CSC12001 – An toàn bảo mật CSDL trong HTTT

Tháng 1/2021

PL/SQL Đơn giản

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Tóm tắt nội dung bài thực hành:

Tài liệu này hướng dẫn sinh viên thực hiện các lệnh PL/SQL đơn giản trong Oracle 11g. Sinh viên có thể thực hiện các ví dụ bên dưới cũng như bài tập bằng công cụ SQL Plus hoặc SQL Developer trong gói cái đặt Oracle.*

Bộ môn **Hệ thống thông tin**
Khoa Công nghệ thông tin
ĐH Khoa học tự nhiên TP HCM

MỤC LỤC

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

1 Mục tiêu và tóm tắt nội dung

Tài liệu này hướng dẫn sinh viên thực hiện các lệnh PL/SQL đơn giản trong Oracle 11g. Sinh viên có thể thực hiện các ví dụ bên dưới cũng như bài tập bằng công cụ SQL* Plus hoặc SQL Developer trong gói cái đặt Oracle.

Ngoài ra, sinh viên cần tham khảo chi tiết hơn trong các tài liệu tham khảo của Oracle:

- PL/SQL Language Reference: http://docs.oracle.com/cd/B28359_01/appdev.111/b28370/toc.htm
- PL/SQL Packages and Type Reference: http://docs.oracle.com/cd/B28359_01/appdev.111/b28419/toc.htm

2 Các kiểu dữ liệu thông dụng

https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Data-Types.html#GUID-1BABC478-FB47-4962-9B0C-8B8BD059E733

- Kiểu dữ liệu chuỗi:
  - char(n), nchar(n), varchar2(n), nvarchar2(n)
- Kiểu dữ liệu số:
  - number, number(p,s), float(p), long, …
- Kiểu dữ liệu thời gian:
  - datetime, date, time, timestamp, …
- ….

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

3 Lệnh truy vấn cơ bản

3.1 Câu lệnh truy vấn đơn giản

```
Cú pháp
    SELECT  [DISTINCT]  {*, column [alias],...}
    FROM    table;

Với:
    SELECT      --Hiển thị nội dung của một hay nhiều cột
    DISTINCT    --Phân biệt nội dung giữa các dòng dữ liệu trả về
                --Lấy tất cả các cột trong bảng
    column      --Tên cột dữ liệu cần trả về
    alias       --Phần tiêu đề của cột dữ liệu trả về
    FROM table  --Tên bảng chứa dữ liệu truy vấn

Ví dụ:
    SELECT
    FROM emp;
```

3.2 Các thành phần khác của mệnh đề SELECT

- Biểu thức toán học
- Column alias
- Các column được ghép chuỗi
- Literal

Biểu thức toán học

Trong mệnh đề SELECT biểu thức toán học có thể các giá trị (column hoặc hàng số), các toán tử, các hàm.

Các toán tử được dùng là (+), (-), (*), (/). Độ ưu tiên của các toán tử giống trong phần số học.

Ví dụ:

```
SELECT ename, sal*12, comm FROM emp;
SELECT ename, (sal+250)*12 FROM emp;
```

Tiêu đề của cột (column alias)

Trong mệnh đề SELECT, column alias là phần nhãn hiển thị của column khi lấy số liệu ra. Trong column alias không được có dấu cách và viết cách sau tên column một dấu cách. Column alias được chấp nhận có dấu cách khi được đặt trong dấu nháy kép (“”).

Ví dụ: (ANUAL chính là column alias)

Bộ môn HTTT - Khoa CNTT

```
SELECT ename, SAL*12 "ANUAL", comm
FROM emp;
```

Các column được ghép chuỗi

Toán tử ghép tiếp chuỗi (||) cho phép ghép tiếp dữ liệu trong các cột khác nhau của cùng một dòng dữ liệu với nhau thành một chuỗi. Ta có thể có nhiều toán tử ghép chuỗi trong cùng một column alias.

Ví dụ:

```
SELECT name||post_number, vistor, page_number
FROM website;
```

NAME||POST_NUMBER	VISITOR	PAGE_NUMBER
cafeitvn.com500	10000	20
cafeitvn.com-2500	10000	14
cafeitvn.com-3500	10000	15
cafeitvn.com-4500	10000	20

Ghép tiếp chuỗi ký tự

Trong mệnh đề SELECT, ta có thể thực hiện ghép tiếp bất kỳ ký tự nào, biểu thức hay số nào mà không phải là column hoặc column alias.

Ví dụ:

```
SELECT name || 'has' || vistor || 'visitor' website_info
FROM website;
```

WEBSITE_INFO
cafeitvn.com has 10000visitor
cafeitvn.com-2 has 10000visitor
cafeitvn.com-3 has 10000visitor
cafeitvn.com-4 has 10000visitor

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

3.3 Phân biệt giá trị dữ liệu trả về

Trong thực tế nhiều khi giá trị dữ liệu trên các dòng dữ liệu kết xuất trùng nhau. Gây nhiều bất tiện. Để có thể lấy được chỉ các dòng dữ liệu phân biệt với nhau. Ta sử dụng mệnh đề DISTINCT trong câu lệnh truy vấn.

```
SELECT DISTINCT deoptno FROM dept
```

3.4 Giá trị NULL

Cột có giá trị rỗng (NULL) là cột chưa được gán giá trị, nói cách khác nó chưa được khởi tạo giá trị. Các cột với bất cứ kiểu dữ liệu nào cũng có thể có trị NULL, trừ khi được nó là khóa hay có ràng buộc toàn vẹn NOT NULL. Trong biểu thức có bất kỳ giá trị NULL nào kết quả cũng là NULL.

```
SELECT ename, sal*12 + comm ANUAL_SAL
FROM emp;
```

NULL trong các hàm của SQL

Trong các hàm làm việc với từng cột hay hàm vô hướng (scalar function). Các hàm loại này trả về trị null khi có tham số NULL, trừ hàm NVL và TRANSLATE có thể trả về giá trị thực.

Cú pháp của hàm NVL:

```
NVL(DATECOLUMN,'01-01-2001')
NVL(NUMBERCOLUMN, 9)
NVL(CHARCOLUMN, 'STRING')
NVL(comm, 0) -- trả về 0 khi comm là null

SELECT ename, sal*12 + NVL(comm, 0) ANUAL_SAL
FROM emp;
```

Trong các hàm làm việc với nhóm các cột (group function): Hầu hết các hàm làm việc trên nhóm bỏ qua trị null, ví dụ như khi sử dụng hàm AVG để tính trung bình cho một cột có các giá trị 1000, NULL, NULL, NULL, 2000. Khi đó trung bình được tính là (1000+2000)/2=1500, như vậy trị null bị bỏ qua chứ không phải xem là trị 0.

Bộ môn TTT - Khoa CNTT - ĐH KHTN

NULL trong các biểu thức so sánh, điều kiện

Để kiểm tra có phải NULL hay không dùng các toán tử IS NULL hoặc IS NOT NULL. Nếu trong biểu thức so sánh có trị null tham gia và kết quả của biểu thức phụ thuộc vào trị null thì kết quả là không xác định, tuy nhiên trong biểu thức DECODE, hai giá trị null được xem là bằng nhau trong phép so sánh.

Oracle xem các biểu thức với kết quả không xác định tương đương với FALSE. (Ví dụ: comm = NULL) có kết quả không xác định và do đó biểu thức so sánh xem như cho kết quả FALSE. Trong câu lệnh sau không có mẫu tin nào được chọn

```
SELECT * FROM emp WHERE comm=NULL;
SELECT * FROM emp WHERE comm IS NULL;
```

Nếu muốn chọn các nhân viên có comm là NULL thì phải dùng toán tử IS NULL.

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

4 SQL*PLUS, công cụ tương tác lệnh SQL với CSDL

Kết nối tới CSDL

Cú pháp:

```
Conn[ect] <user_name>/<password>[@<database];

Với:
    user_name: Tên truy cập
    password: Mật khẩu truy cập
    database: Tên CSDL truy cập

Ví dụ:
    Conn Tester/tester@DB1
```

Hiển thị cấu trúc bảng dữ liệu

Cú pháp:

```
Desc[ribe] <table_name>

Với:
    table_name: Tên bảng cần hiển thị cấu trúc

Ví dụ:
    Desc Dept;
```

Bộ môn HTTT - KHMT - ĐH KHTN

5 Truy vấn dữ liệu có điều kiện

5.1 Mệnh đề WHERE

Cú pháp:

```
SELECT [DISTINCT ] {*, column [alias],...}
FROM table
[WHERE condition (s)];

Với:
    column      tên cột dữ liệu trả về
    alias       tiêu đề của cột dữ liệu trả về
    table       tên bảng truy vấn dữ liệu
    condition   mệnh đề điều kiện để lọc dữ liệu trả về
```

Mệnh đề WHERE dùng để đặt điều kiện cho toàn bộ câu lệnh truy vấn. Trong mệnh đề WHERE có thể có các thành phần: tên column, toán tử so sánh, tên column, hằng số hoặc danh sách các giá trị.

5.2 Sử dụng các toán tử điều kiện

```
[NOT] BETWEEN x AND y
```

Ví dụ: chọn nhân viên có lương nằm trong khoảng 2000 và 3000

```
SELECT * FROM emp WHERE sal BETEEN 2000 AND 3000;
```

```
IN (danh sách)
```

Ví dụ: Chọn nhân viên có lương bằng một trong 2 giá trị 1400 hoặc 3000

```
SELECT * FROM emp WHERE sal IN (1400, 3000);
```

Ví dụ: Tìm tên phòng ban nếu phòng đó có nhân viên làm việc.

```
SELECT dname
FROM dept
WHERE EXISTS (  SELECT *
                FROM emp
                WHERE dept.deptno = emp.deptno);
```

```
x [NOT] LIKE y
```

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Ví dụ: Tìm nhân viên có tên bắt đầu bằng chuỗi SMITH

```
SELECT * FROM emp WHERE ename LIKE 'SMITH_';
```

6 Sắp xếp dữ liệu trả về

6.1 Mệnh đề ORDER BY

Cú pháp:

```
SELECT [DISTINCT ] {*, column [alias],...}
FROM table;
[WHERE condition]
[ORDER BY expr/position [DESC/ASC]];
```

Mệnh đề ORDER BY dùng để sắp xếp số liệu được hiển thị và phải đặt ở vị trí sau cùng của câu lệnh truy vấn.

Ví dụ:

```
SELECT   ENAME,   JOB,   SAL*12,   DEPTNO
FROM EMP
ORDER BY ENAME;
```

Mệnh đề ORDER BY mặc định sắp xếp theo thứ tự tăng dần ASC[ENDING]: Số thấp trước, Ngày nhỏ trước, Ký tự theo bảng chữ cái

Để sắp xếp theo thứ tự ngược lại (giảm dần) đặt từ khoá DESC[ENDING] sau column cần sắp thứ tự.

6.2 Sắp xếp nhiều cột dữ liệu trả về

Mệnh đề Order còn có thể sắp xếp nhiều column. Các column cần sắp xếp được viết thứ tự sau mệnh đề ORDER BY và cách bởi dấu phẩy (,). Column nào gần mệnh đề ORDER BY hơn có mức độ ưu tiên khi sắp xếp cao hơn. Chỉ định cách thức sắp xếp ASC/DESC được viết sau column cách bởi một dấu cách.

Ví dụ:

```
SELECT DEPTNO, JOB, ENAME, SAL
FROM EMP
ORDER BY DEPTNO, SAL DESC ;
```

Order giá trị NULL

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Riêng đối với giá trị NULL, nếu sắp xếp theo thứ tự ASCENDING sẽ nằm ở các vị trí cuối cùng. Chú ý:Có thể chỉ định sắp xếp theo thứ tự các column trong mệnh đề SELECT.

Ví dụ:

```
SELECT DEPTNO, JOB, ENAME, SAL
FROM EMP
ORDER BY 2;
```

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

7 Hàm SQL

7.1 Các hàm thao tác trên kiểu dữ liệu ký tự

Hàm SQL	Diễn giải
CONCAT(char1, char2)	Cho kết hợp của 2 chuỗi ký tự, tương tự như sử dụng toán tử ||
INITCAP(char)	Cho chuỗi với ký tự đầu các từ là ký tự Hoa
LOWER(char)	Cho chuỗi ký tự viết thường (không viết hoa)
LPAD(char1, n [,char2])	Cho chuỗi ký tự có chiều dài bằng n. Nếu chuỗi char1 ngắn hơn n thì thêm vào bên trái chuỗi char2 cho đủ n ký tự. Nếu chuỗi char1 dài hơn n thì giữ lại n ký từ tính từ trái sang
LTRIM(char1, n [,char2])	Bỏ các ký tự trống bên trái
NLS_INITCAP(char)	Cho chuỗi với ký tự đầu các từ là chữ hoa, các chữ còn lại là chữ thường
REPLACE(char, search_string [,replacement_string])	Thay tất cả các chuỗi search_string có trong chuỗi char bằng chuỗi replacement_string.
RPAD(char1, n [,char2])	Giống LPAD(char1, n [,char2]) nhưng căn phải.
RTRIM(char1, n [,char2])	Bỏ các ký tự trống bên phải
SOUNDEX(char)	Cho chuỗi đồng âm của char.
SUBSTR(char, m [,n])	Cho chuỗi con của chuỗi char lấy từ vị trí m về phải n ký tự, nếu không chỉ n thì lấy cho đến cuối chuỗi
TRANSLATE(char, from, to)	Cho chuỗi trong đó mỗi ký tự trong chuỗi from thay bằng ký tự tương ứng trong chuỗi to, những ký tự trong chuỗi from không có tương ứng trong chuỗi to sẽ bị loại bỏ.
UPPER(char)	Cho chuỗi chữ hoa của chuỗi char
ASCII(char)	Cho ký tự ASCII của byte đầu tiên của chuỗi char
INSTR(char1, char2 [,n[,m]])	Tìm vị trí chuỗi char2 trong chuỗi char1 bắt đầu từ vị trí n, lần xuất hiện thứ m.
LENGTH(char)	Cho chiều dài của chuỗi char

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Ví dụ hàm LOWER(char)

```
SELECT LOWER(DNAME), LOWER('SQL COURSE')
FROM DEPT;
```

Ví dụ hàm UPPER(char)

```
SELECT ENAME
FROM EMP
WHERE ENAME = UPPER('Smith');
```

7.2 Các hàm thao tác trên kiểu dữ liệu thời gian

Hàm SQL	Diễn giải
MONTH_BETWEEN(d1, d2)	Cho biết só tháng giữa ngày d1 và d2.
ADD_MONTHS(d,n)	Cho ngày d thêm n tháng.
NEXT_DAY(d, char)	Cho ngày tiếp theo ngày d có thứ chỉ bởi char
LAST_DAY(d)	Cho ngày cuối cùng trong tháng chỉ bởi d.

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

8 Truy vấn nâng cao

8.1 Cú pháp lệnh PL/ SQL

- Mỗi lệnh SQL kết thúc bằng dấu (;)
- Lệnh định nghĩa CSDL (DDL) không được sử dụng trong PL/SQL
- Lệnh SELECT trả về nhiều dòng có thể gây exception
- Lệnh DML có thể tác động trên nhiều dòng

Ví dụ:

```
x := 1;
INSERT INTO        emp (id, name)
VALUES (50, 'GARNOR');
BEGIN
            SELECT name FROM dept INTO :DEPT.NAME;
EXCEPTION
            WHEN others THEN
                  Message(SQLERRM);
END;

UPDATE emp
            SET sal := sal*1.2
            WHERE dept_id = 10;
```

8.2 Khối lệnh PL/ SQL

Ngôn ngữ PL/SQL tổ chức các lệnh theo từng khối lệnh. Một khối lệnh PL/SQL cũng có thể có các khối lệnh con khác ở trong nó.

Cấu trúc đầy đủ của một khối lệnh PL/SQL bao gồm:

```
DECLARE /* Phần khai báo - Không bắt buộc */
            Khai báo các biến sử dụng trong phần thân
BEGIN /* Phần thân */
Đoạn lệnh thực hiện;
EXCEPTION /* Phần xử lý lỗi - Không bắt buộc */
            Xử lý lỗi xảy ra;
END;
```

Ví dụ:

```
DECLARE
            empno        NUMBER(4):=7788;
BEGIN
            UPDATE emp
SET sal = 9000
                  WHERE empno = 0001;
END;
```

Bộ môn ... - Khoa CNTT - ĐHKHTN

8.3 Lệnh lập trình PL/ SQL đơn giản

Lệnh IF: Thực hiện câu lệnh theo điều kiện.

```
IF <điều kiện 1> THEN
      Công việc 1;
[ELSIF <điều kiện 2> THEN
      Công việc 2;
]

[ELSE
      Công việc n + 1;
]
END IF;
```

Ví dụ 1:

```
IF ename = 'SCOTT' THEN
      beam_me_up := 'YES';
      COMMIT;
ELSE
      beam_me_up := 'NO';
      ROLLBACK;
END IF;
```

Ví dụ 2:

```
IF choice= 1 THEN
      action := 'Run payroll';
ELSIF choice=2 THEN
      action:='Run';
ELSIF choice=3 THEN
      action:='Backup';
ELSE
      action:='Invalid';
END IF;
```

Bộ môn HTTT - CNTT - ĐH KHTN

Lệnh 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;
```

Ví dụ:

```
x := 0;
y := 1000;
LOOP
      x := x + 1;
      y := y - x;
      EXIT x > y;
END LOOP;
```

Lệnh lặp LOOP có định trước: Ngay khi bắt đầu vòng lặp, ta đã xác định được số lần lặp.

Cú pháp:

```
LOOP Index IN Cận dưới .. Cận trên
      Công việc;
END LOOP;
```

Ví dụ:

```
x := 0;
LOOP Index IN 1 .. 100
      x := x + 1;
END LOOP;
```

Lệnh lặpWHITE

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Cú pháp:

```
WHILE Điều kiện LOOP
      Công việc;
END LOOP;
```

Ví dụ:

```
WHILE length(:Address) < 50 LOOP
      :Address := :Address || ' ';
END LOOP;
```

Lệnh GOTO

Cú pháp:

```
GOTO        Nhãn;
```

Ví dụ:

```
BEGIN
      <<Nhãn>>
      công việc;

      GOTO Nhãn;

END;
```

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

9 Cursor

Cursor là kiểu biến có cấu trúc, cho phép ta xử lý dữ liệu gồm nhiều dòng. Số dòng phụ thuộc vào câu lệnh truy vấn dữ liệu sau nó. Trong quá trình xử lý, ta thao tác với cursor thông qua từng dòng dữ liệu. Dòng dữ liệu này được định vị bởi một con trỏ. Với việc dịch chuyển con trỏ, ta có thể lấy được toàn bộ dữ liệu trả về.

Các bước sử dụng biến cursor:

Khai báo --> mở cursor --> lấy dữ liệu để xử lý --> đóng cursor

Khai báo:

```
CURSOR Tên cursor( danh sách biến) IS Câu
lệnh truy vấn;
```

Ví dụ 1:

```
CURSOR      c_Dept      IS
      SELECT deptno, dname
            FROM dept
            WHERE deptno>10;
```

Ví dụ 2:

```
CURSOR      c_Dept(p_Deptno NUMBER)        IS
      SELECT deptno, dname
            FROM dept
            WHERE deptno>10;
```

Mở cursor:

```
OPEN Tên cursor | Tên cursor( danh sách biến);
```

Ví dụ 1:

```
OPEN        c_Dept;
```

Ví dụ 2:

```
OPEN        c_Dept(10);
```

Lấy dữ liệu:

```
FETCH Tên cursor INTO Tên biến;
```

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

Ví dụ:

```
FETCH c_Dept INTO v_Dept;
```

Đóng cursor:

```
CLOSE Tên cursor;
```

Các thuộc tính:

%isopen	trả lại giá trị True nếu cursor đang mở
%notfound	trả lại giá trị True nếu lệnh fetch hiện thời trả lại không có row
%found	trả lại giá tri true cho đến khi fetch không còn row nào
%rowcount	trả lại số row đã được thực hiện bằng lệnh fetch

Ví dụ:

```
DECLARE
      -- Khai báo cursor để truy vấn dữ liệu
      CURSOR c_Emp IS
SELECT *
FROM emp
WHERE dept_id = 10;

      -- Khai báo biến cursor tương ứng để chứa dòng dữ liệu
      v_Emp c_EMP%rowtype;

BEGIN
-- Mở cursor
OPEN c_Emp;
LOOP
      -- Lấy dòng dữ liệu từ cursor
            FETCH c_Emp INTO v_Emp;

-- Thoát khỏi vòng lặp nếu đã lấy hết dữ liệu trong cursor
            EXIT WHEN c_Emp%notfound;

            -- Bổ sung dữ liệu vào Emp_ext theo dữ liệu lấy được từ
      cursor
      INSERT INTO Emp_ext (empno, ename, job)
VALUES (v_Emp.empno, v_Emp.ename, v_Emp.job); END
LOOP;
-- Đóng cursor
CLOSE c_Emp;
END;
```

Bộ môn CNTT - ĐH KHTN

10 Các kiểu dữ liệu thông dụng

10.1 Kiểu dữ liệu TABLE

```
TYPE Tên_kiểu_Table IS
      TABLE OF Tên kiểu dữ liệu [NOT NULL] INDEX BY BINARY_INTEGER;

Tên biến     Tên_kiểu_Table;
```

Ví dụ:

```
TYPE t_Name IS
      TABLE OF Emp.Ename%TYPE INDEX BY BINARY_INTEGER;

v_First_name       t_Name;
v_Last_name        t_Name;
```

10.2 Kiểu dữ liệu RECORD

```
TYPE Tên_kiểu_Record IS
      RECORD OF (
            Col1          Tên kiểu [NOT NULL{:=|DEFAULT} biểu thức],
            Col2          Tên kiểu [NOT NULL{:=|DEFAULT} biểu
                          thức]...);

Tên biến     Tên_kiểu_Record;
```

Ví dụ:

```
TYPE t_Emp IS
        RECORD OF (
                    empno       number(4)          not null,
                    ename       char(10),
                    job         char(9),
                    mgr         number(4),
                    hiredate    date default sysdate,
                    sal         number(7,2),
                    comm        number(7,2),
                    deptno      number(2)          not null);
    v_Emp_record  t_Emp;
```

Bộ môn HTTT - Khoa ... - ĐH ...

10.3 Lệnh SELECT…INTO

Cú pháp:

```
SELECT col1, col2...
       INTO  var1,  var2...  [cursor_var]
       FROM table1, table2...
       [WHERE  condition1,  condition2...  ]
       [GROUP BY col1, col2 ...]
       [HAVING  condition1,  condition2...]
       [FOR UPDATE];
```

Với:

INTO var1, var2... [cursor_var] Biến lưu giữ các giá trị trong table lấy từ lệnh select.

Ví dụ:

SELECT deptno, loc INTO v_deptno, v_loc

FROM dept

WHERE dname = 'SALES';

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

11 Procedure Builder

Procedure builder là một thành phần được tích hợp vào môi trường phát triển ứng dụng của Oracle. Nó cho phép người sử dụng có thể soạn thảo, biên dịch, kiểm tra và dò lỗi đối với các hàm, thủ tục hay package viết bởi ngôn ngữ PL/SQL ở cả Client và Server.

11.1 Các thành phần trong procedure builder

Thành phần	Diễn giải
Object Navigator	Điều khiển truy nhập các hàm, thủ tục PL/SQL. Thực hiện thao tác dò lỗi (debug) trên các khối lệnh SQL và PL/SQL.
PL/SQL Interpreter	Dò lỗi mã nguồn PL/SQL
Program Unit Edito	Tạo và soạn thảo các mã nguồn khối lệnh PL/SQL.
Store Program Unit Editor	Tạo và soạn thảo các mã nguồn khối lệnh PL/SQL trong các Store Program thuộc Server.
Database Trigger Edditor	Tạo và soạn thảo các mã nguồn khối lệnh PL/SQL trong các Trigger thuộc Server

11.2 Tạo hàm, thủ tục tren CLIENT

Đối với hàm, thủ tục hay package trên client, ta có thể tạo và biên dịch ngay chúng. Oracle Builder hỗ trợ trình thông dịch cho phép kiểm tra lỗi của đoạn chương trình vừa thực hiện.

Việc tạo hàm, thủ tục được thực hiện theo ba bước:

- Khai báo tên hàm hay thủ tục
- Soạn thảo nội dung của hàm hay thu tục
- Biên dịch hàm hay thủ tục vừa tạo và xác định các lỗi nếu có.

11.3 Tạo hàm, thủ tục trên SERVER

Procedure Builder chỉ cho phép tạo mới, sửa chữa và lưu lại các thay đổi đối với các hàm và thủ tục trên Server, không hỗ trợ việc biên dịch và phát hiện lỗi.

Ta thực hiện việc tạo hàm, thủ tục trên server theo hai bước:

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

- Tạo hàm, thủ tục
- Soạn thảo và ghi lại nội dung của hàm, thủ tục

11.4 Dò lỗi đối với các hàm, thủ tục

Với Procedure Builder, ta có thể thực hiện chạy các hàm, thủ tục theo từng bước. Qua đó, ta có thể phát hiện được các lỗi xảy ra trong chương trình, nếu có.

Cấu trúc của màn hình PL/SQL Interpreter được chia làm ba phần chính:

- Phần mã nguồn hàm, thủ tục
- Phần điều khiển
- Phần tương tác trực tiếp với dữ liệu

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

12 Store Procedure

Cú pháp:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
      [(argument1      [mode1]     datatype1,
        argument2      [mode2]     datatype2,
        ...)]
      IS | AS
BEGIN
      PL/SQL Block;
END;
```

Với

procedure_name	Tên thủ tục
argument	Tên tham số
mode	Loại tham số: IN hoặc OUT hoặc IN OUT, mặc định là IN
datatype	Kiểu dữ liệu của tham số
PL/SQL Block	Nội dung khối lệnh SQL và PL/SQL trong thủ tục

Ví dụ:

```
CREATE OR REPLACE PROCEDURE change_sal
   (p_Percentage  IN number, p_Error  OUT varchar2, )
IS
   v_User_exp  Exception;
BEGIN
   IF p_Percentage < 0 THEN
         RAISE v_User_exp;
   END IF;
   UPDATE emp
   SET sal = sal*p_Percentage/100;
EXCEPTION
   WHEN v_User_exp THEN
         p_Error := 'Lỗi: Phần trăm nhỏ hơn 0';
      RETURN;
   WHEN others THEN
         p_Error := 'Lỗi: ' || SQLERRM;
END;
```

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

13 Function

Cú pháp:

```
CREATE [OR REPLACE] FUNCTION function_name
      [(argument1            [mode1]     datatype1,
        argument2            [mode2]     datatype2,
        ...)]
RETURN                 datatype
IS | AS
BEGIN
      PL/SQL Block;
END;
```

Với

- function_name Tên hàm
- argument Tên tham số
- mode Loại tham số: IN hoặc OUT hoặc IN OUT, mặc định là IN
- datatype Kiểu dữ liệu của tham số
- PL/SQL Block Nội dung khối lệnh SQL và PL/SQL trong thủ tục

Ví dụ:

```
CREATE OR REPLACE FUNCTION get_sal
      (p_Emp_id        IN number)
RETURN varchar2
IS
BEGIN
      SELECT sal
            FROM emp
            WHERE emp_id = p_Emp_id;

      RETURN      null;
EXCEPTION
      WHEN others THEN
            RETURN      'Lỗi: ' || SQLERRM;
END;
```

Thực hiện FUNCTION: Quá trình lưu giữ và biên dịch một hàm cũng tương tự như đối với một thủ tục. Quá trình gọi và thực hiện một hàm được diễn ra theo ba bước:

Bộ môn HTTT - Khoa CNTT - ĐH KHTN

- Việc gọi hàm được thực hiện ngay khi tên hàm trong biểu thức được tham chiếu tới
- Một biến host (host variable) được tự động tạo ra để lưu giữ giá trị trả về của hàm.
- Thực hiện nội dung trong phần thân hàm, lưu lại giá trị

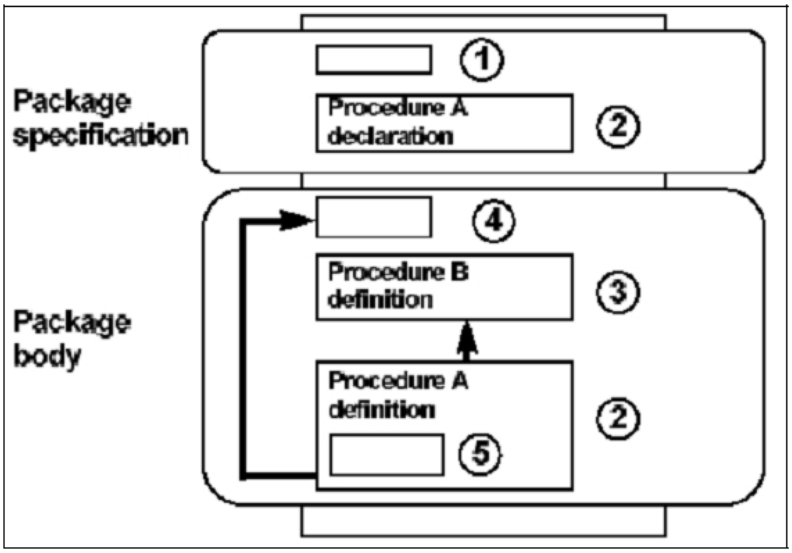
Bộ môn HTTT - Khoa CNTT - ĐH KHTN

14 Package

Package là một tập hợp các kiểu dữ liệu, biến lưu giữ giá trị và các thủ tục, hàm có cùng một mối liên hệ với nhau, được gộp chung lại. Đặc điểm nổi bật nhất của package là khi một phần tử trong package được gọi thì toàn bộ nội dung của package sẽ được nạp vào trong hệ thống. Do đó, việc gọi tới các phần tử khác trong package sau này sẽ không phải mất thời gian nạp vào hệ thống nữa. Từ đó, nâng cao tốc độ thực hiện lệnh của toàn bộ hàm, thủ tục có trong package.

14.1 Cấu trúc

Một package được cấu trúc làm hai phần. Phần mô tả (specification) định nghĩa các giao tiếp có thể có của package với bên ngoài. Phần thân (body) là các cài đặt cho các giao tiếp có trong phần mô tả ở trên.



Trong cấu trúc của package bao gồm 05 thành phần:

- Public variable (biến công cộng): là biến mà các ứng dụng bên ngoài có thể thamchiếu tới được.
- Public procedure (thủ tục công cộng): bao gồm các hàm, thủ tục của package có thể triệu gọi từ các ứng dụng bên ngoài.

Bộ môn HTTT - ĐH KHTN

- Private procedure (thủ tục riêng phần): là các hàm, thủ tục có trong package và chỉ có thể được triệu gọi bởi các hàm hay thủ tục khác trong package mà thôi.
- Global variable (biến tổng thể): là biến được khai báo dùng trong toàn bộ package, ứng dụng bên ngoài tham chiếu được tới biến này.
- Private variable (biến riêng phần): là biến được khai báo trong một hàm, thủ tục thuộc package. Nó chỉ có thể được tham chiếu đến trong bản thân hàm hay thủ tục đó.

14.2 Tạo package

Cú pháp khai báo phần mô tả package:

```
CREATE [OR REPLACE] PACKAGE package_name IS
| AS
     public type and các item declarations
     subprogram specifications
END package_name;
```

Với:

package_name	Tên package
type and item declarations	Phần khai báo các biến, hằng, cursor, ngoại lệ và kiểu sử dụng trong toàn bộ package
subprogram specifications	Khai báo các hàm, thủ tục PL/SQL

Cú pháp khai báo phần thân package:

```
CREATE [OR REPLACE] PACKAGE BODY package_name
IS | AS
     private type and item declarations
     subprogram bodies
END package_name;
```

Với:

- package_name Tên package

Bộ môn HTTT - Khoa CNTT ĐH KHTN

- type and item declarations Phần khai báo các biến, hằng, cursor, ngoại lệ và kiểu
- subprogram specifications Khai báo các hàm, thủ tục PL/SQL

Ví dụ

```
-- Phần khai báo của package
CREATE OR REPLACE PACKAGE comm_package
IS
      v_comm      number := 10; -- Khai báo biến có giá trị khởi tạo

      -- Khai báo thủ tục để giao tiếp với bên ngoài
      PROCEDURE reset_comm (p_comm IN number);
END comm_package;

-- Phần thân của package
CREATE OR REPLACE PACKAGE BODY comm_package
IS
      -- Hàm riêng phần chỉ sử dụng trong package
      FUNCTION          validate_comm
            (v_comm     IN number)
      RETURN BOOLEAN
      IS
            v_max_comm number;
      BEGIN
            SELECT max(comm) INTO v_max_comm
                  FROM        emp;
            IF v_comm > v_max_comm THEN
                  RETURN      FALSE;
            ELSE
                  RETURN      TRUE;
            END IF;
      END validate_comm;

      -- Thủ tục giao tiếp với bên ngoài
      PROCEDURE reset_comm
            (p_comm     IN number)
      IS
```

```
            v_valid     BOOLEAN;
      BEGIN
            v_valid := validate_comm(p_comm);
            IF v_valid = TRUE THEN
                  v_comm := p_comm;
            ELSE
            RAISE_APPLICATION_ERROR(-20210,'Invalid     comm');
            END IF:
      END reset_comm;
END comm_package;
```

Bộ môn HTTT - Khoa CNTT - ĐH KH

15 Trigger

Khi tạo database trigger, ta cần lưu ý tới một số tiêu chí như:

- Thời gian thực hiện: BEFORE, AFTER
- Hành động thực hiện: INSERT, UPDATE, DELETE
- Đối tượng tác động: bảng dữ liệu
- Loại trigger thực hiện: trên dòng lệnh hay trên câu lệnh
- Mệnh đề điều kiện thực hiện
- Nội dung của trigger

Cú pháp

```
CREATE [OR REPLACE] TRIGGER trigger_name
      timing event1 [OR event2 OR event3]
ON table_name
BEGIN
      PL/SQL Block;
END;
```

Ví dụ

```
CREATE OR REPLACE TRIGER secure_emp
BEFORE INSERT ON emp
BEGIN
      IF TO_CHAR(sysdate,'DY') IN ('SAT','SUN')
            OR TO_CHAR(sysdate,'HH24') NOT BETWEEN '08' AND '18'
            THEN
            RAISE_APPLICATION_ERROR (-20500,
                  'Thời gian làm việc không phù hợp');
       END IF;
END;

CREATE OR REPLACE TRIGER audit_emp_values
AFTER DELETE OR INSERT OR DELETE ON emp FOR
EACH ROW
BEGIN
      INSERT INTO audit_emp_values (user_name, timestamp, id,
                        old_last_name, new_last_name, old_title,
                        new_tile, old_salary, new_salary)
      VALUES (USER, SYSDATE, :old.empno, :old.ename,
                        :new.ename, :old.job, :new.job, :old.sal,
                        :new.sal);
END;
```

Bộ môn Khoa - ĐH KHTN