Phương thức và kế thừa

Mục tiêu:

- Khái quát được phương thức và kế thừa
- Biết cách xây dựng phương thức trong Oracle
- Áp dụng được phương thức và kế thừa vào các trường hợp thực tiễn

1

Nội dung

- Đặt vấn đề
- Phương thức
- Kế thừa
- Luyện tập
- Tóm tắt

2

Đặt vấn đề

- Mô hình quan hệ - đối tượng = bảng + một số đặc trưng của mô hình hướng đối tượng
  - Sử dụng bảng là kiến trúc nền
  - Kiểu dữ liệu mới: kiểu đối tượng, kiểu REF, VARRAY, NESTED TABLE
  - Phương thức
  - Kế thừa
  - …
- => Có khả năng biểu diễn dữ liệu phức tạp và hỗ trợ đa truy cập, tìm kiếm thuận lợi
- Ngôn ngữ SQL2003

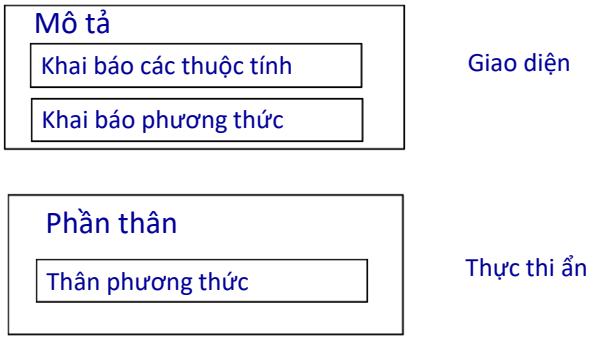
ORACLE DATABASE

Kiểu đối tượng

- Kiểu đối tượng = Kiểu người dùng định nghĩa = UDT
- Kiểu đối tượng xây dựng từ thuộc tính, phương thức
- Phương thức trong Oracle:
  - Khai báo phương thức khi định nghĩa kiểu đối tượng, sử dụng CREATE TYPE
  - Định nghĩa phương thức (phần thân) được xây dựng sau khi định nghĩa đối tượng và sử dụng CREATE TYPE BODY
  - Phương thức được viết bằng PL/SQL hoặc các ngôn ngữ lập trình C++, java, …

Object Type *person_typ*	
Attributes	**Methods**
idno	get_idno
first_name	display_details
last_name	
email	
phone	

Thuộc tính và phương thức của UDT



5

Phương thức

- Ví dụ về khai báo một phương thức:

```
CREATE OR REPLACE TYPE KieuNguoi AS OBJECT(
Hoten VARCHAR2(30),
Dienthoai VARCHAR2(20),
Namsinh NUMBER,
MEMBER FUNCTION tuoi RETURN NUMBER -- khai báo phương thức
);
/
```

- Ví dụ về định nghĩa phần thân cho phương thức

```
CREATE OR REPLACE TYPE BODY  KieuNguoi AS
MEMBER FUNCTION tuoi RETURN NUMBER IS
BEGIN
  RETURN EXTRACT(YEAR FROM SYSDATE)-namsinh;
END tuoi;
END;
/
```


6

Phương thức

- Phân loại phương thức:
  - CONSTRUCTOR
    - Các hàm thiết lập được xây dựng sẵn, như trong C++.
  - MEMBER
    - Dùng để truy nhập vào dữ liệu của đối tượng, tác đông vào thể hiện của đối tượng
    - Lời gọi: Thể_hiện_của_đối_tượng.tên_phương_thức([ds tham số])
    - Có các loại: MEMBER function (MEMBER procedure), MAP MEMBER function và ORDER MEMBER function
  - STATIC
    - Xác định trên kiểu đối tượng
    - Lời gọi: Tên_kiểu_đối_tượng.tên_phương_thức([ds tham số])
    - Có phạm vi thao tác toàn cục trên kiểu dữ liệu

Phương thức constructor – Hàm thiết lập

- Trong Oracle mỗi đối tượng có 1 hàm thiết lập ẩn, được định nghĩa bởi hệ thống, là tên của kiểu đối tượng
- Hàm thiết lập trả về một thể hiện mới (đối tượng mới) của kiểu đối tượng và khởi tạo các giá trị thuộc tính cho đối tượng đó
- Ví dụ:

  Declare

  p KieuNguoi;

  p := NEW KieuNguoi('Giap', NULL, 1990);

  Phương thức constructor với tên KieuNguoi(thuộc_tính_1, thuộc_tính_2, thuộc_tính_3) được gọi để tạo ra một đối tượng mới, có các giá trị 'Giap', Null, 1990 cho các thuộc tính hoten, dienthoai, namsinh và gán đối tượng đó với biến p

Phương thức Member

- Phương thức Member được dùng để truy nhập tới các giá trị của một đối tượng
- Giả sử có kiểu đối tượng KieuNguoi

```
CREATE OR REPLACE TYPE KieuNguoi AS OBJECT(
Hoten VARCHAR2(30),
Dienthoai VARCHAR2(20),
Namsinh NUMBER,
MEMBER FUNCTION tuoi RETURN NUMBER);
/

CREATE OR REPLACE TYPE BODY  KieuNguoi IS
MEMBER FUNCTION tuoi RETURN NUMBER IS
BEGIN
  RETURN  EXTRACT(YEAR FROM SYSDATE)-namsinh;
END tuoi;
END;
/
```


Phương thức Member

```
--tạo bảng tblSinhVien

CREATE TABLE tblSinhVien
(
mssv NUMBER(5) PRIMARY KEY,
thongtin KieuNguoi
);
/
```

Truy nhập phương thức Member

- <thể_hiện_đối_tượng>.<Tên_phương_thức> [(danh sách các tham số,)]
- Ví dụ

```
INSERT INTO tblSinhVien VALUES
(1, KieuNguoi('GIAP',NULL, 1990));

SELECT sv.mssv, sv.thongtin.hoten,
sv.thongtin.namsinh FROM tblSinhVien sv;

SELECT sv.mssv,
  sv.thongtin.hoten,sv.thongtin.tuoi()
FROM tblSinhVien sv;
```


Phương thức member MAP, member ORDER

- Để so sánh và sắp xếp các đối tượng của một kiểu đối tượng (class) chúng ta cần mô tả cơ sở của việc so sánh chúng (tức là mô tả khía cạnh cần so sánh)
- Sử dụng map member method hoặc order member method để so sánh các đối tượng
- Được khai báo bằng từ khóa map member method hoặc order member method trong một kiểu đối tượng
- Phần thân của phương thức được xây dựng tương tự member method
- Lưu ý rằng: mỗi kiểu đối tượng chỉ có nhiều nhất 1 map method hoặc 1 order method

Phương thức member MAP

- Phương thức map method trả về một giá trị thông qua việc tính toán các giá trị trên các thuộc tính của đối tượng và do vậy map method có thể sử dụng để so sánh, sắp xếp các đối tượng trong một kiểu đối tượng
- Phương thức Map được gọi tự động để đánh giá việc so sánh các đối tượng, chẳng hạn obj_1 > obj_2 và do đó có thể đi kèm với DISTINCT, GROUP BY, UNION, hay ORDER BY

  Khi có obj_1 > obj_2

  tương đương với

  obj_1.mapMethod() > obj_2.mapMethod()

Phương thức Member và Member Map

Ví dụ:

```
CREATE OR REPLACE TYPE BODY kieu_hcn AS
MAP MEMBER FUNCTION dientich RETURN NUMBER IS
--MEMBER FUNCTION dientich RETURN NUMBER IS
 BEGIN
    RETURN cd*cr;
 END dientich;
END;
/
CREATE OR REPLACE TYPE kieu_hcn AS OBJECT (
  cd NUMBER,
  cr NUMBER,
 MAP MEMBER FUNCTION dientich RETURN NUMBER);
 --MEMBER FUNCTION dientich RETURN NUMBER);
/
```

Phương thức Member và Member Map

```
DECLARE
  p kieu_hcn;  q kieu_hcn;
 BEGIN
  p := NEW kieu_hcn(10,5);
  q := NEW kieu_hcn(10,10);
-- if (p.dientich()<q.dientich()) then
 if (p<q) then
   DBMS_OUTPUT.PUT_LINE('Dien tich p nho hon dien tich q');
 end if;
 if (p>q) then
  DBMS_OUTPUT.PUT_LINE('Dien tich P LON HON dien tich Q');
 end if;
 if (p=q) then
  DBMS_OUTPUT.PUT_LINE('Dien tich P BANG DIEN TICH Q');
 end if;
END;
/
```


Phương thức Member order

- Phương thức Order được sử dụng để so sánh trực tiếp đối tượng với đối tượng, cho biết đối tượng đang xét lớn hơn/nhỏ hơn/bằng đối tượng khác
- Phương thức Order trả về một số nguyên <0, =0, >0 khi giá trị của đối tượng SELF là <, =, > giá trị của đối tượng làm đối cho phương thức này
- Phương thức order có thể đi với WHERE, ORDER BY

Phương thức Member order

- Ví dụ:

```
CREATE or REPLACE TYPE kieu_hinh_tron AS OBJECT(
  x NUMBER,
  y NUMBER,
  bk NUMBER,
ORDER MEMBER FUNCTION so_sanh(c kieu_hinh_tron)
RETURN INTEGER ); /
```


Phương thức Member order

```
CREATE OR REPLACE TYPE BODY kieu_hinh_tron AS
ORDER MEMBER FUNCTION so_sanh(c kieu_hinh_tron) RETURN INTEGER IS
   BEGIN
        IF bk < c.bk THEN
               RETURN -1;
        END IF;
        IF bk > c.bk THEN
               RETURN 1;
        END IF;
        IF bk = c.bk THEN
               RETURN 0;
        END IF;
   END;
END;
```

Phương thức Member order

```
CREATE TABLE tblHinh_tron OF kieu_hinh_tron;


INSERT INTO tblHinh_tron VALUES (kieu_hinh_tron(10, 10, 3));
INSERT INTO tblHinh_tron VALUES (kieu_hinh_tron(40, 20, 8));
INSERT INTO tblHinh_tron VALUES (kieu_hinh_tron(10, 50, 4));

SELECT ht.x, ht.y, ht.bk
FROM tblHinh_tron ht
WHERE VALUE(ht) < (kieu_hinh_tron(40, 25, 5));
```


Kế thừa

- Kế thừa
  - Lớp mới (sub-class) có thể mở rộng, kế thừa những lớp có sẵn (super-class). Sub-class có thể bổ sung những thuộc tính, phương thức mới
  - Quan hệ kế thừa là quan hệ kiểu "thành viên" ("is-a"), nghĩa là mỗi đối tượng của lớp con cũng là đối tượng của lớp cha và ngược lại đối tượng của lớp cha đại diện cho mội đối tượng của lớp con.
  - Quan hệ kế thừa tuân theo qui luật 100%, nghĩa là các đối tượng của lớp con đều có tất cả các mối quan hệ giống như đối tượng của lớp cha đối với lớp khác
  - SQL3 không hỗ trợ đa thừa kế, ta chỉ xét đơn kế thừa

Kế thừa

- Định nghĩa kế thừa

```
CREATE OR REPLACE TYPE <Lớp_con> UNDER <Lớp_cha>
(thuộc_tính, phương_thức lớp con)
```

Chú ý: Định nghĩa lớp cha phải có từ khóa NOT FINAL thì lớp con mới có thể kế thừa

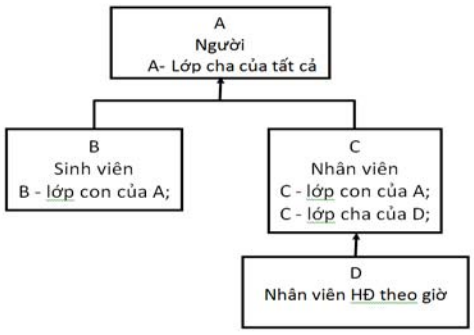
Ví dụ

```
CREATE OR REPLACE TYPE Type_Nguoi AS OBJECT(
Maso NUMBER,
Hoten VARCHAR2(30),
Diachi VARCHAR2(50)) NOT FINAL;/


CREATE OR REPLACE TYPE Type_SinhVien UNDER type_Nguoi(
Nganhhoc VARCHAR2(30));/
```


Kế thừa

- Ví dụ về kế thừa:



Kế thừa

- Giả sử Người có 3 thuộc tính: maso, hoten, diachi
- Sinh viên có thêm thuộc tính: nganhhoc
- Nhân viên có thêm thuộc tính: msnv, luong
- Nhân viên tính giờ có thêm thuộc tính: sogiolam

```
CREATE OR REPLACE TYPE Type_Nguoi AS OBJECT(
Maso NUMBER,
Hoten VARCHAR2(30),
Diachi VARCHAR2(50)) NOT FINAL;/

CREATE OR REPLACE TYPE Type_SinhVien UNDER type_Nguoi(
Nganhhoc VARCHAR2(30));/

CREATE OR REPLACE TYPE Type_NhanVien UNDER type_Nguoi(
Msnv NUMBER,
Luong NUMBER) NOT FINAL;/

CREATE OR REPLACE TYPE Type_NVgio UNDER type_NhanVien(
Sogiolam NUMBER);/
```


Kế thừa

- Bổ sung 1 dòng (đối tượng) vào bảng, tương tự bổ sung 1 dòng vào bảng có cột kiểu UDT

```
CREATE TABLE tblNguoi OF type_Nguoi;

INSERT INTO tblNguoi VALUES (type_Nguoi(2015, 'BINH',
'HT'));

INSERT INTO tblNguoi VALUES (type_SinhVien(1, 'GIAP',
'NA', 'CNTT'));

INSERT INTO tblNguoi VALUES (type_NhanVien(2, 'AT',
'NA', 2015, 50000000));

INSERT INTO tblNguoi VALUES (type_NVgio(4, 'DINH',
'TH', 2016, NULL,100 ));
```

Kế thừa

- Đọc thông tin

Ví dụ:

```
SELECT * FROM tblNguoi;
```

Đưa ra tất cả các dòng (đối tượng) trong bảng tblNguoi nhưng chỉ hiển thị các thuộc tính của kiểu type_Nguoi

(kiểu thành viên, is-a)

- Đọc dữ liệu từ bảng sử dụng hàm VALUE (bí_danh_bảng)

Hàm VALUE() trả về các dòng được lưu trữ trong bảng đối tượng qua bí danh bảng;

Các thể hiện đối tượng này có kiểu tương tự kiểu của bảng đối tượng

Ví dụ:

```
SELECT * FROM tblNguoi;
```

tương đương với

```
SELECT VALUE(n).hoten, value(n).diachi,
VALUE(n).maso FROM tblNguoi n;
```


Kế thừa

- Đưa ra các thuộc tính chung ở bảng của các thể hiện ở lớp con (sub-class): kết hợp SELECT <các thuộc tính> FROM <tên bảng> với mệnh đề WHERE VALUE(bí_danh_bảng) IS OF (Kiểu_lớp_con);

- Ví dụ:

Cho biết maso, hoten, diachi của các Nhân viên (tất cả nhân viên - các thể hiện của các lớp con của lớp type_NhanVien)

```
SELECT n.maso, n.hoten, n.diachi
FROM tblNguoi N
WHERE VALUE(N) IS OF (type_NhanVien);
```

Cho biết maso, hoten, diachi của các Nhân viên làm việc theo giờ

```
SELECT n.maso, n.hoten, n.diachi
FROM tblNguoi N
WHERE VALUE(N) IS OF (type_NVgio);
```

Kế thừa

- Đọc thông tin ở lớp con (sub-class) sử dụng hàm TREAT()

  Cú pháp: TREAT (<Biểu_thức> AS <Kiểu dữ liệu mới>)

  Mục đích: "Thay đổi" kiểu dữ liệu đã khai báo của biểu_thức thành "kiểu dữ liệu mới", tức là xem <Biểu_thức> như <Kiểu dữ liệu mới>

  Ví dụ:

  SELECT TREAT(VALUE(N) AS type_SinhVien).hoten, TREAT(VALUE(N) AS type_SinhVien).nganhhoc

  FROM tblNguoi N

  -- Xem mỗi thể hiện trong bảng tblNguoi như một thể hiện của kiểu SinhVien. Thể hiện nào không là Kiểu sinh viên sẽ không xem được thông tin (giá trị Null)

Kế thừa – truy xuất

- Đọc thông tin của các đối tượng ở lớp con kết hợp TREAT() và IS OF <(kiểu_dữ_liệu_lớp_con)>

- Ví dụ:

  SELECT TREAT(VALUE(N) AS type_SinhVien).hoten,
  TREAT(VALUE(N) AS type_SinhVien).nganhhoc
  FROM tblNguoi N
  WHERE VALUE(N) IS OF (type_SinhVien)

Kế thừa

- Ví dụ về TÌM KIẾM:

```
SELECT n.maso, n.hoten, n.diachi,
TREAT(VALUE(N) AS type_NhanVien).msnv,
TREAT(VALUE(N) AS type_NhanVien).luong
FROM tblNguoi N
WHERE VALUE(N) IS OF (type_NhanVien);
-- Câu lệnh trên làm gì?

SELECT n.hoten, TREAT(VALUE(N) AS
type_SinhVien).nganhhoc
FROM tblNguoi N
WHERE VALUE(N) IS OF (type_SinhVien);

SELECT n.hoten, TREAT(VALUE(N) AS
type_SinhVien).nganhhoc
FROM tblNguoi N
WHERE VALUE(N) IS NOT OF (type_SinhVien);
```


Kế thừa

- Ví dụ về DELETE:

```
--xoa sinh vien co ma so la 1
--thuoc tinh maso cua bang
DELETE FROM tblNguoi
WHERE maso=1;

-- xoa tat ca sinh vien
DELETE FROM tblNguoi N
WHERE VALUE(N) IS OF (type_Sinhvien)

--xoa sinh vien co nganh hoc la 'cntt'
DELETE FROM tblNguoi N
WHERE VALUE(N) IS OF (type_Sinhvien)
AND TREAT(VALUE(N) AS
type_Sinhvien).nganhhoc='CNTT';
```