

Sử dụng PL/SQL với đối tượng



Mục tiêu:

- Khái quát được PL-SQL với kiểu đối tượng
- Áp dụng PL-SQL truy xuất kiểu đối tượng
- Cơ chế nạp chồng và ghi đè với kế thừa bằng PL-SQL



Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

1

1

Nội dung



- Giới thiệu PL/SQL
- PL/SQL với kiểu đối tượng
- Cơ chế nạp chồng trong kế thừa với PL/SQL
- Các ví dụ
- Tóm tắt



Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

2

2

Giới thiệu PL/SQL



- SQL là ngôn ngữ của mô hình dữ liệu quan hệ
 - SQL – Structured Query Language
 - T-SQL (Transact SQL)
- PL-SQL (Procedural Language SQL)
 - SQL + SQL 2003 + mở rộng trên ORACLE
 - Có khả năng truy vấn trên mô hình dữ liệu quan hệ, quan hệ-đối tượng
- PL – SQL với kiểu đối tượng, kế thừa...

ORACLE®
DATABASE

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

3

3

Giới thiệu PL/SQL



- PL/SQL (Procedural Language/Structured Query Language) là một mở rộng của SQL trên Oracle
- PL-SQL cung cấp các cấu trúc lập trình, khai báo biến, kiểu dữ liệu, phép toán và cách thức tạo thủ tục, hàm, package, triggers và những thành phần được lưu trữ trong cơ sở dữ liệu cho cả mô hình quan hệ và mô hình quan hệ - đối tượng
- Kết thúc mỗi lệnh sử dụng dấu ;
- Phép gán giá trị cho biến :=

ORACLE®
DATABASE

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

4

4

Giới thiệu PL/SQL



- Tổ chức khối lệnh trong PL-SQL
 - Declare**
 - Phân khai báo, không bắt buộc
 - Begin**
 - Phần thân của khối lệnh
 - Đoạn lệnh thực hiện
 - Phần xử lý lỗi - Không bắt buộc
 - End;**
- Một số cấu trúc lập trình trong PL-SQL
 - Lệnh If-elsif-else
 - Lệnh lặp LOOP, FOR LOOP, WHILE
 - ...
- Function, Trigger, Cursor, Procedure, Package...

ORACLE®
DATABASE

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

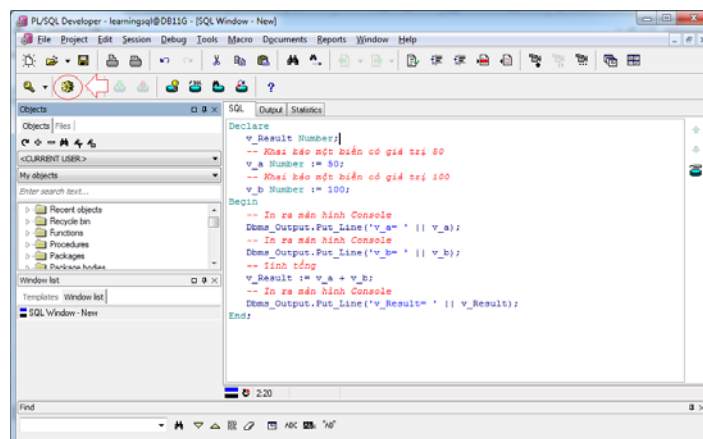
5

5

Giới thiệu PL/SQL



- Công cụ PL/SQL Developer có thể được dùng để viết chương trình với PL-SQL



Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

6

6

Khai báo & khởi tạo đối tượng với PL/SQL



- Giả sử kiểu đối tượng `address_typ` được định nghĩa như sau:

```
CREATE TYPE address_typ AS OBJECT (  
    street VARCHAR2(30),  
    city VARCHAR2(30),  
    state CHAR(30));  
/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

7

7

Khai báo & khởi tạo đối tượng với PL/SQL



- Giả sử kiểu đối tượng `employee_typ` được định nghĩa:

```
CREATE TYPE employee_typ AS OBJECT (  
    employee_id NUMBER(6),  
    first_name VARCHAR2(20),  
    last_name VARCHAR2(25),  
    address address_typ,  
    MAP MEMBER FUNCTION get_idno RETURN NUMBER,  
    MEMBER PROCEDURE display_address ( SELF IN OUT NOCOPY  
    employee_typ ) );  
/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

8

8

Khai báo & khởi tạo đối tượng với PL/SQL



- employee_typ có phần thân các phương thức như sau:

```
CREATE TYPE BODY employee_typ AS
    MAP MEMBER FUNCTION get_idno RETURN NUMBER IS
BEGIN
    RETURN employee_id;
END;

    MEMBER PROCEDURE display_address ( SELF IN OUT NOCOPY
employee_typ ) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE(first_name || ' ' || last_name);
    DBMS_OUTPUT.PUT_LINE(address.street);
    DBMS_OUTPUT.PUT_LINE(address.city || ', ' || address.state);
END;
END;
/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

9

9

Khai báo & khởi tạo đối tượng với PL/SQL



- Khai báo và khởi tạo giá trị cho đối tượng

```
DECLARE
    emp employee_typ; -- Khởi tạo một đối tượng Null
BEGIN
    emp := employee_typ(2021, 'Le', 'Xuan', address_typ('Xa A',
'Huyen B', 'Tinh C'));
    emp.display_address();
END;
/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

10

10

Thao tác với đối tượng trong PL/SQL



- Truy cập thuộc tính sử dụng ký pháp “dấu chấm”

DECLARE

emp employee_typ;

BEGIN

emp := employee_typ(2021, 'Le', 'Xuan', address_typ('Xa A', 'Huyen B', 'Tinh C'));

DBMS_OUTPUT.PUT_LINE(emp.first_name || ' ' || emp.last_name);

DBMS_OUTPUT.PUT_LINE(emp.address.street);

DBMS_OUTPUT.PUT_LINE(emp.address.city || ', ' || emp.address.state);

END;

/

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

11

11

Thao tác với đối tượng trong PL/SQL



- Truy cập phương thức

CREATE TABLE employee_tab OF employee_typ;

DECLARE

emp employee_typ;

BEGIN

INSERT INTO employee_tab VALUES (employee_typ(2021, 'Le', 'Xuan', address_typ('Xa A', 'Huyen B', 'Tinh C')));

INSERT INTO employee_tab VALUES (employee_typ(1, 'Tran', 'Ha', address_typ('Xa M', 'Huyen N', 'Tinh P')));

END;

/

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

12

12

Thao tác với đối tượng trong PL/SQL



- Truy cập phương thức

```
DECLARE

    emp employee_typ;

BEGIN

    SELECT VALUE(e) INTO emp FROM employee_tab e
    WHERE e.employee_id = 1;

    emp.display_address();

END;

/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

13

13

Thao tác với đối tượng trong PL/SQL



- Sửa và xóa

```
DECLARE

    emp employee_typ;

BEGIN

    INSERT INTO employee_tab VALUES (employee_typ(2020, 'Nguyen',
    'New', address_typ('xa new', 'Huyen New', 'tinh New')) );

    UPDATE employee_tab e SET e.address.street = '182 Le Duan'
    WHERE e.employee_id = 2020;

    --DELETE FROM employee_tab e WHERE e.employee_id = 2020;

END;/

SELECT VALUE(e).first_name,VALUE(e).last_name, VALUE(e).address.street
from employee_tab e;
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

14

14

Thao tác với đối tượng trong PL/SQL



- Cập nhật dữ liệu sử dụng REF

```
DECLARE
    emp employee_typ;
    emp_ref REF employee_typ;
BEGIN
    SELECT REF(e) INTO emp_ref FROM employee_tab e WHERE
    e.employee_id = 2020;
    UPDATE employee_tab e
    SET e.address = address_typ('Dai lo Vinh - Cua lo', 'Cua lo', 'Nghe an')
    WHERE REF(e) = emp_ref;
END;/
SELECT VALUE(e).first_name, VALUE(e).last_name, VALUE(e).address.street
from employee_tab e;
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

15

15

Overloading, Overriding trong PL/SQL



- Overloading trong PL/SQL

- Thêm 1 phương thức mới cho một lớp cùng tên với phương thức được kế thừa

- Overriding trong PL/SQL

- Định nghĩa lại hành vi của phương thức được kế thừa cho một lớp
 - Không giống với overloading, không cần tạo phương thức mới mà chỉ định nghĩa lại hành vi của phương thức được kế thừa với từ khóa OVERRIDING

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

16

16

Overriding với kế thừa trong PL/SQL



- Overriding trong PL/SQL

```
CREATE OR REPLACE TYPE super_t AS OBJECT
(n NUMBER, MEMBER FUNCTION func RETURN NUMBER) NOT final;
/
CREATE OR REPLACE TYPE BODY super_t AS
MEMBER FUNCTION func RETURN NUMBER IS BEGIN RETURN 1; END; END;
/
CREATE TYPE sub_t UNDER super_t
(n2 NUMBER,
OVERRIDING MEMBER FUNCTION func RETURN NUMBER) NOT final;
/
CREATE OR REPLACE TYPE BODY sub_t AS
OVERRIDING MEMBER FUNCTION func RETURN NUMBER IS BEGIN RETURN 2; END;
END;
/
CREATE OR REPLACE TYPE final_t UNDER sub_t
(n3 NUMBER);/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

17

17

Overriding với kế thừa trong PL/SQL



- Overriding trong PL/SQL

```
DECLARE
  v0 super_t := super_t(2021);
  v1 sub_t := sub_t(1,2);
  v2 final_t := final_t(1,2,3);
BEGIN
  DBMS_OUTPUT.PUT_LINE('answer: ' || v0.func);
  DBMS_OUTPUT.PUT_LINE('answer: ' || v1.func);
  DBMS_OUTPUT.PUT_LINE('answer: ' || v2.func);
END;
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

18

18

Overriding với kế thừa trong PL/SQL



- Overriding trong PL/SQL với các phương thức được định nghĩa trong **PACKAGE**

```
CREATE OR REPLACE TYPE super_t AS OBJECT
(n NUMBER) NOT final;
/
CREATE OR REPLACE TYPE sub_t UNDER super_t
(n2 NUMBER) NOT final;
/
CREATE OR REPLACE TYPE final_t UNDER sub_t
(n3 NUMBER);
/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

19

19

Overriding với kế thừa trong PL/SQL



```
CREATE OR REPLACE PACKAGE p IS
    FUNCTION func (arg super_t) RETURN NUMBER;
    FUNCTION func (arg sub_t) RETURN NUMBER;
END;/
CREATE OR REPLACE PACKAGE BODY p IS
    FUNCTION func (arg super_t) RETURN NUMBER IS BEGIN RETURN 1; END;
    FUNCTION func (arg sub_t) RETURN NUMBER IS BEGIN RETURN 2; END;
END;/
DECLARE
    v0 super_t := super_t(2021);
    v1 sub_t := sub_t(1,2);
    v2 final_t := final_t(1,2,3);
BEGIN
    DBMS_OUTPUT.PUT_LINE(p.func(v0)); -- prints 1
    DBMS_OUTPUT.PUT_LINE(p.func(v1)); -- prints 2
    DBMS_OUTPUT.PUT_LINE(p.func(v2)); -- prints 2
END;/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

20

20

Package in Oracle



- 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 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 => Nâng cao tốc độ thực hiện lệnh
- Cấu trúc của Package: 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

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

21

21

```
CREATE OR REPLACE TYPE person_typ AS OBJECT
(name VARCHAR2(25), age NUMBER);
/
CREATE TYPE hobbies_var AS VARRAY(10) OF
VARCHAR2(25);
/
CREATE OR REPLACE PACKAGE teams
AUTHID CURRENT_USER AS
  PROCEDURE create_table (tab_name VARCHAR2);
  PROCEDURE insert_row (tab_name VARCHAR2, p
person_typ, h hobbies_var);
  PROCEDURE print_table (tab_name VARCHAR2);
END;
/
```

Sử dụng PL-SQL với đối tượng – TS. Phan Anh Phong

22

22



```

CREATE OR REPLACE PACKAGE BODY teams AS
PROCEDURE create_table (tab_name VARCHAR2) IS
BEGIN
EXECUTE IMMEDIATE 'CREATE TABLE ' || tab_name ||
' (pers person_typ, hobbs hobbies_var)';
END;
PROCEDURE insert_row (
tab_name VARCHAR2,
p person_typ,
h hobbies_var) IS
BEGIN
EXECUTE IMMEDIATE 'INSERT INTO ' || tab_name ||
' VALUES (:1, :2)' USING p, h;
END;
PROCEDURE print_table (tab_name VARCHAR2) IS
TYPE refcurtyp IS REF CURSOR;
v_cur refcurtyp;
p person_typ;
h hobbies_var;
BEGIN
OPEN v_cur FOR 'SELECT pers, hobbs FROM ' || tab_name;
LOOP
FETCH v_cur INTO p, h;
EXIT WHEN v_cur%NOTFOUND;
-- print attributes of 'p' and elements of 'h'
DBMS_OUTPUT.PUT_LINE('Name: ' || p.name || ' - Age: ' || p.age);
FOR i IN h.FIRST..h.LAST
LOOP
DBMS_OUTPUT.PUT_LINE('Hobby(' || i || '): ' || h(i));
END LOOP;
END LOOP;
CLOSE v_cur;
END;
END;

```

23



--Calling Procedures from the TEAMS Package

```

DECLARE
team_name VARCHAR2(15);
BEGIN
team_name := 'Notables';
TEAMS.create_table(team_name);
TEAMS.insert_row(team_name, person_typ('John', 31),
hobbies_var('skiing', 'coin collecting', 'tennis'));
TEAMS.insert_row(team_name, person_typ('Mary', 28),
hobbies_var('golf', 'quilting', 'rock climbing', 'fencing'));
TEAMS.print_table(team_name);
END;
/

```

24