
    CONSTRAINT price_audit_fk_products  
    REFERENCES products(product_id),  
    old_price NUMBER(5, 2),  
    new_price NUMBER(5, 2));
```

✓ Tạo Trigger

```
CREATE OR REPLACE TRIGGER before_product_price_update
BEFORE UPDATE OF price
ON products
FOR EACH ROW WHEN (new.price < old.price * 0.75)
BEGIN
    dbms_output.put_line('product_id = ' || :old.product_id);
    dbms_output.put_line('Old price = ' || :old.price);
    dbms_output.put_line('New price = ' || :new.price);
    dbms_output.put_line('The price reduction is more than
25%');
    -- insert row into the product_price_audit table
    INSERT INTO product_price_audit ( product_id, old_price,
new_price)
    VALUES (:old.product_id, :old.price, :new.price);
END before_product_price_update;
```

✓ Firing a Trigger: để thấy được output từ một trigger, bạn cần phải chạy câu lệnh:

```
SET SERVEROUTPUT ON
UPDATE products
SET price = price * .7
WHERE product_id IN (5, 10);

product_id = 10
Old price = 15.99
New price = 11.19
The price reduction is more than 25%
product_id = 5
Old price = 49.99
New price = 34.99
The price reduction is more than 25%
2 rows updated.
```

✓ Disable and Enable Trigger

Có thể cấm một trigger hoạt động và ngược lại bằng câu lệnh ALTER TRIGGER.

```
ALTER TRIGGER before_product_price_update DISABLE;
```

```
ALTER TRIGGER before_product_price_update ENABLE;
```

VI. Bài tập

- Các bài tập trong bài lab này có sử dụng đến bảng Message đã được mô tả trong bài lab 4.

1. Đoạn mã sau đây cần phải được thi hành trong một vòng lặp với các giá trị khác nhau của v tại mỗi bước lặp (tầm từ 1 đến 10).

```
UPDATE message SET numcol2 = 100
WHERE numcol1 = v;
```

Nếu bất kỳ quá trình UPDATE nào mà không có hoặc có nhiều hơn 1 hàng thì thoát khỏi vòng lặp (Có thể dùng SQL%ROWCOUNT để kiểm tra).

2. Sửa đổi khối bạn đã viết trong bài tập bài 2. Định nghĩa lại biến PL/SQL là NUMBER(1). Điều gì sẽ xảy ra nếu giá trị gán vào là 42. Thêm một bộ kiểm soát exception vào khối để lưu lại các câu chú giải thích trong MESSAGE cho bất kỳ kiểu exception xảy ra nào. Chạy khối một lần nữa.

3. Dùng explicit cursor và các thuộc tính của nó:

Cho bảng Dept gồm các thuộc tính ID phòng ban, tên phòng ban và địa điểm của phòng ban ở các chi nhánh khác nhau.

```
Dept ( ID, dname, loc)
```

Xử lý mỗi hàng của bảng 'Dept', di chuyển phòng SALES đến địa điểm Dallas và các phòng khác đến New York. Ngoài ra nó đếm số phòng ban tại mỗi địa điểm.

4. Tạo ra một file cript SQL*Plus chấp nhận một tham số đơn là kiểu nghề nghiệp lúc chạy chương trình :

Ví dụ : @UNIT3_FILE MANAGER

Trong khối PL/SQL, sẽ dùng lệnh SELECT lấy các hàng từ bảng 'emp' với điều kiện 'job' là tham số nhập vào (Tham khảo đến tham số bằng '&1'). Gửi một chú giải đến

bảng MESSAGE tùy vào việc có hàng, không hàng hoặc một vài hàng được trả về.

Ví dụ: 'Jobtype found once'

'Jobtype found more than once'

'Jobtype not found'

Lưu jobtype trong bảng MESSAGE và COMMIT giao dịch để chú giải được tạo ra.

cuu duong than cong . com

cuu duong than cong . com