TRƯỜNG CAO ĐẲNG NGHỀ CÔNG NGHIỆP HÀ NỘI

Tác giả: Vũ Thị Kim Phượng

Bùi Quang Ngọc

GIÁO TRÌNH

Cơ sở dữ liệu hướng đối tượng

(Lưu hành nội bộ)

Hà Nội năm 2012

Mục lục

BÀI 1: GIỚI THIỆU LỊCH SỬ PHÁT TRIỂN

1. Lịch sử phát triển của Cơ sở dữ liệu hướng đối tượng

Các loại cấu trúc cơ sở dữ liệu và mối liên hệ giữa chúng đóng vai trò rất lớn trong việc xác định tính hiệu quả của hệ quản trị cơ sở dữ liệu. Vì vậy, thiết kế cơ sở dữ liệu trở thành hoạt động chính trong môi trường cơ sở dữ liệu.

Việc thiết kế cơ sở dữ liệu được thực hiện đơn giản hơn nhiều khi ta sử dụng các mô hình. Các mô hình là sự trừu tượng đơn giản của các sự kiện trong thế giới thực. Các trừu tượng như vậy cho phép ta khảo sát các đặc điểm của các thực thể và các mối liên hệ được tạo ra giữa các thực thể đó. Việc thiết kế các mô hình tốt sẽ đưa ra các cơ sở dữ liệu tốt và trên cơ sở đó sẽ có các ứng dụng tốt. Ngược lại, mô hình không tốt sẽ đưa đến thiết kế cơ sở dữ liệu tồi và dẫn đến các ứng dụng không đúng.

Một mô hình cơ sở dữ liệu là một tập hợp các khái niệm dùng để biểu diễn các cấu trúc của cơ sở dữ liệu. Cấu trúc của một cơ sở dữ liệu là các kiểu dữ liệu, các mối liên kết và các ràng buộc phải tuân theo trên các dữ liệu. Nhiều mô hình còn có thêm một tập hợp các phép toán cơ bản để đặc tả các thao tác trên cơ sở dữ liệu.

Các loại mô hình cơ sở dữ liệu:

- Các mô hình dữ liệu bậc cao hoặc mô hình dữ liệu mức quan niệm cung cấp các khái niệm gắn liền với cách cảm nhận dữ liệu của nhiều người sử dụng.
- Các mô hình dữ liệu bậc thấp hoặc các mô hình dữ liệu vật lý cung cấp các khái niệm mô tả chi tiết về việc dữ liệu được lưu trữ trong máy tính như thế nào.
- Các mô hình dữ liệu thể hiện (mô hình dữ liệu mức logic), chúng cung cấp những khái niệm mà người sử dụng có thể hiểu được và không xa với cách tổ chức dữ liệu bên trong máy tính.

Trong một mô hình dữ liệu cần phải phân biệt rõ giữa mô tả của cơ sở dữ liệu và bản thân cơ sở dữ liệu.

Sau đây, chúng ta sẽ điểm qua lịch sử phát triển của các mô hình cơ sở dữ liệu:

- Vào những năm sáu mươi, thế hệ đầu tiên của cơ sở dữ liệu ra đời dưới dạng mô hình thực thể kết hợp *(Entity Relationship Model)*, mô hình mạng *(Network Model)* và mô hình phân cấp *(Hierachical Model)*.
- Vào những năm bảy mươi, thế hệ thứ hai của cơ sở dữ liệu ra đời. Đó là mô hình dữ liệu quan hệ *(Relational Data Model)* do EF. Codd phát minh. Mô hình này có cấu trúc logic chặt chẽ. Đây là mô hình đã và đang được sử dụng rộng khắp trong công tác quản lý trên phạm vi toàn cầu. Việc nghiên cứu mô hình dữ liệu quan hệ nhằm vào lý thuyết chuẩn hoá các quan hệ và là một công cụ quan trọng trong việc phân tích thiết kế các hệ cơ sở dữ liệu hiện nay. Mục đích của nghiên cứu này nhằm bỏ đi các phần tử không bình thường của quan hệ khi thực hiện các phép cập nhật, loại bỏ các phần tử dư thừa.
- Sang thập kỷ tám mươi, mô hình cơ sở dữ liệu thứ ba ra đời, đó là mô hình cơ sở dữ liệu hướng đối tượng *(Object Oriented Data Model)*, mô hình cơ sở dữ liệu phân tán, mô hình cơ sở dữ liệu suy diễn,…

Thực tế chưa có mô hình dữ liệu nào là tốt nhất. Tốt nhất phụ thuộc vào yêu cầu truy xuất và khai thác thông tin của đơn vị quản lý nó. Nó được sử dụng ở đâu và vào lúc nào là

tốt nhất. Tuy nhiên, người ta thường dựa vào các tiêu chí sau để nói rằng mô hình dữ liệu tốt nhất:

- Mục đích: Phần lớn các mô hình dữ liệu sử dụng hệ thống ký hiệu để biểu diễn dữ liệu và làm nền tảng cho các hệ ứng dụng và ngôn ngữ thao tác dữ liệu. Các mô hình thực thể quan hệ không có hệ thống ký hiệu để xây dựng các phép toán thao tác dữ liệu, mà sử dụng để thiết kế lược đồ khái niệm, cài đặt trong một mô hình dữ liệu với một hệ quản trị cơ sở dữ liệu nào đó.
- Hướng giá trị hay hướng đối tượng: Các mô hình dữ liệu quan hệ và mô hình logic là các mô hình dữ liệu hướng giá trị. Trong các mô hình dữ liệu hướng giá trị có tính khai báo (declarativeness) và có tác động đến các ngôn ngữ được nó hỗ trợ. Các mô hình mạng, phân cấp, mô hình dữ liệu hướng đối tượng cung cấp đặc tính nhận dạng đối tượng, nên có thể xem chúng là các mô hình hướng đối tượng. Mô hình thực thể quan hệ cũng được có đặc tính nhận dạng hướng đối tượng.
- Tính dư thừa: Tất cả các mô hình dữ liệu đều có khả năng hỗ trợ lưu trữ dữ liệu vật lý và hạn chế sự dư thừa dữ liệu. Tuy nhiên các mô hình dữ liệu hướng đối tượng giải quyết sự dư thừa tốt hơn, bằng cách tạo ra sử dụng con trỏ trỏ đến nhiều vị trí khác nhau.
- Giải quyết mối quan hệ nhiều – nhiều: Phần lớn trong các mô hình cơ sở dữ liệu có chứa các mối quan hệ nhiều – nhiều, một – nhiều hay quan hệ môt – một. Một quan hệ có nhiều phần tử của các quan hệ khác và ngược lại. Tuy nhiên trong mô hình dữ liệu mạng không chấp nhận mối quan hệ nhiều – nhiều.

2. Nền tảng của dữ liệu hướng đối tượng.

Cơ sở dữ liệu hướng đối tượng và hệ quản trị hướng đối tượng (Object Oriented DataBase Management Systems – OO DBMS) mô tả các kiểu dữ liệu được xây dựng bằng phương pháp tạo bản ghi và tạo tập hợp. Các quan hệ được xây dựng từ các bộ bằng thao tác tạo một tập hợp các bản ghi có khuôn dạng thống nhất.

Che dấu dữ liệu (Encapsulation): Nghĩa là khi có yêu cầu truy xuất đến các đối tượng thuộc kiểu đặc biệt, phải qua các thủ tục đã được định nghĩa cho các đối tượng đó. Chẳng hạn định nghĩa stack như là một kiểu và định nghĩa các thao tác PUSH, POP áp dụng cho stack.

Đặc tính nhận dạng đối tượng (Object Indentity) là khả năng phân biệt các đối tượng. Nghĩa là cấu trúc các kiểu cơ bản như nhau. Các kiểu cơ bản là chuỗi ký tự, số.

Thực tế cho thấy cơ sở dữ liệu hướng đối tượng có các ưu điểm:

- Cho phép xét các liên kết đối tượng dưới dạng các phép lưu trữ với các đối tượng.
- Các đối tượng dùng chung giữa nhiều người sử dụng.
- Khả năng phát triển kho tri thức bằng cách thêm các đối tượng mới và các phép xử lý kèm theo.
- Phát triển hệ quản trị cơ sở dữ liệu dựa trên việc xử lý các đối tượng phức tạp, giao diện chương trình, đối tượng động và trừu tượng.

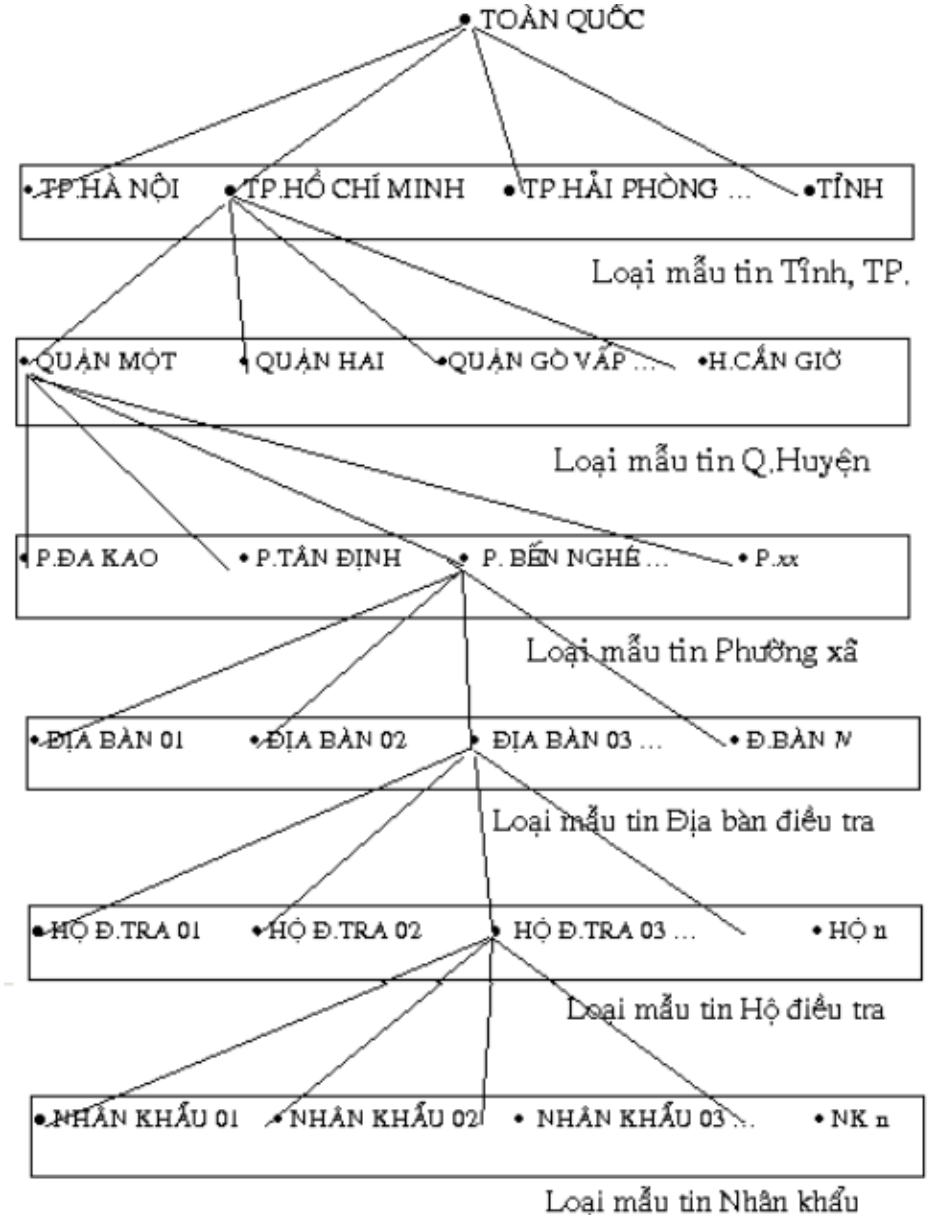
BÀI 2: MÔ HÌNH DỮ LIỆU HƯỚNG ĐỐI TƯỢNG ODMG – CÁC THÀNH PHẦN ĐẶC TRƯNG

1. Khái niệm về mô hình dữ liệu

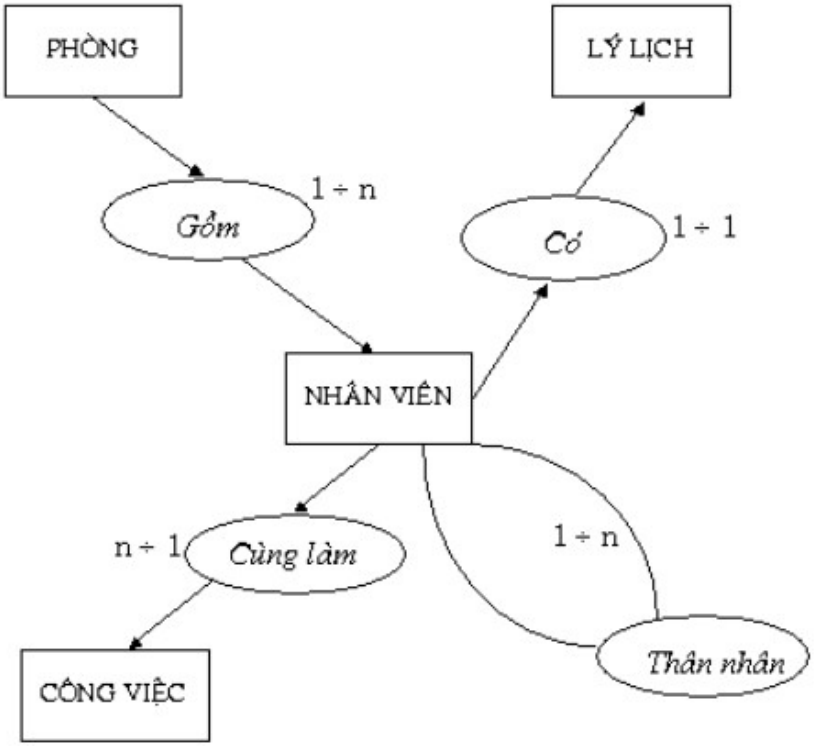
Mô hình dữ liệu là một hệ thống hình thức toán học, bao gồm:

- Hệ thống các ký hiệu biểu diễn dữ liệu.
- Tập hợp các phép toán thao tác trên cơ sở dữ liệu.

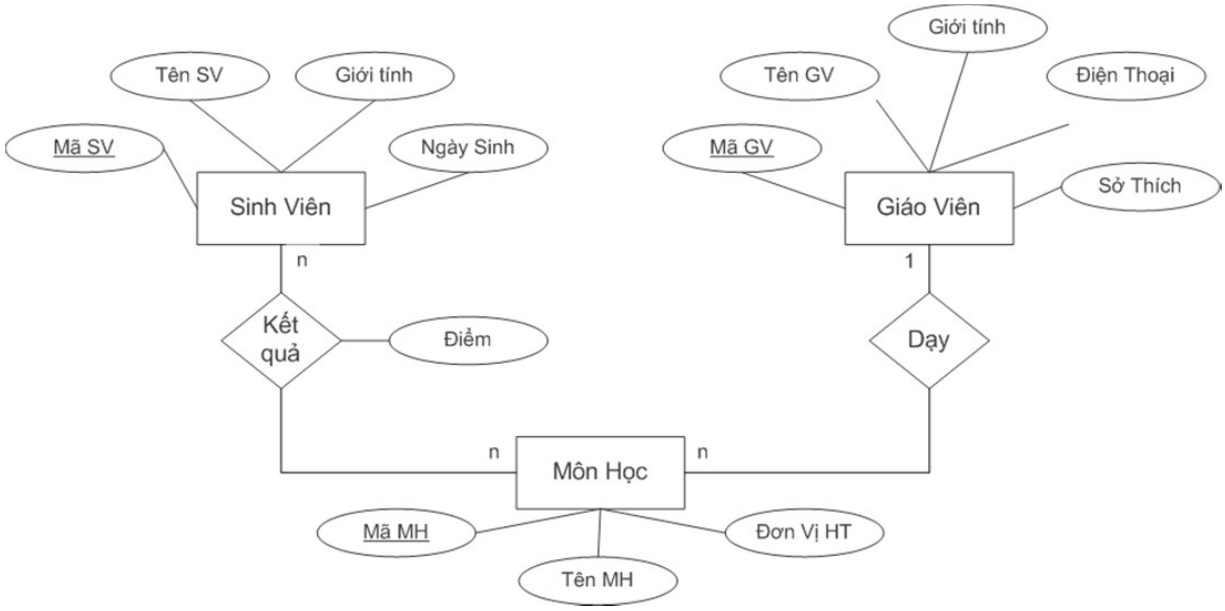
Ví dụ: Các mô hình dữ liệu:



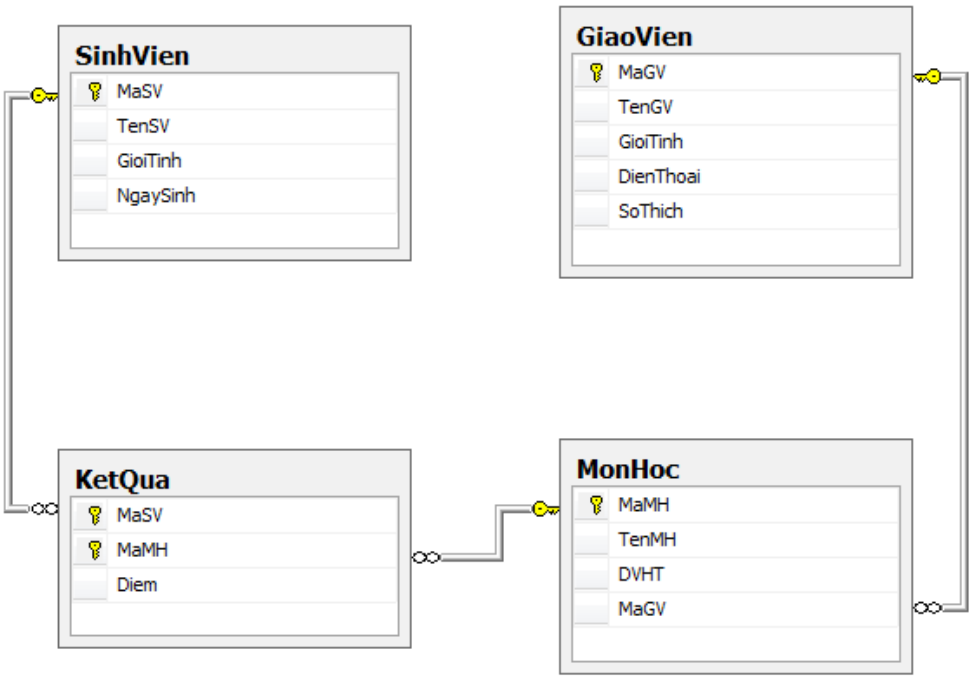
Hình 2.1: Mô hình phân cấp (Hierachical Model)



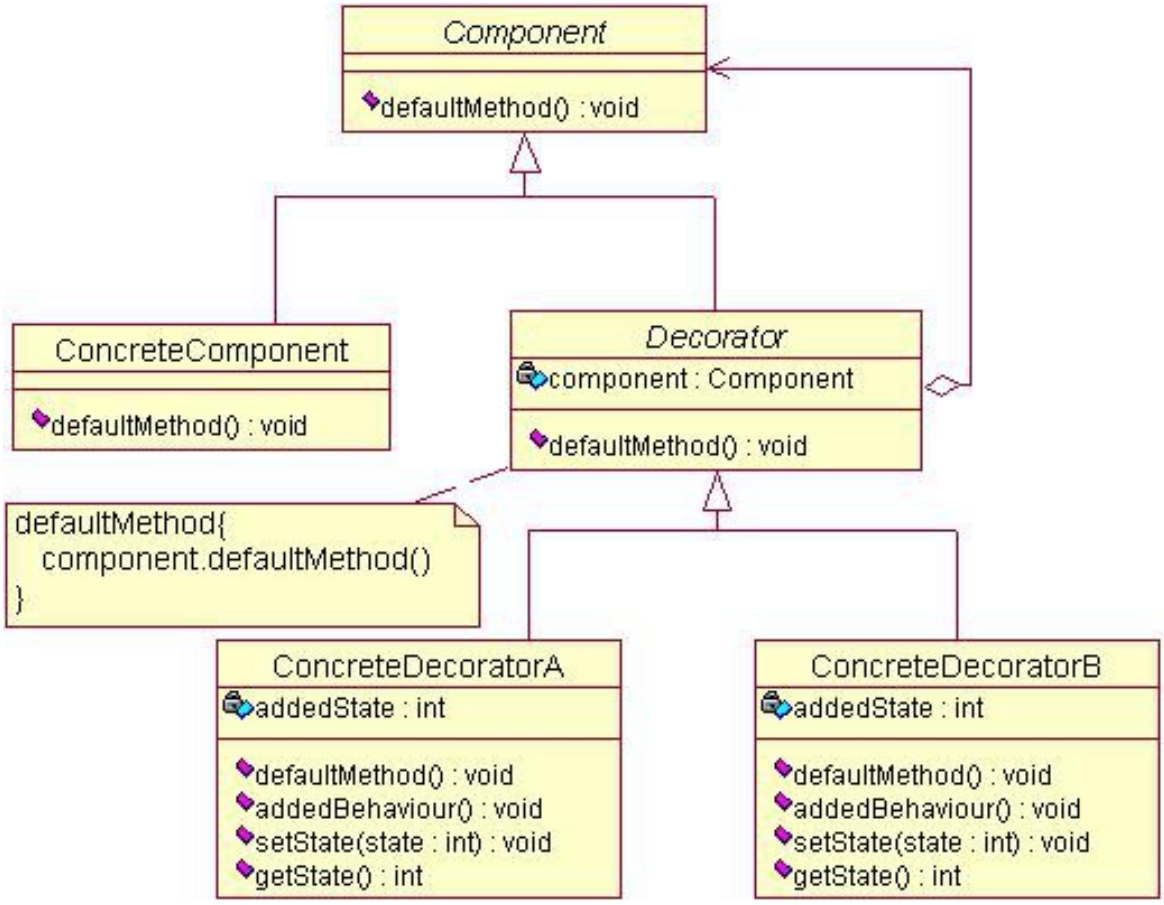
Hình 2.2: Mô hình mạng (Network Model)



Hình 2.3: Mô hình thực thể kết hợp (Entity Relationship Model)



Hình 2.4: Mô hình dữ liệu quan hệ (Relational Data Model)



Hình 2.5: Mô hình dữ liệu hướng đối tượng (Object Oriented Data Model)

3. Mô hình dữ liệu hướng đối tượng – ODMG

Mô hình dữ liệu hướng đối tượng - OODM (*Object Oriented Data Model*) ra đời từ cuối những năm 80 và đầu những năm 90.

Đây là loại mô hình tiên tiến nhất hiện nay dựa trên cách tiếp cận hướng đối tượng đã quen thuộc trong các phương pháp lập trình hướng đối tượng, nó sử dụng các khái niệm như lớp (*class*), sự kế thừa (*inheritance*), kế thừa bội(multi-inheritance).

Đặc trưng cơ bản của cách tiếp cận này là tính đóng gói (*encapsulation*), tính đa hình (*polymorphism*) và tính tái sử dụng (*Reusability*).

Lược đồ ODMG *(Object Database Management Group)*: ODMG đề xuất một cơ sở dữ liệu tiêu chuẩn với mục tiêu thống nhất mô hình đối tượng hạt nhân của nhiều hệ quản trị cơ sở dữ liệu đối tượng khác nhau.

ODMG đưa ra một chuẩn mới cho OODM:

- Một mô hình đối tượng (OM)
- Một ngôn ngữ định nghĩa đối tượng (ODL)
- Một ngôn ngữ hỏi đối tượng với cú pháp tựa SQL
- Ràng buộc ngôn ngữ C++(Java/Smalltalk)

Những kết cấu chính được đặc tả bởi mô hình dữ liệu của ODMG:

- Đối tượng và literal
- Kiểu
- Các kiểu con và tính kế thừa
- Ngoại diên
- Khoá
- Kiểu sưu tập và kiểu có cấu trúc

Đối tượng bền vững (đối tượng cơ sở dữ liệu): là các đối tượng tiếp tục tồn tại khi thủ tục hay quá trình tạo ra chúng đã kết thúc. Chúng được cấp phát bộ nhớ và được lưu trữ bởi hệ quản trị cơ sở dữ liệu hướng đối tượng.

Đối tượng không bền (chuyển tiếp): chỉ tồn tại bên trong thủ tục hay quá trình tạo ra chúng. Chúng được cấp phát bộ nhớ bởi hệ thống thời gian chạy của ngôn ngữ lập trình.

3.1. Mô hình hóa các đối tượng

- Đối tượng (Object): Bộ sưu tập các yếu tố DL có cấu trúc, được đồng nhất bởi một dẫn trỏ (tham chiếu) duy nhất.
  - Mọi đối tượng đều được đặc trưng bằng một tên duy nhất, gọi là OID (Object Indentifier)
  - Hai đối tượng là đồng nhất (O_1==O_2) nếu chúng có cùng OID
  - Hai đối tượng là bằng nhau (O1=O2) nếu chúng có cùng giá trị
  - Các đối tượng đặc trưng bởi các tính chất
- Tính chất (Property): đặc trưng của một đối tượng được chỉ định bằng một tên có thể ứng với một thuộc tính, một hàm hay một đối tượng con thành phần

Ví dụ:

- Thuộc tính đơn: tên của một người,...
- Hàm: Hàm tuổi (của một người),...
- Thuộc tính kép: các con của một người,...

- Lớp: nhóm các đối tượng có cùng tính chất, được đặc trưng bởi một cấu trúc và tập các phép toán tác dụng lên các đối tượng của lớp bằng cách che dấu cấu trúc
  - Việc đặc tả tiến triển của các lớp đối tượng làm thành một CSDL hướng đối tượng, cho phép mô hình hoá hành vi chung của các đối tượng một cách đơn thể và mở rộng được.

Ví dụ: các con người, các hình tròn,...

3.2. Mô hình hóa tính động

- Phương pháp: thao tác liên kết với một lớp, xử lý hay đưa trả lại trạng thái của một đối tượng hay một phần của đối tượng thuộc lớp
  - Một đối tượng được thao tác bởi phương pháp của lớp và được thấy qua các phương pháp: nguyên lý bọc kín
  - Phương pháp có thể áp dụng được cho nhiều đối tượng thuộc các lớp khác nhau: đa lớp → dùng để mô hình hoá các mối liên kết giữa các lớp

Ví dụ:

class cửa

public: {các thuộc tính thấy được từ bên ngoài lớp}

trạng thái: mở, đóng

chiều cao: real {kiểu thực}

chiều rộng: real

chiều dầy: real

private:

trục: vectơ

góc: real

public operation: {các phương pháp}

mở(lực: real) ... end;

đóng ... end;

- Thông báo: các đối tượng trao đổi (giao lưu thông tin) với nhau bằng thông báo.
  - Thông báo gồm tên của một phương pháp và các tham số của nó
  - Khối tham số cho phép bằng việc gửi đi dẫn gọi một phương pháp công cộng của một đối tượng

- Đối tượng phản ứng lại một thông báo bằng cách thực hiện phương pháp liên kết và đưa trả về các tham số kết quả của phương pháp

Ví dụ: Có thể gửi thông báo tới một đối tượng p của lớp cửa:

p: mở(30)

p: đóng

p: chiều rộng.read

p: chiều rộng.write

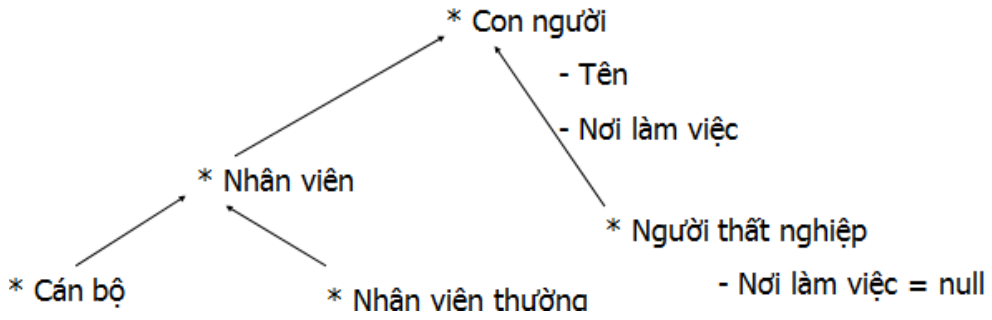
- Demon: Là thao tác trên các đối tượng được khởi phát bởi hệ thống khi có xuất hiện một điều kiện đặc biệt

Ví dụ: Demon có thể được thêm vào lớp cửa nhằm duy trì tự động trạng thái của nó: **if** góc > 10^o **then** trạng thái = mở

3.3. Các liên kết ngữ nghĩa giữa các lớp

- Sự tổng quát hoá: liên kết phân cấp giữa hai lớp xác định rằng các đối tượng của lớp trên tổng quát hơn các đối tượng của lớp dưới, các đối tượng của lớp dưới có các tính chất đầy đủ và tinh tế hơn

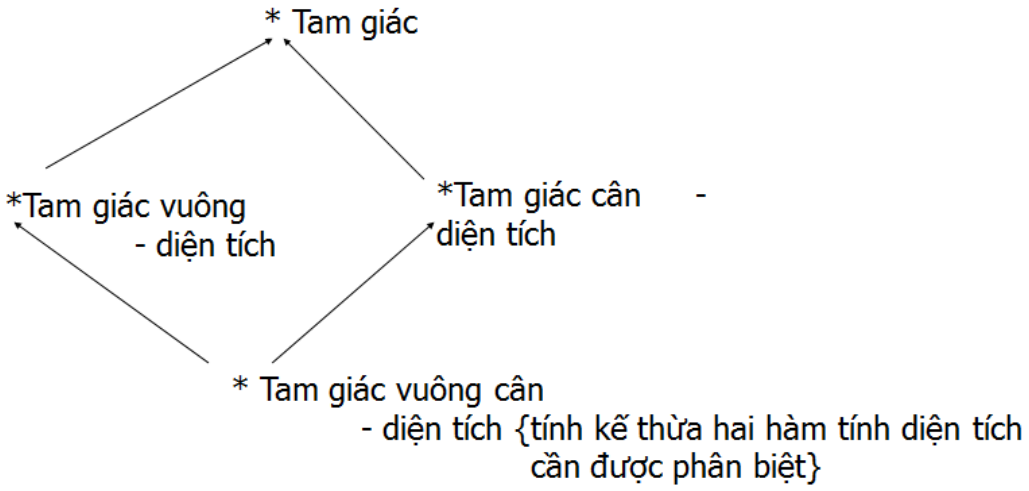
Ví dụ:



- Tính kế thừa: sự truyền tính chất của một lớp tới lớp con của nó
  - Mọi phần tử của lớp con kế thừa các tính chất của lớp trên
  - Một số tính chất của lớp con có thể được làm tinh tế hơn → định nghĩa lại

Ví dụ: thuộc tính "Nơi làm việc" của lớp "Con người" có thể được định nghĩa lại với giá trị null ở mức của lớp "Người thất nghiệp"

- Tính kế thừa bội: cho phép một lớp có nhiều lớp trên trực tiếp
  - Lớp con kế thừa các tính chất và phương pháp của các lớp trên
  - Có thể xảy ra và cần được giải quyết những xung đột về tên các tính chất hay phương pháp



Các mô hình đối tượng thường phân biệt các tính chất được phân chia bởi nhiều lớp và nhóm hợp chúng trong những lớp đặc biệt gọi là các mối liên kết

Mối liên kết là liên hệ cấu trúc cho phép liên kết các lớp đối tượng với nhau bằng các tính chất phân chia

Ví dụ: "Người" và "Sách" là hai lớp gộp một số tính chất (Tên, ..., Tên sách,...), thì có thể định nghĩa mối liên kết Tác giả của như sau:

Người Tác giả của Sách

3.4. Tổ chức các nhóm đối tượng

- Tác tử xây (constructor): lớp cấu trúc, cho phép áp đặt một cấu trúc lên một tập đối tượng và định nghĩa các tính chất cấu trúc đa trị

Các tác tử xây:

- bộ (tuple): cho phép nhóm gộp các thuộc tính (tích Đề các)
- tập (set): cho phép định nghĩa các nhóm không sắp thứ tự, không chứa các phần tử giống nhau
- túi (bag): các tập không sắp thứ tự, có các phần tủ giống nhau
- danh sách (list): cho phép định nghĩa các nhóm có thứ tự, được phép có các phần tử giống nhau
- bảng (table): các nhóm có thứ tự và có chỉ số

Một nhóm đối tượng kế thừa các tính chất của tác tử xây nếu có tác tử xây đứng trước.

Ví dụ: có thể quản lý dễ dàng các danh sách các bảng sau:

class câu

nội dung: list array char;

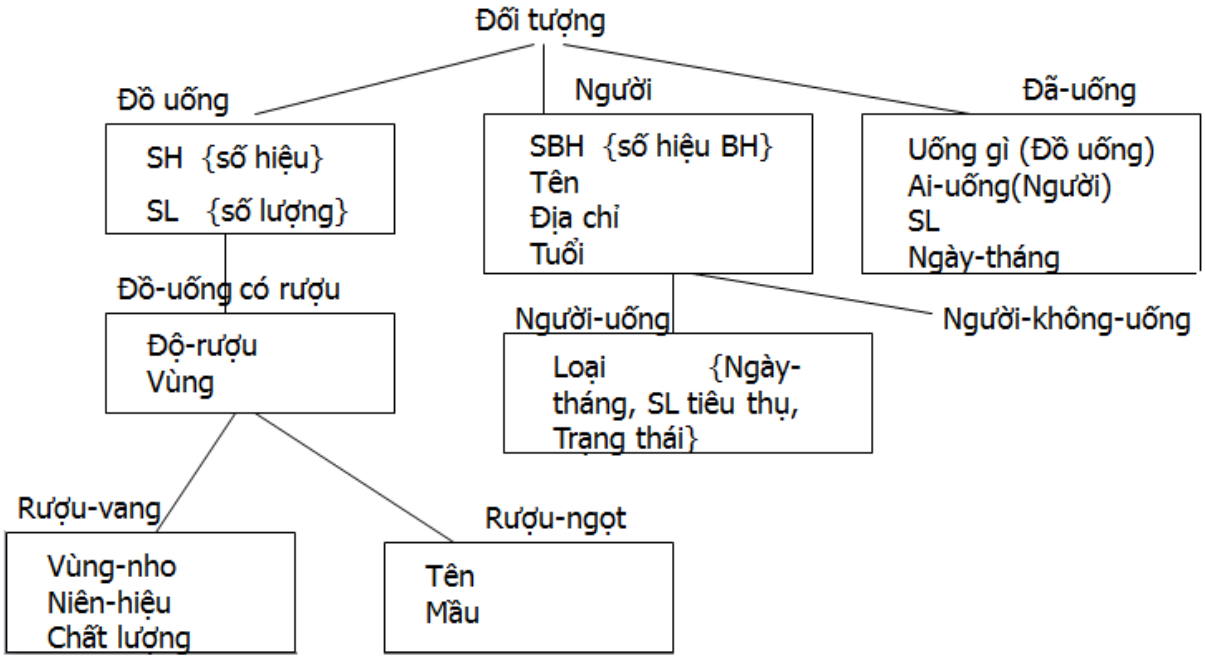
class văn-bản

đoạn : list câu

3.5. Lược đồ

- Lược đồ cơ sở dữ liệu hướng đối tượng mô tả các thành phần sau:
  - Mô tả các lớp. Mỗi lớp bao gồm các tính chất (tuỳ theo tình hình được tổ chức thành các nhóm bởi các toán tử xây) và các phương pháp.

- Mô tả các mối liên kết giữa các lớp.

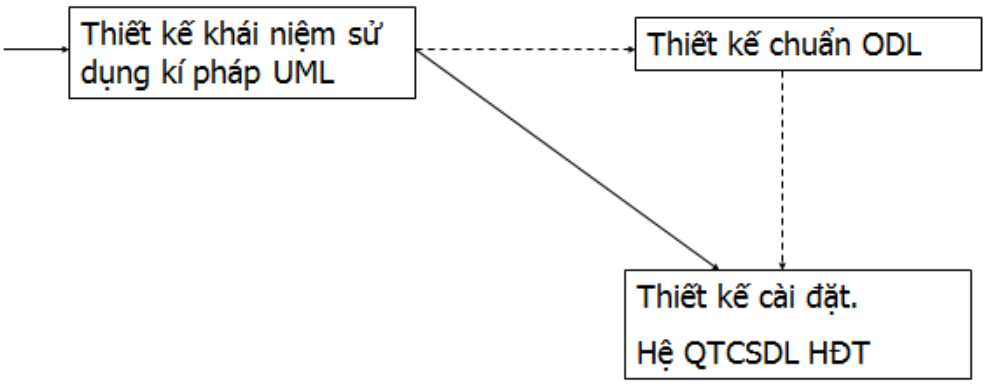


Hình 2.6: Lược đồ đối tượng của cơ sở dữ liệu rượu vang

4. Phương pháp xây dựng mô hình dữ liệu hướng đối tượng

4.1. Phương pháp chuyển đổi

- **Định nghĩa *lược đồ khái niệm*** trong một ngôn ngữ (mô hình) sao cho gần với người dùng và độc lập với cài đặt cuối cùng. Mô hình được dùng trong bước này phải có khả năng biểu diễn mọi yêu cầu của người dùng (UML – Unified Modeling Language).
- **Dịch chuyển trực tiếp sang cài đặt** cuối cùng trong một hệ QTCSDL hướng đối tượng xác định.
- **Có thể qua một bước trung gian** để có một lược đồ được mô tả trong ODL (Object Definition Language), biểu diễn các chi tiết thiết kế độc lập với sản phẩm cuối cùng

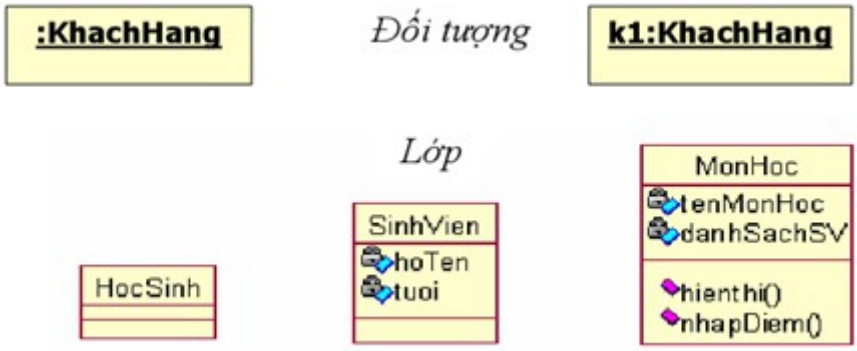


Hình 2.7: Quá trình thiết kế một lược đồ cơ sở dữ liệu hướng đối tượng

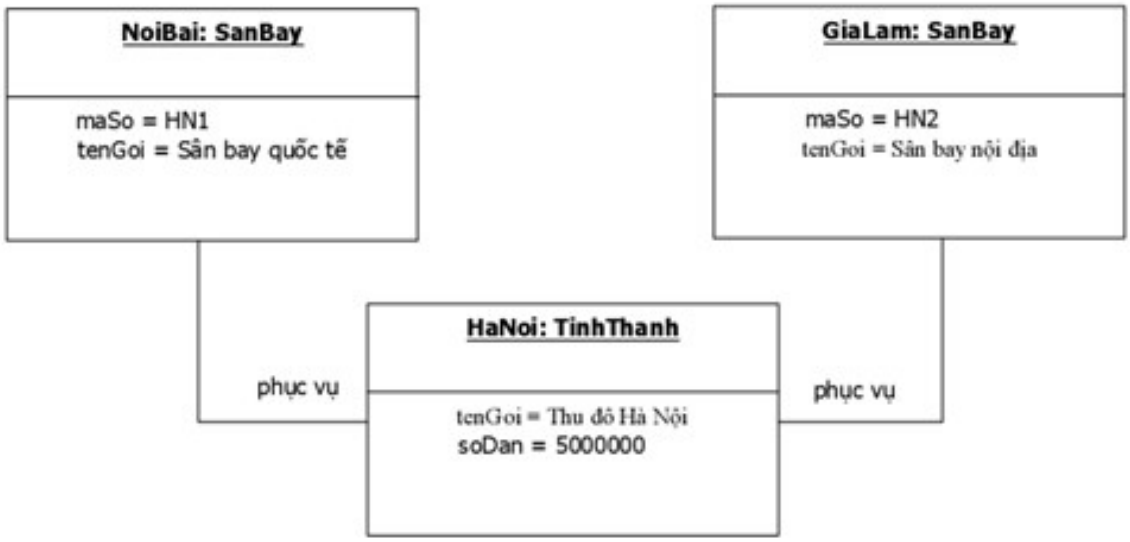
4.2. Phương pháp phân tích và xây dựng trực tiếp

a. **Thiết kế khái niệm (UML)**

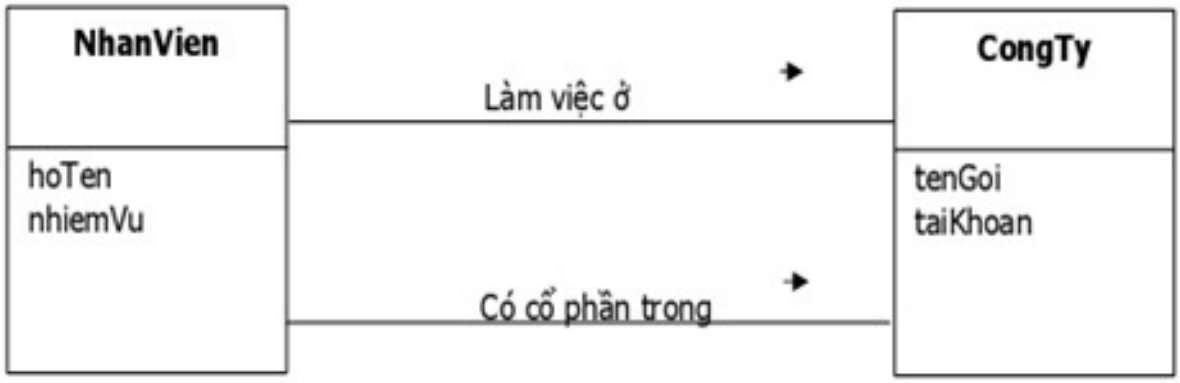
- Ký pháp UML:



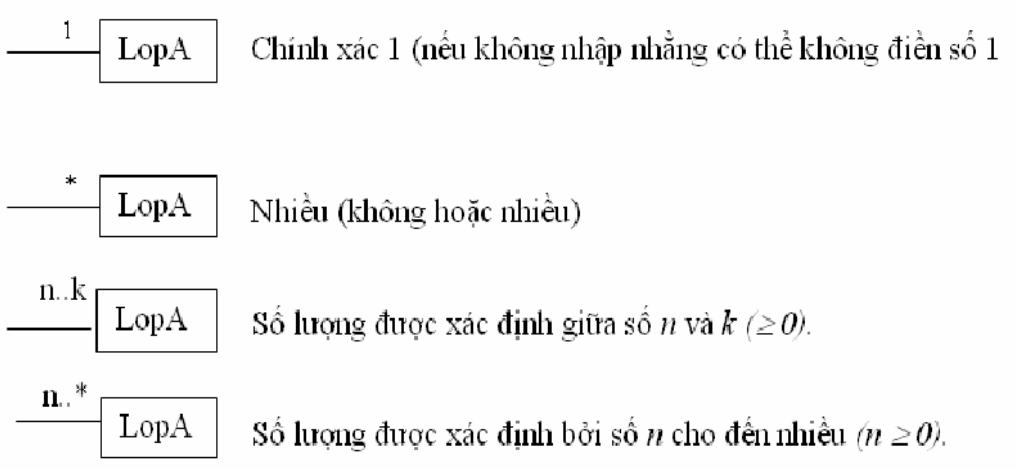
- ‘+’ đứng trước tên thuộc tính, hàm xác định tính công khai (public). Trong Rose kí hiệu là ổ khoá không bị khoá.
- ‘#’ đứng trước tên thuộc tính, hàm xác định tính được bảo vệ (protected). Trong Rose kí hiệu là ổ khoá bị khoá nhưng có chìa để bên cạnh.
- ‘-’ đứng trước tên thuộc tính, hàm xác định tính sở hữu riêng (private). Trong Rose kí hiệu là ổ khoá bị khoá nhưng không có chìa để bên cạnh.



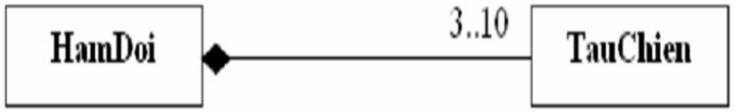
Hình 2.8: Liên kết giữa các đối tượng



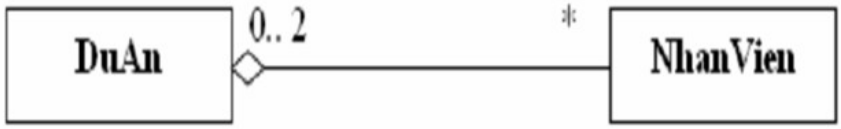
Hình 2.9: Quan hệ kết hợp giữa các lớp



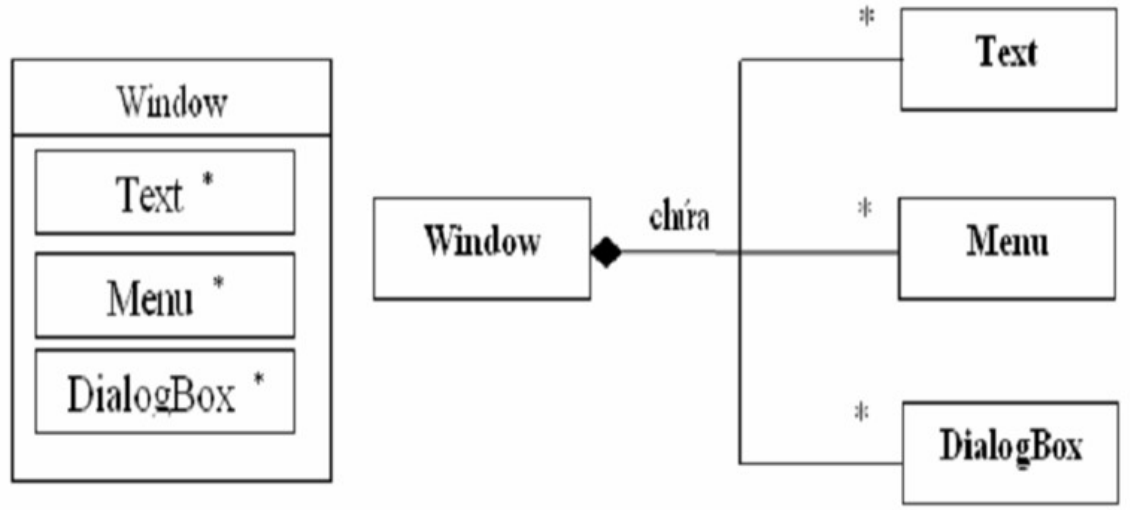
Hình 2.10: Biểu diễn các bội số



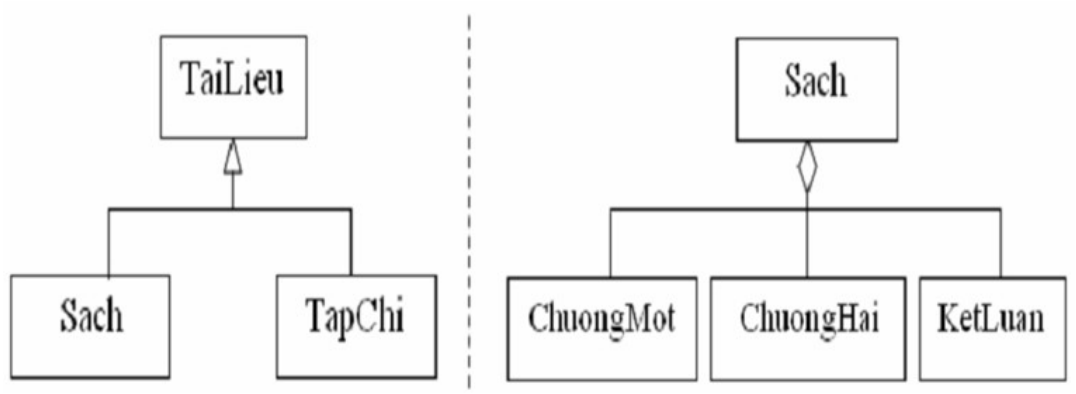
Hình 2.11: Quan hệ kết tập thông thường



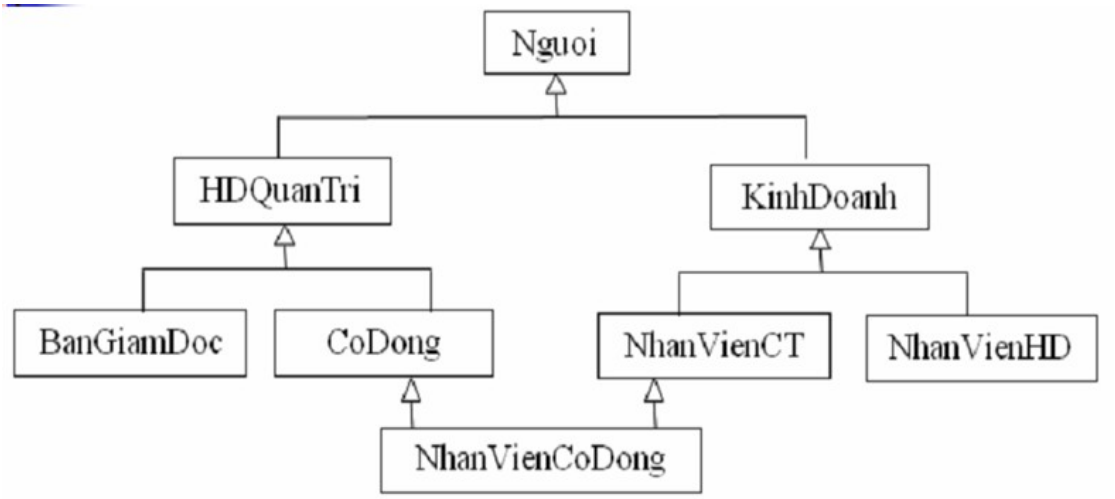
Hình 2.12: Quan hệ kết tập chia sẻ



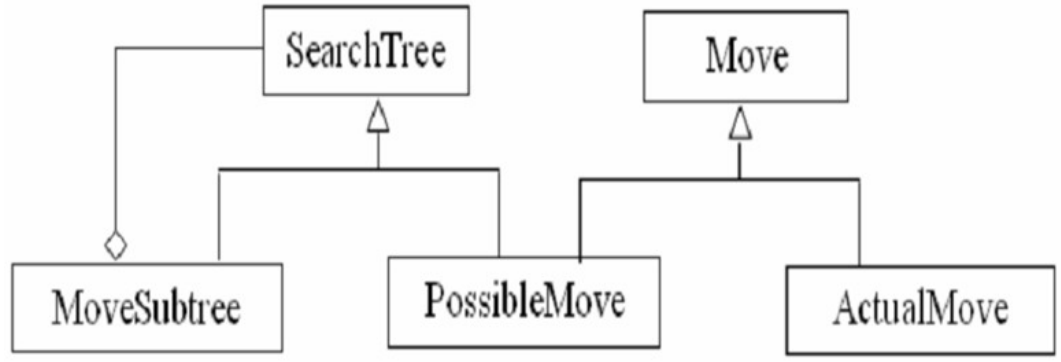
Hình 2.13: Quan hệ kết tập hợp thành



Hình 2.14: Quan hệ tổng quát hóa



Hình 2.15: Kế thừa bội từ 2 lớp khác nhau, có chung lớp cơ sở



Hình 2.16: Kế thừa bội không có chung lớp cơ sở

b. **Thiết kế lược đồ tiêu chuẩn (ODMG):** Chuyển một lược đồ khái niệm biểu thị trong kí pháp UML về một lược đồ ODMG (Object Database Management Group).

   Sự tổng quát hoá được ngầm định trong UML là tổng quát hoá rời nhau, không đầy đủ và được hỗ trợ trực tiếp bởi mô hình dữ liệu ODMG thông qua mối quan hệ EXTEND.

   - Mỗi lớp bền vững UML được dịch thành một lớp ODL.
   - Mỗi giao diện UML được dịch thành một giao diện ODL.
   - Mỗi thuộc tính được dịch sang một thuộc tính. Nếu là thuộc tính đa trị được dịch sang một kiểu sưu tập (collection type)

- Các mối liên kết được định nghĩa là các mối quan hệ trong ODL.
    - Số bội (multiplicity) (bao gồm số bội cực đại và cực tiểu) biểu diễn có bao nhiêu đối tượng của một lớp có thể được kết hợp với một đối tượng xác định của lớp có liên quan.
    - Trong ODMG, số bội cực đại hỗ trợ định nghĩa mối quan hệ. Nếu số bội cực đại lớn hơn 1, mối quan hệ được định nghĩa bởi kiểu sưu tập (tập, danh sách hay túi)
- UML hỗ trợ hai cách biểu diễn gộp nhập:
    - Gộp nhập phần tử - bộ sưu tập (member-collection aggregation): biểu diễn một bộ sưu tập các đối tượng, tất cả thuộc cùng một lớp và cùng với nhau làm thành một lớp mới. Ví dụ một bộ sưu tập các cây làm thành một rừng.
    - Gộp nhập bộ phận – toàn thể (part-whole aggregation): biểu diễn một lớp có cấu trúc gồm hai lớp hầu như khác nhau

c. **Thiết kế lược đồ cài đặt (Poet 4.0):** Dịch lược đồ thiết kế chuẩn thành lược đồ cài đặt trong POET 4.0

5. **Các thành phần đặc trưng của kiểu dữ liệu hướng đối tượng**

5.1. Kiểu dữ liệu hướng đối tượng

- Kiểu lớp và giao diện :
    - Một kiểu xác định các tính chất chung (các thuộc tính và liên kết) và hành vi (thao tác) của một tập các phần tử. Các giá trị của những tính chất của một đối tượng có thể thay đổi bất kì lúc nào.
    - Một kiểu có một đặc tả ngoài và một hay nhiều cài đặt. ODL hỗ trợ đặc tả ngoài với ba kết cấu: giao diện, lớp và literal
        - Một định nghĩa của giao diện là một đặc tả chỉ định nghĩa hành vi trừu tượng của một kiểu đối tượng.
        - Định nghĩa của lớp là một đặc tả định nghĩa dáng điệu trừu tượng và trạng thái trừu tượng của một kiểu đối tượng.
        - Định nghĩa của literal chỉ định nghĩa trạng thái trừu tượng của một literal.
    - Việc cài đặt của một kiểu đối tượng phải được thực hiện bởi một ràng buộc ngôn ngữ
- Kiểu con và tính kế thừa
    - Mô hình dữ liệu ODMG hỗ trợ 2 loại liên kết kế thừa:
        - Liên kết is-a (biểu diễn bởi :): định nghĩa tính kế thừa hành vi giữa các kiểu đối tượng, hoặc là giao diện hoặc là lớp.
        - Liên kết EXTENDS (biểu diễn bởi từ extend) chỉ tính kế thừa trạng thái. Nó chỉ áp dụng cho kiểu đối tượng.
    - Như vậy, chỉ có các lớp có thể kế thừa trạng thái, các literal thì không.
- Kiểu sưu tập (collections)
    - Một sưu tập là một kiểu có số phần tử biến đổi, tất cả đều cùng kiểu.

- Mô hình dữ liệu ODMG hỗ trợ các kiểu sưu tập (đối tượng hay literal): tập, túi, danh sách, từ điển và bảng.

- Kiểu có cấu trúc (structured types)
  - Là kiểu có số cố định phần tử, có thể thuộc nhiều kiểu khác nhau.
  - Mô hình ODMG hỗ trợ các kiểu có cấu trúc (đối tượng hay literal): date, interval, time và timestamp.
  - Ngoài ra ODMG còn cho phép người dùng định nghĩa các kiểu có cấu trúc mới.
- Ngoại diên (extents) của một kiểu là nhóm (bộ sưu tập) của tất cả các đối tượng (thể hiện-instances) của kiểu.
- Khoá (keys): là một hay một tập thuộc tính xác định duy nhất mỗi đối tượng của một kiểu (giống khái niệm khoá dự tuyển của mô hình quan hệ).

5.2. Tính chất của các đối tượng

- Tính bền vững của các đối tượng:
  - Các đối tượng cần nằm chắc chắn trên phương tiện nhớ như đ a từ, khi được một chương trình tạo ra.
  - Đối tượng bền vững: là đối tượng được lưu giữ trong CSDL, có thời gian tồn tại dài hơn thời gian của chương trình tạo ra đối tượng đó.
  - Đối tượng tạm thời: là đối tượng được lưu trong bộ nhớ trong; do vậy thời hạn tồn tại của nó không quá thời hạn của chương trình tạo ra đối tượng đó.
- Tính khai thác tương tranh:
  - CSDL đối tượng cho phép các giao tác dùng chung. Việc khoá giao tác, khoá dữ liệu cần hạn chế để đảm bảo tính tương hợp về dữ liệu.
- Tính tin cậy của đối tượng:
  - Những đối tượng có thể khôi phục lại khi có sai sót xảy ra. Các giao tác cần chia nhỏ để đảm bảo hoặc chúng được thực hiện hoàn toàn, hoặc không thực hiện tí gì
- Tính tiện lợi tra cứu:
  - Người ta yêu cầu tìm được các đối tượng theo giá trị của thuộc tính đối tượng.
  - Do vậy cần quản lý tận giá trị thuộc tính, các kết quả của phương pháp, các liên hệ giữa các đối tượng.
- Chức năng khác:
  - Phân bố các đối tượng
  - Những mô hình về các giao tác
  - Những thế hệ của các đối tượng

5.3. Quản lý tính bền vững của các đối tượng

Một mô hình CSDL đối tượng cho phép xác định các loại dữ liệu của đối tượng. Trong môi trường lập trình, các đối tượng cần được xây dựng và bị huỷ bỏ trong bộ nhớ nhờ các chức năng đặc biệt, gọi là bộ tạo dạng và bộ huỷ bỏ.

- **Tạo dựng đối tượng**: chức năng gắn với một lớp cho phép tạo nên và khởi động một đối tượng trong bộ nhớ.
- **Huỷ bỏ đối tượng**: chức năng gắn với một lớp cho phép huỷ một đối tượng ra khỏi bộ nhớ.

Vấn đề đặt ra trong CSDL hướng đối tượng là đảm bảo tính bền vững của các đối tượng trên (ra theo cách có thể tận lại được nó. Một giải pháp thường dùng để bảo vệ các đối tượng trên đ a gồm việc đặt tên mỗi đối tượng bền vững và trang bị một chức năng cho phép một đối tượng đã trên đ a là bền vững.

- **Thừa kế tính bền vững**: Kỹ thuật cho phép xác định chất lượng của đối tượng là bền vững do thừa kế từ lớp gốc, khiến cho các đối tượng được kích hoạt hay ngừng hoạt động.
- **Tính bền vững do tham chiếu**

Kỹ thuật cho phép xác chất lượng bền vững của đối tượng nhờ từ khoá, tức gốc của bền vững, hoặc nhờ việc nó được đối tượng bền vững khác tham chiếu đến.

BÀI 3: NGÔN NGỮ TRUY VẤN DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

1. Giới thiệu một số ngôn ngữ truy vấn dữ liệu đối tượng

Ngôn ngữ truy vấn dữ liệu đối tượng OQL *(Object Query Language)* cho phép thao tác của những mở rộng của lớp như:

Đường dẫn truy xuất dữ liệu

Thuộc tính, phương thức hay liên kết

6. Ngôn ngữ truy vấn dữ liệu đối tượng OQL tương thích với đa hệ quản trị

OQL không chỉ tương thích với đa hệ quản trị như: MS SQL server, Oracle, Postgres... mà còn tương thích với nhiều ngôn ngữ lập trình: C++, Java, Smalltalk...

7. Cú pháp OQL

1.1. Quy ước

OQL sử dụng các mệnh đề giống như ngôn ngữ SQL

1.2. Ngữ pháp OQL

Cú pháp:

SELECT DISTINCT <expression>
FROM <list of collections>
WHERE <expression>
GROUP BY <attributes>
HAVING <condition>
ORDER BY <expression>

Trong đó:

Danh sách tập dữ liệu (list of collections):

Định nghĩa biến trong câu truy vấn

FROM Props AS L

Định nghĩa biểu thức (clause IN): với những truy vấn hoàn thiện hay cho phép truy vấn những quan hệ tạm thời

Cho phép thao tác:

Mở rộng của một lớp

Có thể là một biểu thức mà kết quả cho ra một tập hợp

Biểu thức có thể là:

Biểu thức cơ bản: một lớp, một thuộc tính, một hàm…

Biểu thức xây dựng: hỗ trợ xây dựng struct, list, bag, set, array hay biểu thức trên những tập hợp hay những đối tượng

Những biểu thức nguyên tử: nối chuỗi, trích chuỗi con…

Tính chất

Số lượng lớn các biểu thức

Hoàn thiện về ngôn ngữ

8. **Bài tập ví dụ minh họa**

Ví dụ 1: Tìm kiếm NAS, LastName và Street của những chủ sở hữu ở Montréal

```
SELECT L.NAS, L.LastName, L.Address.Str
FROM Props L
WHERE L.Address.City = "Montréal"
```

Ví dụ 2: Tìm số và địa chỉ của những căn hộ chung cư của chủ sở hữu Pierre Tremblay

```
SELECT A.No, A.Address
FROM Props AS L, L.Posses AS A
WHERE L.LastName="Tremblay"
AND L.FirstName="Pierre"
```

Phép kết nối được diễn đạt trong mệnh đề from nhờ vào định nghĩa tập hợp mới

Ví dụ 3:

Tìm kiếm đồng thời một đối tượng: Những HQTCSDL hướng đối tượng cho phép truy cập đồng thời một đối tượng duyệt qua sự phân cấp và sử dụng định danh đối tượng (OID)

Truy xuất thông qua định danh đối tượng: Tìm giá của căn hộ chung cư có định danh đối tượng là o1

o1 price

Truy xuất bằng duyệt từng phần tử của tập hợp các đối tượng của lớp appartment

Hiển thị giá của căn hộ chung cư

```
{
for (p in Appartment)
printf (p->price)
}
```

Ví dụ 4:

Duyệt qua sự phân cấp

Biểu thức nối kết không tường minh: tính diện tích phòng tắm của căn hộ chung cư tại địa chỉ 31 Pins

```
SELECT p.surface
FROM a IN Appartments,
p IN a.Rooms !! jointure implicit !!
WHERE a.address = "31 Pins"
AND p.type = "Shower Room"
```

Biểu thức nối kết tường minh: Tìm chủ sở hữu mà họ ở cùng thành phố với nhau

```
SELECT p1.lastname, p2.lastname, p1.city
```

FROM p1 IN Props,

p2 IN Props

WHERE p1.nas <> p2.nas !! jointure explicit !!

AND p1.city = p2.city !! jointure explicit !!

Tìm phòng diện tích nhỏ hơn 5m2

SELECT Rooms*

WHERE surface < 5

Tìm phòng diện tích nhỏ hơn 5m2, không phải phòng tắm

SELECT Rooms* difference ShowerRoom

WHERE surface < 5

Ví dụ 5:

Sử dụng các phương thức

Có thể sử dụng phương thức của đối tượng trong truy vấn

Tiếp cận hàm, tiếp cận theo phương châm hàm hoàn toàn có thể được sử dụng những thuộc tính được định nghĩa như những hàm

Một số phương thức có thể được định nghĩa từ lời củaphương thức khác

Tính diện tích căn hộ chung cư giá 30 000$

O2 select surface(a)

from a in appartment

where a.price = 30 000

BÀI 4: HỆ QUẢN TRỊ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

1. Giới thiệu một số hệ quản trị dữ liệu đối tượng

1.3. Một số hệ quản trị cơ sở dữ liệu hướng đối tượng

Cơ sở dữ liệu hướng đối tượng: dữ liệu cũng được lưu trữ trong các bản dữ liệu nhưng các bảng có bổ sung thêm các tính năng hướng đối tượng như lưu trữ thêm các hành vi, nhằm thể hiện hành vi của đối tượng. Mỗi bảng xem như một lớp dữ liệu, một dòng dữ liệu trong bảng là một đối tượng.

Các hệ quản trị có hỗ trợ cơ sở dữ liệu hướng đối tượng như: MS SQL server, Oracle, Postgres...

1.4. Các tính năng bắt buộc của hệ quản trị cơ sở dữ liệu hướng đối tượng

a. **Đối tượng phức**

- Định nghĩa:
  - Một đối tượng phức là đối tượng có một hai nhiều thuộc tính tham chiếu đến đối tượng khác hoặc đến một tập các giá trị.
  - Một đối tượng có thể chứa nhiều đối tượng con
- *Ví dụ:* Một căn hộ bao gồm nhiều phòng

o451 : tuple (No : 10,
Address : "71 Pins",
Nb-Room : 4;
Rooms : set (o5, o6, o7, o8)
Price : 35 000)

b. **Định danh đối tượng**

- Định nghĩa:

Tất cả các đối tượng phải có 1 định danh (OID) duy nhất cho phép tham chiếu đến nó. Định danh này bất biến trong suốt chu trình sống của đối tượng.

- *Ví dụ:* Định danh của đối tượng "Appartment số 10" là: o451
- Khái niệm liên quan:

Tính đồng nhất của đối tượng: Hai đối tượng là một nếu chúng có cùng định danh

- HQTCSDL HĐT phải hỗ trợ OID và tính đồng nhất của đối tượng

c. **Đóng gói**

- Định nghĩa:

Không được thao tác trực tiếp trên giá trị của đối tượng, chỉ có thể thao tác thông qua các phương thức được cho phép. Đây là nguyên lý của tính đóng gói.

- *Ví dụ:*
  - Không được thao tác trực tiếp trên các thuộc tính của đối tượng
  - Sử dụng các hàm để lấy (hoặc đặt lại) giá trị của thuộc tích
- HQTCSDL HĐT phải hỗ trợ tính năng đóng gói

d. **Kiểu hoặc lớp**

- Định nghĩa:
  - Kiểu mô tả cách mà các giá trị được thao tác, cho phép kiểm tra tính hợp lệ của kiểu đối với phép toán nào đó
  - Lớp bao gồm tập các đối tượng có cùng các thuộc tính và hành vi chung. Lớp mô tả cấu trúc đối tượng và hành vi của nó và cho phép tạo ra đối tượng của lớp này.
- HQTCSDL HĐT phải hỗ trợ Kiểu hoặc Lớp hoặc hỗ trợ cả 2

e. **Định nghĩa chồng và ràng buộc muộn**

- Overriding (định nghĩa lại): phương thức được định nghĩa cho mỗi trường hợp sử dụng
- Overloading (nạp chồng): sử dụng một tên chung (của phương thức) cho nhiều bản cài đặt khác nhau
- Late binding (ràng buộc muộn): hệ thống xác định phương thức nào được sử dụng lúc thực thi (chứ không phải lúc biên dịch)
- HQTCSDL HĐT phải cho phép định nghĩa lại các phương thức

f. **Quản lý cạnh tranh**

g. **Tính Đầy đủ và khả năng mở rộng**

- Đầy đủ: Khả năng biểu diễn của ngôn ngữ lập trình hỗ trợ tất cả các kiểu thao tác
- Khả năng mở rộng: khả năng mở rộng tập kiểu cơ sở cho phép xây dựng kiểu mới

h. **Lưu trữ lâu dài và thao tác dữ liệu**

- Lưu trữ lâu dài
  - Khả năng lưu trữ đối tượng càng trong suốt càng tốt
  - Lưu trữ lâu dài phải độc lập với kiểu đối tượng
- Ngôn ngữ thao tác dữ liệu
  - Ngôn ngữ cấp cao
  - Độc lập với ứng dụng
  - Thích nghi cho việc tối ưu hóa câu truy vấn

i. **Tin cậy và cạnh tranh**

- Tin cậy: cơ chế hoạt động của hệ thống phải được định nghĩa để có thể đối mặt với sự cố
- Cạnh tranh: hỗ trợ chia sẻ dữ liệu giữa các người dùng khác nhau
- Các cơ chế này phải thích nghi cho từng kiểu đối tượng khác nhau

j. **Kế thừa**

- Định nghĩa:

Sự phân cấp kiểu hoặc lớp cho phép mô tả kiểu con hoặc lớp con thừa kế các đặc tính (cấu trúc và hành vi) của kiểu cha hoặc lớp cha.

- *Ví dụ:*
  - Person (người), Prop (người chủ sở hữu), Locator (người thuê nhà)
- HQTCSDL HĐT phải hỗ trợ thừa kế.

k. **Quản lý bộ nhớ ngoài**

- Cho phép lưu trữ và truy xuất dữ liệu:
  - Đối tượng lớn
  - Đối tượng phức
- Cung cấp cơ chế truy xuất hiệu quả
  - Chỉ mục
  - Gom nhóm dữ liệu
  - Quản lý vùng đệm
  - Phương pháp truy xuất

1.5. Các thành phần của hệ quản trị cơ sở dữ liệu hướng đối tượng

- Tính năng tùy chọn
  - Đa thừa kế
  - Kiểm tra và suy diễn kiểu
  - Phân tán
  - Giao dịch thiết kế
  - Quản lý phiên bản
- Tính năng mở
  - Họ ngôn ngữ lập trình
  - Hệ thống biểu diễn
  - Hệ thống kiểu
  - Tính đồng nhất của hệ thống

1.6. Chuẩn của hệ quản trị cơ sở dữ liệu hướng đối tượng

- OM: mô hình dữ liệu đối tượng
- ODL: ngôn ngữ định nghĩa dữ liệu
- OQL: ngôn ngữ thao tác dữ liệu
- Bindings: quan hệ với ngôn ngữ lập trình

9. **Cài đặt, cấu hình tích hợp với môi trường phát triển ứng dụng**

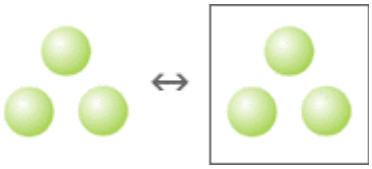
Sử dụng LINQ

10. **Khai thác công cụ (Versant, DB4o)**

1.7. Giới thiệu hệ cơ sở dữ liệu hướng đối tượng DB4O

Db4o là cơ sở dữ liệu hướng đối tượng nguồn mở tương tác với môi trường phát triển Java và .Net để *store* và *retrieve* bất kỳ đối tượng ứng dụng nào chỉ với *one line of code,*loại trừ tới nhu cầu *predefine* hoặc bảo trì một mô hình dữ liệu riêng biệt, cứng rắn.

Những người cách tân muốn triển khai *db4o database engine* tại những giữa những thiết bị điều khiển bằng dữ liệu phát sinh và những ứng dụng về *cache user-generated* và *client-side data*, cho phép những đặc tính mới hấp dẫn và đạt được sự thực hiện và tính linh hoạt chưa hề thấy.



Hình 4.1: One-Line-of-Code Database loại trừ Complexity

Bí mật của db4o là sự thành công, sự tích hợp dễ dàng với đối với java và .NET. Việc lưu trữ những đối tượng dữ liệu chính xác với cách mà họ định nghĩa bởi ứng dụng. Bởi vậy mà db4o dễ dàng hợp nhất vào trong ứng dụng và thực hiện những nhiệm vụ đáng tin cậy .

Dưới đây là một số đặc tính của cơ sở dữ liệu hướng đối tượng db4o.

Platforms

Java	- J2EE. - J2SE. - J2ME with reflection: CDC, PersonalProfile, Symbian, JavaFX Mobile and Zaurus; on demand(2): J2ME w/o reflection (CLDC and MIDP), including RIM/Blackberry and Palm OS. - Android (Open Handset Alliance). - Harmony. - Supported Frameworks: Spring, OSGi, Tomcat, JPOX.
.NET	- .NET Framework (1.0(1), 2.0, 3.0, 3.5). - .NET Compact Framework 1.0(1), 2.0. - Windows (XP, Vista). - Windows Mobile / PocketPC. - Mono(1). - Supported Frameworks: Castle, Eiffel, Spring.net.
Cross-platform	- Connect to .NET server with Java client. - Connect to Java server with .NET client.

Languages

Java	JDK 1.1.x - JDK 6.0
.NET	. All managed .NET languages (C#, VB.NET, ASP.NET, Boo, Managed

	C++ etc.)

Commands

Sessions	JDK 1.1.x - JDK 6.0
Database files	Create, open, close, and delete
Transactions	Commit, and Rollback
Objects	Store, retrieve, update (incl. cascaded), replicate, delete (incl. cascaded)
Messaging	TCP/IP

Transparency

Language constructs	- Primitive types - Strings - Arrays - Multi-dimensional arrays - Inner classes - Java/C# collections - Classes without public constructors - .NET structs - Blobs (stored outside of DB file)
Non-Intrusive	- Without deriving from a specific base class. - Without implementing a specific interface. - Without modifications to source code. - Without implementing Serializable.
Private Fields	Storable
File I/O	Pluggable
Reflector	- Pluggable - Generic
Aliases	Class aliasing for class-to-class mappings

Query Languages / APIs

Object oriented	- Native Queries (NQ) - Query By Example (QbE) - S.O.D.A.

SQL	Via replication to many relational databases
XML	With Third-Party products (e.g., Xstream)

Modes / Concurrency

Operation Modes	- Local - Client/Server
Threads	Multiple
Transactions	Multiple, parallel
Semaphores	Available
Read-Only Mode	Available

Scalability and Performance

Performance benchmark	Up to 55x faster than Hibernate/MySQL
Examples of Scalability	- Stores 200,000 objects/second - Stores 300,000 objects on a PDA
In Memory Mode	Available
Client-side	Single-process execution available
Server-side	Server-side query execution available
DB-aware Collections	Available
Object Caching	Available
Pagination	erver-side cursors (lazy queries)
Indexing	- BTree field indexes - BTree query processor

1.8. Các loại lớp trong hệ thống đối tượng của DB4o

*Class Types :*Một hệ thống hướng đối tượng được xây dựng từ những lớp có ảnh hưởng và tương tác với nhau. Một số thực thể class trong hệ thống mà bao gồm các thực thể trừu tượng hóa trong thế giới thực và những lớp khác thì đảm nhiệm vai trò khác trong hệ thống. Các loại class được chia thành :

- ✓ *Entity classes:* Đây là mô hình thực thể hay dữ liệu trong hệ thống.
- ✓ *Boundary classes :* Đây thực chất cung cấp các giao diện để tương tác giữa hệ thống với môi trường bên ngoài, thông thường là giao diện người dùng, giao diện có thể là các *GUI Component* hay *web page.*

- ✓ *Control classes :* Lớp điều khiển những luồng trong hệ thống, điển hình là đưa đầu vào từ giao diện người dùng và sử dụng những lớp thực thể để thực hiện '*business logic'* và lần lượt cung cấp giao diện với kết quả mà người dùng sử dụng trả lại.

1.9. Object Indentity

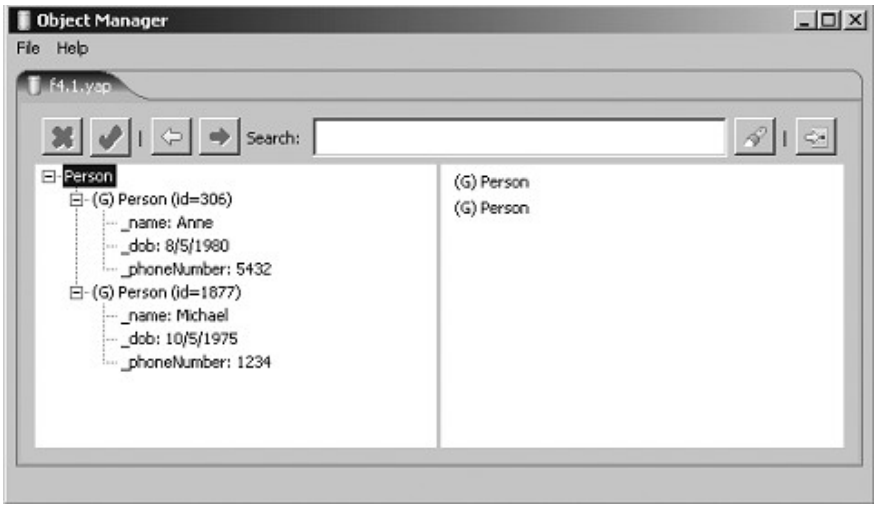
Trong cơ sở dữ liệu đối tượng và quan hệ đưa nhưng cách tiếp cận khác về ý tưởng *indentity. Indentity* xác định những thực thể được phân biệt với một thực thể khác. Trong cơ sở dữ liệu quan hệ, thì thực thể được *indentity* bởi khóa chính và khóa ngoại của dữ liệu. Những mối quan hệ được sử dụng trong những câu truy vấn như yêu cầu *join* giữa hai *table* với nhau. *Indentity* phụ thuộc vào giá trị của trường khóa.

Trong sự tương phản thì một OODBMS lưu trữ một định danh đối tượng *OID* bên trong mỗi đối tượng. *OID* xác định một đối tượng duy nhất trong OODBMS và cũng được dùng để chỉ tham chiếu tới một đối tượng khác tham chiếu tới đối tượng đó.*OID* thông thường là một số và không được hiển thị đối với người sử dụng và cơ sở dữ liệu.

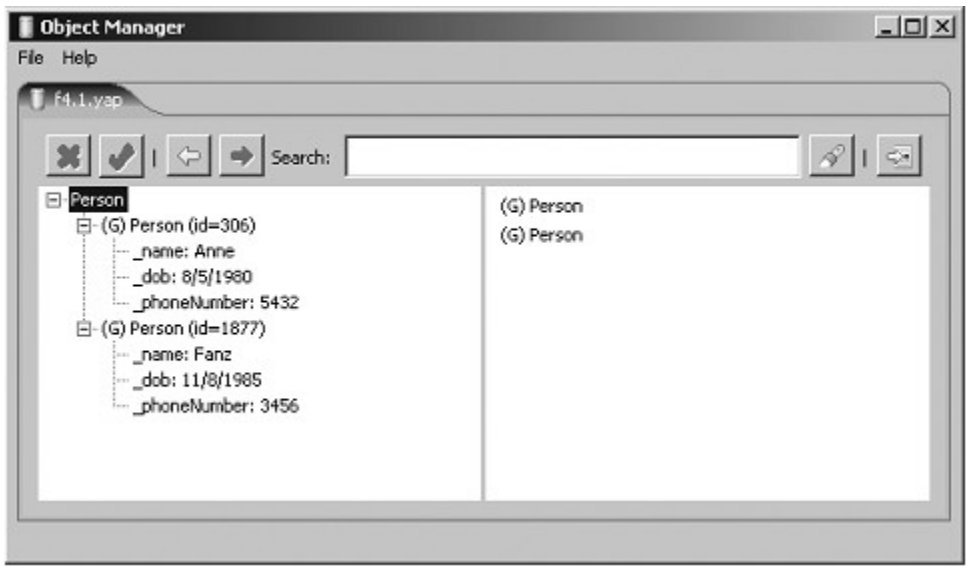
*Hình 4.*2 mô tả hai đối tượng được lưu trữ trong cơ sở dữ liệu db4o, dùng công cụ *objectManager* để duyệt dưới dạng đồ thị. Cả hai đối tượng là những thể hiện của một lớp *person* đã được gán *OIDs* 306 và 1877 tương ứng. *OID* này được gán bởi chương trình cơ sở dữ liệu khi đối tượng đó được lưu vào cơ sở dữ liệu. Trong db4o thì *OID* là một con trỏ vật lý trỏ tới *database file*.

Ở đây ta có một nhận định rằng, *OID* không liên quan tới bất kỳ một thuộc tính nào của đối tượng hay tới class của đối tượng. Có hai vấn đề liên quan tới nó:

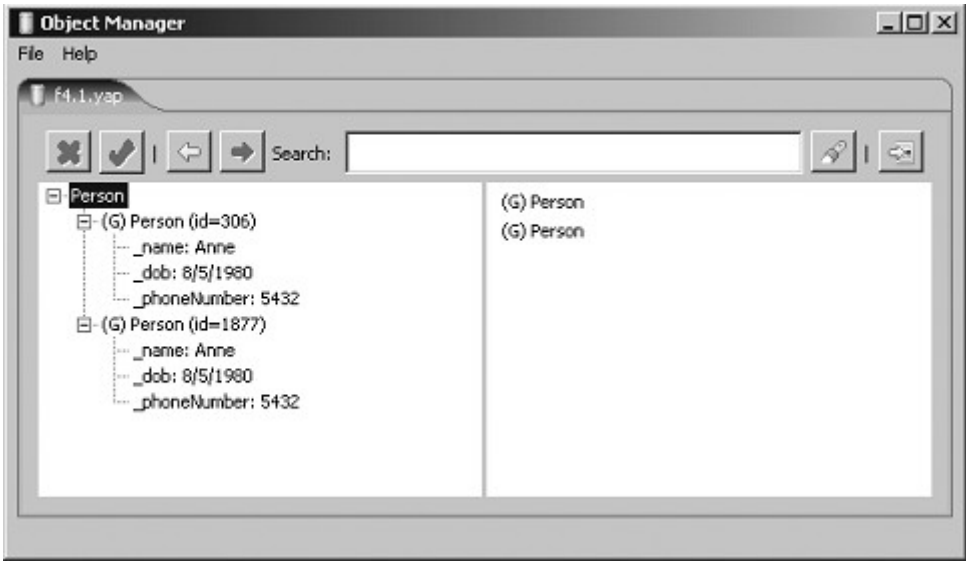
- ✓ Hai đối tượng có thể phân biệt hoàn toàn dù rằng tất cả các thuộc tính của nó đều giống nhau nhưng chúng có *OID* hoàn toàn khác nhau, dễ dàng tạo ra được hai đối tượng với các thuộc tính đồng nhất. Với db4o ta không thể định nghĩa key để đảm bảo tính duy nhất trong hệ thống, cái này thì ứng dụng sẽ quản lý.
- ✓ Một đối tượng được duy trì cùng với đối tượng thậm chí khi trạng thái của nó nhận giá trị mới hoàn toàn.



Hình 4.2: Db4o database với hai objects, được view bởi ObjectManager



Hình 4.3: Ví dụ databse db4o với tất cả các thuộc tính thay đổi



Hình 4.4. Hai đối tượng person với các thuộc tính

1.10. Lược đồ cơ sở dữ liệu trong db4o (database schema)

Bây giờ ta có thể nhìn thấy nội dung của một cơ sở dữ liệu đối tượng lưu trữ một đối tượng *person* rất đơn giản. Để làm được điều này một thì một *database schema* được định nghĩa hỗ trợ việc lưu trữ dữ liệu. *Database schema* là một cấu trúc của một hệ thống cơ sở dữ liệu. Phần này sẽ mô tả những *object* trong *database schema* và một số khía cạnh khác nữa.

Properties và *Relationships:* Trong cơ sở dữ liệu quan hệ thì *schema* được định nghĩa là các *table* và quan hệ giữa các *table.* Trong cơ sở dữ liệu hướng đối tượng thì *schema* được định nghĩa bởi các *classes* và các thuộc tính của *class* và quan hệ giữa các đối tượng.

Database schema dùng để định nghĩa ngôn ngữ dữ liệu (DDL). Trong cơ sở dữ liệu quan hệ thì việc sử dụng DDL như cú pháp tạo table như *CREATE TABLE* để định nghĩa cấu trúc của *table* . Cũng giống như vậy cơ sở dữ liệu đối tượng sử dụng ngôn ngữ định nghĩa đối tượng (ODL) để mô tả cấu trúc của đối tượng, nó là một phần trong tiêu chuẩn của ODMG. Ví dụ một *databas schema* sử dụng ODL định nghĩa như sau :

```
class Person
{
   attribute string name;
   attribute string dob;
   attribute string phoneNumber;
}
```

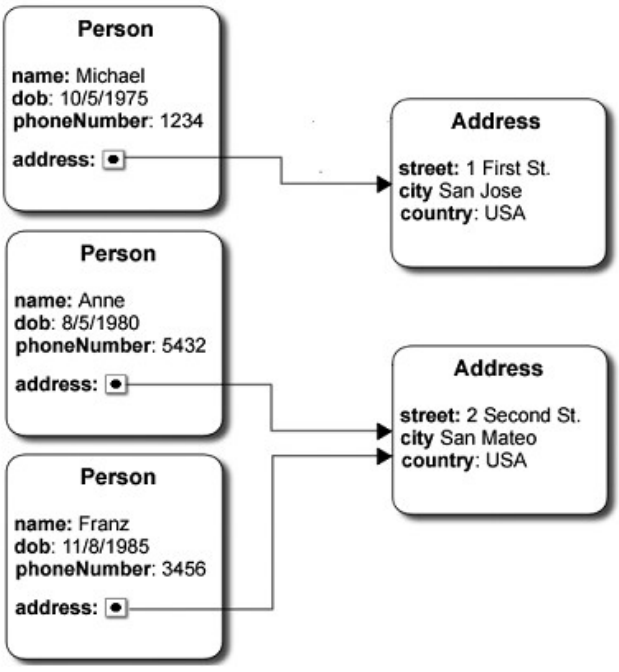
Chú ý rằng mặc dù định nghĩa lớp này trông hơi tương tự như định nghĩa một lớp trong java hay C#, nó không thực sự chỉ ra là của bất kỳ ngôn ngữ lập trình hướng đối tượng nào cả. Nó đơn giản là dùng để định nghĩa một *database schema*. Để truy cập cơ sở dữ liệu từ một ứng dụng thì cần phải sử dụng một ngôn ngữ lập trình hướng đối tượng để *binding* mà cơ sở dữ liệu maps vào các *class* của ngôn ngữ java hay C# và cho phép thực hiện các thao tác và query.

1.11. Object Relationships

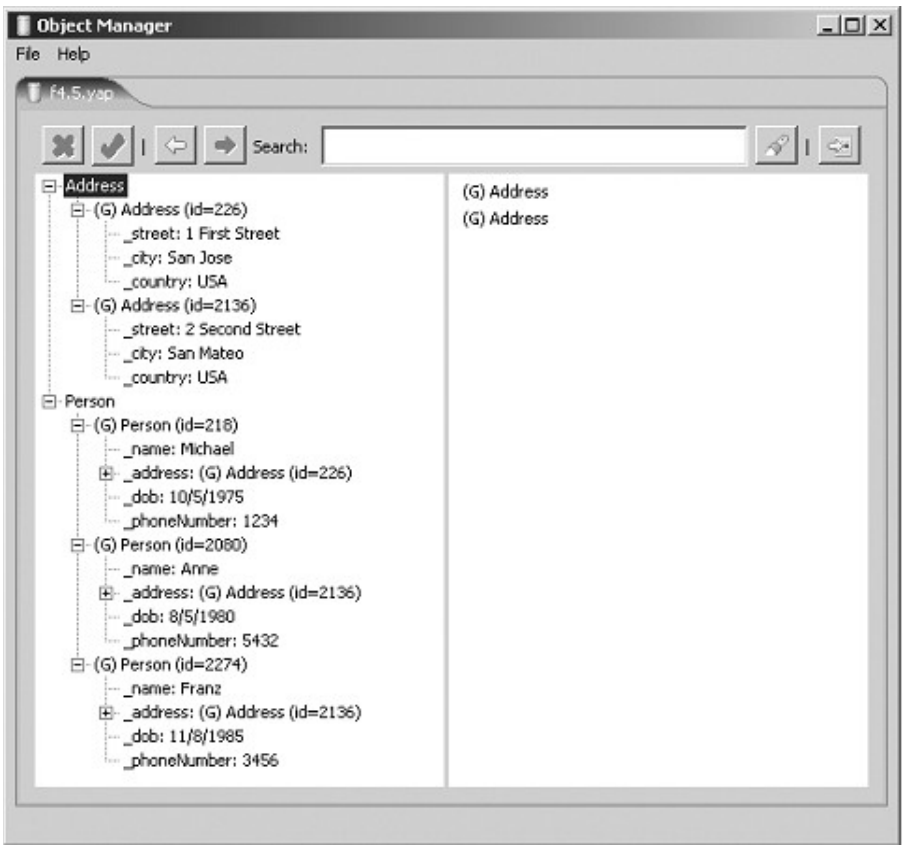
Mô hình dữ liệu đối tượng về cơ bản thì cũng như *object model*, vì thế các mối quan hệ của đối tượng được duy trì.

Storing OIDs to Represent Relationships những mối quan hệ giữa các đối tượng được lưu vào bộ nhớ *memory* và được duy trì bởi các đối tượng tham chiếu tới nhau. Một đối tượng được lưu trữ và cơ sở dữ liệu hướng đối tượng thì đối tượng tượng đó luôn được tham chiếu sẵn, vì thế những đối tượng này được duy trì những mối quan hệ bằng cách lưu trữ *OID* của bất kỳ đối tượng nào liên quan hay tham chiếu tới đối tượng đó.

Trong *hình 3.x* mô tả một đối tượng được lưu trữ, nó được minh họa dưới dạng đồ thị đối tượng (*object graph*). Có ba đối tượng *person* và hai đối tượng *address* ,hình 3.x cho thấy một cơ sở dữ liệu db4o chứa năm đối tượng này.Phạm vi của class *person* là ba đối tượng còn phạm vi của class *address* chỉ có hai đối tượng. Đối tượng *person* ở đây lưu trữ thêm một thuộc tính bổ sung đó là thuộc tính *address,*thuộc tính này sẽ tham chiếu đến đối tượng có trong cơ sở dữ liệu và hơn nữa nó là một ký tự giống như các thuộc tính khác

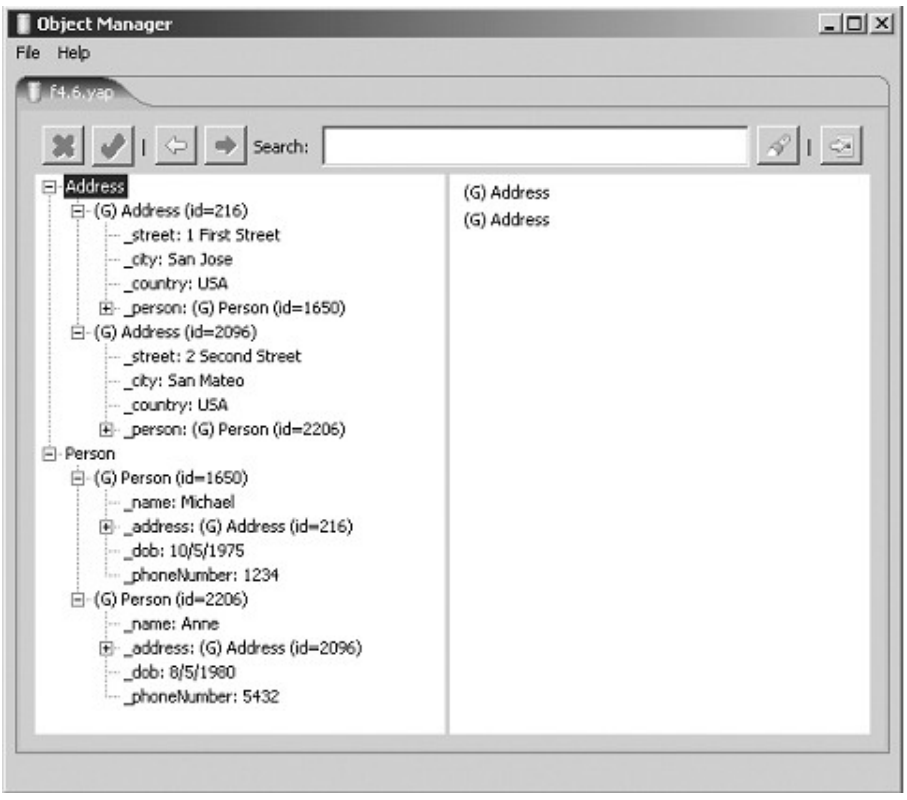


Hình 4.5: Một số object graph của objects

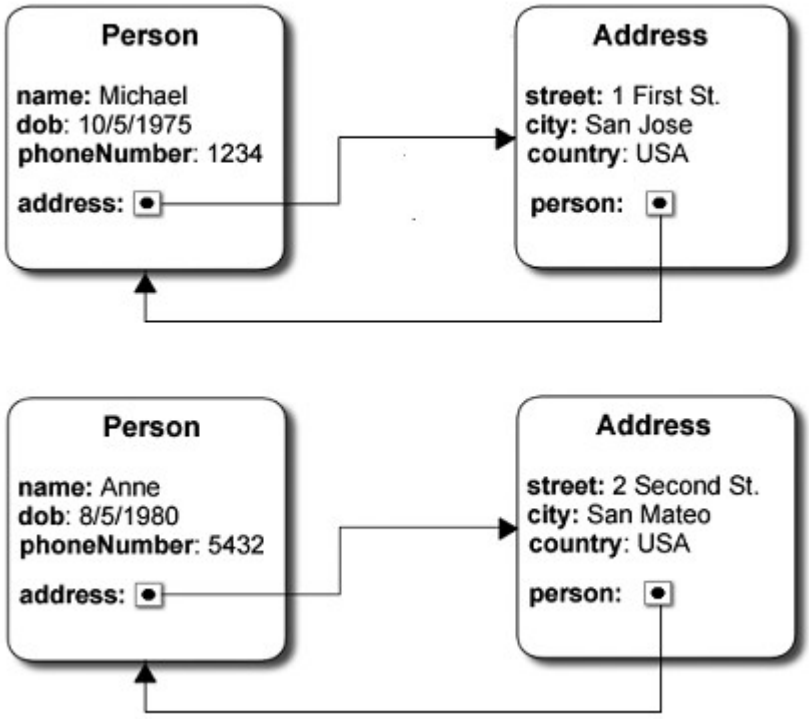


Hình 4.6: Các đối tượng trong database db4o

Inverse Relationships : Nếu ta nhìn vào đối tượng *address* thì làm sao ta có thể biết được *person* nào sống ở đó, điều đó là không thể, vì những đối tượng *address* không cất giữ *OID* tham chiếu đến đối tượng *person* . Mặt khác nếu ta tìm địa chỉ của một *person* nào đó thì hoàn toàn có thể xác định được là do đối tượng *person* lưu trữ *OID* của đối tượn *address*



Hình 4.7: Các đối tượng liên quan trong quan hệ ngược



Hình 4.8: Object graph của objects trong quan hệ ngược

Defining the Relationships : Ta có thể dùng ODL để định nghĩa những mối quan hệ bên trong một mô hình cơ sở dữ liệu ODMG. ODL định nghĩa mối quan hệ ngược lại để giám sát việc thi hành sự toàn vẹn mối quan hệ. Hình 3.x định nghĩa class *person* và *address.*

```
class Person
{
   attribute string name;
   attribute string dob;
   attribute string phoneNumber;
   relationship Address address
      inverse Address :: person;
}
class Address
{
   attribute string street;
   attribute string city;
   attribute string country;
   relationship Person person
      inverse Person :: address;
}
```

Hình 4.9: Định nghĩa class Person và Address trong ODL

Có nghĩa là đối tượng *person* có quan hệ tới đối tượng *address* và quan hệ giữa hai đối tượng này có tên là *address*, và mối quan hệ ngược *address* tới *person*

Integrity : như chúng ta đã thấy *ODL* yêu cầu những mối quan hệ ngược để giúp đỡ đảm bảo sự toàn vẹn mối quan hệ, tương tự đối với cơ sở dữ liệu quan hệ là việc tạo khóa ngoại để tham chiếu tới các *entity* khác, ràng buộc toàn vẹn ở đây là có thể một *person* không thể không có *address*

1.12. Các kiểu quan hệ của đối tượng

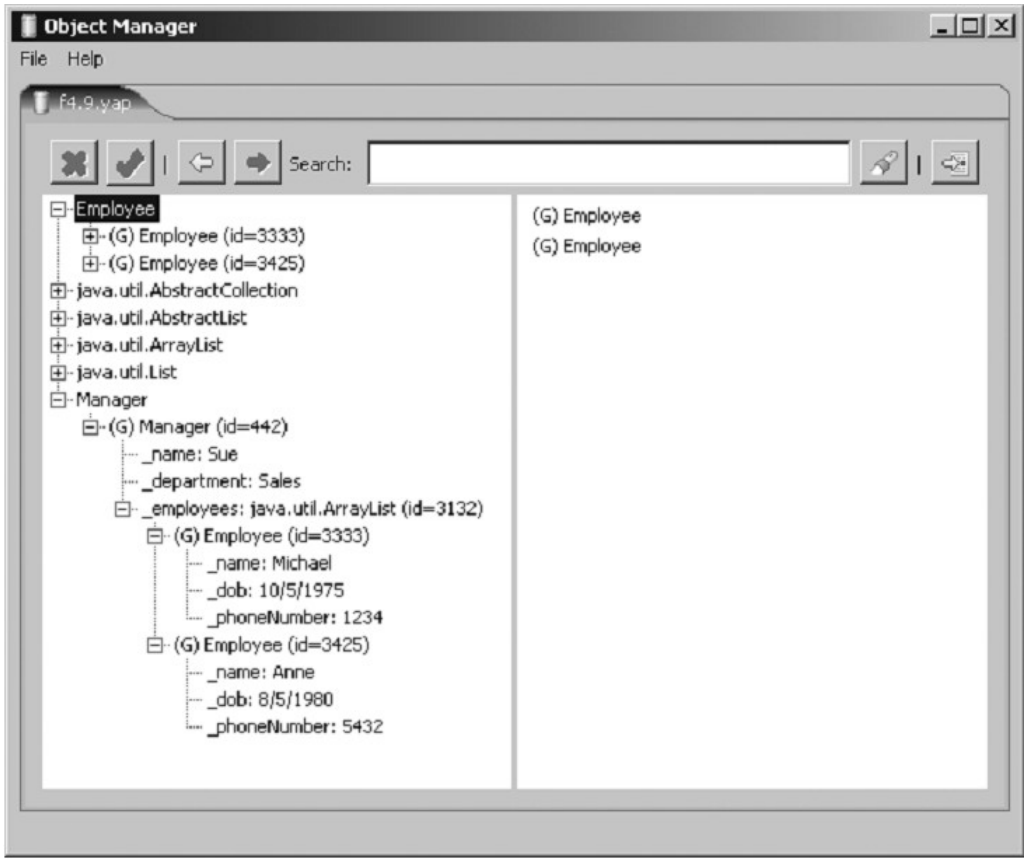
Trong phần này sẽ trình bày đến mối quan hệ của đối tượng trong ODB đó là quan hệ *one-to-many* và *many-to-many* hay *aggregation* và *inheritance.*

Quan hệ *one-to-many* : khi lưu trữ một đối tượng thì vấn đề thường liên quan tới tập hợp các đối tượng các đối tượng, chẳng hạn như một *manager* có nhiều *employee*. Trong ODL thì định nghĩa mối quan hệ một nhiều được mô tả như hình dưới đây.

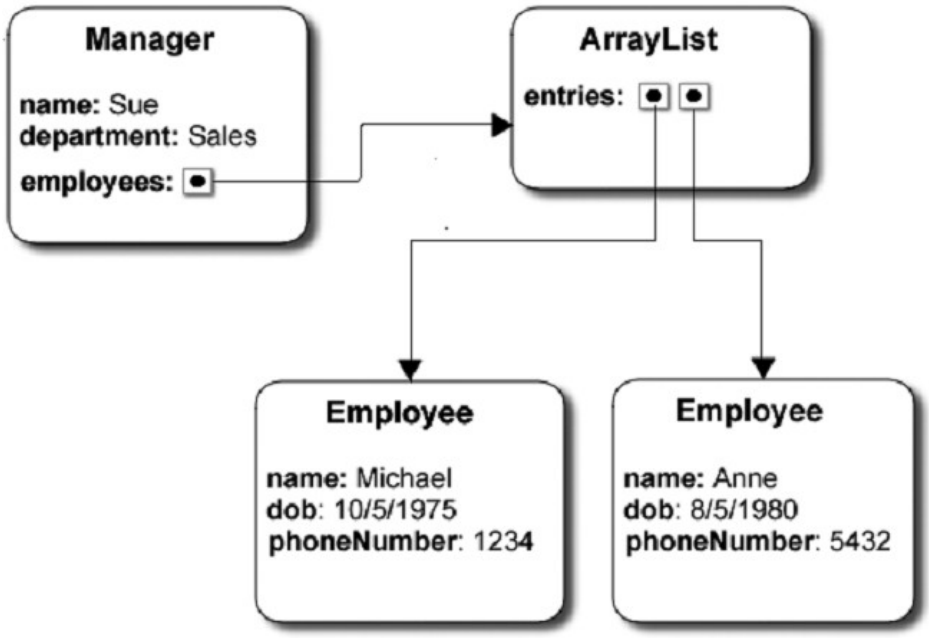
```
Class Manager{
  attribute string name;
  attribute string department;
  relationship set<Employee> employees
     inverse Employee :: manager;
}
class Employee
{
   attribute string name;
   attribute string dob;
   attribute string phoneNumber;
   relationship Manager manager
      inverse Manager :: employees;
}
```

Hình 4.10: Định nghĩa clas Manager và Employee trong ODL

Với db4o, những định nghĩa lớp trong lược đồ ODB có thể có hay không các mối quan hệ ngược lại.



Hình 4.11: Quan hệ One-to-many



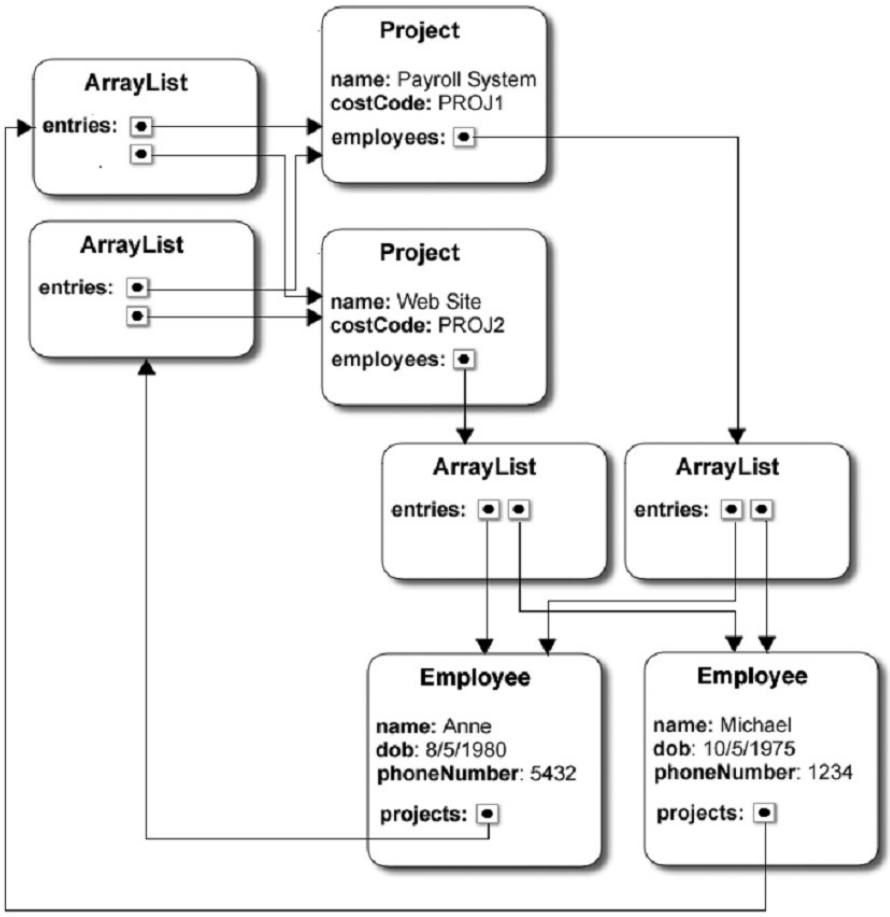
Hình 4.12: Object graph trong quan hệ one-to-many

Quan hệ *many-to-many* :Một mối quan hệ *many-to-many* được sinh ra nếu như *employee* có thể có nhiều *project* và ngược lại trong *project* có thể có nhiều *employee* khi đó quan hệ *many-to-many* được sinh ra. Kiểu mối quan hệ này chỉ khả dĩ trong một cơ sở dữ liệu quan hệ nếu như hai bảng *join* với nhau, và được thẳng thắn hơn trong cơ sở dữ liệu hướng đối tượng. Mỗi một đối tượng đơn đều được cất giữ bởi một *OID* để xác định cho các đối tượng liên quan. Trong OQL mối quan hệ *many-to-many* được mô tả như hình sau :

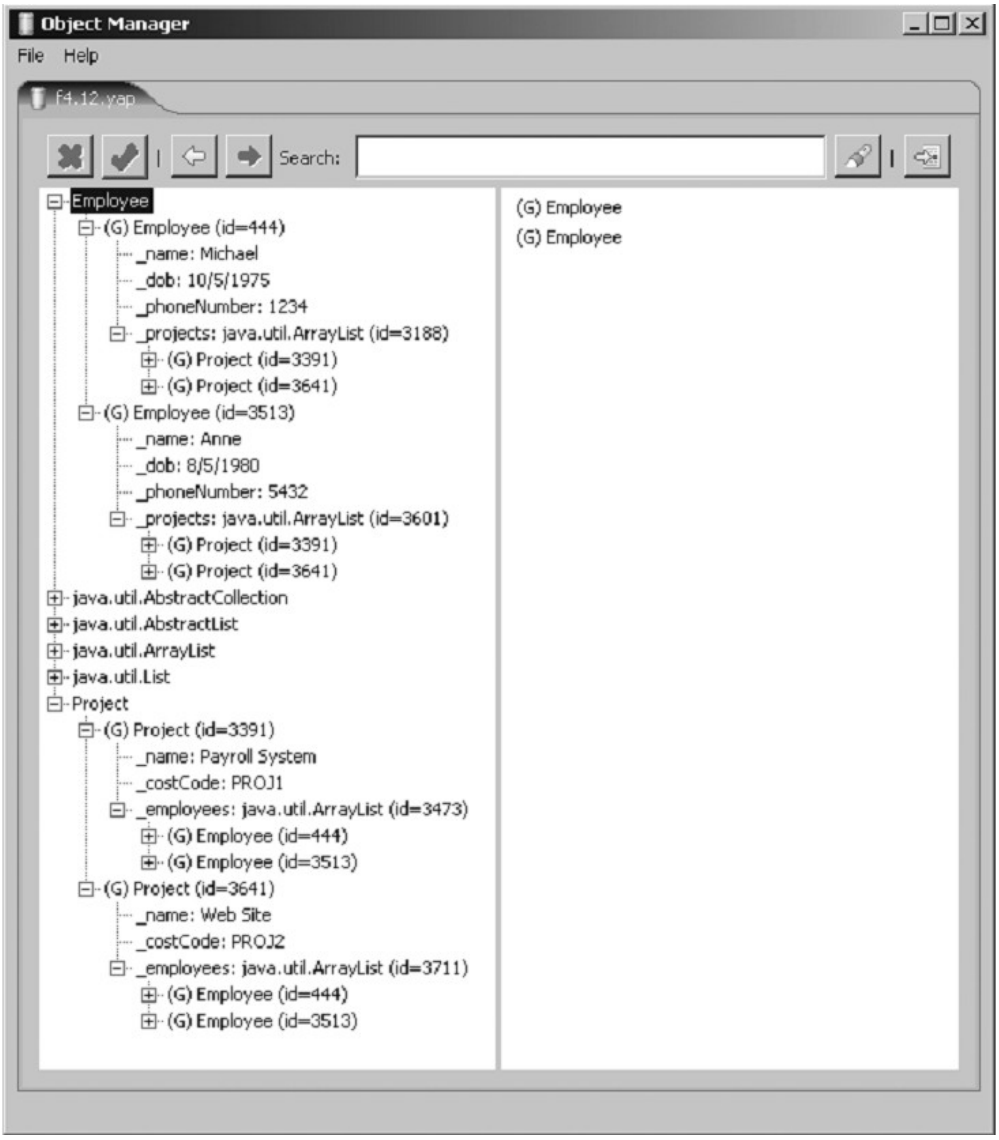
```
class Project
{
   attribute string name;
   attribute string costCode;
   relationship set<Employee> employees
      inverse Employee :: projects;
}
class Employee
{
   attribute string name;
   attribute string dob;
   attribute string phoneNumber;
   relationship set<Project> projects
      inverse Project :: employees;
}
```

Hình 4.13: Định nghĩa class Project và Employee trong ODL

Với db4o thì những định nghĩa lớp là thành phần cấu tạo lên lược đồ ODB. Trong trường hợp này mối quan hệ ngược lại trở nên rất hữu ích, nhưng đó là sự chính xác không cần thiết. Với ví dụ này nếu như ta cần tìm *employee* trong *project* và không một *project* nào có *employee* thì khi đó không cần đối tượng *employee* lưu trữ *OID* để tham chiếu tới đối tượng *project*. Vì vậy trong quan hệ *many-to-many* thì đối tượng *project* cần lưu trữ *OID* để tham chiếu tới đối tượng *employee* và đối tượng *employee* cũng vậy.



Hình 4.14: Minh họa object graph với quan hệ many-to-many



Hình 4.15: Quan hệ Many-to-many sử dụng mối quan hệ ngược

11. **Truy vấn dữ liệu đối tượng trực tiếp bằng tool hoặc dos**

Có thể truy vấn dữ liệu đối tượng trực tiếp bằng tool hoặc thông qua lệnh DOS

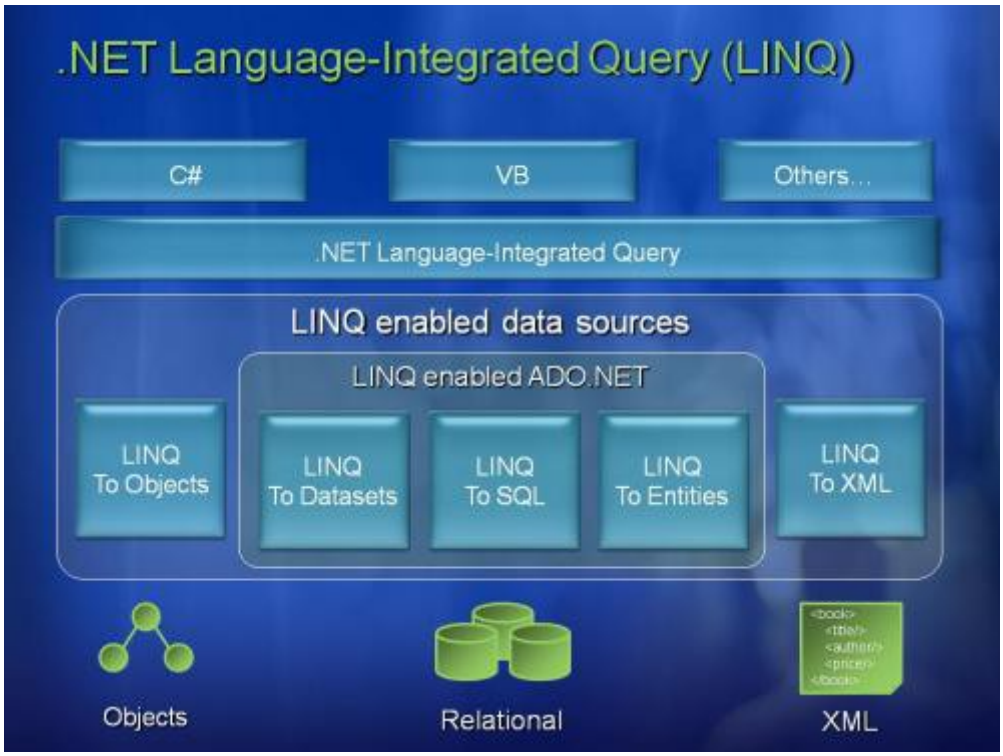
BÀI 5: TÍCH HỢP DỮ LIỆU HƯỚNG ĐỐI TƯỢNG TRÊN MÔI TRƯỜNG PHÁT TRIỂN ỨNG DỤNG .NET

1. Cài đặt tích hợp môi trường

Để giảm gánh nặng thao tác trên nhiều ngôn ngữ khác nhau và cải thiện năng suất lập trình, Microsoft đã phát triển giải pháp tích hợp dữ liệu cho .NET Framework có tên gọi là LINQ (Language Integrated Query), đây là thư viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic.NET (có thể mở rộng cho các ngôn ngữ khác) cung cấp khả năng truy vấn trực tiếp dữ liệu Object, cơ sở dữ liệu và XML.

Điểm mạnh của LINQ là "viết truy vấn cho rất nhiều các đối tượng dữ liệu", từ cơ sở dữ liệu, XML, Data Object … thậm chí là viết truy vấn cho một biến mảng đã tạo ra trước đó. Vì thế ta có các khái niệm như là LinQ to SQL, LinQ to XML,….

Các thành phần của LINQ



Hình 5.1: Kiến trúc của LINQ trong .NET Framework 3.5

LINQ to Objects

"LINQ to Objects" ở đây có nghĩa là nói đến cách sử dụng LINQ đối với các đối tượng Collection mà đã được thực thi giao diện IEnumerable hoặc IEnumerable<T> tức những Collection có thể "liệt kê" ra được. Đây là trường hợp sử dụng đơn giản nhất của LINQ khi làm việc với dữ liệu.

LINQ to SQL

LINQ to SQL là một phiên bản hiện thực hóa của O/RM (object relational mapping) có bên trong .NET Framework 3.5, nó cho phép bạn mô hình hóa một cơ sở dữ liệu dùng các lớp .NET. Sau đó bạn có thể truy vấn cơ sở dữ liệu dùng LINQ, cũng như cập nhật/thêm/xóa dữ liệu từ đó.

LINQ to SQL hỗ trợ đầy đủ transaction, view và các stored procedure (SP). Nó cũng cung cấp một cách dễ dàng để thêm khả năng kiểm tra tính hợp lệ của dữ liệu và các quy tắc vào trong mô hình dữ liệu của bạn.

LINQ to XML

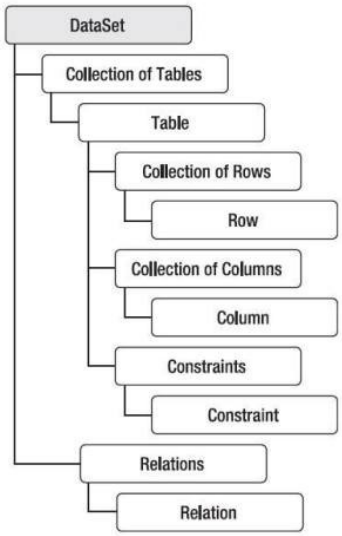
Sử dụng LINQ với mục đích truy vấn file XML và tiện lợi truy vấn hơn nhiều so với việc dùng XmlDocument, Xpath và Xquery như trước kia.

LINQ to Datasets

DataSet trong ADO.NET là một bước phát triển lớn trong việc phát triển ứng dụng cơ sở dữ liệu đa hệ. Khi lấy và chỉnh sửa dữ liệu, duy trì liên tục kết nối tới Data Source trong khi chờ user (người dùng) yêu cầu thì rõ ràng là tốn tài nguyên máy rất nhiều.

DataSet giúp ích ở đây rất lớn. Vì DataSet cho phép lưu trữ dữ liệu và chỉnh sửa tại 'local cache', hay gọi là offline mode. Có thể xem xét và xử lý thông tin trong khi ngắt kết nối. Sau khi chỉnh sửa và xem xong thì tạo một kết nối và update dữ liệu từ local vào Data Source.

Dữ liệu trong DataSet được lưu trữ dưới dạng một Collection các Tables và bạn cần phải xử lý thông qua các lớp DataTable (DataRow và DataColumn).



Hình 5.2: Kiến trúc DataSet

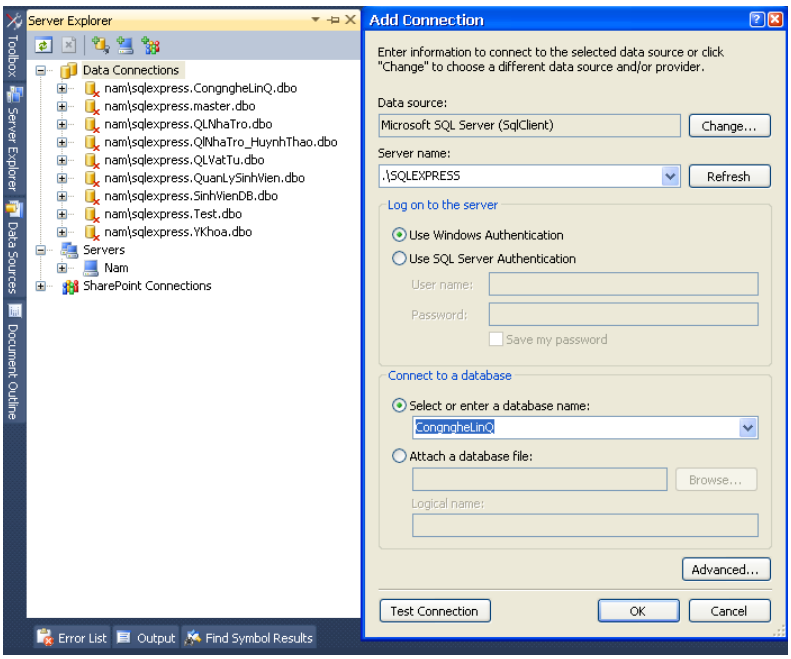
LINQ to DataSet cho phép người lập trình sử dụng DataSets như một nguồn dữ liệu bình thường bằng các cú pháp truy vấn căn bản của LINQ

2. **Cài đặt mô hình dữ liệu hướng đối tượng bằng ngôn ngữ .Net (Visual, C#)**

Kết nối đến CSDL

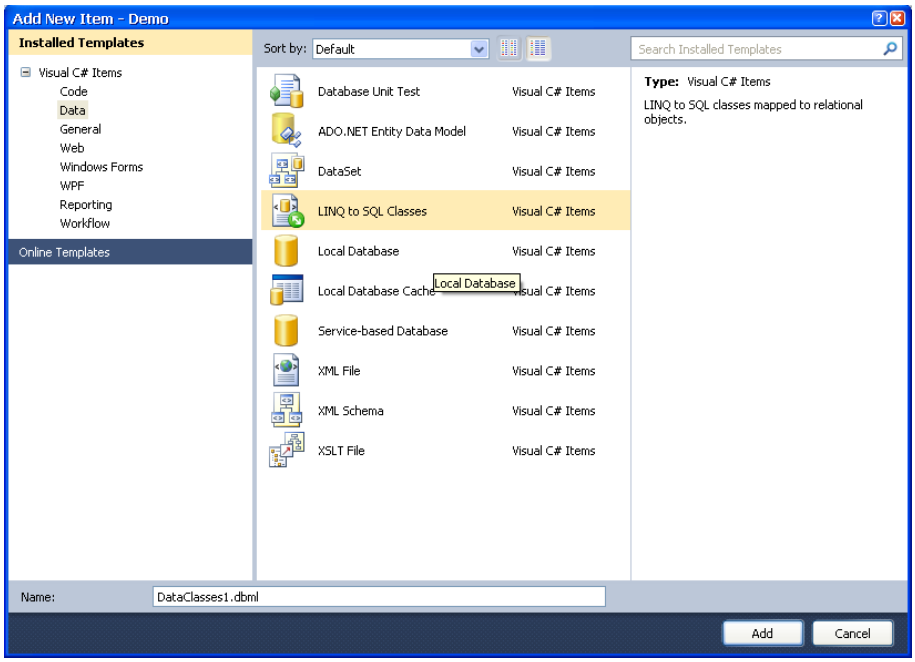
- View -> Server Explorer

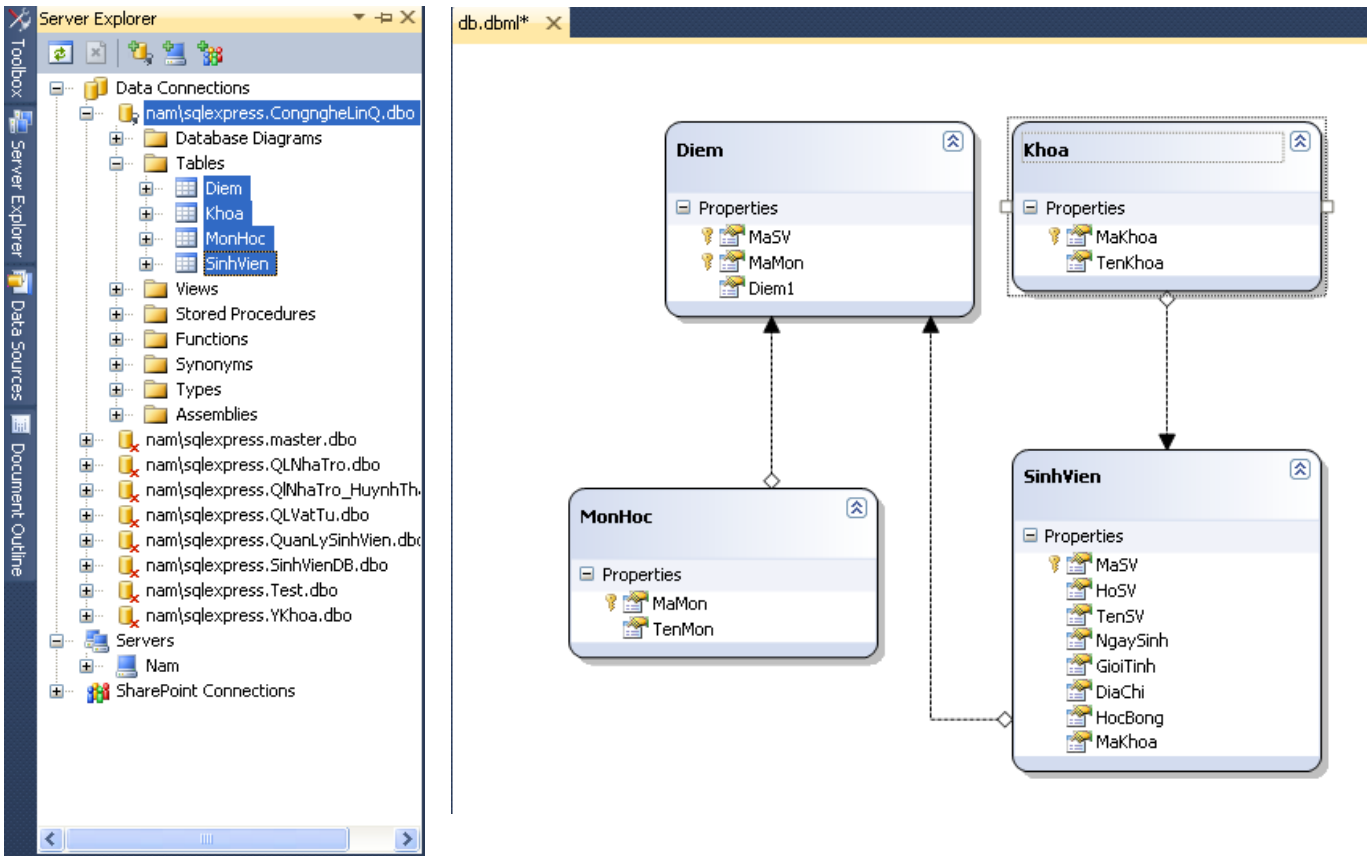
- Kích phải chuột lên DataConnection -> Add Connection



Hình 5.3: Kết nối đến cơ sở dữ liệu

Tạo ra mô hình dữ liệu LINQ TO SQL

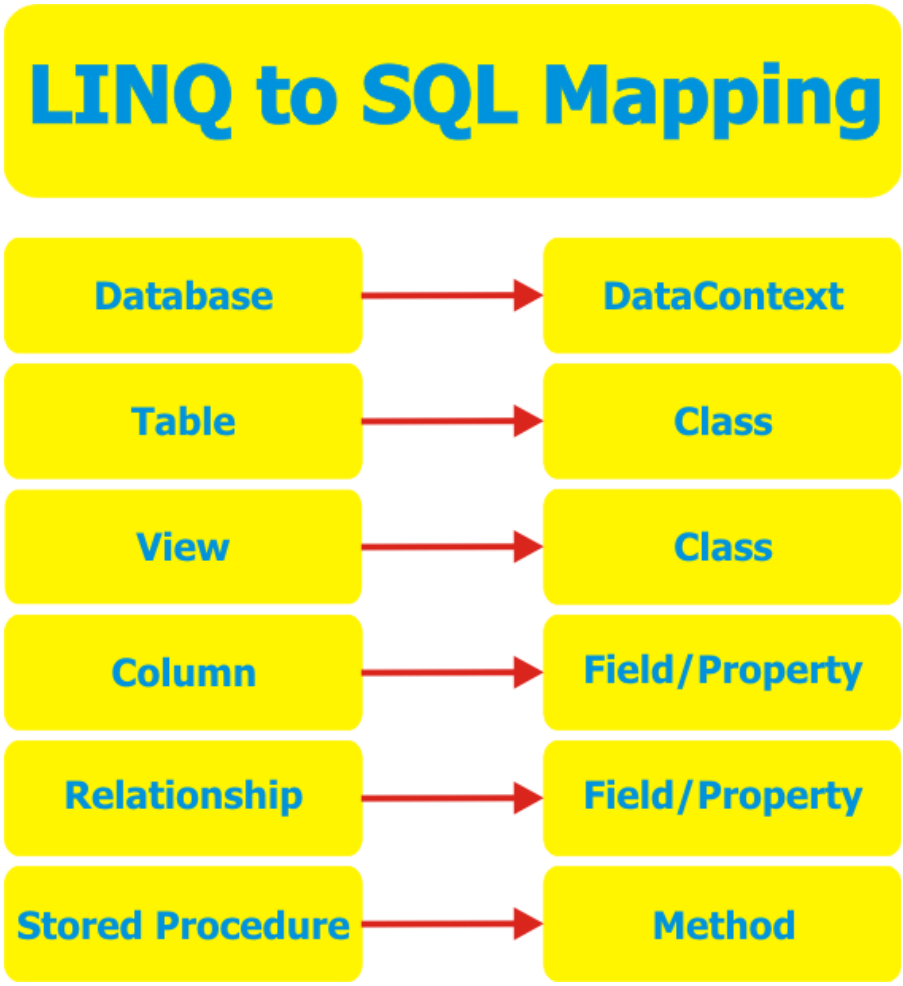




Hình 5.4: Tạo mô hình LinQ to SQL

3. **Biên dịch lược đồ (diagram)**

Mô hình ánh xạ

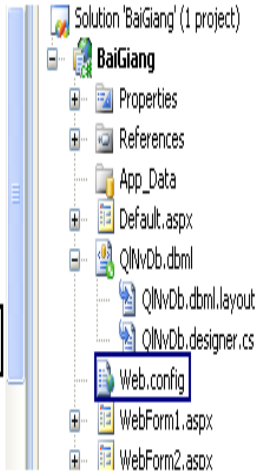


Hình 5.5: Mô hình ánh xạ LinQ to SQL

4. **Xây dựng ứng dụng cho phép truy xuất dữ liệu đối tượng**

Hiệu chỉnh đường kết nối trong file web.config:

```
<configuration>
  <configSections>...
    <appSettings/>
    <connectionStrings>
        <add name="qlnvConnectionString" connectionString=
        "Data Source=.\sqlexpress;Initial Catalog=qlnv;Integrated Security=True"
            providerName="System.Data.SqlClient" />
    </connectionStrings>
```



Các thành phần của lớp DataContext:

```
[System.Data.Linq.Mapping.DatabaseAttribute(Name="qlnv")]
public partial class QlNvDbDataContext : System.Data.Linq.DataContext
{
private static System.Data.Linq.Mapping.MappingSource mappingSource =
    new AttributeMappingSource();
#region Extensibility Method Definitions
partial void OnCreated();
partial void InsertDonVi(DonVi instance);
partial void UpdateDonVi(DonVi instance);
partial void DeleteDonVi(DonVi instance);
#endregion
public QlNvDbDataContext() :
  base(global::System.Configuration.ConfigurationManager.ConnectionStrings[
      "qlnvConnectionString"].ConnectionString, mappingSource)...
 public System.Data.Linq.Table<DonVi> DonVis
    {
        get
        {
            return this.GetTable<DonVi>();
        }
    }
```

```
[Table(Name="dbo.DonVi")]
public partial class DonVi : INotifyPropertyChanging, INotifyPropertyChanged
{   private static PropertyChangingEventArgs emptyChangingEventArgs =
    new PropertyChangingEventArgs(String.Empty);
    private string _MaDonVi;
    private string _TenDonVi;
#region Extensibility Method Definitions
partial void OnLoaded();
partial void OnValidate(System.Data.Linq.ChangeAction action);
partial void OnCreated();
partial void OnMaDonViChanging(string value);
partial void OnMaDonViChanged();
partial void OnTenDonViChanging(string value);
partial void OnTenDonViChanged();
#endregion
public DonVi()
    {OnCreated();    }
[Column(Storage="_MaDonVi", DbType="NVarChar(50) NOT NULL", CanBeNull=false,
    IsPrimaryKey=true)]
```

```
public string MaDonVi
    {get{return this._MaDonVi;  }
    set
        {
            if ((this._MaDonVi != value))
            {
                this.OnMaDonViChanging(value);
                this.SendPropertyChanging();
                this._MaDonVi = value;
                this.SendPropertyChanged("MaDonVi");
                this.OnMaDonViChanged();
            }
        }
    }
    [Column(Storage="_TenDonVi", DbType="NVarChar(50)")]
    public string TenDonVi
    {
        get
        {
```

```
[Function(Name="dbo.TimNv")]
public ISingleResult<TimNvResult> TimNv([Parameter(DbType="NVarChar(50)")] string tendv)
{
    IExecuteResult result = this.ExecuteMethodCall(this,
        ((MethodInfo)(MethodInfo.GetCurrentMethod())), tendv);
    return ((ISingleResult<TimNvResult>)(result.ReturnValue));
}
```

Các truy vấn:

```
QlNvDbDataContext db = new QlNvDbDataContext();
var q =
        from p in db.NhanViens
        where p.MaDonVi == "dv1"
        select new { p.MaNv, p.HeSoLuong };
```

```
var q=
    db.NhanViens
    .Where(p=>p.MaDonVi =="dv1")
    .Select(p=>new {p.MaNv,p.HeSoLuong } );
```