

## Chương III

### Các hệ cơ sở dữ liệu hướng đối tượng

Nguyễn Kim Anh  
anhnk-fit@mail.hut.edu.vn  
Bộ môn Hệ thống Thông tin, SoICT

1

## Nội dung

- Các vấn đề liên quan đến các DBMS quan hệ
- Các DBMS thể hệ thứ ba
- Mô hình dữ liệu hướng đối tượng
- Hệ quản trị CSDL hướng đối tượng (OODBMS)
- Ngôn ngữ định nghĩa đối tượng (ODL)
- Ngôn ngữ truy vấn đối tượng (OQL)

2

### Các DBMS quan hệ

#### Các vấn đề

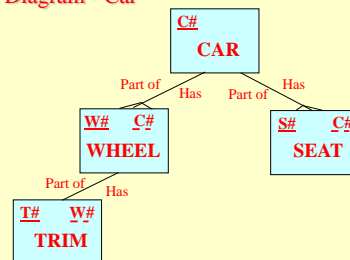
- ⇒ Sự biểu diễn nghèo nàn các thực thể 'thế giới thực'
- ⇒ Chống chéo ngữ nghĩa
- ⇒ Hỗ trợ kiểm soát các ràng buộc toàn vẹn và các nghiệp vụ nghèo nàn
- ⇒ Cấu trúc dữ liệu thuần nhất
- ⇒ Các thao tác hạn chế
- ⇒ Khó điều khiển các truy vấn đệ qui
- ⇒ Trở ngại không phù hợp
- ⇒ Khó điều khiển 'các giao dịch dài'

3

### Các DBMS quan hệ

#### Các đối tượng thế giới thực

#### ER Diagram - Car



4

## Các DBMS quan hệ

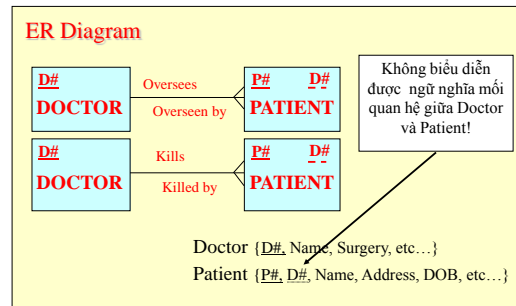
### Các đối tượng thể giới thực

Chuẩn hóa	
Car { <u>C#</u> , Chassis#, NoWheels, NoSeats, etc...}	
Wheel { <u>W#</u> , C#, Size, Pressure, etc...}	
Trim { <u>T#</u> , W#, Material, Cost, etc...}	
Seat { <u>S#</u> , C#, Material, Size, Cost., etc...}	
Tìm tất cả các chi tiết về Car, chúng ta phải thực hiện nhiều phép <b>JOIN</b> với chi phí rất cao	
<pre> Select * From Car, Wheel, Trim, Seat Where Car.C# = Wheel.C# And Car.C# = Seat.C# And Wheel.W# = Trim.W#;</pre>	

5

## Các DBMS quan hệ

### Chồng chéo ngữ nghĩa



6

## Các DBMS quan hệ

### Cấu trúc dữ liệu thuần nhất

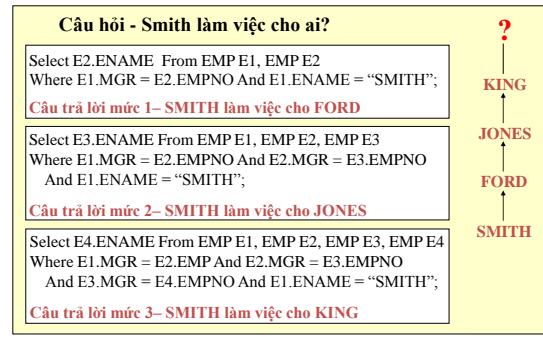
ACCOUNT TABLE			
ACCOUNT	CUSTOMER	BRANCH	BALANCE
200	JONES	STRATFORD	1000.00
324	GRAY	BARKING	200.00
345	SMITH	STRATFORD	23.17
350	GREEN	BARKING	340.14
400	ONO	BARKING	500.00
456	KHAN	STRATFORD	333.00

Tất cả các dòng đều có cùng số thuộc tính  
Tất cả các giá trị trong 1 cột đều có cùng 1 kiểu  
Tất cả các giá trị trong 1 cột đều là nguyên tố

7

## Các DBMS quan hệ

### Các truy vấn đệ qui



8

## Các DBMS quan hệ

### Trở ngại không phù hợp

SQL là một ngôn ngữ khai báo dựa trên tập mà không hoàn thiện về khả năng tính toán!

Các ứng dụng CSDL cần sử dụng một ngôn ngữ thủ tục dựa trên bản ghi hoàn thiện về khả năng tính toán như C, C++, Java, và PL/SQL.

Chúng ta cần phải ánh xạ tập dữ liệu thành các bản ghi sử dụng các cấu trúc dữ liệu như các con trỏ.

Điều này đòi hỏi chi phí cao về thời gian thực hiện ứng dụng và sự nỗ lực về lập trình mà chiếm khoảng 30% chi phí thực hiện các dự án!

9

## Các DBMS thế hệ thứ ba

### Mục đích

- **Hỗ trợ các đối tượng hoạt động phức tạp**

Cho phép mô hình hóa cả dữ liệu và hành vi ở một mức phức tạp nào đó.

- **Cải thiện khả năng mở rộng**

Cho phép mở rộng một cách linh hoạt cả các kiểu dữ liệu cho phép và các hành vi kết hợp với các kiểu này.

- **Giảm trở ngại không phù hợp**

Đảm bảo có một sự hợp nhất giữa mô hình dữ liệu được sử dụng bởi DBMS và các kiểu dữ liệu được sử dụng bởi ngôn ngữ lập trình truy cập các dữ liệu này.

10

## Mô hình dữ liệu hướng đối tượng

### Đối tượng nguyên tố là gì?

Một đối tượng nguyên tố luôn chứa một giá trị cố định và được sử dụng giống như 1 hằng trong ngôn ngữ lập trình.

Một đối tượng nguyên tố không thể thay đổi trạng thái của nó.

Ví dụ về các kiểu nguyên tố và các đối tượng nguyên tố

**Integer** - e.g. 1, 2, 3, -5, -45, etc.....

**Float** - e.g. 1.52, -0.3456, 2.000, etc...

**Boolean** - i.e. True or False

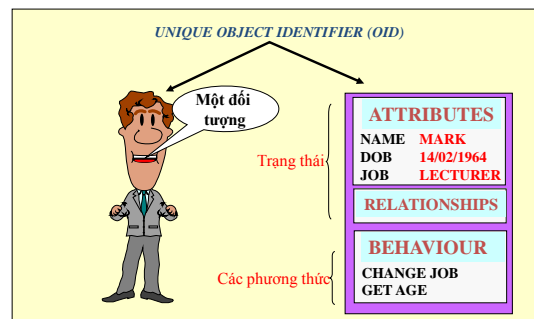
**Char** - e.g. a, b, c, @, #, !, etc...

**String** - e.g. "Mark", "Database Systems"

11

## Mô hình dữ liệu hướng đối tượng

### Một đối tượng có thể thay đổi



12

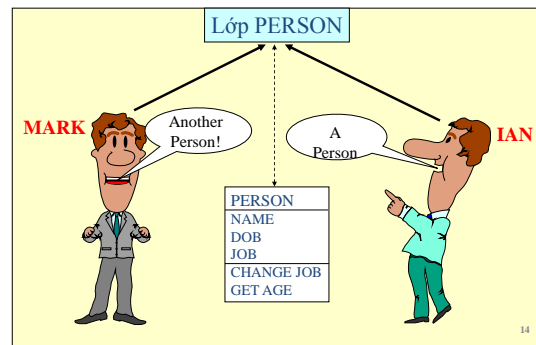
## Mô hình dữ liệu hướng đối tượng

Một lớp là gì?



## Mô hình dữ liệu hướng đối tượng

Một lớp là gì?



## Mô hình dữ liệu hướng đối tượng

Định danh đối tượng (OID) là gì ?

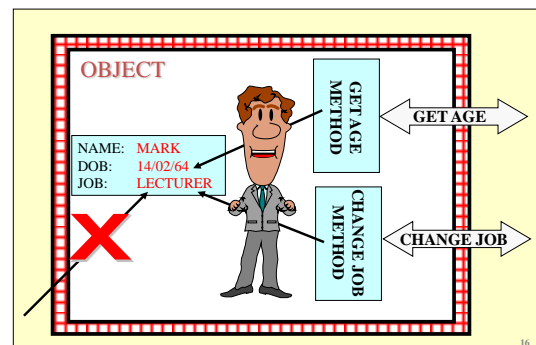
Mỗi đối tượng được xác định duy nhất từ tất cả các đối tượng khác. Khi một đối tượng được tạo lần đầu, nó được gán 1 giá trị để xác định nó. Giá trị này là OID của đối tượng.

- ⇒ Do hệ thống sinh ra.
- ⇒ Duy nhất đối với đối tượng này.
- ⇒ Bất biến hay không thể thay đổi.
- ⇒ Độc lập đối với các giá trị thuộc tính của nó.
- ⇒ Không thấy được đối với người dùng.

15

## Mô hình dữ liệu hướng đối tượng

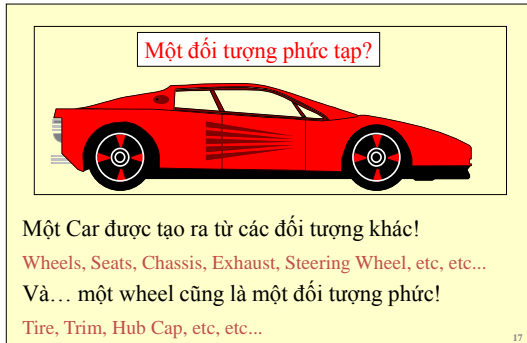
Đóng gói là gì?



16

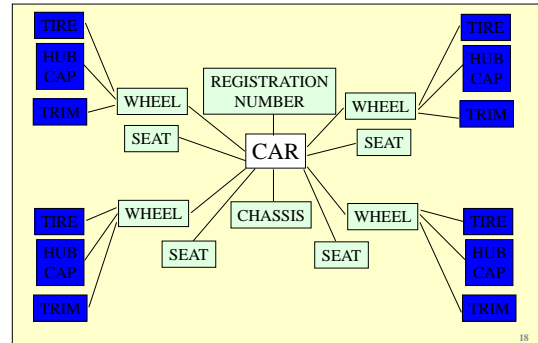
## Mô hình dữ liệu hướng đối tượng

Một đối tượng phức tạp là gì?



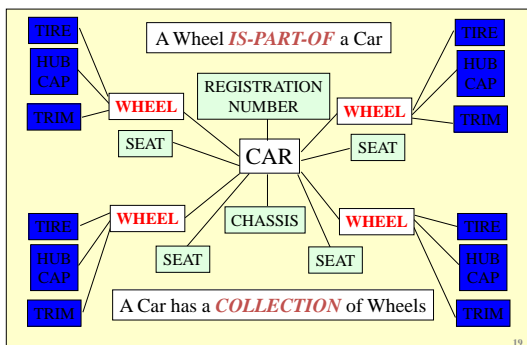
## Mô hình dữ liệu hướng đối tượng

Một đối tượng phức tạp là gì?



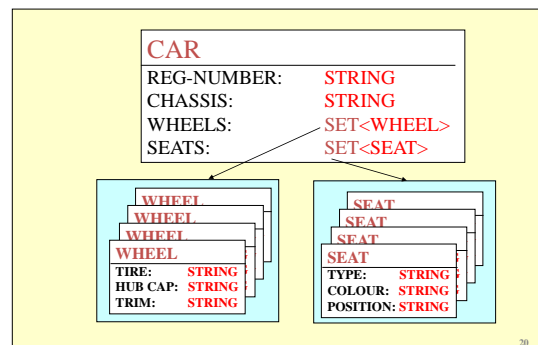
## Mô hình dữ liệu hướng đối tượng

Một đối tượng phức tạp là gì?



## Mô hình dữ liệu hướng đối tượng

Một đối tượng phức tạp là gì?

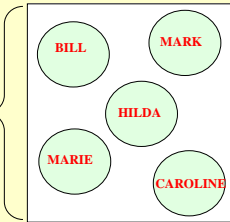


## Mô hình dữ liệu hướng đối tượng

### SET là gì?

**SET** – Một tập các đối tượng phân biệt không có thứ tự có cùng kiểu.  
 Ví dụ: Customers : SET <Customer>;

Một thể hiện của  
CUSTOMERS



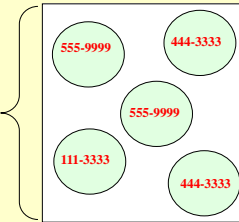
21

## Mô hình dữ liệu hướng đối tượng

### BAG là gì?

**BAG** - Một tập các đối tượng không có thứ tự có cùng kiểu.  
 Ví dụ: Phone\_calls : BAG <TelephoneNumber>;

Một thể hiện của  
PHONE\_CALLS



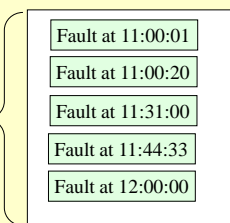
22

## Mô hình dữ liệu hướng đối tượng

### LIST là gì?

**LIST** - Một tập các đối tượng có thứ tự có cùng kiểu.  
 Ví dụ: MachineFaults : LIST <Fault>;

Một thể hiện của  
MachineFaults



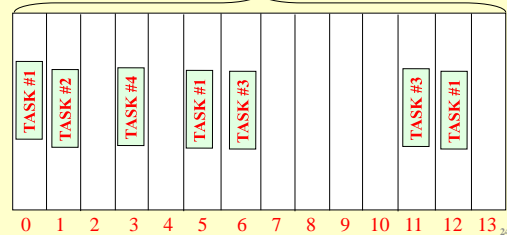
23

## Mô hình dữ liệu hướng đối tượng

### ARRAY là gì?

**ARRAY** – mỗi đối tượng được lưu trữ tại một vị trí đặc biệt  
 Ví dụ: StudySchedule : ARRAY <Task>;

An instance of StudySchedule



24

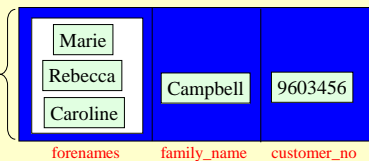
## Mô hình dữ liệu hướng đối tượng

### STRUCTURE là gì?

Một số xác định các thành phần được đặt tên, mỗi thành phần có thể chứa một đối tượng có một kiểu cụ thể.

Ví dụ: CustomersDetails : **STRUCTURE** <  
 forenames : List < String >,  
 family\_name : String,  
 customer\_no : Integer >

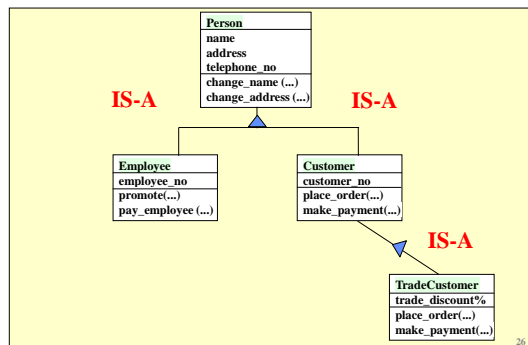
Một thể hiện của  
CustomersDetails



25

## Mô hình dữ liệu hướng đối tượng

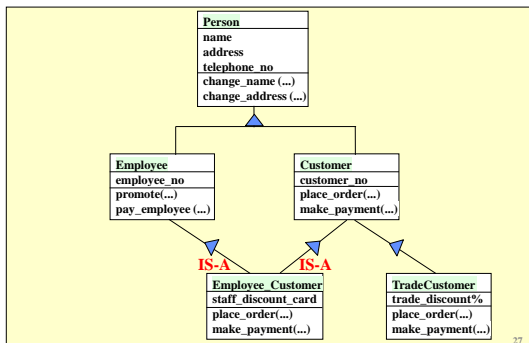
### Kế thừa là gì?



26

## Mô hình dữ liệu hướng đối tượng

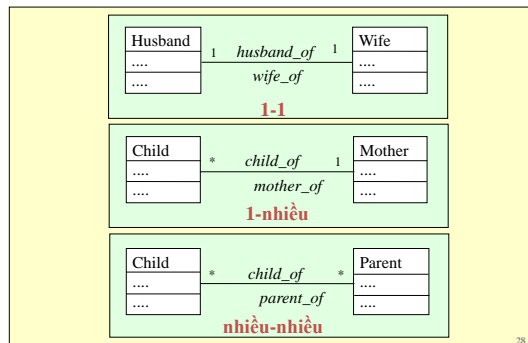
### Đa kế thừa là gì?



27

## Mô hình dữ liệu hướng đối tượng

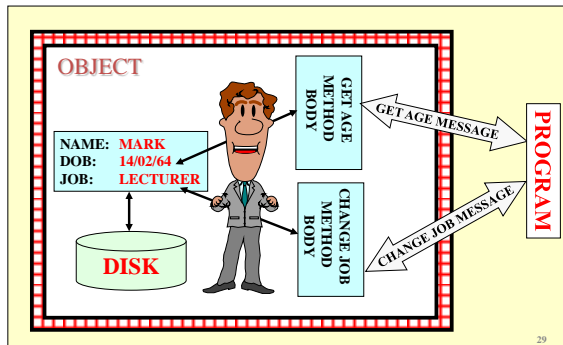
### Các quan hệ đối tượng là gì?



28

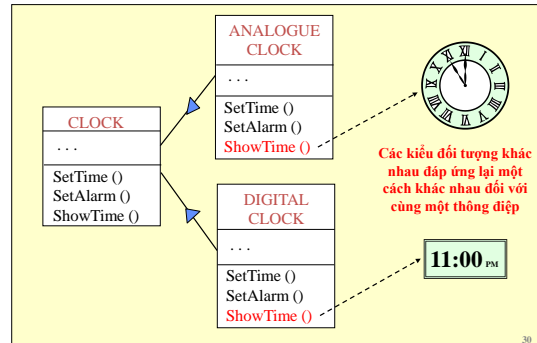
## Mô hình dữ liệu hướng đối tượng

### METHODS và MESSAGES?



## Mô hình dữ liệu hướng đối tượng

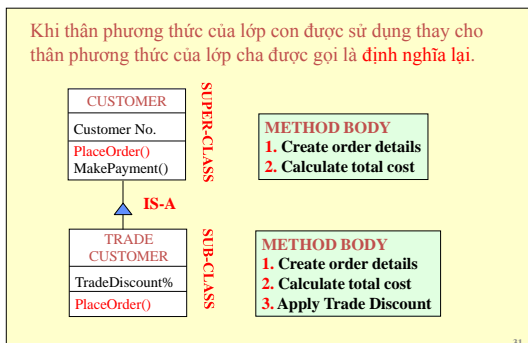
### Đa hình là gì?



## Mô hình dữ liệu hướng đối tượng

### Định nghĩa lại là gì?

Khi thân phương thức của lớp con được sử dụng thay cho thân phương thức của lớp cha được gọi là **định nghĩa lại**.



## Mô hình dữ liệu hướng đối tượng

### Liên kết muộn là gì?

Liên kết muộn hay liên kết động là khả năng xác định thân phương thức nào được thực hiện phụ thuộc vào kiểu đối tượng của hệ thống trong thời gian thực hiện.





## Mô hình dữ liệu hướng đối tượng

### Các khái niệm cơ bản

- Mỗi thực thể thế giới thực được mô hình hóa là một đối tượng. Mỗi đối tượng được kết hợp với một định danh duy nhất (OID)
- Mỗi đối tượng được mô tả với một tập thuộc tính thể hiện (các biến thể hiện) và các phương thức.
  - Giá trị của một thuộc tính có thể là 1 đối tượng hay một tập các đối tượng
  - Một đối tượng phức có thể được xây dựng bởi sự kết tập của các đối tượng khác.
  - Tập các thuộc tính của đối tượng và tập các phương thức biểu diễn cấu trúc và hành vi của đối tượng.

33

## Mô hình dữ liệu hướng đối tượng

### Các khái niệm cơ bản

- Các giá trị thuộc tính của một đối tượng biểu diễn trạng thái của đối tượng.
  - Trạng thái được truy cập và thay đổi bằng cách gửi các thông điệp đến đối tượng kéo theo các phương thức tương ứng.

34

## Mô hình dữ liệu hướng đối tượng

### Các khái niệm cơ bản

- Các đối tượng chia sẻ cùng cấu trúc và hành vi được nhóm lại thành các lớp.
  - ✓ Một lớp biểu diễn một khuôn mẫu đối với một tập các đối tượng tương tự.
  - ✓ Mỗi đối tượng là thể hiện của một lớp nào đó.
- Một lớp có thể được định nghĩa như một sự đặc biệt hóa của một hay một số lớp nào đó.
  - ✓ Một lớp được định nghĩa như một sự đặc biệt hóa là một lớp con và kế thừa các thuộc tính và các phương thức từ một (hay nhiều) lớp cha của nó.

35

## DBMS hướng đối tượng (OODBMS)

### OODBMS là gì?

Các DBMS hướng đối tượng (OODBMSs) luôn cố gắng kết hợp thế mạnh của các ngôn ngữ lập trình hướng đối tượng với các công nghệ đã rất phát triển của DBMS.

OODBMS	
<u>Ngôn ngữ lập trình HDT</u>	<u>DBMSs</u>
<ul style="list-style-type: none"> <li>Các đối tượng phức tạp</li> <li>Định danh đối tượng</li> <li>Các phương thức và thông điệp</li> <li>Kế thừa</li> <li>Đa hình</li> <li>Khả năng mở rộng</li> <li>Hoàn thiện khả năng tính toán</li> </ul>	<ul style="list-style-type: none"> <li>Sự tồn tại lâu dài</li> <li>Quản lý đĩa</li> <li>Chia sẻ dữ liệu</li> <li>Độ tin cậy</li> <li>Độ an toàn</li> <li>Truy vấn theo yêu cầu</li> </ul>

36

## DBMS hướng đối tượng

### OODBMS cần hỗ trợ?

- Các đối tượng nguyên tố và phức tạp
- Các phương thức và các thông điệp
- Định danh đối tượng
- Đơn kế thừa
- Đa hình – nạp chồng và liên kết muộn
- Sự tồn tại lâu dài
- Các đối tượng được chia sẻ

#### OODBMS có thể hỗ trợ

- Đa kế thừa
- Các thông điệp ngoại lệ (Exception messages)
- Phân tán
- Các giao dịch dài
- Các phiên bản

37

## DBMS hướng đối tượng

Một OODBMS là một DBMS hỗ trợ trực tiếp một mô hình dựa trên phương pháp hướng đối tượng.

- Cung cấp một ngôn ngữ đối với định nghĩa sơ đồ và thao tác trên các đối tượng cũng như sơ đồ của chúng.
- Cung cấp cách lưu trữ lâu dài các đối tượng.
- Cung cấp một ngôn ngữ truy vấn, các khả năng đánh chỉ mục, etc.

38

## Ngôn ngữ định nghĩa đối tượng (Object Definition Language -ODL)

- Đặc tả sơ đồ logic đối với một CSDL hướng đối tượng
- Dựa trên các đặc tả của Nhóm quản trị CSDL hướng đối tượng (Object Database Management Group - ODMG)

39

## Định nghĩa lớp

- **class** – từ khóa định nghĩa các lớp
- **attribute** – từ khóa đối với các thuộc tính
- **operations** – bao gồm kiểu trả về, tên, các tham số trong cặp ngoặc
- **relationship** – từ khóa thiết lập quan hệ

40

## Định nghĩa thuộc tính

- Giá trị thuộc tính có thể là:
  - OID hoặc literal (nguyên bản)
- Các kiểu literal
  - Nguyên tố – 1 hằng không thể phân tách được
  - Sưu tập – nhiều literal hay các kiểu đối tượng
  - Cấu trúc – 1 số xác định các phần tử có tên, trong đó mỗi phần tử có thể là 1 literal hay kiểu đối tượng
- Miền thuộc tính
  - Các giá trị cho phép đối với 1 thuộc tính
  - enum – liệt kê các giá trị cho phép

41

## Các kiểu sưu tập

- **Set** - Một tập các đối tượng phân biệt không có thứ tự có cùng kiểu.
- **Bag** - Một tập các đối tượng không có thứ tự có cùng kiểu
- **List** - Một tập các đối tượng có thứ tự có cùng kiểu
- **Array** - Một tập có thứ tự có kích thước động, định vị bởi vị trí
- **Dictionary** - Một dãy không có thứ tự các cặp (key-value) không lặp lại

42

## Định nghĩa các cấu trúc

Structure = kiểu người dùng định nghĩa với các thành phần

**struct** từ khóa

Ví dụ:

```
struct Address {
    String street_address
    String city;
    String state;
    String zip;
};
```

43

## Định nghĩa các thao tác

- Kiểu trả về
- Tên
- Các đối số trong cặp ngoặc

44

## Định nghĩa các quan hệ

- Chỉ cho phép định nghĩa các quan hệ 1 ngôi và 2 ngôi
- Các quan hệ là 2 chiều
  - Được cài đặt bằng cách sử dụng từ khóa **inverse**
- Các quan hệ được đặc tả với ODL:
  - relationship** chỉ định lớp được định nghĩa ở bên nhiều
  - relationship set** chỉ định lớp được định nghĩa ở bên 1 và các (nhiều) thể hiện của lớp tham chiếu không có thứ tự
  - relationship list** chỉ định lớp được định nghĩa ở bên 1 và các (nhiều) thể hiện của lớp tham chiếu có thứ tự

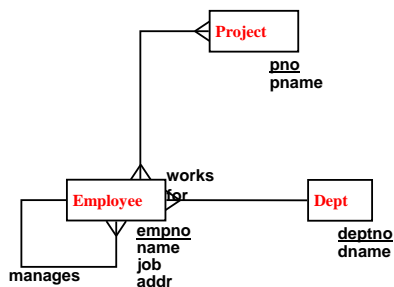
45

## Tạo các thể hiện đối tượng

- Khởi tạo các thuộc tính:
  - Cheryl student (name: "Cheryl Davis", dateOfBirth:4/5/77);
- Khởi tạo các thuộc tính đa giá trị:
  - Dan employee (emp\_id: 3678, name: "Dan Bellon", skills {"Database design", "OO Modeling"});
- Thiết lập các liên kết đối với quan hệ
  - Cheryl student (takes: {OOD99F, Telecom99F, Java99F});

46

## Ví dụ: Mô hình ER



## Ví dụ: ODL

```

class Empl {
    attribute int empno;
    attribute string name;
    attribute string job;
    attribute Address addr;
    relationship Dept dept inverse Dept::empls;
    relationship Empl mgr inverse manages;
    relationship Set<Empl> managees
        inverse mgr;
    relationship Set<Project> assignments
        inverse Project::members;
}

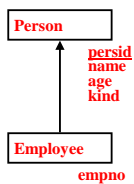
class Dept {
    attribute int deptno;
    attribute string dname;
    relationship Set<Empl> empl
        inverse Employee::dept;
}

class Project {
    attribute int pno;
    attribute string pname;
    relationship Set<Empl> members
        inverse Employee::assignments;
}
  
```

## Kế thừa trong ODL

```
class Person {
    attribute int persid,
    attribute string name,
    attribute int age,
    attribute string kind,
    ...
}

class Employee extends Person
{
    attribute int empno,
    ...
}
```



## Định nghĩa các quan hệ

• Một số các OODBMS bảo trì tự động các quan hệ ngược (inverse relationships).

–Trong Empl

- relationship **Set<Project>** assignments  
inverse **Project::members**;

–Khi 1 project được thêm vào assignments của 1 empl, empl này có thể được bổ sung tự động vào members của project đó.

• Các quan hệ 1:M và M:M có thể được biểu diễn bởi một kiểu sưu tập nào đó (set, bag, list,...).

–Trong Empl

- relationship **List<Project>** assignments  
inverse **Project::members**;

–Điều này cho phép các project của 1 empl được bảo trì theo một thứ tự quan trọng nào đó đối với empl này → khó bảo trì tự động các quan hệ ngược (nếu 1 empl được thêm vào 1 project, project này cần được bổ sung vào assignments của empl theo thứ tự nào????)

## Thuộc tính phức so với thuộc tính tham chiếu

**Person w**  
**attribute OrganStruct**

111-33-6174	SMITH
Kidney	23 cm
Liver	29 cm
Spleen	25 cm

Thuộc tính phức  
chứa trong đối tượng

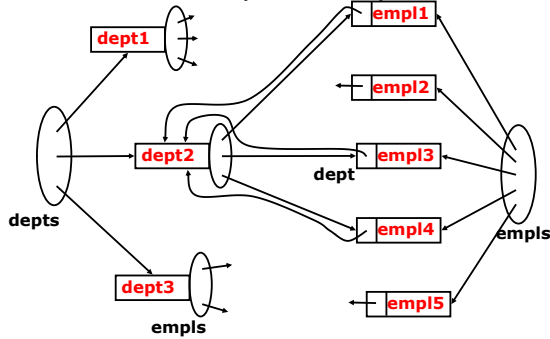
111-33-6174	SMITH	
Kidney	23 cm	
Liver	29 cm	
Spleen	25 cm	

Thuộc tính tham chiếu  
(có thể chia sẻ)

## Thuộc tính phức so với thuộc tính tham chiếu

- Các quan hệ luôn biểu diễn các tham chiếu có thể chia sẻ đối với các đối tượng
- Các thuộc tính phức có thể chứa các đối tượng (các cấu trúc hay các sưu tập chứa các đối tượng) nhưng không thể được chia sẻ

## Các sưu tập và các quan hệ



## Tổ chức lưu trữ đối với OODB

- Tất cả các đối tượng được lưu trữ một cách độc lập → có thể tăng một cách đáng kể chi phí duyệt một thể hiện (OODB tương đương với 1 bảng)
- Nhiều OODB's tự động nhóm lại các đối tượng được sử dụng cùng nhau trong cùng trang nhớ (v.d: một phòng & các nhân viên của phòng)
- Một số OODB's cho phép người sử dụng liệt kê/mô tả một cách tường minh các đối tượng nên nhóm lại cùng nhau
  - Các đối tượng được định danh bởi OIDs không phải ROWIDs → Các đối tượng từ các lớp khác nhau có thể chia sẻ cùng một khối!

## Đánh chỉ mục (non-standard)

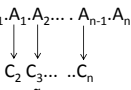
- Một số OODB's hỗ trợ tạo các chỉ mục trên các thể hiện & các sưu tập ổn định
- Giả sử emps là thể hiện đối với Employee (tập các nhân viên)
  - Attribute index  
Create index addr on emps
  - Multi-attribute index  
Create index job, addr on emps
- Đánh chỉ mục có thể đặc biệt hữu ích khi được sử dụng với các sưu tập ổn định so với các thể hiện

## Object Query Language (OQL)

- Ngôn ngữ chuẩn theo ODMG
- Tương tự với SQL-92
- Một số khác biệt:
  - Joins sử dụng tên quan hệ của lớp:
    - Select e.name from Empl e, **e.dept** d where d.dname = "Technique" and e.job = "Database Design";
  - Sử dụng sưu tập set trong 1 câu truy vấn
    - Select emp\_id, name from Empl where **"Database Design"** in **assignments**;
    - Select emp\_id, name from Empl where some **assignments** = **"Database Design"**;

## Object Query Language (OQL)

- Cú pháp dựa trên cú pháp của SQL (select, from, where) :
 

```
select <structured query result>
from <class [class variable]> [, <path>....]
where <path expressions>
```
- Ngôn ngữ truy vấn hướng đường dẫn
  - Đường dẫn :  $C_1.A_1.A_2 \dots A_{n-1}.A_n$ 

  - Biểu thức đường dẫn:  $C_1.A_1.A_2 \dots A_{n-1}.A_n = v$

## SQL so với OQL

- SQL
  - Trả về các tập kết quả dưới dạng bảng hay các giá trị vô hướng
  - Tìm kiếm qua nhiều bảng bằng cách sử dụng các phép joins
- OQL
  - Trả về các sưu tập gồm nhiều phần tử hay một phần tử đơn
  - Tìm kiếm qua nhiều sưu tập sử dụng các đường dẫn (đôi khi sử dụng joins)

## Tham chiếu/đường dẫn

- Cho một nhân viên e
    - e.name – tên của e
    - e.dept – phòng của e
    - e.dept.deptno – số hiệu phòng của phòng của e
    - e.dept.dname – tên của phòng của e
  - Cho một phòng d
    - d.empls – tập các nhân viên làm việc trong phòng d
- ~~d.empls.age – không hợp lệ vì d.empls là một tập~~

## Đường dẫn so với Joins

Đưa ra tên và phòng của Nhân viên với điều kiện phòng ở Boston

SQL:  
 SELECT e.name, d.dname  
 FROM Empl e NATURAL JOIN Dept d  
 WHERE d.loc = 'Boston'

OQL:  
 SELECT e.name, e.dept.dname  
 FROM Empl e  
 WHERE e.dept.loc = "Boston"

## Joins thông thường

Đưa ra các cặp nhân viên mà lương của họ chênh nhau dưới \$100

```
SELECT e1.name, e2.name
FROM Empl e1, Empl e2
WHERE e1.empno <> e2.empno
      AND abs(e1.sal - e2.sal) < 100
```

## Lựa chọn các sưu tập kết quả

Cho Set<Empl> emps

```
SELECT e.name
FROM emps e
WHERE e.age > 40
```

→ Bag<string>  
(nên trả về Set<string> nếu name là khóa)

---

Cho List<Empl> rankemps

```
SELECT e.name
FROM rankemps e
WHERE e.age > 40
```

→ Bag<string>  
(nên trả về List<string>)

## Lựa chọn các phần tử sưu tập

Cho Set<Empl> emps

```
SELECT e
FROM emps e
WHERE e.age > 40
```

→ Set<Emp>

---

Cho List<Empl> rankemps

```
SELECT e
FROM rankemps e
WHERE e.age > 40
```

→ Bag<Emp>  
(nên trả về List<Emp>)

## Sinh kết quả Sets, Lists & Bags

```
SELECT e.name
FROM Empl e
WHERE e.age > 40
```

→ Bag<string>

---

```
SELECT DISTINCT e.name
FROM Empl e
WHERE e.age > 40
```

→ Set<string>

---

```
SELECT e.name
FROM Empl e
WHERE e.age > 40
ORDER BY e.age
```

→ List<string>



Sinh kết quả sưu tập của Structs

Đổi tên thuộc tính

```
SELECT who: e.name, e.age
FROM Empl e
WHERE e.age > 40
```

→ Bag<struct( string who; int age )>

```
SELECT struct(
  who: e.name, age: e.age )
FROM Empl e
WHERE e.age > 40
```

Cấu trúc có thể đặc tả tường minh, nhưng tất cả các thuộc tính phải được đặt tên tường minh

Sinh kết quả Objects

```
SELECT struct(
  name: e.name, age: e.age )
FROM Empl e
WHERE e.age > 40
```

→ Bag<struct( string name; int age )>

```
SELECT Person(
  name: e.name, age: e.age - 20 )
FROM Empl e
WHERE e.age > 40
```

→ Bag<Person>

Table15-1 – OODBMS Products

Table 15-1 OODBMS Products		
Company	Product	Web Site
Computer Associates	Jasmine	http://www.cai.com/products/jasmine.htm
Franz	AllegroSCL	http://www.franz.com
Gemstone Systems	GemStone	http://www.gemstone.com
neoLogic	neoAccess	http://neoLogic.com
Object Design	ObjectStore	http://www.odi.com
Objectivity	Objectivity/DB	http://www.objectivity.com
POET Software	POET Object Server	http://www.poet.com
Versant	Versant ODBMS	http://www.versant.com
Other Links Related to OODBMS Products		
Barry & Associates		http://www.odbmfacts.com
Doug Barry's The Object Database Handbook		http://wiley.com
Object database newsgroup		news://comp.databases.object
Rick Cattell's The Object Database Standard		http://www.mkp.com
ODMG 3.0		
Object Database Management Group		http://www.odmg.org
Chaudhri and Zican's Succeeding with Object Databases		http://www.wiley.com/compbooks/chaudhri