HQT CSDL hướng đối tượng (OODBMS: Object-Oriented DataBase Management System)

Phạm Thị Ngọc Diễm
ptndiem@cit.ctu.edu.vn

Bộ môn HTTT
Khoa CNTT&TT - Đại học Cần Thơ

Tháng 12 năm 2012

Nội dung

Nội dung

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)
- Sản xuất nhờ máy tính (CAM: Computer-Aided Manufactoring)

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)
- Sản xuất nhờ máy tính (CAM: Computer-Aided Manufactoring)
- Công nghệ phần mềm

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)
- Sản xuất nhờ máy tính (CAM: Computer-Aided Manufactoring)
- Công nghệ phần mềm
- Hệ thống đa phương tiện (Multimedia system)

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)
- Sản xuất nhờ máy tính (CAM: Computer-Aided Manufactoring)
- Công nghệ phần mềm
- Hệ thống đa phương tiện (Multimedia system)
- Hệ thống thông tin địa lý (GIS)

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)
- Sản xuất nhờ máy tính (CAM: Computer-Aided Manufactoring)
- Công nghệ phần mềm
- Hệ thống đa phương tiện (Multimedia system)
- Hệ thống thông tin địa lý (GIS)
- Tìm kiếm và tích hợp dữ liệu của các Website.

Sự phát triển của các ứng dụng I

Những ứng dụng mới

- Thiết kế nhờ máy tính (CAD: Computer-Aided Design)
- Sản xuất nhờ máy tính (CAM: Computer-Aided Manufactoring)
- Công nghệ phần mềm
- Hệ thống đa phương tiện (Multimedia system)
- Hệ thống thông tin địa lý (GIS)
- Tìm kiếm và tích hợp dữ liệu của các Website.
- Etc.

Sự phát triển của các ứng dụng II

=> Những nhu cầu mới

- Những đối tượng có cấu trúc ngày càng phức tạp,

Sự phát triển của các ứng dụng II

=> Những nhu cầu mới

- Những đối tượng có cấu trúc ngày càng phức tạp,
- Những kiểu dữ liệu mới,

Sự phát triển của các ứng dụng II

=> Những nhu cầu mới

- Những đối tượng có cấu trúc ngày càng phức tạp,
- Những kiểu dữ liệu mới,
- Những giao dịch kéo dài,

Sự phát triển của các ứng dụng II

=> Những nhu cầu mới

- Những đối tượng có cấu trúc ngày càng phức tạp,
- Những kiểu dữ liệu mới,
- Những giao dịch kéo dài,
- Việc phát triển của các hệ thống thông tin không thỏa mãn nhu cầu người dùng

Sự phát triển của các ứng dụng II

=> Những nhu cầu mới

- Những đối tượng có cấu trúc ngày càng phức tạp,
- Những kiểu dữ liệu mới,
- Những giao dịch kéo dài,
- Việc phát triển của các hệ thống thông tin không thỏa mãn nhu cầu người dùng
- Etc.

Sự phát triển của HQT CSDL

- 1960s: HQT CSDL phân cấp (IMS), HQT CSDL mạng (CODASYL)

Sự phát triển của HQT CSDL

- 1960s: HQT CSDL phân cấp (IMS), HQT CSDL mạng (CODASYL)
- 1970s: HQT CSDL quan hệ
  - Mô hình đơn giản
  - Toàn vẹn dữ liệu
  - An toàn dữ liệu
  - Tính nguyên tố của các giao dịch
  - Độ tin cậy của dữ liệu
  - Etc.

Sự phát triển của HQT CSDL

- 1960s: HQT CSDL phân cấp (IMS), HQT CSDL mạng (CODASYL)
- 1970s: HQT CSDL quan hệ
  - Mô hình đơn giản
  - Toàn vẹn dữ liệu
  - An toàn dữ liệu
  - Tính nguyên tố của các giao dịch
  - Độ tin cậy của dữ liệu
  - Etc.
- 1986 : Sự xuất hiện HQT CSDL hướng đối tượng

Sự phát triển của HQT CSDL

- 1960s: HQT CSDL phân cấp (IMS), HQT CSDL mạng (CODASYL)
- 1970s: HQT CSDL quan hệ
  - Mô hình đơn giản
  - Toàn vẹn dữ liệu
  - An toàn dữ liệu
  - Tính nguyên tố của các giao dịch
  - Độ tin cậy của dữ liệu
  - Etc.
- 1986 : Sự xuất hiện HQT CSDL hướng đối tượng
- 1993 : Sự ra đời của chuẩn ODMG cho HQT CSDL HĐT

Sự phát triển của HQT CSDL

- 1960s: HQT CSDL phân cấp (IMS), HQT CSDL mạng (CODASYL)
- 1970s: HQT CSDL quan hệ
  - Mô hình đơn giản
  - Toàn vẹn dữ liệu
  - An toàn dữ liệu
  - Tính nguyên tố của các giao dịch
  - Độ tin cậy của dữ liệu
  - Etc.
- 1986 : Sự xuất hiện HQT CSDL hướng đối tượng
- 1993 : Sự ra đời của chuẩn ODMG cho HQT CSDL HĐT
- 1998 : Chuẩn UML cho thiết kế những ứng dụng HĐT

Sự phát triển của HQT CSDL

- 1960s: HQT CSDL phân cấp (IMS), HQT CSDL mạng (CODASYL)
- 1970s: HQT CSDL quan hệ
  - Mô hình đơn giản
  - Toàn vẹn dữ liệu
  - An toàn dữ liệu
  - Tính nguyên tố của các giao dịch
  - Độ tin cậy của dữ liệu
  - Etc.
- 1986 : Sự xuất hiện HQT CSDL hướng đối tượng
- 1993 : Sự ra đời của chuẩn ODMG cho HQT CSDL HĐT
- 1998 : Chuẩn UML cho thiết kế những ứng dụng HĐT
- 1999 : Chuẩn SQL3 cho HTQ CSDL quan hệ - đối tượng

Nội dung

Ưu điểm của HQT CSDL quan hệ

- Mô hình dữ liệu đơn giản và dễ hiểu đối với người sử dụng,

Ưu điểm của HQT CSDL quan hệ

- Mô hình dữ liệu đơn giản và dễ hiểu đối với người sử dụng,
- Mô hình dựa trên cơ sở lý thuyết vững chắc, nó cho phép định nghĩa các phương pháp thiết kế các sơ đồ (schema)(lý thuyết chuẩn hóa) và ngôn ngữ SQL (một ngôn ngữ khai báo đã trở thành ngôn ngữ chuẩn),

Ưu điểm của HQT CSDL quan hệ

- Mô hình dữ liệu đơn giản và dễ hiểu đối với người sử dụng,
- Mô hình dựa trên cơ sở lý thuyết vững chắc, nó cho phép định nghĩa các phương pháp thiết kế các sơ đồ (schema)(lý thuyết chuẩn hóa) và ngôn ngữ SQL (một ngôn ngữ khai báo đã trở thành ngôn ngữ chuẩn),
- Phù hợp với những ứng dụng quản lý cổ điển,

Ưu điểm của HQT CSDL quan hệ

- Mô hình dữ liệu đơn giản và dễ hiểu đối với người sử dụng,
- Mô hình dựa trên cơ sở lý thuyết vững chắc, nó cho phép định nghĩa các phương pháp thiết kế các sơ đồ (schema)(lý thuyết chuẩn hóa) và ngôn ngữ SQL (một ngôn ngữ khai báo đã trở thành ngôn ngữ chuẩn),
- Phù hợp với những ứng dụng quản lý cổ điển,
- Etc.

Nhược điểm của HQT CSDL quan hệ I

- Biểu diễn các thực thể của thế giới thực không phong phú
- Dữ liệu và xử lý tách rời nhau
- Quá tải về ngữ nghĩa: quan hệ (relation) vừa có thể được tạo ra từ thực thể (entity) , vừa có thể được tạo ra từ liên kết (relationship) trong mô hình thực thể- liên kết (ERD: Entity-Relationshipc Diagram).
- Ít hỗ trợ cho các ràng buộc toàn vẹn và ràng buộc của xí nghiệp: Các trình ứng dụng phải tự bảo đảm lấy, nhờ vào chuẩn cho SQL.

Nhược điểm của HQT CSDL quan hệ II

- Cấu trúc dữ liệu đồng nhất: theo chiều dọc và chiều ngang:
  - Các bộ có các thuộc tính giống nhau
  - Các giá trị trong cùng thuộc tính phải cùng miền giá trị
  - Giao tại một dòng và một cột phải là một trị nguyên tố (không chia nhỏ ra được nữ)
  - Nhiều HQT CSDL cho phép lưu trữ và hiển thị BLOB (Binary Large Object: chứa thông tin nhị phân, biểu diễn ảnh, phim, một đối tượng bất kỳ không cấu trúc) nhưng không cho sử dụng hoặc hiển thị một phần của nó. BLOB không có tính đệ qui và không biểu diễn được khía cạnh hành xử (behaviour aspect) của đối tượng.
    => quá hạn chế đối với các đối tượng trong thế giới thực, vốn có cấu trúc phức.

Nhược điểm của HQT CSDL quan hệ III

- Các thao tác (operation) bị hạn chế:
  - Chỉ có các thao tác trên tập hợp và trên các bộ
  - Không cho phép thêm mới phép toán
- Khó kiểm soát các câu hỏi đệ qui:
  - Không thể chỉ dùng SQL để thực hiện các câu hỏi đệ qui
  - Chính là nhờ ngôn ngữ lập trình trong trình ứng dụng hỗ trợ câu hỏi đệ qui
- Không tương ứng giữa ngôn ngữ SQL và ngôn ngữ lập trình:
  - SQL: ngôn ngữ khai báo, sử dụng cùng lúc nhiều dòng dữ liệu
  - Ngôn ngữ lập trình: ngôn ngữ thủ tục, chỉ có thể sử dụng 1 dòng dữ liệu/ thời điểm
  - Hai loại ngôn ngữ này có cách biểu diễn dữ liệu khác nhau => trình ứng dụng phải chuyển đổi qua lại 2 cách biểu diễn này => không hiệu quả.

Nhược điểm của HQT CSDL quan hệ IV

- Các vấn đề khác về tính cạnh tranh, sự thay đổi các sơ đồ và truy cập nghèo nàn trong RDBMS
  a) Kiểm soát cạnh tranh không còn hợp thời với các giao dịch kéo dài trên các đối tượng phức hợp.
  b) Khó thay đổi các sơ đồ :người quản trị CSDL phải can thiệp vào để thay đổi cấu trúc CSDL, và trình ứng dụng phải đổi để thích nghi với các cấu trúc mới
  => chậm, làm cản trở công nghệ mới
  => hầu hết các HTTT xí nghiệp đành tự trói mình với cấu trúc CSDL đã có

Nhược điểm của HQT CSDL quan hệ V

c) Truy cập nghèo nàn: HQT CSDL quan hệ dùng các truy cập kết hợp (associative access: dựa trên sự kết hợp các tân từ để chọn) mà ít dùng các truy cập thông thương (navigational access: dựa trên sự chuyển động giữa các bộ)
=> Không sử dụng con trỏ (pointer).)

Chú ý

Các mục a) và b) xảy ra không chỉ trên các HQT CSDL quan hệ.

Nội dung

Tiếp cận hướng đối tượng I

- *Đề nghị một tập các phương pháp và các công cụ cho phép thiết kế và thực hiện các phần mềm có cấu trúc và có thể được sử dụng lại bởi sự hợp thành của các thành phần độc lập nhau.*
- **Mục tiêu**: Tăng hiệu suất làm việc của các lập trình viên bằng cách cho phép tạo ra các phần mềm có cấu trúc, có thể mở rộng, có thể tái sử dụng và bảo trì dễ dàng.
- Các khái niệm chính: đối tượng, đóng gói, kế thừa, etc.
- Ngôn ngữ lập trình HĐT: C++, Java, etc.
  => Các CSDL HĐT ra đời đáp ứng những *nhu cầu mới* và phát sinh từ sự kết hợp của hai lĩnh vực: CSDL và ngôn ngữ lập trình HĐT.

Tiếp cận hướng đối tượng II

HQT CSDL HĐT = CSDL + Ngôn ngữ lập trình HĐT

HQT CSDL

- Biểu diễn thế giới thực
- Toàn vẹn dữ liệu
- Quản lý đĩa
- Chia sẻ dữ liệu
- Độ tin cậy của dữ liệu
- An toàn dữ liệu
- Ngôn ngữ hỏi
- Độc lập logic/vật lý

NN lập trình HĐT

- Phát triển
- Cấu trúc phức tạp
- Định danh đối tượng
- Đóng gói
- Lớp
- Kế thừa
- Định nghĩa lại
- Thư viện lớp

Nội dung

Định nghĩa I

Mô hình dữ liệu HĐT

Mô hình dữ liệu HĐT (OODM: Object-Oriented Data Model) là một mô hình luận lý của dữ liệu để biểu diễn ngữ nghĩa của các dữ liệu dùng trong một ngôn ngữ lập trình hướng đối tượng.

CSDL HĐT

CSDL HĐT(OODB : Object-Oriented Database) là một CSDL trong đó các thành phần dữ liệu là các đối tượng bền vững và chia sẻ được, định nghĩa bởi một OODM.

Định nghĩa II

HQT CSDL HĐT

HQT CSDL HĐT (OODBMS: Object-Oriented DataBase Management System) là một hệ thống hỗ trợ và cung cấp các công cụ cho việc thiết kế và môi trường thực thi của một OODB.

- Không có mô hình dữ liệu hướng đối tượng nào tương đương với mô hình dữ liệu nền của hệ CSDL quan hệ.

- Mỗi hệ thống cung cấp định nghĩa riêng về OODBMS.

Ví dụ Hệ CSDL HĐT

OODBMS	Nhà sản xuất
Gemstone	Gemstone System Inc.
Objectivity/DB	Objectivity Inc.
Object Store	Progress Software Corporation
FastObjects by Poet	Poet Software Corporation
Jasmine Object Database	Computer Associates/ Fujitshu Limited
Versant	Versant Corporation
Etc.	

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;
- Dữ liệu và các xử lý không còn tách biệt nhau, các phương thức (method) là một thành phần khai báo của các lớp đối tượng;

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;
- Dữ liệu và các xử lý không còn tách biệt nhau, các phương thức (method) là một thành phần khai báo của các lớp đối tượng;
- Tính kế thừa;

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;
- Dữ liệu và các xử lý không còn tách biệt nhau, các phương thức (method) là một thành phần khai báo của các lớp đối tượng;
- Tính kế thừa;
- Các đối tượng đều có một định danh để phân biệt nó với bất kỳ đối tượng nào khác, ngay cả khi chúng có cùng giá trị;

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;
- Dữ liệu và các xử lý không còn tách biệt nhau, các phương thức (method) là một thành phần khai báo của các lớp đối tượng;
- Tính kế thừa;
- Các đối tượng đều có một định danh để phân biệt nó với bất kỳ đối tượng nào khác, ngay cả khi chúng có cùng giá trị;
- Tương thích giữa các ngôn ngữ lập trình và ngôn ngữ thao tác dữ liệu (DML).

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;
- Dữ liệu và các xử lý không còn tách biệt nhau, các phương thức (method) là một thành phần khai báo của các lớp đối tượng;
- Tính kế thừa;
- Các đối tượng đều có một định danh để phân biệt nó với bất kỳ đối tượng nào khác, ngay cả khi chúng có cùng giá trị;
- Tương thích giữa các ngôn ngữ lập trình và ngôn ngữ thao tác dữ liệu (DML).
- Tập các kiểu dữ liệu phải mở rộng được;

Các đặc trưng của Hệ CSDL HĐT I

Các hệ CSDL HĐT được đặc trưng bởi các điểm chính:

- Biểu diễn các cấu trúc dữ liệu phức tạp;
- Dữ liệu và các xử lý không còn tách biệt nhau, các phương thức (method) là một thành phần khai báo của các lớp đối tượng;
- Tính kế thừa;
- Các đối tượng đều có một định danh để phân biệt nó với bất kỳ đối tượng nào khác, ngay cả khi chúng có cùng giá trị;
- Tương thích giữa các ngôn ngữ lập trình và ngôn ngữ thao tác dữ liệu (DML).
- Tập các kiểu dữ liệu phải mở rộng được;
- Gắn kết động (dynamic binding).

Các đặc trưng của Hệ CSDL HĐT II

Ngoài ra các hệ CSDL HĐT cũng phải thỏa mãn các yêu cầu chung cho tất cả các HQT CSDL:

- Tính bền vững dữ liệu;

Các đặc trưng của Hệ CSDL HĐT II

Ngoài ra các hệ CSDL HĐT cũng phải thỏa mãn các yêu cầu chung cho tất cả các HQT CSDL:

- Tính bền vững dữ liệu;
- Kiểm soát được CSDL rất lớn;

Các đặc trưng của Hệ CSDL HĐT II

Ngoài ra các hệ CSDL HĐT cũng phải thỏa mãn các yêu cầu chung cho tất cả các HQT CSDL:

- Tính bền vững dữ liệu;
- Kiểm soát được CSDL rất lớn;
- Hỗ trợ tính cạnh tranh;

Các đặc trưng của Hệ CSDL HĐT II

Ngoài ra các hệ CSDL HĐT cũng phải thỏa mãn các yêu cầu chung cho tất cả các HQT CSDL:

- Tính bền vững dữ liệu;
- Kiểm soát được CSDL rất lớn;
- Hỗ trợ tính cạnh tranh;
- Phục hồi CSDL;

Các đặc trưng của Hệ CSDL HĐT II

Ngoài ra các hệ CSDL HĐT cũng phải thỏa mãn các yêu cầu chung cho tất cả các HQT CSDL:

- Tính bền vững dữ liệu;
- Kiểm soát được CSDL rất lớn;
- Hỗ trợ tính cạnh tranh;
- Phục hồi CSDL;
- Truy vấn dữ liệu đơn giản, etc.

Nội dung

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng
- Tương thích với ngôn ngữ lập trình HĐT

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng
- Tương thích với ngôn ngữ lập trình HĐT
- Hỗ trợ các giao dịch kéo dài

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng
- Tương thích với ngôn ngữ lập trình HĐT
- Hỗ trợ các giao dịch kéo dài
- Có khả năng áp dụng các trình ứng dụng chuyên sâu về CSDL

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng
- Tương thích với ngôn ngữ lập trình HĐT
- Hỗ trợ các giao dịch kéo dài
- Có khả năng áp dụng các trình ứng dụng chuyên sâu về CSDL
- Cải tiến hiệu suất

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng
- Tương thích với ngôn ngữ lập trình HĐT
- Hỗ trợ các giao dịch kéo dài
- Có khả năng áp dụng các trình ứng dụng chuyên sâu về CSDL
- Cải tiến hiệu suất
- Sử dụng tham chiếu

Ưu điểm của OODBMS

- Khả năng mô hình hoá phong phú với nhiều kiểu dữ liệu có cấu trúc phức tạp và các kiểu dữ liệu mới (hình ảnh, âm thanh, etc.)
- Có khả năng mở rộng
- Tương thích với ngôn ngữ lập trình HĐT
- Hỗ trợ các giao dịch kéo dài
- Có khả năng áp dụng các trình ứng dụng chuyên sâu về CSDL
- Cải tiến hiệu suất
- Sử dụng tham chiếu
- Ngôn ngữ hỏi có tính biểu hiện cao hơn

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu
- Thiếu mô hình dữ liệu chung

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu
- Thiếu mô hình dữ liệu chung
- Thiếu chuẩn chung

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu
- Thiếu mô hình dữ liệu chung
- Thiếu chuẩn chung
- Hiệu suất bị giới hạn

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu
- Thiếu mô hình dữ liệu chung
- Thiếu chuẩn chung
- Hiệu suất bị giới hạn
- Phức tạp

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu
- Thiếu mô hình dữ liệu chung
- Thiếu chuẩn chung
- Hiệu suất bị giới hạn
- Phức tạp
- Thiếu hỗ trợ về khung nhìn

Nhược điểm của OODBMS

- Thiếu cơ sở lý thuyết cho mô hình dữ liệu
- Thiếu mô hình dữ liệu chung
- Thiếu chuẩn chung
- Hiệu suất bị giới hạn
- Phức tạp
- Thiếu hỗ trợ về khung nhìn
- Thiếu hỗ trợ về an toàn bảo mật

Nội dung

Chuẩn ODMG I

CSDL quan hệ

- Cơ sở lý thuyết
- => Các HQT CSDL quan hệ ra đời

CSDL HĐT

- Sự ra đời của các HQT CSDL HĐT
- => Chuẩn ODMG

=> ODMG được chấp nhận nhiều nhất, nhóm được nhiều nhà cung cấp HQT CSDL HĐT (Poet, Objectivity, Versant, GemStone, etc.), nhiều người sử dụng và các nhà nghiên cứu vì nó :
- Cho phép đặc tả mô hình chuẩn về ngữ nghĩa cho các đối tượng CSDL và ;
- Hỗ trợ khả năng tương tác (interoperability) giữa các OODBMS.

Chuẩn ODMG II

OMDG (1993)

- Là chuẩn cho các HQT CSDL "thuần" HĐT.

Chuẩn ODMG II

OMDG (1993)

- Là chuẩn cho các HQT CSDL "thuần" HĐT.
- Mô tả mô hình đối tượng (object model) và đặc tả của một sơ đồ (schema) trên mô hình đối tượng,

Chuẩn ODMG II

OMDG (1993)

- Là chuẩn cho các HQT CSDL "thuần" HĐT.
- Mô tả mô hình đối tượng (object model) và đặc tả của một sơ đồ (schema) trên mô hình đối tượng,
- Đề nghị ngôn ngữ định nghĩa (ODL) và truy vấn đối tượng (OQL),

Chuẩn ODMG II

OMDG (1993)

- Là chuẩn cho các HQT CSDL "thuần" HĐT.
- Mô tả mô hình đối tượng (object model) và đặc tả của một sơ đồ (schema) trên mô hình đối tượng,
- Đề nghị ngôn ngữ định nghĩa (ODL) và truy vấn đối tượng (OQL),
- Cung cấp các API để định nghĩa và thao tác trên các đối tượng bằng C++, Smalltalk, và Java,

Chuẩn ODMG II

OMDG (1993)

- Là chuẩn cho các HQT CSDL "thuần" HĐT.
- Mô tả mô hình đối tượng (object model) và đặc tả của một sơ đồ (schema) trên mô hình đối tượng,
- Đề nghị ngôn ngữ định nghĩa (ODL) và truy vấn đối tượng (OQL),
- Cung cấp các API để định nghĩa và thao tác trên các đối tượng bằng C++, Smalltalk, và Java,
- Ánh xạ từ một thiết kế mức quan niệm sang một thiết kế mức logic theo hướng đối tượng.

Chuẩn SQL 3

SQL 3 (1999)

- Do tổ chức ISO đề nghị,

Chuẩn SQL 3

SQL 3 (1999)

- Do tổ chức ISO đề nghị,
- Mở rộng SQL cho CSDL quan hệ với các tính năng HĐT và các tính năng khác như: đa phương tiên, không gian và thời gian, truy vấn đệ quy, etc.

Chuẩn SQL 3

SQL 3 (1999)

- Do tổ chức ISO đề nghị,
- Mở rộng SQL cho CSDL quan hệ với các tính năng HĐT và các tính năng khác như: đa phương tiên, không gian và thời gian, truy vấn đệ quy, etc.
- Tương thích với SQL 2 (1992) (các CSDL quan hệ có thể được sinh ra bằng cách sử dụng SQL 3).

Chuẩn SQL 3

SQL 3 (1999)

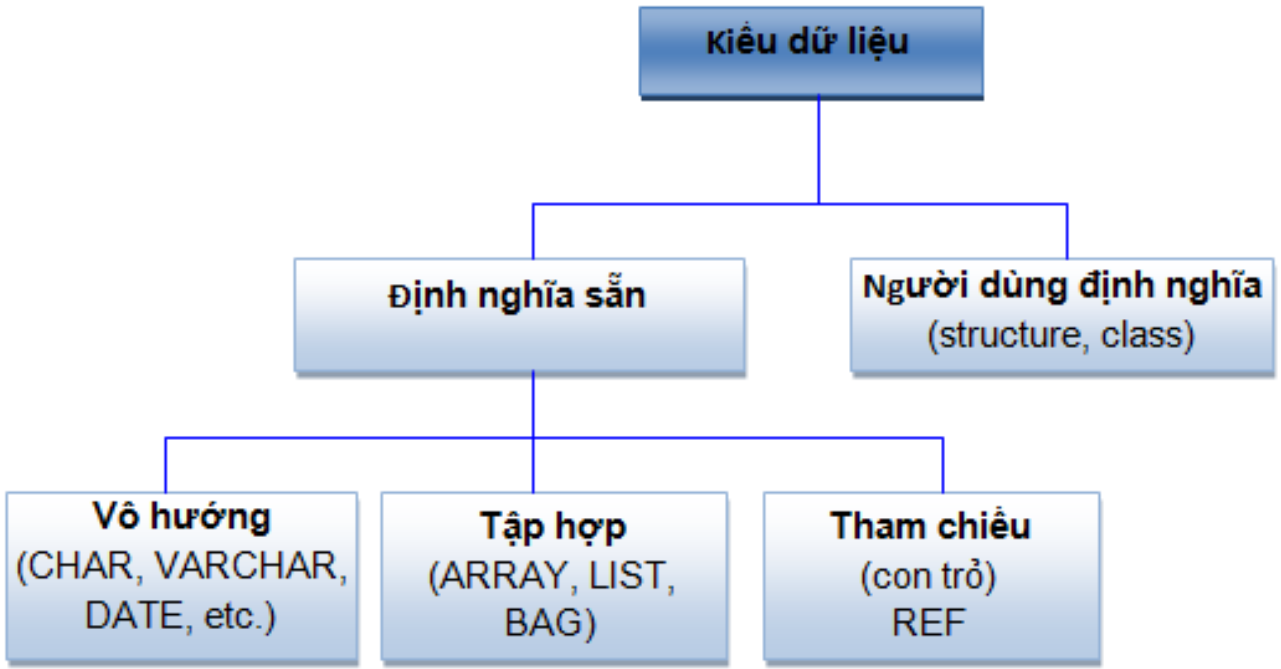
- Do tổ chức ISO đề nghị,
- Mở rộng SQL cho CSDL quan hệ với các tính năng HĐT và các tính năng khác như: đa phương tiên, không gian và thời gian, truy vấn đệ quy, etc.
- Tương thích với SQL 2 (1992) (các CSDL quan hệ có thể được sinh ra bằng cách sử dụng SQL 3).
- HQT CSDL quan hệ của các công ty lớn như Oracle, IBM, Sybase hỗ trợ SQL 3.

Nội dung

Các khái niệm chính

Thế giới thực	**OODB**
Đối tượng	Đối tượng Lớp đối tượng
Tính chất	Thuộc tính Phương thức
Quan hệ	Quan hệ cấu thành (composition) Tổng quát hóa/chuyên biệt hóa Kế thừa

Các kiểu dữ liệu



Đối tượng có cấu trúc phức tạp I

Mục tiêu: *Biểu diễn trực tiếp đối tượng của thế giới thực.*

Ví dụ:

Thế giới thực : **NGƯỜI**

- Họ
- Tên
- Địa chỉ (số, đường, tỉnh/TP)
- Con (Tên, phái, ngày sinh)

CSDL quan hệ:

NGUOI(idN, ho, ten, so, duong,tinh-TP)

CON(idC, ten, phai, ngaysinh, *idN*)

Đối tượng có cấu trúc phức tạp II

CSDL HĐT: Chỉ một đối tượng

```
CLASS Nguoi
    { ATTRIBUTE ho : STRING ;
    ATTRIBUTE ten : STRING ;
    ATTRIBUTE Diachi : STRUCT adr
          { duong : STRING ;
           so : STRING ;
           tinhTP : STRING }
    ATTRIBUTE CacCon : LIST STRUCT Con
          { ten : STRING ;
           phai: ENUM {'M', 'F'} ;
           ngaysinh : DATE }
}
```

Đối tượng có cấu trúc phức tạp III

Thuộc tính có thể có cấu trúc phức hoặc đa trị.

Sử dụng hàm tạo (constructor) :

- Cấu trúc phức : **STRUCT**
- Đa trị : Hàm tạo tập hợp :
  1. Tập hợp : **SET**
  2. Tập hợp với phần tử trùng : **BAG**
  3. Mảng một chiều : **ARRAY**
  4. Danh sách : **LIST**
  5. Tự điển: **DICTIONARY**

Đối tượng có cấu trúc phức tạp IV

Các kiểu được định nghĩa bởi ứng dụng

- Các hàm tạo cấu trúc phức tạp được sử dụng để:
  - Định nghĩa các lớp đối tượng : CLASS có cấu trúc phức tạp
  - Định nghĩa các kiểu dữ liệu: TYPEDEF thích hợp với ứng dụng
- Tương tự như lớp các đối tượng, các kiểu dữ liệu được định nghĩa bởi ứng dụng cũng có:
  - Cấu trúc phức tạp
  - Các phương thức (method)

Đối tượng có cấu trúc phức tạp V

Ví dụ: Các kiểu dữ liệu (1)

TYPEDEF **T_Diachi** STRUCT
{ ATTRIBUTE so : STRING ;
ATTRIBUTE duong : STRING ;
ATTRIBUTE tinhTP: STRING ;
ATTRIBUTE nuoc: STRING
}

Đối tượng có cấu trúc phức tạp VI

Ví dụ: Các kiểu dữ liệu (2)

CLASS **Sinhvien**
{ ATTRIBUTE MSSV : STRING ;
ATTRIBUTE ho : STRING ;
ATTRIBUTE ten : STRING ;
ATTRIBUTE diachi: **T_Diachi** ;
ATTRIBUTE ngaysinh: DATE ;
ATTRIBUTE phai : ENUM {'M', 'F'} ;
ATTRIBUTE monhoc : SET STRUCT {
ATTRIBUTE tenmon : STRING ;
ATTRIBUTE diem : FLOAT }
}

Định danh đối tượng (OID: Object Identifier) I

- **Mục tiêu:** *Định danh các đối tượng độc lập với giá trị và địa chỉ*
- Mỗi đối tượng có OID riêng và phải:
  - Tồn tại trong suốt chu kỳ sống của đối tượng,
  - Không thay đổi trong suốt chu kỳ sống của đối tượng,
  - Duy nhất trong CSDL.
- Các OID được quản lý bởi OODBMS
- OID là phương tiện hiệu quả để tham chiếu một đối tượng.
- Các oid do hệ thống quản lý, người dùng không thể truy xuất, thay đổi, in giá trị.

Định danh đối tượng (OID: Object Identifier) II

Định danh vs. Khóa chính

- Trong RDBMS: OID= khóa chính :
  - Phụ thuộc giá trị,
  - Vấn đề nếu : cập nhật khóa chính hoặc thay đổi thuộc tính khóa.
- Trong OODBMS: khoá chính không dùng làm OID, vì:
  - Khoá chính chỉ duy nhất trong 1 quan hệ, không phải trên toàn hệ thống.
  - Khoá chính được chọn từ các thuộc tính => phụ thuộc vào trạng thái của đối tượng.
- Trong NN lập trình: tên biến.

Định danh đối tượng (OID: Object Identifier) III

So sánh hai đối tượng

Cần phân biệt 2 trường hợp:

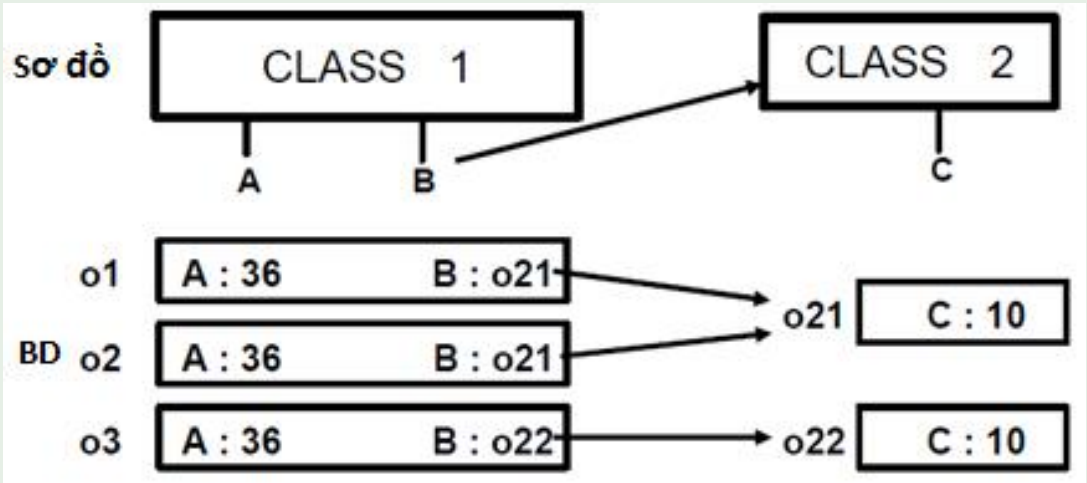
- 2 đối tượng là một (identical): cùng OID => o1 == o2
- 2 đối tượng bằng nhau (equal): khác OID, nhưng giá trị các thuộc tính bằng nhau:
  - Bằng cạn (shallow equality) : cùng giá trị: khi hai đối tượng o1 và o2 tham chiếu đến cùng đối tượng => o1 = o2
  - Bằng sâu (deep equality): khi hai đối tượng o1 và o2 tham chiếu đến hai đối tượng c1 và c2, tất cả các thuộc tính của hai đối tượng o1 và o2 phải bằng nhau (giá trị), tương tự cho c1 và c2 => o1 =* o2

Cho 2 đối tượng o1 và o2 của cùng lớp, ta có:

- o1 == o2 => o1 = o2
- o1 = o2 => o1 =* o2

Định danh đối tượng (OID: Object Identifier) IV

Ví dụ về so sánh hai đối tượng



o1.B == o2.B ? o1 == o2 ?

o1 = o2 ? o1 = o3 ?

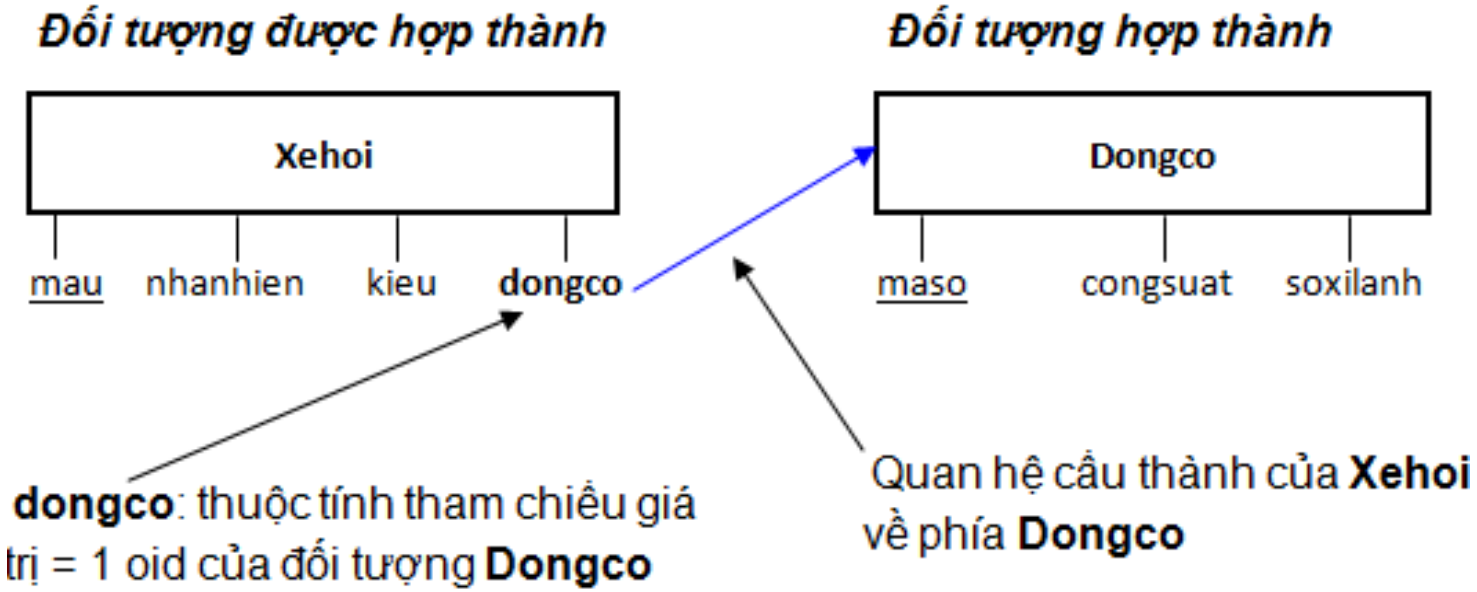
o1 =* o3 ?

o21 = o22 ? o21 == o22 ?

Quan hệ cấu thành I

Mục tiêu: *Giới thiệu mối quan hệ cấu thành giữa các đối tượng trong thế giới thực*

Ví dụ:



Quan hệ cấu thành II

Ví dụ

CLASS **Xehoi** {
  mau : STRING ;
  nhanhieu : STRING ;
  kieu : STRING ;
  dongco : **Dongco** }

CLASS **Dongco** {
  maso : STRING ;
  congsuat : FLOAT ;
  soXilanh : INT }

Chú ý

2 kiểu thuộc tính :
- Giá trị: (miền giá trị = STRING, INT,... kiểu cấu trúc)
- Tham chiếu (miền giá trị = 1 lớp đối tượng)

Quan hệ cấu thành III

Các ràng buộc toàn vẹn của quan hệ hợp thành

- Có hướng
- Có bản số bất kỳ dạng (min, max)
- Được biểu diễn trên sơ đồ bằng một mũi tên
- Được cài đặt bởi các thuộc tính tham chiếu trỏ đến các đối tượng hợp thành
- Các đối tượng hợp thành có thể được chia sẻ hoặc không
- Các đối tượng hợp thành có thể phụ thuộc hoặc không.
  - Ràng buộc phụ thuộc xác định thứ tự mà các đối tượng được tạo ra (và hủy đi): đầu tiên là đối tượng được hợp thành hay đối tượng hợp thành.
  - Một vài OODBMS đề nghị quan hệ đảo : Đối tượng được hợp thành phụ thuộc vào các đối tượng hợp thành.

Quan hệ cấu thành IV

Quan hệ đảo

Để có quan hệ hai chiều trong CSDL : Sử dụng quan hệ hợp thành và quan hệ đảo của nó

Ví dụ

```
CLASS Sinhvien
  { ATTRIBUTE mssv : STRING ;
  ATTRIBUTE ho : STRING ;
  ATTRIBUTE ten : STRING ;
  ATTRIBUTE dsmonhoc : SET STRUCT {
      ATTRIBUTE mon : Monhoc ;
      ATTRIBUTE diem : FLOAT }
  }
```

Quan hệ cấu thành V

Ví dụ (tt)

CLASS **Monhoc**
{ ATTRIBUTE tenmon : STRING ;
ATTRIBUTE dssv : SET Sinhvien }

=> Thuộc tính **dssv** của **Monhoc** là thuộc tính tham chiếu đảo của thuộc tính **dsmonhoc** của lớp **Sinhvien**

=> Để bảo đảm tính chặt chẽ của các thuộc tính tham chiếu đảo. Một vài OODBMS cho phép khai báo các thuộc tính đảo bằng cách sử dụng INVERSE :

CLASS **Monhoc**
{ ATTRIBUTE tenmon : STRING ;
ATTRIBUTE dssv : SET Sinhvien INVERSE Sinhvien.dsmonhoc.mon}

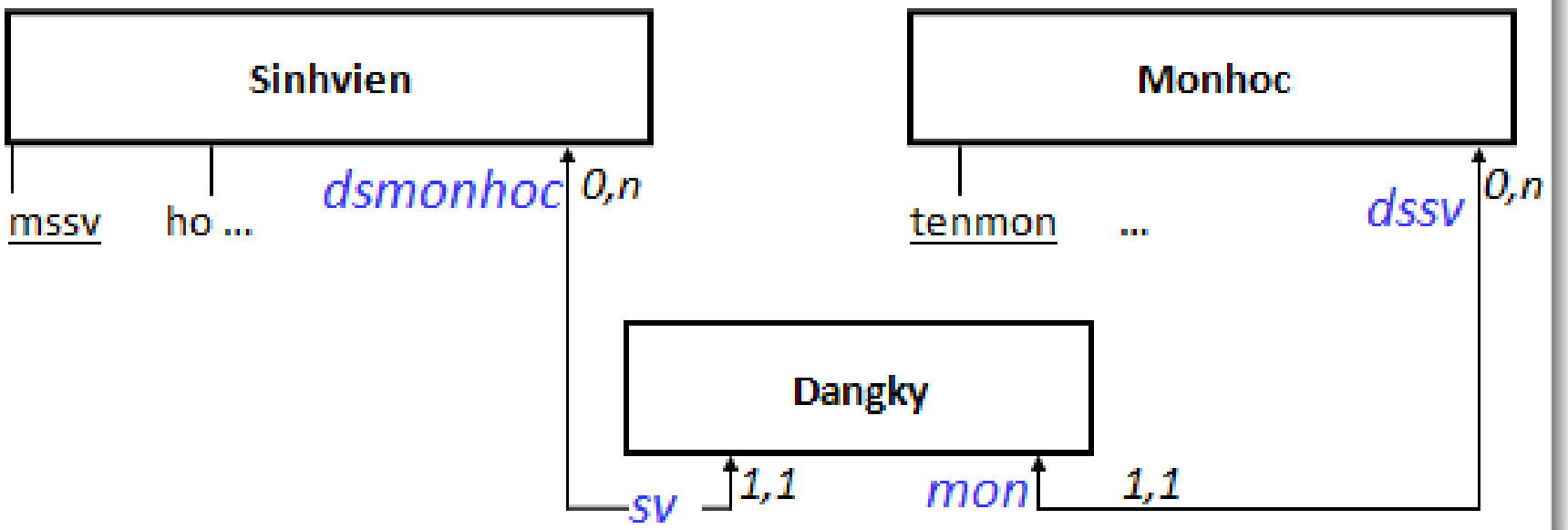
Quan hệ cấu thành VI

Quan hệ cấu thành và ODMG

- ODMG chỉ cho phép các thuộc tính tham chiếu là các thuộc tính của cấp độ đầu tiên của các lớp => thuộc tính hợp thành các thuộc tính phức không thể là các thuộc tính tham chiếu.
- Ví dụ : Lớp **Sinhvien** trên đây sẽ không thể khai báo theo OMDG vì thuộc tính tham chiếu **mon** là thuộc tính hợp thành.
- =>tạo lớp trung gian

Quan hệ cấu thành VII

Quan hệ cấu thành và ODMG - Ví dụ



Quan hệ cấu thành VIII

Quan hệ cấu thành và ODMG - Ví dụ

CLASS **Sinhvien**
{ ATTRIBUTE mssv : STRING ;
ATTRIBUTE ho : STRING ;
ATTRIBUTE ten : STRING ;
ATTRIBUTE dsmonhoc : SET Dangky }
CLASS **Dangky**
{ ATTRIBUTE mon : Monhoc ;
ATTRIBUTE diem : FLOAT ;
ATTRIBUTE sv : Sinhvien INVERSE Sinhvien.dsmonhoc}
CLASS **Monhoc**
{ ATTRIBUTE tenmon : STRING ;
ATTRIBUTE dssv : SET Sinhvien INVERSE Dangky.mon}

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x
  - UPDATE Xehoi
    WHERE mau = 'Golf GTI'
    SET dongco = x

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x
  - UPDATE Xehoi
    WHERE mau = 'Golf GTI'
    SET dongco = x
  - => x phải là một trong các oid của lớp được tham chiếu

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x
  - UPDATE Xehoi
    WHERE mau = 'Golf GTI'
    SET dongco = x
  - => x phải là một trong các oid của lớp được tham chiếu
- Xóa một đối tượng hợp thành

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x
  - UPDATE Xehoi
    WHERE mau = 'Golf GTI'
    SET dongco = x
  - => x phải là một trong các oid của lớp được tham chiếu
- Xóa một đối tượng hợp thành
  - OODBMS phải đặt giá trị NULL cho các thuộc tính tham chiếu của những đối tượng hợp thành

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x
  - UPDATE Xehoi
    WHERE mau = 'Golf GTI'
    SET dongco = x
  - => x phải là một trong các oid của lớp được tham chiếu
- Xóa một đối tượng hợp thành
  - OODBMS phải đặt giá trị NULL cho các thuộc tính tham chiếu của những đối tượng hợp thành
  - Nhưng hiếm khi điều này được thực hiện ...

Quan hệ cấu thành IX

Toàn vẹn tham chiếu

- OODBMS kiểm tra các phép gán :
  - Thuộc tính tham chiếu = x
  - UPDATE Xehoi
    WHERE mau = 'Golf GTI'
    SET dongco = x
  - => x phải là một trong các oid của lớp được tham chiếu
- Xóa một đối tượng hợp thành
  - OODBMS phải đặt giá trị NULL cho các thuộc tính tham chiếu của những đối tượng hợp thành
  - Nhưng hiếm khi điều này được thực hiện ...
  - SELECT v.dongco.maso
    FROM v IN Xehoi
    WHERE mau = 'Golf GTI' => Có thể bị treo !!!

Kế thừa I

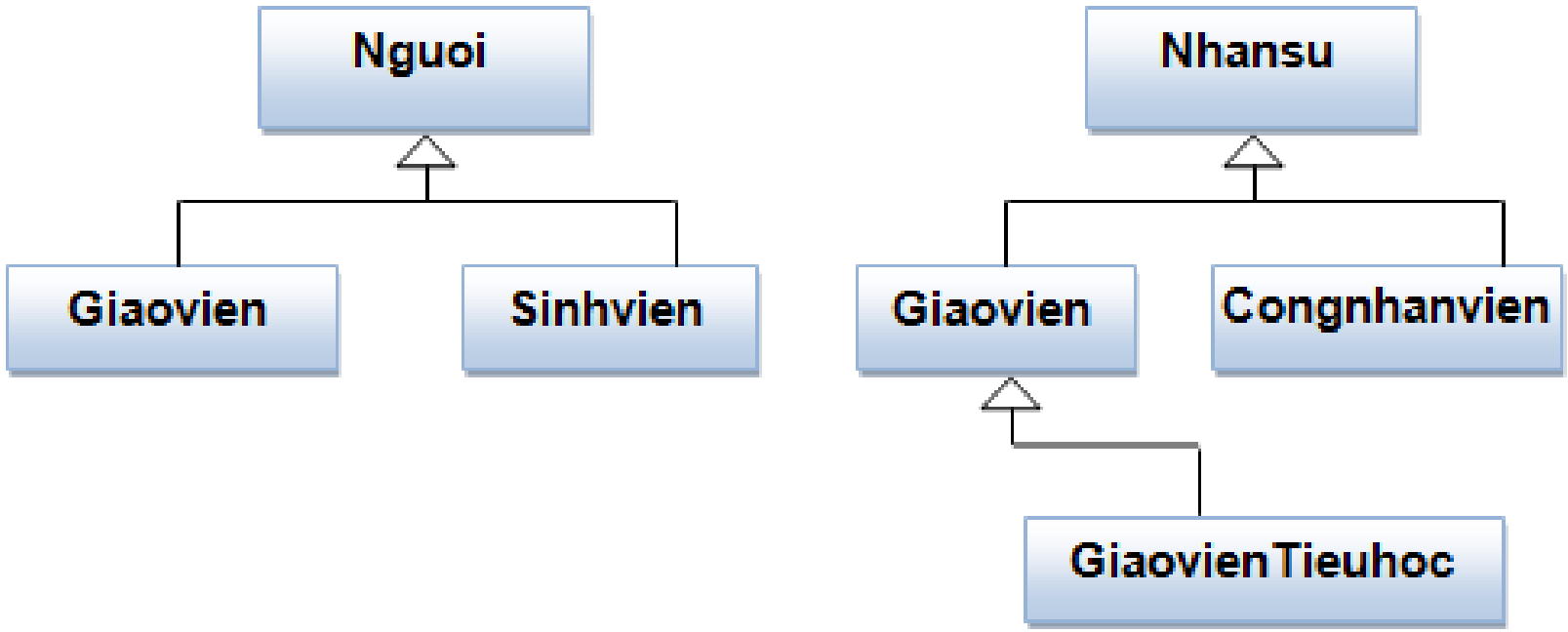
- **Mục tiêu:** *Biểu diễn một tập các đối tượng có các tính chất chung*

Kế thừa I

- **Mục tiêu:** *Biểu diễn một tập các đối tượng có các tính chất chung*
- Quan hệ **is-a** hay quan hệ phân cấp tổng quát hóa/chuyên biệt hóa hay quan hệ kế thừa,

Kế thừa I

- **Mục tiêu:** *Biểu diễn một tập các đối tượng có các tính chất chung*
- Quan hệ **is-a** hay quan hệ phân cấp tổng quát hóa/chuyên biệt hóa hay quan hệ kế thừa,
- **Ví dụ:**



Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:

Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:
  - Tất cả đối tượng của một SC cũng là đối tượng của lớp PC,

Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:
  - Tất cả đối tượng của một SC cũng là đối tượng của lớp PC,
  - **Tính thay thế**: luôn có thể sử dụng đối tượng của lớp SC ở vị trí của lớp PC,

Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:
  - Tất cả đối tượng của một SC cũng là đối tượng của lớp PC,
  - **Tính thay thế**: luôn có thể sử dụng đối tượng của lớp SC ở vị trí của lớp PC,
  - Tính kế thừa : lớp con SC kế thừa các thuộc tính giá trị, các thuộc tính tham chiếu và các phương thức của lớp cha PC của nó.

Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:
  - Tất cả đối tượng của một SC cũng là đối tượng của lớp PC,
  - **Tính thay thế**: luôn có thể sử dụng đối tượng của lớp SC ở vị trí của lớp PC,
  - Tính kế thừa : lớp con SC kế thừa các thuộc tính giá trị, các thuộc tính tham chiếu và các phương thức của lớp cha PC của nó.
- Lớp con SC có thể có:

Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:
  - Tất cả đối tượng của một SC cũng là đối tượng của lớp PC,
  - **Tính thay thế**: luôn có thể sử dụng đối tượng của lớp SC ở vị trí của lớp PC,
  - Tính kế thừa : lớp con SC kế thừa các thuộc tính giá trị, các thuộc tính tham chiếu và các phương thức của lớp cha PC của nó.
- Lớp con SC có thể có:
  - Các thuộc tính riêng của nó,

Kế thừa II

Tính chất của quan hệ is-a

- Quan hệ nhị phân có hướng giữa lớp cha (PC : parent class) và lớp con (SC: subclass) :**SC->PC**. Quan hệ này phải thỏa mãn ba quy tắc sau:
  - Tất cả đối tượng của một SC cũng là đối tượng của lớp PC,
  - **Tính thay thế**: luôn có thể sử dụng đối tượng của lớp SC ở vị trí của lớp PC,
  - Tính kế thừa : lớp con SC kế thừa các thuộc tính giá trị, các thuộc tính tham chiếu và các phương thức của lớp cha PC của nó.
- Lớp con SC có thể có:
  - Các thuộc tính riêng của nó,
  - Các thuộc tính hoặc phương thức được định nghĩa lại,

Kế thừa III

Ràng buộc toàn vẹn trên quan hệ is-a

- Tách rời (disjoint) : một tập con các thể hiện của PC là thể hiện của *nhiều nhất* một lớp SC.

Kế thừa III

Ràng buộc toàn vẹn trên quan hệ is-a

- Tách rời (disjoint) : một tập con các thể hiện của PC là thể hiện của *nhiều nhất* một lớp SC.
- Phủ (coverage) : một tập con các thể hiện của PC là thể hiện của *ít nhất* một lớp SC.

Kế thừa III

Ràng buộc toàn vẹn trên quan hệ is-a

- Tách rời (disjoint) : một tập con các thể hiện của PC là thể hiện của *nhiều nhất* một lớp SC.
- Phủ (coverage) : một tập con các thể hiện của PC là thể hiện của *ít nhất* một lớp SC.
- Phân hoạch (partition): thỏa mãn cả tách rời và phủ.

Định nghĩa lại thuộc tính I

Định nghĩa lại thuộc tính trong một lớp con (1)

- Định nghĩa mới cho thuộc tính

Định nghĩa lại thuộc tính I

Định nghĩa lại thuộc tính trong một lớp con (1)

- Định nghĩa mới cho thuộc tính
- Giới hạn miền trị của thuộc tính:

Định nghĩa lại thuộc tính I

Định nghĩa lại thuộc tính trong một lớp con (1)

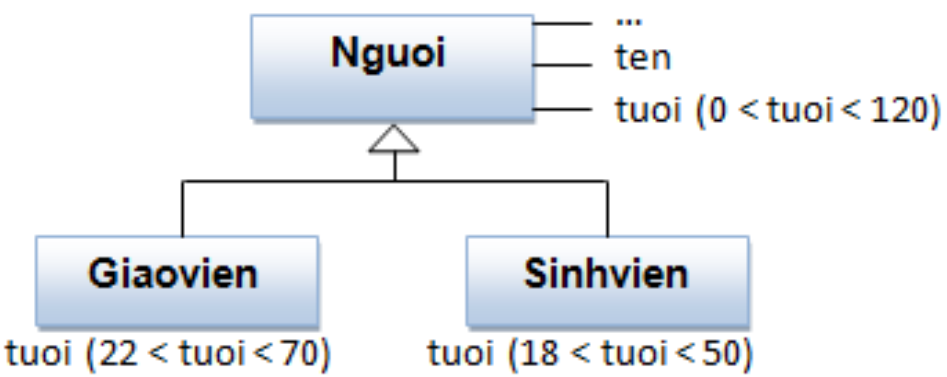
- Định nghĩa mới cho thuộc tính
- Giới hạn miền trị của thuộc tính:
  - Miền giá trị hoặc bản số được giới hạn, chỉ nhận một tập con của miền giá trị trong lớp cha PC)

Định nghĩa lại thuộc tính I

Định nghĩa lại thuộc tính trong một lớp con (1)

- Định nghĩa mới cho thuộc tính
- Giới hạn miền trị của thuộc tính:
  - Miền giá trị hoặc bản số được giới hạn, chỉ nhận một tập con của miền giá trị trong lớp cha PC)
  - Bổ sung thuộc tính phức
- Không tồn tại trong tất cả các OODBMS

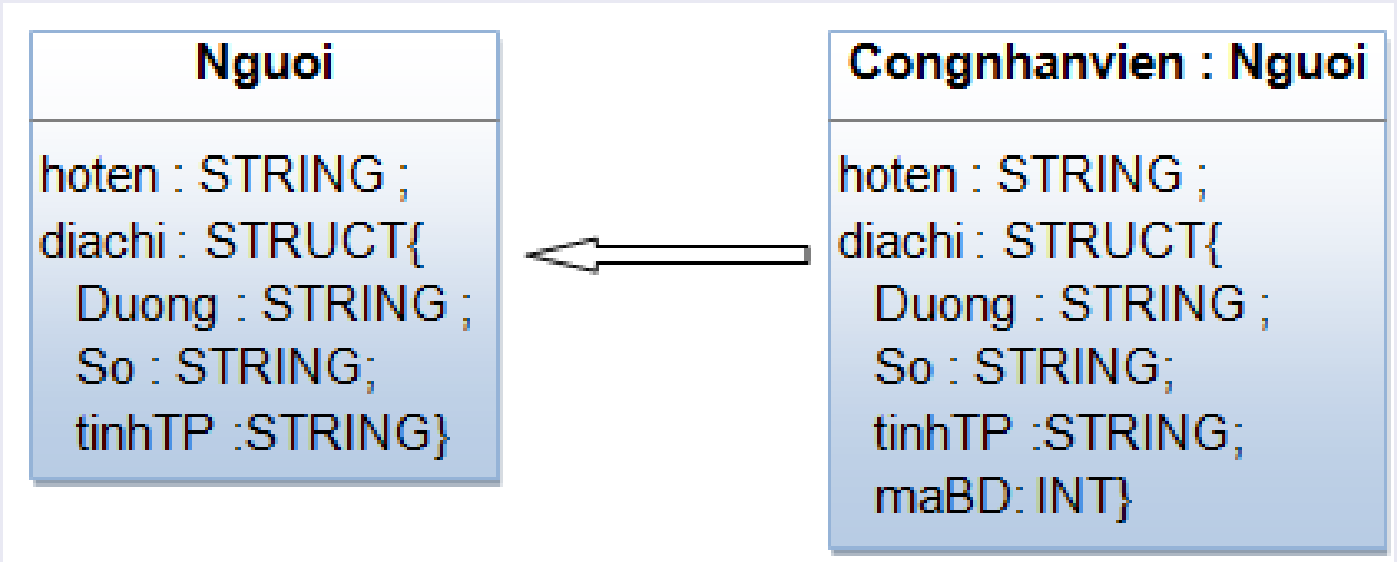
- Ví dụ : Giới hạn miền trị của thuộc tính



Định nghĩa lại thuộc tính II

Định nghĩa lại thuộc tính trong một lớp con (2)

- Ví dụ : Hoàn chỉnh thuộc tính phức



Chú ý

Trong ODMG, quan hệ is-a được biểu diễn bằng dấu hai chấm (:).

Hạn chế của sự phân cấp

- Kế thừa động ?
  => Một sinh viên có thể trở thành một giáo viên ?

Hạn chế của sự phân cấp

- Kế thừa động ?
  => Một sinh viên có thể trở thành một giáo viên ?
- => Phần lớn OODBMS cho phép định nghĩa sự phân cấp tĩnh

Hạn chế của sự phân cấp

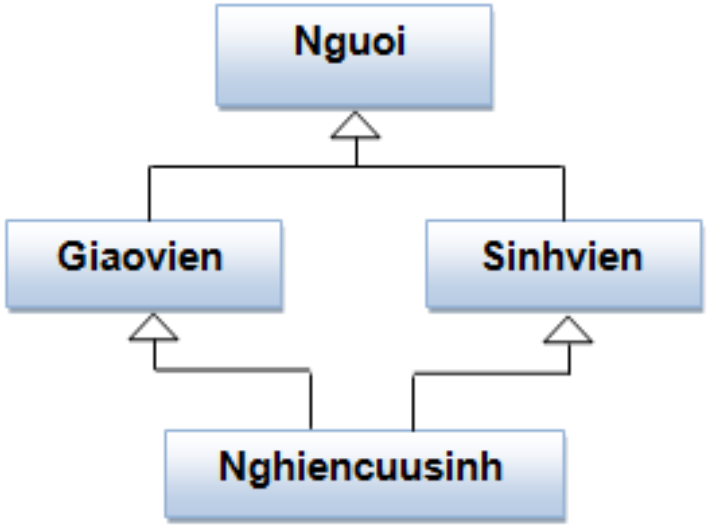
- Kế thừa động ?
  => Một sinh viên có thể trở thành một giáo viên ?
- => Phần lớn OODBMS cho phép định nghĩa sự phân cấp tĩnh
- Phần lớn OODBMS giới hạn khả năng (tách rời/phủ) phân cấp tổng quát hóa/chuyên biệt hóa

Đa kế thừa I

- Một lớp con SC có thể có nhiều lớp cha PC

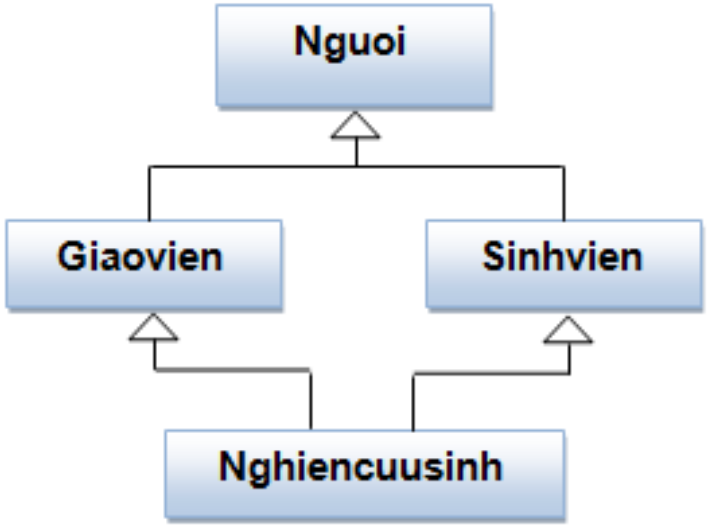
Đa kế thừa I

- Một lớp con SC có thể có nhiều lớp cha PC
- Ví dụ



Đa kế thừa I

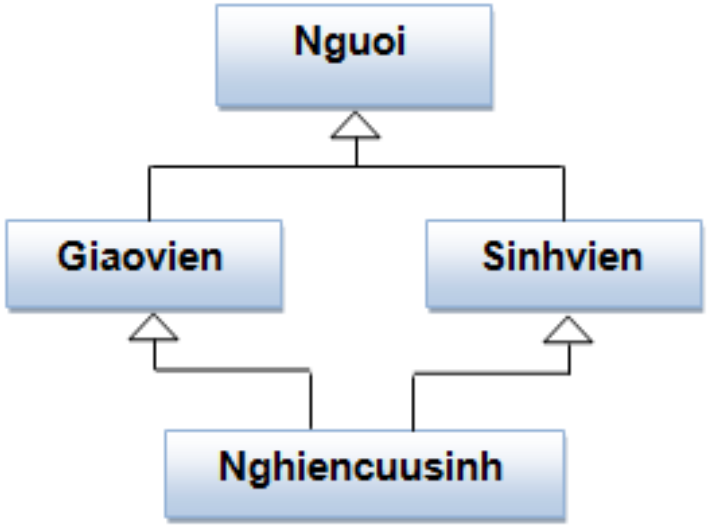
- Một lớp con SC có thể có nhiều lớp cha PC
- Ví dụ



- Xung đột khi kế thừa thuộc tính và phương thức :

Đa kế thừa I

- Một lớp con SC có thể có nhiều lớp cha PC
- Ví dụ



- Xung đột khi kế thừa thuộc tính và phương thức :
- => Hai thuộc tính/phương thức của hai lớp cha có thể có cùng tên và không giống nhau.
  => Lớp con sử dụng thuộc tính/phương thức của lớp cha nào ?

Đa kế thừa II

Giải quyết xung đột

Tùy vào từng OODBMS:

- Hoặc không giải quyết trường hợp này, không cho phép định nghĩa 1 sơ đồ dữ liệu như thế.
  => Đặt tên lại thuộc tính/phương thức.

Đa kế thừa II

Giải quyết xung đột

Tùy vào từng OODBMS:

- Hoặc không giải quyết trường hợp này, không cho phép định nghĩa 1 sơ đồ dữ liệu như thế.
  => Đặt tên lại thuộc tính/phương thức.
- Hoặc tự động đặt trước tên của các thuộc tính hoặc phương thức bằng tên của lớp cha (bởi hệ thống).

Đa kế thừa II

Giải quyết xung đột

Tùy vào từng OODBMS:

- Hoặc không giải quyết trường hợp này, không cho phép định nghĩa 1 sơ đồ dữ liệu như thế.
  => Đặt tên lại thuộc tính/phương thức.
- Hoặc tự động đặt trước tên của các thuộc tính hoặc phương thức bằng tên của lớp cha (bởi hệ thống).
- Hoặc áp dụng một sự lựa chọn (bởi hệ thống) dựa theo một quy tắc, ví dụ chọn lớp cha đầu tiên được khai báo.

Đa kế thừa II

Giải quyết xung đột

Tùy vào từng OODBMS:

- Hoặc không giải quyết trường hợp này, không cho phép định nghĩa 1 sơ đồ dữ liệu như thế.
  => Đặt tên lại thuộc tính/phương thức.
- Hoặc tự động đặt trước tên của các thuộc tính hoặc phương thức bằng tên của lớp cha (bởi hệ thống).
- Hoặc áp dụng một sự lựa chọn (bởi hệ thống) dựa theo một quy tắc, ví dụ chọn lớp cha đầu tiên được khai báo.
- Hoặc người thiết kế quyết định vào lúc định nghĩa sơ đồ dữ liệu (tĩnh) hoặc vào lúc truy xuất (động).

Cài đặt kế thừa

- Ngôn ngữ hỏi : SELECT * FROM **Nguoi** => Kết quả ???
  => Theo định dạng nào ???

Cài đặt kế thừa

- Ngôn ngữ hỏi : SELECT * FROM **Nguoi** => Kết quả ???
  => Theo định dạng nào ???
- Lưu trữ một đối tượng.

Cài đặt kế thừa

- Ngôn ngữ hỏi : SELECT * FROM **Nguoi** => Kết quả ???
  => Theo định dạng nào ???
- Lưu trữ một đối tượng.
  - Với kế thừa : 1 đối tượng=1 "mẫu tin" (trong lớp con sâu nhất)

Cài đặt kế thừa

- Ngôn ngữ hỏi : SELECT * FROM **Nguoi** => Kết quả ??? => Theo định dạng nào ???
- Lưu trữ một đối tượng.
  - Với kế thừa : 1 đối tượng=1 "mẫu tin" (trong lớp con sâu nhất)
  - Không kế thừa :1 đối tượng=1 "mẫu tin"/lớp (lớp đó và các lớp cha)

"Dân số" và sự bền vững I

- **Mục tiêu:**
  - CSDL: quản lý một tập các đối tượng bền vững : "dân số" (population)
  - NN LT HĐT : cho phép người sử dụng thao tác cùng cách những đối tượng tạm thời và những đối tượng bền vững

"Dân số" và sự bền vững I

- **Mục tiêu:**
  - CSDL: quản lý một tập các đối tượng bền vững : "dân số" (population)
  - NN LT HĐT : cho phép người sử dụng thao tác cùng cách những đối tượng tạm thời và những đối tượng bền vững
- **DBMS trước đây**, 1 kiểu/quan hệ (CSDL quan hệ):

"Dân số" và sự bền vững I

- **Mục tiêu:**
  - CSDL: quản lý một tập các đối tượng bền vững : "dân số" (population)
  - NN LT HĐT : cho phép người sử dụng thao tác cùng cách những đối tượng tạm thời và những đối tượng bền vững
- **DBMS trước đây**, 1 kiểu/quan hệ (CSDL quan hệ):
  - Định nghĩa cấu trúc cho các dữ liệu tiềm năng của kiểu/quan hệ.

"Dân số" và sự bền vững I

- **Mục tiêu:**
  - CSDL: quản lý một tập các đối tượng bền vững : "dân số" (population)
  - NN LT HĐT : cho phép người sử dụng thao tác cùng cách những đối tượng tạm thời và những đối tượng bền vững
- **DBMS trước đây**, 1 kiểu/quan hệ (CSDL quan hệ):
  - Định nghĩa cấu trúc cho các dữ liệu tiềm năng của kiểu/quan hệ.
  - Khai báo một thùng chứa (container) chứa tất cả các thể hiện hiện có và bền vững của kiểu/quan hệ.
    => Thùng chứa này được gọi là "dân số" (population) của kiểu/quan hệ.

"Dân số" và sự bền vững II

- **NN LT HĐT**:

"Dân số" và sự bền vững II

- **NN LT HĐT**:
  - 1 kiểu/lớp chỉ mô tả cấu trúc cho các đối tượng cùng kiểu.

"Dân số" và sự bền vững II

- **NN LT HĐT**:
  - 1 kiểu/lớp chỉ mô tả cấu trúc cho các đối tượng cùng kiểu.
  - Các đối tượng là tạm thời (chu kỳ sống của chúng = chu kỳ sống của chương trình trừ khi chúng được lưu vào các tập tin).

"Dân số" và sự bền vững II

- **NN LT HĐT**:
  - 1 kiểu/lớp chỉ mô tả cấu trúc cho các đối tượng cùng kiểu.
  - Các đối tượng là tạm thời (chu kỳ sống của chúng = chu kỳ sống của chương trình trừ khi chúng được lưu vào các tập tin).
- **OODBMS xuất phát từ quan điểm CSDL**:

"Dân số" và sự bền vững II

- **NN LT HĐT**:
  - 1 kiểu/lớp chỉ mô tả cấu trúc cho các đối tượng cùng kiểu.
  - Các đối tượng là tạm thời (chu kỳ sống của chúng = chu kỳ sống của chương trình trừ khi chúng được lưu vào các tập tin).
- **OODBMS xuất phát từ quan điểm CSDL**:
  - Kết hợp mỗi lớp với "dân số" bền vững của nó .

"Dân số" và sự bền vững II

- **NN LT HĐT**:
  - 1 kiểu/lớp chỉ mô tả cấu trúc cho các đối tượng cùng kiểu.
  - Các đối tượng là tạm thời (chu kỳ sống của chúng = chu kỳ sống của chương trình trừ khi chúng được lưu vào các tập tin).
- **OODBMS xuất phát từ quan điểm CSDL**:
  - Kết hợp mỗi lớp với "dân số" bền vững của nó .
  - ODMG: người thiết kế chọn việc định nghĩa hay không một thùng chứa vào lúc khai báo lớp nhờ vào mệnh đề : **EXTENT <populationName>**

"Dân số" và sự bền vững III

- **OODBMS xuất phát từ quan điểm OOLP**:

"Dân số" và sự bền vững III

- **OODBMS xuất phát từ quan điểm OOLP**:
  - Lớp chỉ định nghĩa cấu trúc dữ liệu.

"Dân số" và sự bền vững III

- **OODBMS xuất phát từ quan điểm OOLP**:
  - Lớp chỉ định nghĩa cấu trúc dữ liệu.
  - OODBMS không kết hợp bất kỳ "dân số" nào. Người sử dụng tạo thùng chứa để lưu trữ các đối tượng.

"Dân số" và sự bền vững III

- **OODBMS xuất phát từ quan điểm OOLP**:
  - Lớp chỉ định nghĩa cấu trúc dữ liệu.
  - OODBMS không kết hợp bất kỳ "dân số" nào. Người sử dụng tạo thùng chứa để lưu trữ các đối tượng.
  - => Định nghĩa biến hoặc đối tượng kiểu tập hợp (SET, LIST, BAG hoặc ARRAY) để lưu trữ các oid của các đối tượng mà người sử dụng muốn nhóm lại

"Dân số" và sự bền vững III

- **OODBMS xuất phát từ quan điểm OOLP**:
  - Lớp chỉ định nghĩa cấu trúc dữ liệu.
  - OODBMS không kết hợp bất kỳ "dân số" nào. Người sử dụng tạo thùng chứa để lưu trữ các đối tượng.
  - => Định nghĩa biến hoặc đối tượng kiểu tập hợp (SET, LIST, BAG hoặc ARRAY) để lưu trữ các oid của các đối tượng mà người sử dụng muốn nhóm lại
  - Ví dụ:

```
m : Giaovien;
tapGiaovien : SET Giaovien ; /* khai báo*/
tapGiaovien.insert_element(m) ;  /* thêm*/
SELECT x.ten FROM x IN tapGiaovien  /* sử dụng*/
```

"Dân số" và sự bền vững IV

- **OODBMS xuất phát từ quan điểm OOLP (tt)**:

"Dân số" và sự bền vững IV

- **OODBMS xuất phát từ quan điểm OOLP (tt)**:
  - OODBMS quản lý 2 kiểu đối tượng: bền vững và tạm thời.

"Dân số" và sự bền vững IV

- **OODBMS xuất phát từ quan điểm OOLP (tt)**:
  - OODBMS quản lý 2 kiểu đối tượng: bền vững và tạm thời.
    - Chỉ thị/lệnh cho phép 1 đối tượng trở thành bền vững (ODMG).
      => **Ví dụ: Name** giamdoc : Nguoi ;

"Dân số" và sự bền vững IV

- **OODBMS xuất phát từ quan điểm OOLP (tt)**:
  - OODBMS quản lý 2 kiểu đối tượng: bền vững và tạm thời.
    - Chỉ thị/lệnh cho phép 1 đối tượng trở thành bền vững (ODMG).
      => **Ví dụ: Name** giamdoc : Nguoi ;
    - Dựa vào quy tắc *"Đối tượng được cấu thành là bền vững thì các đối tượng cấu thành là bền vững"* (giải pháp bởi Gemstone và O2).
      => **Ví dụ:** thêm chỉ thị thêm đối tượng cấu thành vào đối tượng được cấu thành.

"Dân số" và sự bền vững V

Tính chất của sự bền vững

- Cho cùng một lớp: có thể có những đối tượng tạm thời và đối tượng bền vững.

"Dân số" và sự bền vững V

Tính chất của sự bền vững

- Cho cùng một lớp: có thể có những đối tượng tạm thời và đối tượng bền vững.
- Cùng một phương thức có thể được áp dụng cho những đối tượng tạm thời và đối tượng bền vững.

"Dân số" và sự bền vững V

Tính chất của sự bền vững

- Cho cùng một lớp: có thể có những đối tượng tạm thời và đối tượng bền vững.
- Cùng một phương thức có thể được áp dụng cho những đối tượng tạm thời và đối tượng bền vững.
- Một đối tượng bền vững không thể tham chiếu những đối tượng tạm thời.

"Dân số" và sự bền vững V

Tính chất của sự bền vững

- Cho cùng một lớp: có thể có những đối tượng tạm thời và đối tượng bền vững.
- Cùng một phương thức có thể được áp dụng cho những đối tượng tạm thời và đối tượng bền vững.
- Một đối tượng bền vững không thể tham chiếu những đối tượng tạm thời.
- Trạng thái bền vững/tạm thời có thể thay đổi bất kỳ thời điểm nào.

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :
  - Tất cả là bền vững

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :
  - Tất cả là bền vững
  - Lớp: sự bền vững được chỉ ra lúc khai báo lớp

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :
  - Tất cả là bền vững
  - Lớp: sự bền vững được chỉ ra lúc khai báo lớp
  - Thể hiện: sự bền vững được chỉ ra vào lúc tạo thể hiện

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :
  - Tất cả là bền vững
  - Lớp: sự bền vững được chỉ ra lúc khai báo lớp
  - Thể hiện: sự bền vững được chỉ ra vào lúc tạo thể hiện
- Động :

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :
  - Tất cả là bền vững
  - Lớp: sự bền vững được chỉ ra lúc khai báo lớp
  - Thể hiện: sự bền vững được chỉ ra vào lúc tạo thể hiện
- Động :
  - Chỉ ra bởi 1 lệnh ở bất kỳ thời điểm nào
    Ví dụ: tapGiaovien.**save()**

"Dân số" và sự bền vững VI

Kỹ thuật bền vững

Hai mô hình bền vững khác nhau:

- Tĩnh :
  - Tất cả là bền vững
  - Lớp: sự bền vững được chỉ ra lúc khai báo lớp
  - Thể hiện: sự bền vững được chỉ ra vào lúc tạo thể hiện
- Động :
  - Chỉ ra bởi 1 lệnh ở bất kỳ thời điểm nào
    Ví dụ: tapGiaovien.**save()**
  - Bởi việc truy xuất từ gốc của sự bền vững: "Tất cả đối tượng cấu thành của một đối tượng bền vững là bền vững"
    Ví dụ: **PersistList**.insert_last_element(tapgiaovien)

"Dân số" và sự bền vững VII

ODMG

- Sử dụng tiếp cận theo kiểu HQT CSDL cũ.

"Dân số" và sự bền vững VII

ODMG

- Sử dụng tiếp cận theo kiểu HQT CSDL cũ.
  - Tất cả đối tượng là bền vững

"Dân số" và sự bền vững VII

ODMG

- Sử dụng tiếp cận theo kiểu HQT CSDL cũ.
  - Tất cả đối tượng là bền vững
  - Mỗi lớp có 1 (hoặc 0) "dân số"

"Dân số" và sự bền vững VII

ODMG

- Sử dụng tiếp cận theo kiểu HQT CSDL cũ.
  - Tất cả đối tượng là bền vững
  - Mỗi lớp có 1 (hoặc 0) "dân số"
  - Nếu "dân số" tồn tại thị tất cả các đối tượng tự động lưu vào trong nó.

"Dân số" và sự bền vững VII

ODMG

- Sử dụng tiếp cận theo kiểu HQT CSDL cũ.
  - Tất cả đối tượng là bền vững
  - Mỗi lớp có 1 (hoặc 0) "dân số"
  - Nếu "dân số" tồn tại thị tất cả các đối tượng tự động lưu vào trong nó.
- Cú pháp :
  CLASS <className> [EXTENT <populationName>]

"Dân số" và sự bền vững VII

ODMG

- Sử dụng tiếp cận theo kiểu HQT CSDL cũ.
  - Tất cả đối tượng là bền vững
  - Mỗi lớp có 1 (hoặc 0) "dân số"
  - Nếu "dân số" tồn tại thị tất cả các đối tượng tự động lưu vào trong nó.
- Cú pháp :
  CLASS <className> [EXTENT <populationName>]
- Người dùng có thể kết hợp tên bền vững với một vài đối tượng
  Ví dụ: NAME giamdoc : Nhansu */* khai báo biến bền vững có tên*/*
  giamdoc := Nhansu (id: 1111,ho: 'Iksal'...) */* tạo đối tượng giám đốc*/*

Phương thức và sự đóng gói I

Phương thức (1)

- Mỗi lớp được kết hợp với các phương thức cho phép:
  - Truy xuất,
  - Cập nhật,
  - Thao tác

  các đối tượng của lớp (hoặc các trị của kiểu dữ liệu)

Phương thức và sự đóng gói I

Phương thức (1)

- Mỗi lớp được kết hợp với các phương thức cho phép:
  - Truy xuất,
  - Cập nhật,
  - Thao tác

  các đối tượng của lớp (hoặc các trị của kiểu dữ liệu)
- Định nghĩa một lớp các đối tượng:
  - Định nghĩa cấu trúc của các đối tượng,
  - Định nghĩa các phương thức để thao tác trên các đối tượng.

Phương thức và sự đóng gói II

Phương thức (2)

- Định nghĩa một phương thức

Phương thức và sự đóng gói II

Phương thức (2)

- Định nghĩa một phương thức
  - Giao diện (interface):
    - tên phương thức,
    - danh sách các tham số và kiểu của nó,
    - kiểu trả về nếu có

Phương thức và sự đóng gói II

Phương thức (2)

- Định nghĩa một phương thức
  - Giao diện (interface):
    - tên phương thức,
    - danh sách các tham số và kiểu của nó,
    - kiểu trả về nếu có
  - Cài đặt (code): ẩn đối với người sử dụng:
    - các chỉ thị NN lập trình HĐT (C++, Java),
    - các chỉ thị của OODBMS (ngôn ngữ hỏi, lời gọi phương thức chuẩn như tạo, xóa đối tượng),
    - lời gọi phương thức của các lớp đối tượng của CSDL.

Phương thức và sự đóng gói III

Phương thức - Ví dụ

Class **Nguoi**

{ ho : STRING ;
ten : STRING ;
diachi : STRING ;
Void hienthi() ;} ;

Phương thức và sự đóng gói IV

Sự đóng gói

- **Mục tiêu:** *Ẩn đi các chi tiết cài đặt bên trong của các lớp*

Phương thức và sự đóng gói IV

Sự đóng gói

- **Mục tiêu:** *Ẩn đi các chi tiết cài đặt bên trong của các lớp*
  - Tạo cho việc tái sử dụng của các lớp được dễ dàng : chỉ cần biết giao diện (interface)

Phương thức và sự đóng gói IV

Sự đóng gói

- **Mục tiêu:** *Ẩn đi các chi tiết cài đặt bên trong của các lớp*
  - Tạo cho việc tái sử dụng của các lớp được dễ dàng : chỉ cần biết giao diện (interface)
  - Cho phép thay đổi cấu trúc đối tượng, sửa đổi cài đặt của phương thức, thêm phương thức mới mà không cần sửa đổi chương trình ứng dụng, chỉ cần biên dịch lại.

Phương thức và sự đóng gói IV

Sự đóng gói

- **Mục tiêu:** *Ẩn đi các chi tiết cài đặt bên trong của các lớp*
  - Tạo cho việc tái sử dụng của các lớp được dễ dàng : chỉ cần biết giao diện (interface)
  - Cho phép thay đổi cấu trúc đối tượng, sửa đổi cài đặt của phương thức, thêm phương thức mới mà không cần sửa đổi chương trình ứng dụng, chỉ cần biên dịch lại.
- Đặc điểm : từ bên ngoài đối tượng, chỉ khai báo (giao diện) của các phương thức là được nhìn thấy (visible)

Phương thức và sự đóng gói IV

Sự đóng gói

- **Mục tiêu:** *Ẩn đi các chi tiết cài đặt bên trong của các lớp*
  - Tạo cho việc tái sử dụng của các lớp được dễ dàng : chỉ cần biết giao diện (interface)
  - Cho phép thay đổi cấu trúc đối tượng, sửa đổi cài đặt của phương thức, thêm phương thức mới mà không cần sửa đổi chương trình ứng dụng, chỉ cần biên dịch lại.
- Đặc điểm : từ bên ngoài đối tượng, chỉ khai báo (giao diện) của các phương thức là được nhìn thấy (visible)
- Những thành phần được ẩn:
  - Cấu trúc đối tượng,
  - Cài đặt chi tiết của các phương thức.

Phương thức và sự đóng gói V

Ví dụ đóng gói

CLASS **Nhansu**

- Giao diện nhìn thấy được :
  - INT luong()
  - VOID hienthi()

Phương thức và sự đóng gói V

Ví dụ đóng gói

CLASS **Nhansu**

- Giao diện nhìn thấy được :
  - INT luong()
  - VOID hienthi()
- Cài đặt ẩn :
  ATTRIBUTE id : STRING ;
  ATTRIBUTE hoten : STRING ;
  ATTRIBUTE diachi : STRING ;
  ATTRIBUTE luongThang : INT ;
  RELATIONSHIP donvi : Donvi INVERSE Donvi.dsnhansu

Phương thức và sự đóng gói VI

Ví dụ đóng gói (tt)

- Cài đặt ẩn (tt) :

```
luong ()
    { return luongThang; }
hienthi ()  {
    PRINT('id:', self.id) ; PRINT('Họ tên :', self.hoten) ;
    PRINT('Địa chỉ :', self.diachi) ;
    PRINT(Lương tháng :', self.luongthang) ; }
```

Đóng gói

Chỉ có chính đối tượng (tức là các chỉ thị của các phương thức) có thể truy xuất các thuộc tính.

Phương thức và sự đóng gói VII

Chú ý

Trong ngôn ngữ hỏi OQL của ODMG:

- Nếu các câu truy vấn không được sử dụng lại thì đóng gói không có tác dụng.

Phương thức và sự đóng gói VII

Chú ý

Trong ngôn ngữ hỏi OQL của ODMG:

- Nếu các câu truy vấn không được sử dụng lại thì đóng gói không có tác dụng.
- => Đóng gói ít nhiều có giới hạn của nó, ví dụ :
  - Trong NN lập trình HĐT: dùng đóng gói,
  - Trong các câu truy vấn: không dùng đóng gói.

Đa hình

- Nhiều phương thức có code khác nhau có cùng tên.

Đa hình

- Nhiều phương thức có code khác nhau có cùng tên.
- Việc lựa chọn phương thức nào đúng tùy thuộc vào chức năng của lớp đối tượng được gọi.

Đa hình

- Nhiều phương thức có code khác nhau có cùng tên.
- Việc lựa chọn phương thức nào đúng tùy thuộc vào chức năng của lớp đối tượng được gọi.
- Hai kiểu lựa chọn phổ biến:
  - Vào lúc biên dịch (ví dụ như ngôn ngữ C, Pascal),
  - Vào lúc thực thi (ví dụ như ngôn ngữ Smalltalk) => gắn kết động (dynamic binding/late binding)

Đa hình

- Nhiều phương thức có code khác nhau có cùng tên.
- Việc lựa chọn phương thức nào đúng tùy thuộc vào chức năng của lớp đối tượng được gọi.
- Hai kiểu lựa chọn phổ biến:
  - Vào lúc biên dịch (ví dụ như ngôn ngữ C, Pascal),
  - Vào lúc thực thi (ví dụ như ngôn ngữ Smalltalk) => gắn kết động (dynamic binding/late binding)
- Hai kiểu đa hình phổ biến: định nghĩa lại (overriding) và định nghĩa chồng (overloading)

Định nghĩa lại phương thức I

Khái niệm này gắn liền với quan hệ kế thừa.

- **Mục tiêu** : *Làm cho cài đặt bên trong một phương thức phù hợp với đặc điểm của lớp con.*

Định nghĩa lại phương thức I

Khái niệm này gắn liền với quan hệ kế thừa.

- **Mục tiêu** : *Làm cho cài đặt bên trong một phương thức phù hợp với đặc điểm của lớp con.*
- Giao diện của phương thức không thay đổi.

Định nghĩa lại phương thức I

Khái niệm này gắn liền với quan hệ kế thừa.

- **Mục tiêu** : *Làm cho cài đặt bên trong một phương thức phù hợp với đặc điểm của lớp con.*
- Giao diện của phương thức không thay đổi.
- **Ví dụ:**
  - Lớp **Nhansu** : luong()= self.luongThang
  - Lớp **Giaovien**: luong()= self.luongThang + self.phucap
  - Lớp **Giaovientieuhoc**: luong()= self.luongThang + self.phucap + self.phucapuudai

Định nghĩa lại phương thức I

Khái niệm này gắn liền với quan hệ kế thừa.

- **Mục tiêu** : *Làm cho cài đặt bên trong một phương thức phù hợp với đặc điểm của lớp con.*
- Giao diện của phương thức không thay đổi.
- **Ví dụ:**
  - Lớp **Nhansu** : luong()= self.luongThang
  - Lớp **Giaovien**: luong()= self.luongThang + self.phucap
  - Lớp **Giaovientieuhoc**: luong()= self.luongThang + self.phucap + self.phucapuudai
- Không định nghĩa lại => tên các phương thức phải khác nhau
  - Lớp **Nhansu** : luong()
  - Lớp **Giaovien**: luongGv()
  - Lớp **Giaovien**: luongGvTieuhoc()

Định nghĩa lại phương thức II

Ví dụ - Ngôn ngữ hỏi

- Không định nghĩa lại

```
SELECT p.luong()      FROM p IN tapNhansu
WHERE NOT (p IN tapGiaovien)

SELECT p.luongGv()      FROM p IN tapGiaovien
WHERE NOT (p IN tapGiaovienTieuhoc)

SELECT p.luongGvTieuhoc()
FROM p IN tapGiaovienTieuhoc
```

Định nghĩa lại phương thức II

Ví dụ - Ngôn ngữ hỏi

- Không định nghĩa lại

  SELECT p.luong() FROM p IN tapNhansu
  WHERE NOT (p IN tapGiaovien)

  SELECT p.luongGv() FROM p IN tapGiaovien
  WHERE NOT (p IN tapGiaovienTieuhoc)

  SELECT p.luongGvTieuhoc()
  FROM p IN tapGiaovienTieuhoc

- Với định nghĩa lại, cùng tên phương thức nhưng code khác nhau

  SELECT p.luong() FROM p IN tapNhansu
  =>Vấn đề lựa chọn phương thức ???

Định nghĩa lại phương thức III

Giải pháp

- SELECT p.luong()
  FROM p IN tapNhansu

Định nghĩa lại phương thức III

Giải pháp

- SELECT p.luong()
  FROM p IN tapNhansu
- Phương thức luong() được định nghĩa trong nhiều lớp con

Định nghĩa lại phương thức III

Giải pháp

- SELECT p.luong()
  FROM p IN tapNhansu
- Phương thức luong() được định nghĩa trong nhiều lớp con
- =>Phương thức luong() của lớp nào được chọn ???

Định nghĩa lại phương thức III

Giải pháp

- SELECT p.luong()
  FROM p IN tapNhansu
- Phương thức luong() được định nghĩa trong nhiều lớp con
- =>Phương thức luong() của lớp nào được chọn ???
- Giải pháp 1 : Phương thức của lớp được khai báo
  - Chọn tĩnh vào lúc biên dịch
  - Ví dụ: cùng công thức tính lương cho tất cả các lớp (= luongThang)

Định nghĩa lại phương thức III

Giải pháp

- SELECT p.luong()
  FROM p IN tapNhansu
- Phương thức luong() được định nghĩa trong nhiều lớp con
- =>Phương thức luong() của lớp nào được chọn ???
- Giải pháp 1 : Phương thức của lớp được khai báo
  - Chọn tĩnh vào lúc biên dịch
  - Ví dụ: cùng công thức tính lương cho tất cả các lớp (= luongThang)
- Giải pháp 2 : Gắn kết động

Định nghĩa lại phương thức IV

Gắn kết động (dynamic binding)

- Gắn kết là quá trình chọn phương thức thích hợp, dựa trên kiểu của đối tượng.

Định nghĩa lại phương thức IV

Gắn kết động (dynamic binding)

- Gắn kết là quá trình chọn phương thức thích hợp, dựa trên kiểu của đối tượng.
- Quá trình gắn kết động là việc xác định kiểu đối tượng được trì hoãn đến lúc thực thi chương trình, chứ không phải ngay khi biên dịch.

Định nghĩa lại phương thức IV

Gắn kết động (dynamic binding)

- Gắn kết là quá trình chọn phương thức thích hợp, dựa trên kiểu của đối tượng.
- Quá trình gắn kết động là việc xác định kiểu đối tượng được trì hoãn đến lúc thực thi chương trình, chứ không phải ngay khi biên dịch.
- Phần lớn các OODBMS chấp nhận giải pháp này.

Định nghĩa lại phương thức IV

Gắn kết động (dynamic binding)

- Gắn kết là quá trình chọn phương thức thích hợp, dựa trên kiểu của đối tượng.
- Quá trình gắn kết động là việc xác định kiểu đối tượng được trì hoãn đến lúc thực thi chương trình, chứ không phải ngay khi biên dịch.
- Phần lớn các OODBMS chấp nhận giải pháp này.
- => Phương thức của lớp con sâu nhất chứa đối tượng (oid) được chọn.

Định nghĩa chồng phương thức

- Giao diện của phương thức thay đổi :
  - Kiểu trả về
  - Số tham số, kiểu tham số

Định nghĩa chồng phương thức

- Giao diện của phương thức thay đổi :
  - Kiểu trả về
  - Số tham số, kiểu tham số
- Làm đơn giản trình ứng dụng: cho phép xác định ngữ cảnh mà chỉ một phương thức thích hợp được gọi tại một thời điểm.

Định nghĩa chồng phương thức

- Giao diện của phương thức thay đổi :
  - Kiểu trả về
  - Số tham số, kiểu tham số
- Làm đơn giản trình ứng dụng: cho phép xác định ngữ cảnh mà chỉ một phương thức thích hợp được gọi tại một thời điểm.
- Định nghĩa lại : gắn kết động => kiểu trả của phương thức lớp con về phải tương thích với kiểu trả về của phương thức lớp cha.

Định nghĩa chồng phương thức

- Giao diện của phương thức thay đổi :
  - Kiểu trả về
  - Số tham số, kiểu tham số
- Làm đơn giản trình ứng dụng: cho phép xác định ngữ cảnh mà chỉ một phương thức thích hợp được gọi tại một thời điểm.
- Định nghĩa lại : gắn kết động => kiểu trả của phương thức lớp con về phải tương thích với kiểu trả về của phương thức lớp cha.
- Định nghĩa chồng : không gắn kết động => kiểu trả về bất kỳ.

Thư viện lớp/kiểu I

Tùy vào từng OODBMS, nó có thể cung cấp những thư viện tương đối hoàn chỉnh.

- Đối tượng (lớp cha cao nhất) do người dùng định nghĩa :
  - create([name: value, ...]) -> o:Oid
  - delete()
  - same_as(o:Oid) -> Boolean

Thư viện lớp/kiểu I

Tùy vào từng OODBMS, nó có thể cung cấp những thư viện tương đối hoàn chỉnh.

- Đối tượng (lớp cha cao nhất) do người dùng định nghĩa :
  - create([name: value, ...]) -> o:Oid
  - delete()
  - same_as(o:Oid) -> Boolean
- Collection:
  - create(<T>) -> c:Collection<T>
  - insert_element(e)
  - remove_element(e)
  - ...

Thư viện lớp/kiểu II

- LIST:
  - insert_element_before(o:T, position:Integer)
  - insert_first_element(o:T)
  - retrieve_element_at(position:Integer) -> element:T
  - ...

Thư viện lớp/kiểu II

- LIST:
  - insert_element_before(o:T, position:Integer)
  - insert_first_element(o:T)
  - retrieve_element_at(position:Integer) -> element:T
  - ...
- SET :
  - union(s2:Set<T>) -> s3:Set<T>
  - intersection(s2:Set<T>) -> s3:Set<T>
  - difference(s2:Set<T>) -> s3:Set<T>
  - is_subset?(s2:Set<T>) -> b:Boolean

Thư viện lớp/kiểu II

- LIST:
  - insert_element_before(o:T, position:Integer)
  - insert_first_element(o:T)
  - retrieve_element_at(position:Integer) -> element:T
  - ...
- SET :
  - union(s2:Set<T>) -> s3:Set<T>
  - intersection(s2:Set<T>) -> s3:Set<T>
  - difference(s2:Set<T>) -> s3:Set<T>
  - is_subset?(s2:Set<T>) -> b:Boolean
- Etc.

Thư viện lớp/kiểu III

- Ưu điểm

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế
  - Giảm hiệu suất của lập trình viên vì họ quen với ngôn ngữ khai báo như SQL

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế
  - Giảm hiệu suất của lập trình viên vì họ quen với ngôn ngữ khai báo như SQL
  - Ngôn ngữ khai báo dễ sử dụng hơn NN lập trình

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế
  - Giảm hiệu suất của lập trình viên vì họ quen với ngôn ngữ khai báo như SQL
  - Ngôn ngữ khai báo dễ sử dụng hơn NN lập trình
  - Người sử dụng không tối ưu hóa được các câu hỏi => hệ thống thực hiện

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế
  - Giảm hiệu suất của lập trình viên vì họ quen với ngôn ngữ khai báo như SQL
  - Ngôn ngữ khai báo dễ sử dụng hơn NN lập trình
  - Người sử dụng không tối ưu hóa được các câu hỏi => hệ thống thực hiện
- => Giải pháp

Thư viện lớp/kiểu III

- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế
  - Giảm hiệu suất của lập trình viên vì họ quen với ngôn ngữ khai báo như SQL
  - Ngôn ngữ khai báo dễ sử dụng hơn NN lập trình
  - Người sử dụng không tối ưu hóa được các câu hỏi => hệ thống thực hiện
- => Giải pháp
  - Ngôn ngữ hỏi thuần đối tượng gần SQL : OQL của ODMG

Thư viện lớp/kiểu III

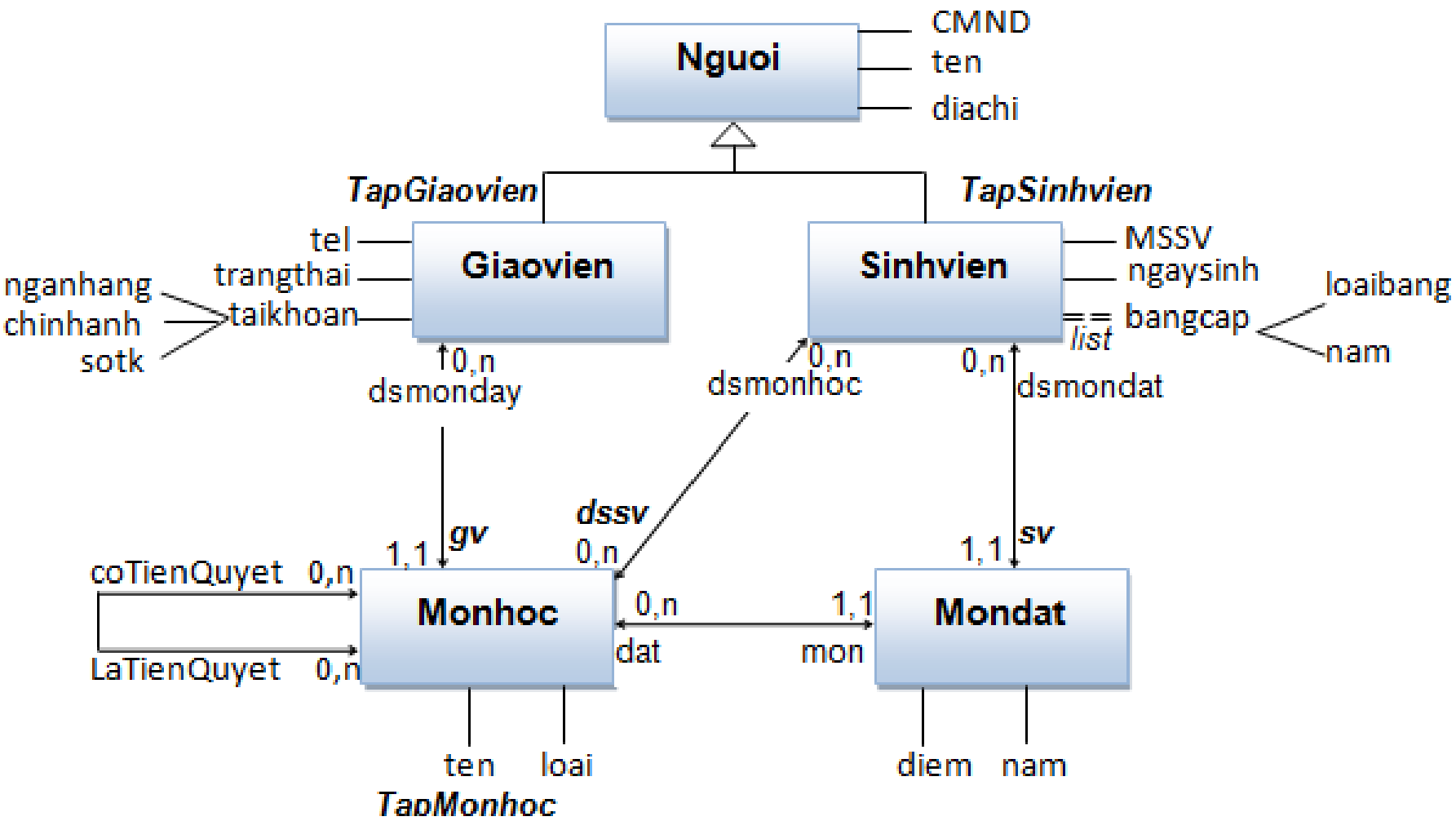
- Ưu điểm
  - Người sử dụng có sẵn các hàm định nghĩa trước như của 1 NNLT
  - Giảm sự không tương thích giữa NNLT và NN thao tác CSDL
- Hạn chế
  - Giảm hiệu suất của lập trình viên vì họ quen với ngôn ngữ khai báo như SQL
  - Ngôn ngữ khai báo dễ sử dụng hơn NN lập trình
  - Người sử dụng không tối ưu hóa được các câu hỏi => hệ thống thực hiện
- => Giải pháp
  - Ngôn ngữ hỏi thuần đối tượng gần SQL : OQL của ODMG
  - Chuẩn SQL 3 được mở rộng từ SQL quan hệ

Nội dung

CSDL Đào Tạo I

- Một trường ĐH muốn quản lý một cơ sở dữ liệu với môn học, giáo viên, sinh viên, đăng ký môn học và kết quả học.
- Mỗi môn học được xác định bởi tên và loại (1, 2 và 3) và có không, một hoặc nhiều môn cùng loại hoặc loại trước đó là môn tiên quyết.
- Giáo viên, được định danh bởi họ và tên, địa chỉ, số điện thoại, trạng thái (giảng viên, phó giáo sư, ...) và thông tin ngân hàng. Mỗi giáo viên có thể dạy một hoặc nhiềum môn, nhưng mỗi môn chỉ được dạy bởi một giáo viên.
- Sinh viên ghi danh vào một hoặc nhiều môn học và trả một khoản phí đăng ký cho mỗi môn học. Sinh viên được đinh danh bởi MSSV và các thông tin họ, tên, địa chỉ, các bằng cấp đã nhận (Bằng cấp và năm) và ngày sinh.
- Với mỗi học sinh, trường lưu lại danh sách các môn học mà sinh viên đã đạt cùng với điểm và năm đạt được.

CSDL Đào Tạo II



Kết luận

- **Mục tiêu đạt được**

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,
    - Thiết kế phương thức,

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,
    - Thiết kế phương thức,
  - Cạnh tranh giữa các tiêu chuẩn

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,
    - Thiết kế phương thức,
  - Cạnh tranh giữa các tiêu chuẩn
  - Thiếu cơ sở lý thuyết

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,
    - Thiết kế phương thức,
  - Cạnh tranh giữa các tiêu chuẩn
  - Thiếu cơ sở lý thuyết
  - Không hổ trợ khung nhìn

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,
    - Thiết kế phương thức,
  - Cạnh tranh giữa các tiêu chuẩn
  - Thiếu cơ sở lý thuyết
  - Không hổ trợ khung nhìn
  - Khó khăn trong việc chuyển từ DBMS kiểu cũ (quan hệ) sang OODBMS

Kết luận

- **Mục tiêu đạt được**
  - Biểu diễn các đối tượng của thế giới thực phong phú hơn,
  - Cho phép việc tái sử dụng,
  - Hiệu quả cho các ứng dụng mới.
- **Tuy nhiên:**
  - Các OODBMS không phù hợp với bất kỳ loại ứng dụng!
  - Phương pháp thiết kế không đầy đủ :
    - Chuẩn hóa cấu trúc CSDL,
    - Thiết kế phương thức,
  - Cạnh tranh giữa các tiêu chuẩn
  - Thiếu cơ sở lý thuyết
  - Không hổ trợ khung nhìn
  - Khó khăn trong việc chuyển từ DBMS kiểu cũ (quan hệ) sang OODBMS
  - Tối ưu hóa câu hỏi do OODBMS thực hiện.