**TOPIC #8C**

**CHUẨN NGÔN NGỮ - THIẾT KẾ CSDL HƯỚNG ĐỐI TƯỢNG**

**Object Database Standards, Languages, and Design**

1. TỔNG QUAN VỀ MÔ HÌNH ĐỐI TƯỢNG CỦA ODMG

Mô hình đối tượng ODMG là mô hình mà ngôn ngữ định nghĩa đối tượng (ODL) và ngôn ngữ truy vấn đối tượng (OQL) là nền tảng. Mô hình này cung cấp các kiểu dữ liệu, hàm dựng kiểu và những khác niệm khác có thể được sử dụng trong ODL để xác định lược đồ CSDL đối tượng. Vì thế, mô hình ODMG là mô hình dữ liệu chuẩn cho CSDL hướng đối tượng, giống như SQL mô tả mô hình dữ liệu chuẩn cho CSDL quan hệ.

* 1. **Object và Literal**

Object và literal là các yếu tố cơ bản của mô hình đối tượng. Object có cả ID nhận biết đối tượng và giá trị, trong khi literal chỉ có giá trị mà không có ID. Giá trị của đối tượng có thể có cấu trúc phức tạp và có thể thay đổi. Tuy nhiên giá trị của literal là cố định và không thay đổi, có thể có kiểu cấu trúc.

* Object có 4 đặc điểm:
  + Identifier: là định danh duy nhất trong hệ thống (OBJECT\_ID).
  + Name: duy nhất trong một CSDL hay một chương trình. Tên của đối tượng là tùy ý. Thông thường, những tên này được sử dụng như là một entry points trong CSDL; bằng cách xác định các đối tượng này theo tên, người sử dụng có thể định vị được các đối tượng khác được tham chiếu từ những đối tượng này.
  + Lifetime: cho biết đây là *persistent object* (database object) hay *transient object* (đối tượng trong quá trình thực thi chương trình và sẽ biến mất sau khi chương trình kết thúc).
  + Structure: cho biết một đối tượng được được xây dựng bằng hàm dựng kiểu như thế nào và nó có phải là đối tượng nguyên tử hay bộ sưu tập (atomic hay collection).
* Literal:
  + Có giá trị nhưng không có ID.
  + Có ba kiểu dữ liệu:
    - Atomic: đã được định nghĩa, giá trị kiểu dữ liệu cơ bản. (short, long, float, boolean, char…)
    - Collection: một bộ sưu tập (array…) của các giá trị hay đối tượng nhưng không có Object\_id. (set<T>, bag<T>, list<T>, array<T> với T là kiểu của các đối tượng hay giá trị, Dictionary<K,V> với K là khóa, V là giá trị)
    - Structured: các giá trị được xây dựng bởi các hàm dựng kiểu (date, các biến kiểu struct …)
* Sau đây là một ví dụ đơn giản về mô hình đơn giản. Mô hình ODMG sử dụng từ khóa interface để định nghĩa một lớp. Trong thực tế, từ khóa *interface* thích hợp hơn bởi vì nó mô tả interface của các loại đối tượng, cụ thể là các thuộc tính, mối quan hệ và các phương thức. Chúng được tạo ra nhằm mục đích phục vụ cho việc định nghĩa các phương thức có thể được **kế thừa (inherited)** bởi các đối tượng do người dùng định nghĩa cho các ứng dụng riêng biệt.

interface Object {

…

boolean same\_as (in Object other\_object);

Object copy();

void delete();

};

interface Date : Object {

enum Weekday

{Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday};

enum Month

{January, February, March, April, May, June, July, August, September, October, November, December};

unsigned short year();

unsigned short month();

unsigned short day();

…

boolean is\_equal (in Date other\_Date);

boolean is\_greater (in Date other\_Date);

…

};

Interface Collection : Object {

…

exception ElementNotFound {any element};

unsigned long cardinality();

boolean is\_empty();

…

boolean contains\_element (in any element);

void insert\_element (in any element);

void remove\_element (in any element) raises(ElementNotFound);

Iterator create\_iterator(in boolean stable);

…

};

interface Iterator {

exception NoMoreElements();

…

boolean is\_stable();

boolean at\_end();

void reset();

any get\_element() raises(NoMoreElement);

void next\_position() raises(NoMoreElement);

…

};

interface Set : Collection {

Set creation\_union(in Set other\_set);

…

boolean is\_subset\_of(in Set other\_set);

…

};

interface Bag : Collection {

unsigned long occurrences\_of(in any element);

Bag create\_union(in Bag other\_bag);

…

};

* Các phương thức được sử dụng cho các đối tượng thông qua dấu chấm.

Vd: trong lớp Object để sử dụng phương thức để so sánh với đối tượng p

**o.same\_as(p)**

hay sao chép đối tượng cho đối tượng p

**p = o.copy()**

* Ngoài ra có thể sử dụng dấu [->]: **o->same\_as(p)** hay **o->copy()**
  1. **Xây dựng Interface cho tập các đối tượng**

Một đối tượng bộ sưu tập kế thừa những phương thức và đặc điểm cơ bản từ Collection interface, ví dụ như:

* O.cardinality() trả về số phần tử trong một bộ.
* O.is\_empty() trả về true nếu bộ O là rỗng.
* O.insert\_element(E) thêm một phần tử E vào bộ O.
* O.remover\_element(E) xóa một phần tử ra khỏi bộ O.
* O.contains\_element(E) trả về true nếu bộ O có chứa phần tử E.
* I = O.create\_iterator() tạo một bản sao đối tượng I từ đối tượng O.
  1. **Atomic (User-Defined) Objects**
* Trong phần này, chúng ta sẽ biết cách tạo lớp đối tượng cho các đối tượng nguyên tử (*atomic object*). Trong ODL, chúng ta sẽ sử dụng từ khóa **class.**
* Trong mô hình đối tượng, bất kì đối tượng người dùng tự định nghĩa mà không phải là đối tượng bộ sưu tập được gọi là đối tượng nguyên tử (**atomic object**). Ví dụ, trong CSDL UNIVERSITY, ta có thể định nghĩa một lớp đối tượng cho các đối tượng Student.
* Hầu hết các đối tượng là các đối tượng có cấu trúc (structured object). Ví dụ, một đối tượng Student sẽ có cấu trúc phức tạp với nhiều thuộc tính, mối quan hệt và phương thức, nhưng vẫn được xem là nguyên tử vì nó không phải là một bộ sưu tập.
* Một đối tượng nguyên tử được định nghĩa như một lớp bằng cách xác định các đặc điểm (**properties**) và các phương thức (**operations**).
* VD:

**class** Employee

( **extent** all\_employees **key** ssn)

{

**attribute** **string** name;

**attribute** **string** ssn;

**attribute** **date** birthdate;

**attribute** **enum** Gender{M, F} sex;

**attribute** **short** age;

**relationship** Department works\_for **inverse** Department::has\_emps;

void reassign\_emp(**in** **string** new\_dname) **raises**(dname, valid);

};

**class** Department

( **extent** all\_departments **key** dname, dnumber)

{

**attribute** **string** dname;

**attribute** **short** dnumber;

**attribute** **struct** Dept\_Mgr {Employee manager, **date** startdate}

mgf;

**attribute** **set**<**string**> location;

**attribute** **struct** Projs {**string** projname, **time** weekly\_hours} projs;

**relationship set**<Employee> has\_emps **inverse**

Employee::works\_for;

**void** add\_emp(**in** **string** new\_enamel **raises**(ename\_not\_valid);

**void** change\_manager(**in** **string** new\_mgr\_name; **in** **date** startdate);

}

* + Properties:
    - Thuộc tính (attributes):
      * Một thuộc tính mô tả một số khía cạnh nào đó của đối tượng. Các thuộc tính có giá trị, thường là các literal có giá trị đơn hay cấu trúc phức tạp và được lưu trong đối tượng. Tuy nhiên, các giá trị thuộc tính có thể là Object\_Ids của các đối tượng khác. Giá trị thuộc tính còn được xác định thông qua các phương thức được sử dụng để tính giá trị thuộc tính.
      * VD: trong ví dụ trên, các thuộc tính của lớp Employee là ***name***, ***ssn***, bir***t***hdate, ***sex*** và ***age***. Các thuộc tính của Department là ***dname***, ***dnumber***, ***mgr***, ***locations*** và ***projs***. Các thuộc tính ***mgr*** và ***projs*** của ***Department*** có cấu trúc phức tạp và được định nghĩa thông qua ***struct***. Vì thế giá trị của ***mgr*** trong mỗi đối tượng ***Department*** sẽ có hai phần: ***manager*** – có giá trị là ***Object***\_Id tham chiếu từ đối tượng E***m***ployee, là người quản lý ***Department***, và ***startdate*** có giá trị là ***date***. Thuộc tính vị trí của ***Department*** được định nghĩa thông qua kiểu ***set***, vì mỗi đối tượng Department có thể có nhiều vị trí.
    - Quan hệ (Relationships):
      * Một quan hệ cho biết hai đối tượng trong CSDL có quan hệ với nhau. Trong mô hình ODMG, chỉ có mối quan hệ nhị phân (mối quan hệ đơn) và mỗi mối quan hệ được biểu diễn bằng một cặp tham chiếu (**pair of inverse references**) thông qua từ khóa ***relationship***.
      * VD: trong ví dụ trên, có một một quan hệ liên quan giữa Employee với Department là nơi nhân viên làm việc – mối quan hệ ***works\_for*** của Employee. Ngược lại, mỗi Department có quan hệ với một tập các Employee làm việc trong Department - mối quan hệ ***has\_emps*** của Department.
      * Từ khóa ***inverse*** xác định mối quan hệ đơn tương ứng ngược chiều. Bằng cách xác định mối quan hệ ngược này, hệ CSDL có thể duy trì tính toàn vẹn tham chiếu của mối quan hệ một cách tự động. Ví dụ, nếu giá trị của works\_for cho Employee e tham chiếu Department d, thì giá trị của has\_emps của Department d phải chứa một tham chiếu đến e trong tập tham chiếu đến Employee. Nếu người thiết kế CSDL muốn có mối quan hệ thể hiện trong một hướng – *only one direction*, thì nó phải được tạo như một thuộc tính hay phương thức. Một ví dụ là giá trị *manager component* của thuộc tính ***mgr*** trong Department.
  + Operations:
    - Mỗi đối tượng có thể có một số phương thức, mỗi phương thức có tên, các kiểu đối số, giá trị trả về. Tên của phương thức là duy nhất trong mỗi lớp đối tượng, nhưng vẫn có thể đặt tên giống cho các phương thức trong các lớp đối tượng khác nhau. Phương thức còn có thể xác định tên của trường hợp ngoại lệ (**exception**) có thể xảy ra trong quá trình thực thi phương thức. Ví dụ, trong lớp Employee có phương thức ***reassign\_emp*** và lớp Department có hai phương thức ***add\_emp*** và ***change\_manager***.
  1. **Interfaces, Classes và Inheritance**

Trong mô hình ODMG, có hai khái niệm tồn tại để xác định các kiểu đối tượng: **interface** và **class**. Ngoài ra, còn có hai kiểu của mối quan hệ kế thừa. Trong phần này, chúng ta sẽ thảo luận sự giống và khác nhau giữa những khái niệm này. Dựa vào các thuật ngữ ODMG, ta sử dụng từ **behavior** để thể hiện cho phương thức *operations*, và **state** để thể hiện cho các đặc tính *properties* (thuộc tính và mối quan hệ).

* **Interface** mô tả hành vi trừu tượng (**abstract behavior**)của một loại đối tượng, cho biết phương thức của đối tượng đó. Mặc dù interface có thể có các đặc tính như thuộc tính và phương thức, nhưng các đặc điểm này sẽ không được kế thừa. Interface còn có tính chất **noninstantiable** – có nghĩa là không thể tạo các đối tượng tương ứng từ định nghĩa interface. Được dùng chủ yếu cho việc xác định các phương thức trừu tượng có thể được kế thừa bởi các lớp và các interface khác, được gọi là **behavior inheritance** và sử dụng kí hiệu **“:”**. Vì thế trong mô hình đối tượng ODMG, behavior inheritance yêu cầu **supertype là interface**, **subtype là class hoặc interface**.
* Class mô tả các hành vi và đặc điểm trừu tượng của một kiểu đối tượng, có tính chất **instantianle –** có nghĩa là có thể tạo đối tượng riêng rẽ tương ứng từ định nghĩa class. Một loại quan hệ kế thừa khác gọi là **EXTENDS** và được xác định bằng từ khóa **extends**, được dùng để **kế thừa cả state và behavior** giữa các class. Trong sự kế thừa **EXTENDS**, cả supertype và subtype đều phải là class. Đa kế thừa thông qua EXTENDS không đượng chấp nhận. Tuy nhiên, đa kế thừa được cho phép cho kế thừa hành vi (behavior) thông qua dấu **“:”**. Vì thế, interface có thể kế thừa hành vi từ một số interface khác. Một class cũng có thể kế thừa hành vi từ một số interface khác thông qua dấu “:”, ngoài ra có thể kế thừa behavior và state từ một class khác thông qua EXTENDS.
  1. **Extents, Keys và Factory Objects**
* ***Extent:*** Trong mô hình ODMG, người thiết kế CSDL có thể thêm phần mở rộng **extent** cho bất kỳ loại đối tượng nào được định nghĩa bằng **class**. Phần mở rộng được đặt tên, và sẽ chứa tất cả các thể hiện đối tượng của lớp. Vì thế, phần mở rộng được xem là một tập các đối tượng chứa tất cả các thể hiện của lớp. Trong ví dụ phần 1.3 ta thấy các lớp ***Employee*** và ***Department*** có phần mở rộng gọi là ***all\_employees*** và ***all\_departments***. Phần mở rộng còn được sử dụng để tự động tuân thủ mối quan hệ set/subset giữa phần mở rộng của supertype và subtype của nó. Nếu hai lớp A và B có phần mở rộng là all\_A và all\_B, và lớp B là subtype của lớp A (có nghĩa là lớp B EXTENDS lớp A) thì bộ sưu tập của các đối tượng trong all\_B phải là subset của tất cả các đối tượng trong all\_A tại mọi thời điểm. Ràng buộc này tự động được tuân thủ bởi hệ CSDL.
* ***Key:*** Lớp với phần mở rộng có thể có một hoặc nhiều khóa. Một key chứa một hay nhiều thuộc tính hay mối quan hệ mà giá trị của nó chứa phải là duy nhất với mỗi đối tượng trong phần mở rộng. Trong phần mở rộng mục 1.3, lớp Employee có thuộc tính ***ssn*** là khóa, lớp Department có hai khóa: ***dname*** và ***dnumber***. Với những khóa kết hợp có nhiều thuộc tính hay mối quan hệ, các đặc tính đó được chứa trong dấu **()**.
* ***Factory Object***: một đối tượng có thể được dùng để phát sinh hay tạo các đối tượng riêng lẽ từ các phương thức của nó. Một số interface của factory object là một phần của mô hình đối tượng ODMG. Interface ***ObjectFactory*** có một phương thức đơn, ***new()***, sẽ trả về một đối tượng mới với một **Object\_Id*.*** Bằng cách kế thừa interface này, người dùng có thể tạo các factory interface cho mỗi kiểu đối tượng nguyên tử, và người lập trình có thể thực thi phương thức ***new*** một cách khác nhau cho mỗi kiểu đối tượng. Ví dụ bên dưới cho thấy một interface DataFactory, có phương thức tạo một **calendar\_date** mới, và một phương thức tạo một đối tượng mà giá trị của nó là **current\_date**… Ta có thể thấy, factory object cơ bản cung cấp phương thức dựng (**constructor operations**) cho các đối tượng mới.
* Ví dụ:

interface ObjectFactory{

Object new();

};

interface DateFactory : ObjectFactory{

exception InvalidDate{};

…

Date calendar\_date ( in unsigned short year,

in unsigned short month,

in unsigned short day)

raises(InvalidDate);

…

Date current();

interface DatabaseFactory{

Database new();

};

interface Database{

void open(in string database\_name);

void close()l

void bind(in any some\_object, in string object\_name);

Object unbind(in string name);

Object lookup(in string object\_name)

raises(ElementNotFound);

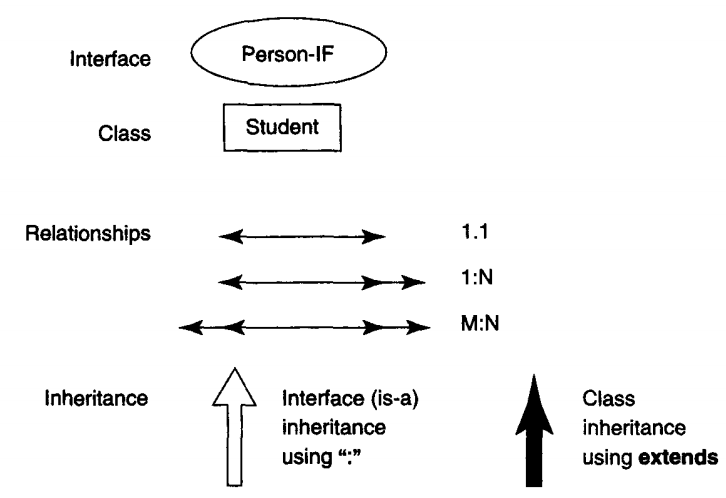
…

};

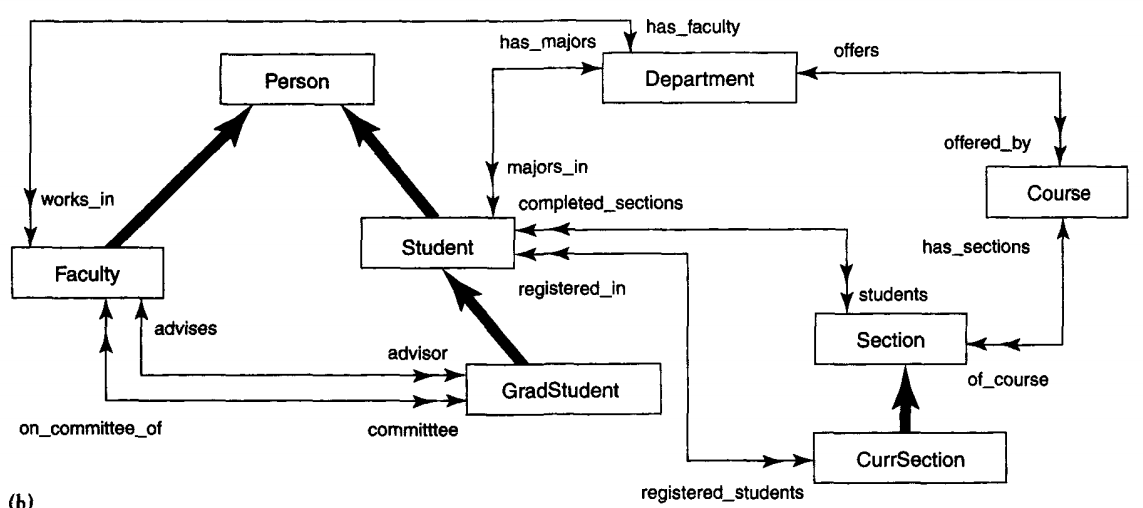
* ***Database***: bởi vì ODBMS có thể tạo nhiều CSDL khác nhau, với những lược đồ khác nhau, nên mô hình dữ liệu ODMG có các interface cho các đối tượng **DatabaseFactory** và **Database**. Mỗi CSDL có một tên CSDL riêng, phương thức **bind** có thể được dùng để gán các tên duy nhất riêng biệt vào các thể hiện đối tượng trong một CSDL cụ thể. Phương thức **lookup** trả về một đối tượng từ CSDL có tên **object\_name**, và phương thức unbind lấy tên ra khỏi các thể hiện đối tượng đã được đặt tên từ CSDL.

1. OBJECT DEFINITION LANGUAGE ODL

* ODL được thiết kế để hỗ trợ các khái niệm ngữ nghĩa của mô hình ODMG và không phụ thuộc vào bất kì ngôn ngữ lập trình nào. Mục đích chính của nó là dùng để tạo các đối tượng – class và interface. Vì thế ODL không phải là một ngôn ngữ lập trình. Người dùng có thể xác định lược đồ trong ODL mà không cần phụ thuộc vào ngôn ngữ lập trình, và sử dụng ngôn ngữ liên kết để xác định các khái niệm ODL có thể **map** với các khái niệm trong ngôn ngữ lập trình như C++, SMALLTALK và JAVA.
* Các kí hiệu trong ODL:



* Ví dụ mô hình CSDL ***UNIVERSITY***:



* Ví dụ tạo một số đối tượng bằng class trong CSDL UNIVERSITY bằng ODL:

**class** Person

(**extent** persons **key** ssn)

{

**attribute** **struct** Pname {**struct** fname,

**string** mname, **string** Inme} name;

**attribute** **string** ssn;

**attribute** **date** birthdate;

**attribute** **enum** Gender{M, F} sex;

…

**Short** age();

};

**class** Faculty **extends** Person

(**extent** faculty)

{

**attribute** **string** rank;

**attribute** **float** salary;

**attribute** **string** phone;

…

**relationship** Department works\_in

**inverse** Department::has\_faculty;

**relationship** **set**<GradStudent> advises

**inverse** GradStudent::advisor;

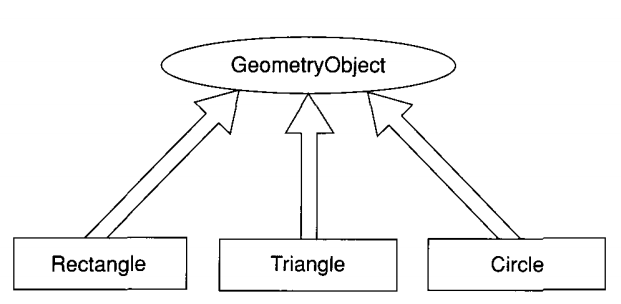
**void** give\_raise (**in** **float** raise);

**void** promote (**in** **string** new\_rank);

};

…

* Ví dụ thể hiện interface và kế thừa:



* Ví dụ tạo các đối tượng trong ví dụ interface và kế thừa:

**interface** GeometryObject {

**attribute enum** Shape {Rectangle, Triangle, Circle, …} shape;

**attribute** **struct** Point{**short** x, **short** y} reference\_point;

**float** perimeter ();

**float** area();

**void** translate (**in short** x\_translation; **in short** y\_translation);

**void** rotate (**in float** angle\_of\_rotation)l

};

**class** Triangle : GeometryObject

(**extent** triangles) {

**attribute** **struct** Point{**short** x, **short** y} reference\_point;

**attribute** **short** length;

**attribute** **short** height;

attribute float orientation\_angle;

};

* Đa kế thừa (multiple inheritance) của interface bởi class được cho phép, giống như việc đa kế thừa của interface từ một interface. Tuy nhiên, với kế thừa EXTENDS (class), đa kế thừa không được sử dụng. Vì thế, một class có thể kế thừa bằng EXTNEDS từ một class.

1. OBJECT QUERY LANGUAGE OQL

OQL là ngôn ngữ truy vấn được đưa ra cho mô hình đối tượng ODMG. Nó được thiết kế để làm việc gần gũi hơn với ngôn ngữ lập trình mà liên kết ODMG đã được định nghĩa, như C++, SMALLTALK và JAVA. Vì thế, truy vấn OQL được thêm vào ngôn ngữ lập trình có thể trả về các đối tượng có kiểu thích hợp với kiểu của ngôn ngữ lập trình. Các phương thức thực thi của class có thể được viết bằng mã trong ngôn ngữ lập trình. Cú phá OQL tương tự như cú pháp của chuẩn SQL, thêm vào các khái niệm ODMG như object identity, complex object, operation, inheritance, polymorphism và relationship.

**3.1. Simple OQL Queries, database entry points, iterator variables**

* Cú pháp cơ bản của OQL là **select… from… where…**, giống như của SQL.

VD: Q0:

**SELECT** d.dname

**FROM** d **in** department

**WHERE** d.college = ‘Engineering’;

* Thông thường, entry point đến CSDL là cần thiết cho mỗi câu truy vấn, là tên đại diện cho đối tượng - ***named persistent object***. Với nhiều câu truy vấn, entry point là tên của phần **extent** của class. Tên phần mở rộng được xem là tên của một thể hiện đối tượng mà kiểu của nó là một bộ sưu tập của các đối tượng trong class. Ví dụ, tên của đối tượng **departments** là kiểu **set**<**Department**>; **persons** là kiểu **set<Person>…**
* Việc sử dụng tên mở rộng – **departments** trong Q0 – như một entry point để thể hiện một tập thể hiện của các đối tượng. Bất cứ khi nào một bộ sưu tập được tham chiếu đến trong OQL, chúng ta nên định nghĩa một biến **iterator variable – d**  trong Q0 – duyệt qua mỗi đối tượng trong bộ sưu tập. Trong nhiều trường hợp, câu truy vấn sẽ chọn các đối tượng từ bộ sưu tập, dựa trên các điều kiện đã được chỉ định trong WHERE. Trong Q0, chỉ các đối tượng d trong bộ sưu tập của **department** thỏa điều kiện **d.college = ‘Engineering’** được chọn cho kết quả. Với mỗi đối tượng được chọn d, giá trị **d.name** được gọi trong kết quả. Vì thế, kiểu của kết quả cho Q0 là **bag<string>** bởi vì kiểu của mỗi giá trị **dname** là string. Trong thực hết, kết quả của truy vấn sẽ là kiểu **bag** cho **select… from…** và kiểu **set** cho **select distinct… from…**.
* Sử dụng ví dụ trong Q0, có 3 cú pháp để định nghĩa iterator variables:
  + d **in** department
  + department d
  + department as d

**3.2. Query Results and Path Expressions**

**3.3. Other Features of OQL**

1. TỔNG QUAN VỀ C++ LANGUAGE BINDING
2. OBJECT DATABASE CONCEPTUAL DESIGN