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 với kế thừa bằng PL-SQL

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

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 quan hệ và quan hệ-đối tượng
- PL – SQL với kiểu đối tượng, kế thừa…

ORACLE DATABASE

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 có thể tạo thủ tục, hàm, package, triggers, những thứ được lưu trữ 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

4

Giới thiệu PL/SQL

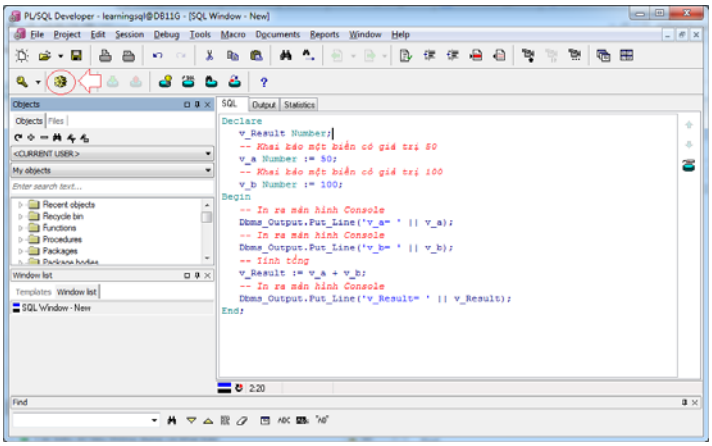
- Tổ chức khối lệnh trong PL-SQL

```
Declare
-- Phần 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…

Giới thiệu PL/SQL

- Công cụ PL/SQL Developer có thể được dung để lập trình với PL-SQL



Khai báo & khởi tạo đối tượng với PL/SQL

- Giả sử có kiểu đối tượng address_typ

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


7

Khai báo & khởi tạo đối tượng với PL/SQL

- Giả sử có kiểu đối tượng employee_typ

```
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 ) );
/
```


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;
/
```


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;
/
```

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;
/
```


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;
/
```

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;
/
```


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;
```


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;
```


Nạp chồng với kế thừa trong PL/SQL

- Overiding 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);/
```

Nạp chồng với kế thừa trong PL/SQL

- Overiding 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;
```


Nạp chồng với kế thừa trong PL/SQL

- Overiding 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);
/
```

Nạp chồng 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;/
```