Procedural Language/Structured Query Language (PL/SQL)

1. Giới thiệu PL/SQL (1)
- (Procedural Language/Structure Query Language)

- Ngôn ngữ thủ tục của Oracle, dùng để xây dựng các ứng dụng.
- 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.
- 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.

1. Giới thiệu PL/SQL (2)
- (Procedural Language/Structure Query Language)

- PL/SQL thêm chức năng vào các công cụ không thủ tục như SQL*Forms và SQL*Report.
- Các lệnh PL/SQL được chia thành nhiều khối lệnh hợp lý (Block), các khối lệnh lồng nhau. Các biến có thể khai báo nội tại (local) bên trong block và điều khiển báo lỗi (exception) được xử lý trong block nơi lỗi phát sinh.
- Một block bao gồm ba phần: phần khai báo là nơi để khai báo biến, phần thi hành lệnh và phần xử lý các ngoại lệ (điều kiện lỗi hoặc cảnh báo).
- Khai báo biến trong PROCEDURE hay FUNCTION: nếu là Block ngoài cùng (đầu tiên) của PROCEDURE, FUNCTION thì không dùng từ khóa DECLARE (Ngược lại với TRIGGER, Block ngoài cùng (đầu tiên) phải có DECLARE)

Ưu điểm của PL/SQL

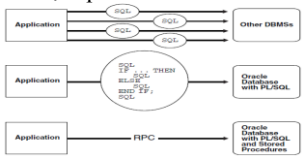
- Tích hợp chặt chẽ với SQL.
- Hiệu suất cao.
- Bảo mật chặt chẽ.

Tích hợp chặt chẽ với SQL.

- PL/SQL kết hợp sức mạnh thao tác dữ liệu của SQL với sức mạnh xử lý của ngôn ngữ thủ tục.
- Ngôn ngữ PL/SQL cho phép làm việc với các column và row mà không xác định kiểu dữ liệu.
- Truy vấn SQL và xử lý tập kết quả dễ dàng trong PL/SQL.
- PL/SQL hỗ trợ đầy đủ các kiểu dữ liệu SQL.

Hiệu suất cao

- Với PL/SQL, một block chứa nhiều câu lệnh có thể được gửi tới database cùng một lúc.
- Chương trình con PL/SQL được biên dịch một lần và lưu trữ ở dạng thực thi, do đó việc gọi chương trình con thì hiệu quả.



Bảo mật chặt chẽ

- Các chương trình con PL/SQL di chuyển code từ client sang server -> bảo vệ nó khỏi sự can thiệp, ẩn các chi tiết bên trong, giới hạn người có thể truy cập.
- Trigger viết bằng PL/SQL có thể kiểm soát hoặc ghi nhận những thay đổi dữ liệu, đảm bảo rằng tất cả các thay đổi tuân theo quy tắc được định trước.

Các tính năng chính của PL/SQL

- **Khối 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

Khối lệnh (block) PL/SQL

- Thành phần cơ bản của một chương trình PL/SQL là block, block nhóm những khai báo và những câu lệnh lại với nhau.
- Block trong PL/SQL được định nghĩa bởi các từ khóa DECLARE, BEGIN, EXCEPTION, và END.
- Block có thể lồng nhau.

PL/SQL Block

DECLARE -- Declarative part (optional)

Declarations of local types, variables, & subprograms

BEGIN --Executable part (required)

Statements (which can use items declared in declarative part)

EXCEPTION -- Exception-handling part (optional)

Exception handlers for exceptions raised in executable part

END;

Các tính năng chính của PL/SQL

- Khối 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

PL/SQL Input và Output

- Hầu hết các PL/SQL input và output (I/O) thông qua câu lệnh SQL để lưu trữ dữ liệu trong các table hoặc truy vấn dữ liệu từ table.
- PL/SQL I/O còn lại được thực hiện thông qua các API, chẳng hạn như PL/SQL package DBMS_OUTPUT.
- Để DBMS_OUTPUT hoạt động trên SQL*Plus, trước tiên phải thực hiện lệnh SET SERVEROUTPUT ON.

PL/SQL Input và Output

```
REM set server output to ON to display output from DBMS_OUTPUT
SET SERVEROUTPUT ON
BEGIN
      DBMS_OUTPUT.PUT_LINE ('These are the tables that ' || USER || ' owns:');
      FOR item IN (SELECT table_name FROM user_tables)
      LOOP
            DBMS_OUTPUT.PUT_LINE(item.table_name);
      END LOOP;
END;
```

Các tính năng chính của PL/SQL

- Khối 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

PL/SQL Biến (variable) và hằng (constant)

- Khai báo biến.
- Gán giá trị cho biến.
- Khai báo hằng số.

Khai báo biến trong PL/SQL

- Biến trong PL/SQL có thể là bất kỳ kiểu dữ liệu nào được sử dụng trong SQL (chẳng hạn char, date, number…) hoặc kiểu dữ liệu chỉ có ở PL/SQL (chẳng hạn boolean, pls_integer).

```
DECLARE
      part_number NUMBER(6); -- SQL data type
      part_name VARCHAR2(20); -- SQL data type
      in_stock BOOLEAN; -- PL/SQL-only data type
      part_price NUMBER(6,2); -- SQL data type
      part_description VARCHAR2(50); -- SQL data type
BEGIN
      NULL;
END;
```

Gán giá trị cho biến

- Để gán giá trị cho biến ta có thể sử dụng các cách sau:
  - Sử dụng toán tử gán :=
  - Bằng cách selecting (hoặc fetching) dữ liệu.
  - Bằng cách truyền cho nó như là một đối số OUT hoặc IN OUT trong chương trình con (subprogram), rồi sau đó gán giá trị trong subprogram.

Gán giá trị cho biến

```
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;
```

Khai báo hằng số trong PL/SQL

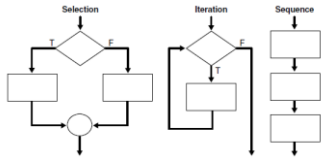
- Khai báo một hằng số trong PL/SQL giống như khai báo biến ngoại trừ việc phải thêm từ khóa CONSTANT và ngay lập tức gán giá trị cho hằng số này.
  - credit_limit CONSTANT NUMBER := 5000.00;

Các tính năng chính của PL/SQL

- Khối 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

Cấu trúc điều khiển trong PL/SQL

- **Quản lý điều kiện (Conditional Control)**
- Quản lý lặp (Iterative Control)
- Quản lý tuần tự (Sequential Control)



Quản lý điều kiện
Sử dụng IF-THEN Statement

```
DECLARE
    sales NUMBER(8,2) := 10400;
    quota NUMBER(8,2) := 10000;
    bonus NUMBER(6,2);
    emp_id NUMBER(6) := 120;
BEGIN
    IF sales > (quota + 200) THEN
        bonus := (sales - quota)/4;

        UPDATE employees SET salary = salary + bonus
        WHERE employee_id = emp_id;
    END IF;
END;
```

Conditional Control
Sử dụng IF-THEN-ELSE Statement

```
DECLARE
    sales NUMBER(8,2) := 10100;
    quota NUMBER(8,2) := 10000;
    bonus NUMBER(6,2);
    emp_id NUMBER(6) := 120;
BEGIN
    IF sales > (quota + 200) THEN
        bonus := (sales - quota)/4;
    ELSE
        bonus := 50;
    END IF;
    UPDATE employees SET salary = salary + bonus
    WHERE employee_id = emp_id;
END;
```

Quản lý điều kiện
IF-THEN-ELSE lồng nhau

```
DECLARE
    sales NUMBER(8,2) := 10100;
    quota NUMBER(8,2) := 10000;
    bonus NUMBER(6,2);
    emp_id NUMBER(6) := 120;
BEGIN
    IF sales > (quota + 200) THEN
        bonus := (sales - quota)/4;
    ELSE
        IF sales > quota THEN
            bonus := 50;
        ELSE
            bonus := 0;
        END IF;
    END IF;
    UPDATE employees SET salary = salary + bonus
    WHERE employee_id = emp_id;
END;
```

Quản lý điều kiện
Sử dụng IF-THEN-ELSIF Statement

```
DECLARE
    sales NUMBER(8,2) := 20000;
    bonus NUMBER(6,2);
    emp_id NUMBER(6) := 120;
BEGIN
    IF sales > 50000 THEN
        bonus := 1500;
    ELSIF sales > 35000 THEN
        bonus := 500;
    ELSE
        bonus := 100;
    END IF;
    UPDATE employees  SET salary = salary + bonus
    WHERE employee_id = emp_id;
END;
```

Quản lý điều kiện
Câu lệnh IF-THEN mở rộng

```
DECLARE
        grade CHAR(1);
BEGIN
    grade := 'B';
    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');
    ELSIF grade = 'D' THEN
        DBMS_OUTPUT. PUT_LINE('Fair');
    ELSIF grade = 'F' THEN
        DBMS_OUTPUT.PUT_LINE('Poor');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No such grade');
    END IF;
END;
```

Quản lý điều kiện
Sử dụng CASE Statement

```
DECLARE
        grade CHAR(1);
BEGIN
    grade := 'B';
    CASE grade
        WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
        WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
        WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
        WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
        WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
        ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
    END CASE;
END;
```

Quản lý điều kiện
Sử dụng Searched CASE Statement

```
DECLARE
        grade CHAR(1);
BEGIN
    grade := 'B';
    CASE
        WHEN grade='A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
        WHEN grade='B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
        WHEN grade='C' THEN DBMS_OUTPUT.PUT_LINE('Good');
        WHEN grade='D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
        WHEN grade='F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
        ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
    END CASE;
END;
```

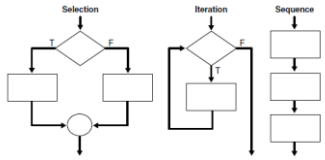
Quản lý điều kiện

- Sử dụng EXCEPTION thay vì mệnh đề ELSE trong CASE Statement

```
DECLARE
        grade CHAR(1);
BEGIN
    grade := 'B';
    CASE
        WHEN grade='A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
        WHEN grade='B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
        WHEN grade='C' THEN DBMS_OUTPUT.PUT_LINE('Good');
        WHEN grade='D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
        WHEN grade='F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
    END CASE;
    EXCEPTION
        WHEN CASE_NOT_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('No such grade');
END;
```

Cấu trúc điều khiển trong PL/SQL

- Quản lý điều kiện (Conditional Control)
- **Quản lý lặp (Iterative Control)**
- Quản lý tuần tự (Sequential Control)



Quản lý lặp

- Sử dụng basic LOOP Statement

```
LOOP
        sequence_of_statements
END LOOP;
```

- Có thể dùng CONTINUE và CONTINUE-WHEN trong một basic loop, nhưng để ngăn chặn một vòng lặp vô tận ta phải sử dụng EXIT hoặc EXIT-WHEN statement.

Quản lý lặp
Sử dụng EXIT Statement

```
DECLARE
        x NUMBER := 0;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
        x := x + 1;
        IF x > 3 THEN
                EXIT;
        END IF;
    END LOOP;
    -- After EXIT, control resumes here
    DBMS_OUTPUT.PUT_LINE (' After loop: x = ' || TO_CHAR(x));
END;
```

Quản lý lặp
Sử dụng EXIT-WHEN Statement

```
DECLARE
        x NUMBER := 0;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
        x := x + 1;
        EXIT WHEN x > 3;
    END LOOP;
    -- After EXIT, control resumes here
    DBMS_OUTPUT.PUT_LINE (' After loop: x = ' || TO_CHAR(x));
END;
```

Quản lý lặp
Sử dụng CONTINUE Statement

```
DECLARE
        x NUMBER := 0;
BEGIN
    LOOP -- After CONTINUE statement, control resumes here
        DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
        x := x + 1;
        IF x < 3 THEN
                CONTINUE;
        END IF;
        DBMS_OUTPUT.PUT_LINE
                ('Inside loop, after CONTINUE: x = ' || TO_CHAR(x));
        EXIT WHEN x = 5;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE (' After loop: x = ' || TO_CHAR(x));
END;
```

Quản lý lặp
Sử dụng CONTINUE-WHEN Statement

```
SET SERVEROUTPUT ON;
DECLARE
        x NUMBER := 0;
BEGIN
    LOOP -- After CONTINUE statement, control resumes here
        DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
        x := x + 1;
        CONTINUE WHEN x < 3;
        DBMS_OUTPUT.PUT_LINE
                ('Inside loop, after CONTINUE: x = ' || TO_CHAR(x));
        EXIT WHEN x = 5;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE (' After loop: x = ' || TO_CHAR(x));
END;
```

Quản lý lặp
Gán nhãn trong PL/SQL Loop

```
DECLARE
    s PLS_INTEGER := 0;
    i PLS_INTEGER := 0;
    j PLS_INTEGER;
BEGIN
    <<outer_loop>>
    LOOP
        i := i + 1;
        j := 0;
        <<inner_loop>>
        LOOP
            j := j + 1;
            s := s + i * j; -- Sum several products
            EXIT inner_loop WHEN (j > 5);
            EXIT outer_loop WHEN ((i * j) > 15);
        END LOOP inner_loop;
    END LOOP outer_loop;
    DBMS_OUTPUT.PUT_LINE ('The sum of products equals: ' || TO_CHAR(s));
END;
```

Quản lý lặp
Sử dụng WHILE-LOOP Statement

```
WHILE condition LOOP
        sequence_of_statements
END LOOP;
```

```
done := FALSE;
WHILE NOT done LOOP
      sequence_of_statements
      done := boolean_expression
END LOOP;
```

Quản lý lặp
Sử dụng FOR-LOOP Statement

```
BEGIN
      FOR i IN 1..3 LOOP
            DBMS_OUTPUT.PUT_LINE (TO_CHAR(i));
      END LOOP;
END;
```

```
BEGIN
      FOR i IN REVERSE 1..3 LOOP
            DBMS_OUTPUT.PUT_LINE (TO_CHAR(i));
      END LOOP;
END;
```

Quản lý lặp
Sử dụng FOR-LOOP Statement

- Trong một FOR loop, biến đếm có thể dùng để đọc nhưng không thể thay đổi, ví dụ:

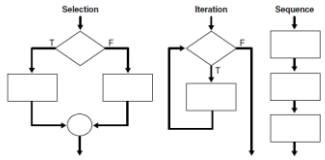
```
BEGIN
      FOR i IN 1..3 LOOP
            IF i < 3 THEN
                  DBMS_OUTPUT.PUT_LINE (TO_CHAR(i));
            ELSE
                  i := 2; --ERROR
            END IF;
      END LOOP;
END;
```

Quản lý lặp
Dynamic Ranges for Loop Bounds

```
DECLARE
      emp_count NUMBER;
BEGIN
      SELECT COUNT(employee_id) INTO emp_count
      FROM employees;
      FOR i IN 1..emp_count LOOP
            INSERT INTO temp VALUES(i, 'to be added later');
      END LOOP;
END;
```

Cấu trúc điều khiển trong PL/SQL

- Quản lý điều kiện (Conditional Control)
- Quản lý lặp (Iterative Control)
- **Quản lý tuần tự (Sequential Control)**



Quản lý tuần tự
Sử dụng GOTO Statement

```
DECLARE
      p VARCHAR2(30);
      n PLS_INTEGER := 37;
BEGIN
      FOR j in 2..ROUND(SQRT(n)) LOOP
            IF n MOD j = 0 THEN
                  p := ' is not a prime number';
                  GOTO print_now;
            END IF;
      END LOOP;
      p := ' is a prime number';
      <<print_now>>
      DBMS_OUTPUT.PUT_LINE(TO_CHAR(n) || p);
END;
```

Quản lý tuần tự
Sử dụng NULL Statement

```
DECLARE
    v_job_id VARCHAR2(10);
    v_emp_id NUMBER(6) := 110;
BEGIN
    SELECT job_id INTO v_job_id FROM employees
    WHERE employee_id = v_emp_id;

    IF v_job_id = 'SA_REP' THEN
        UPDATE employees
        SET commission_pct = commission_pct * 1.2
        WHERE employee_id = v_emp_id;
    ELSE
        NULL; -- Employee is not a sales rep
    END IF;
END;
```

Các tính năng chính của PL/SQL

- Khối 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

Quản lý lỗi trong PL/SQL

- PL/SQL giúp cho việc phát hiện và xử lý lỗi được dễ dàng và gọi là **exceptions.**
- Khi lỗi xảy ra, một exception hình thành và nó dừng công việc hiện tại sau đó chuyển đến bộ phận xử lý lỗi (exception handler).
- Một exception có thể được định nghĩa sẵn (bởi hệ thống) hoặc người dùng định nghĩa.
- Mỗi exception được xử lý bởi một exception handler.

Quản lý lỗi trong PL/SQL
Exception được định nghĩa sẵn

Exception Name	Error Code	Exception Name	Error Code
ACCESS_INTO_NULL	-6530	ZERO_DIVIDE	-1476
CASE_NOT_FOUND	-6592	PROGRAM_ERROR	-6501
COLLECTION_IS_NULL	-6531	ROWTYPE_MISMATCH	-6504
CURSOR_ALREADY_OPEN	-6511	SELF_IS_NULL	-30625
DUP_VAL_ON_INDEX	-1	STORAGE_ERROR	-6500
INVALID_CURSOR	-1001	SUBSCRIPT_BEYOND_COUNT	-6533
INVALID_NUMBER	-1722	SUBSCRIPT_OUTSIDE_LIMIT	-6532
LOGIN_DENIED	-1017	SYS_INVALID_ROWID	-1410
NO_DATA_FOUND	+100	TIMEOUT_ON_RESOURCE	-51
NO_DATA_NEEDED	-6548	TOO_MANY_ROWS	-1422
NOT_LOGGED_ON	-1012	VALUE_ERROR	-6502

Quản lý lỗi trong PL/SQL

```
DECLARE
    stock_price NUMBER := 9.73;
    net_earnings NUMBER := 0;
    pe_ratio NUMBER;
BEGIN
    -- Calculation might cause division-by-zero error.
    pe_ratio := stock_price / net_earnings;
    DBMS_OUTPUT.PUT_LINE('Price/earnings ratio = ' || pe_ratio);
    EXCEPTION -- exception handlers begin Only one of the WHEN blocks is executed.
        WHEN ZERO_DIVIDE THEN -- handles 'division by zero' error
            DBMS_OUTPUT.PUT_LINE('Company must have had zero earnings.');
            pe_ratio := NULL;
        WHEN OTHERS THEN -- handles all other errors
            DBMS_OUTPUT.PUT_LINE('Some other kind of error occurred.');
            pe_ratio := NULL;
END; -- exception handlers and block end here
```

Quản lý lỗi trong PL/SQL

- Ta có thể tránh exception bằng cách kiểm tra mẫu số trước

```
DECLARE
    stock_price NUMBER := 9.73;
    net_earnings NUMBER := 0;
    pe_ratio NUMBER;
BEGIN
    IF(net_earnings = 0) THEN
        pe_ratio:=NULL;
    ELSE net_earnings
        pe_ratio:= stock_price / net_earnings;
    END IF;
END;
```

Quản lý lỗi trong PL/SQL

- Ta có thể tránh exception bằng cách kiểm tra mẫu số trước

```
DECLARE
    stock_price NUMBER := 9.73;
    net_earnings NUMBER := 0;
    pe_ratio NUMBER;
BEGIN
    pe_ratio :=
    CASE net_earnings
        WHEN 0 THEN NULL
        ELSE stock_price / net_earnings
    END;
END;
```

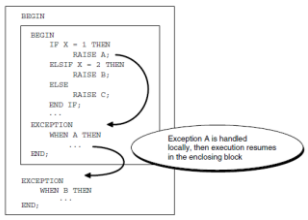
Quản lý lỗi trong PL/SQL

```
DECLARE
    emp_column VARCHAR2(30) := 'last_name';
    table_name VARCHAR2(30) := 'emp';
    temp_var VARCHAR2(30);
BEGIN
    temp_var := emp_column;
    SELECT COLUMN_NAME INTO temp_var FROM USER_TAB_COLS
    WHERE TABLE_NAME = 'EMPLOYEES'
                                AND COLUMN_NAME = UPPER(emp_column);
    -- processing here
    temp_var := table_name;
    SELECT OBJECT_NAME INTO temp_var FROM USER_OBJECTS
    WHERE OBJECT_NAME = UPPER(table_name) AND OBJECT_TYPE = 'TABLE';
    -- processing here
    EXCEPTION
    -- Catches all 'no data found' errors
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE ('No Data found for SELECT on ' || temp_var);
END;
```

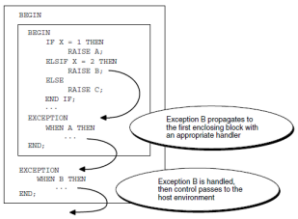
Quản lý lỗi trong PL/SQL

```
DECLARE
    out_of_stock        EXCEPTION;
    number_on_hand      NUMBER := 0;
BEGIN
    IF number_on_hand < 1 THEN
        RAISE out_of_stock; -- raise an exception that you defined
        .........
        .........
    END IF;
    EXCEPTION
        -- handle the error
        WHEN out_of_stock THEN
                DBMS_OUTPUT.PUT_LINE('Encountered out-of-stock error.');
END;
```

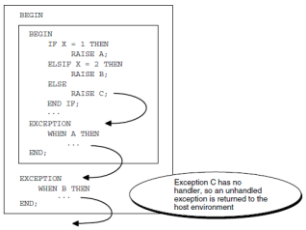
Exception lan truyền như thế nào?



Exception lan truyền như thế nào?



Exception lan truyền như thế nào?



Phạm vi của Exception

```
BEGIN
    DECLARE ---------- sub-block begins
        past_due EXCEPTION;
        due_date DATE := trunc(SYSDATE) - 1;
        todays_date DATE := trunc(SYSDATE);
    BEGIN
        IF due_date < todays_date THEN
            RAISE past_due;
        END IF;
    END; ------------ sub-block ends
    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
END;
```

Chỉ có OTHERS handler có thể bắt được exception

Phạm vi của Exception

```
DECLARE
    past_due EXCEPTION;
    acct_num NUMBER;
BEGIN
    DECLARE ---------- sub-block begins
        past_due EXCEPTION; -- this declaration prevails
        acct_num NUMBER;
        due_date DATE := SYSDATE - 1;
        todays_date DATE := SYSDATE;
    BEGIN
        IF due_date < todays_date THEN
            RAISE past_due; -- this is not handled
        END IF;
    END; ------------ sub-block ends
    EXCEPTION
        -- Does not handle raised exception
        WHEN past_due THEN
            DBMS_OUTPUT.PUT_LINE ('Handling PAST_DUE exception.');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE
              ('Could not recognize PAST_DUE_EXCEPTION in this scope.');
END;
```

Lấy ra thông tin Error Code và Error Message

```
CREATE TABLE errors (
    code NUMBER,
    message VARCHAR2(64),
    happened TIMESTAMP);
```

```
DECLARE
    name EMPLOYEES.LAST_NAME%TYPE;
    v_code NUMBER;
    v_errm VARCHAR2(64);
BEGIN
    SELECT last_name INTO name FROM EMPLOYEES;
    EXCEPTION
        WHEN OTHERS THEN
            v_code := SQLCODE;
            v_errm := SUBSTR(SQLERRM, 1, 64);
            DBMS_OUTPUT.PUT_LINE ('Error code ' || v_code || ': ' || v_errm);
            INSERT INTO errors
            VALUES (v_code, v_errm, SYSTIMESTAMP);
END;
```