

# Cơ sở dữ liệu

Nguyễn Đình Hóa

[dinhhoa@gmail.com](mailto:dinhhoa@gmail.com) / [hoand@ptit.edu.vn](mailto:hoand@ptit.edu.vn)

094-280-7711

# Giới thiệu

- Kiến thức: Nắm được các mức trừu tượng hóa cơ sở dữ liệu, mô hình cơ sở dữ liệu quan hệ , quy trình thiết kế cơ sở dữ liệu, các ngôn ngữ định nghĩa và thao tác dữ liệu
- Kỹ năng: Áp dụng các kiến thức vào việc thiết kế cơ sở dữ liệu, xây dựng các ứng dụng cơ sở dữ liệu

# Tóm tắt nội dung

- Cung cấp những kiến thức cơ bản về cơ sở dữ liệu, các phương pháp tiếp cận và các nguyên tắc thiết kế các hệ cơ sở dữ liệu quan hệ.
- Quá trình xây dựng một hệ CSDL.
- Các phép toán đại số quan hệ cơ bản
- Các câu lệnh của ngôn ngữ định nghĩa và thao tác dữ liệu trên các hệ cơ sở dữ liệu.

# Nội dung

- **Khái niệm cơ bản về cơ sở dữ liệu**
- **Các mô hình cơ sở dữ liệu**
- **Phụ thuộc hàm**
- **Chuẩn hóa dữ liệu**
- **Ngôn ngữ định nghĩa và thao tác dữ liệu**

# Học liệu

- [1] Nguyễn Quỳnh Chi, “**Cơ sở dữ liệu**”, Học viện Công nghệ Bưu chính Viễn thông, 2014.
- [2] Phạm Thế Quế, “**Giáo trình Cơ sở dữ liệu: Lý thuyết và thực hành**”, Nhà xuất bản Bưu điện, Hà Nội, 2004.
- [3] Hector Garcia-Molina, Jeffrey D. Ullman, Jenifer Widom. “**Database Systems: The complete book**”, Pearson Prentice Hall, Upper Saddle River, NJ 07458.

# Tổ chức môn học

- Lý thuyết: 32 tiết (kiểm tra giữa kỳ 2 tiết)
- Chữa bài tập: 8 tiết
- Thực hành: 4 tiết
- Tự học 1 tiết

# Đánh giá

- Tham gia học tập trên lớp: 10%
- Bài tập lớn theo nhóm / bài tập về nhà: 20%
- Kiểm tra giữa kỳ: 10%
- Thi cuối kỳ: 60%

# Bài tập lớn

- Thiết kế cài đặt một hệ cơ sở dữ liệu:
  - Xây dựng hệ CSDL đầy đủ như yêu cầu
  - Thiết kế đúng
  - Biết sử dụng một hệ quản trị CSDL
  - Cài đặt các chức năng cơ bản của hệ thống: nhập dữ liệu, truy vấn dữ liệu, tìm kiếm dữ liệu, báo cáo.
  - Viết báo cáo mô tả các công việc nhóm đã thực hiện một cách chi tiết.

# **CHƯƠNG 1.**

# **KHÁI NIỆM CƠ BẢN VỀ**

# **CƠ SỞ DỮ LIỆU**

**(Phần 1)**

- Phan thị Hà-0948672246

1.Nguyễn Lê Trúc Quỳnh

0329484673

[nltquynh875@gmail.com](mailto:nltquynh875@gmail.com)

2.Ngô Minh Quang

0359693129

[ngominhquang12a2nl@gmail.com](mailto:ngominhquang12a2nl@gmail.com)

3.Vũ Thị Ngọc Lan

0889147911

[vuthingoclan05@gmail.com](mailto:vuthingoclan05@gmail.com)

4.Lê Hoàng Long

0949404106

lehoanglonght12@gmail.com

# **KHÁI NIỆM CƠ BẢN VỀ CSDL**

- ❖ Các khái niệm cơ bản: Cơ sở dữ liệu, Hệ quản trị CSDL, Hệ cơ sở dữ liệu
- ❖ Các hệ CSDL truyền thống
- ❖ Các thành phần của một hệ quản trị CSDL
- ❖ Sự cần thiết của việc thiết kế CSDL
- ❖ Các vai trò trong môi trường CSDL
- ❖ Mô hình trùu tượng 3 lớp
- ❖ Các ngôn ngữ cơ sở dữ liệu
- ❖ Phân loại các hệ CSDL

# KHÁI NIỆM VỀ CSDL

- ❖ Theo nhận thức chung nhất, **cơ sở dữ liệu** (database) đơn giản là một tập thông tin (dữ liệu) có liên quan đến nhau.  
=> định nghĩa này rất mơ hồ, vì theo đây, có thể coi một trang văn bản là một CSDL.
- ❖ Khái niệm “**dữ liệu**” trong CSDL có thể bao gồm một phạm vi rất rộng các đối tượng: chữ số, văn bản, đồ họa, video,...
- ❖ Định nghĩa cụ thể hơn của một CSDL bao gồm một tập các đặc tính không tương minh được xem xét cùng nhau để định nghĩa một CSDL.

# KHÁI NIỆM VỀ CSDL (cont.)

- ❖ CSDL thể hiện các khía cạnh khác nhau của thế giới thực.
- ❖ CSDL được coi là một tập dữ liệu gắn kết logic với nhau. Các dữ liệu ngẫu nhiên không được coi là một CSDL (mặc dù chúng là những ngoại lệ).
- ❖ Một CSDL được thiết kế, xây dựng và sử dụng cho một số mục đích cụ thể. Nó được sử dụng bởi một tập người dùng và ứng dụng cụ thể ngay từ khi mới thiết kế.

# HỆ QUẢN TRỊ CSDL

- ❖ ***Hệ quản trị CSDL*** (DBMS – Database management system) là một hệ thống phần mềm cho phép tạo lập CSDL và điều khiển mọi truy nhập đến CSDL đó.
- ❖ Các đặc điểm (chức năng) quan trọng của một hệ quản trị CSDL:
  1. Cho phép người dùng **tạo mới CSDL**, thông qua ngôn ngữ định nghĩa dữ liệu (DDLs – Data Definition Languages).
  2. Cho phép người dùng **truy vấn cơ sở dữ liệu**, thông qua ngôn ngữ thao tác dữ liệu (DMLs – Data Manipulation Languages).
  3. **Hỗ trợ lưu trữ số lượng lớn dữ liệu**, thường lên tới hàng Gigabytes hoặc nhiều hơn, **trong một thời gian dài**. Duy trì tính bảo mật và tính toàn vẹn trong quá trình xử lý.
  4. **Kiểm soát truy nhập dữ liệu** từ nhiều người dùng tại cùng một thời điểm.

# HỆ CƠ SỞ DỮ LIỆU

- ❖ Một CSDL được quản lý bởi một hệ quản trị CSDL thường được gọi là một **hệ cơ sở dữ liệu**.
- ❖ Khi làm việc với Hệ CSDL: gồm 4 thành phần:
  1. **CSDL hợp nhất**: có 2 tính chất là tối thiểu hóa dư thừa và được chia sẻ.
  2. **Người dùng**: là những người có nhu cầu truy nhập vào CSDL (người dùng cuối, người viết chương trình ứng dụng, người quản trị CSDL).
  3. **Phần mềm hệ quản trị CSDL**.
  4. **Phần cứng**: gồm các thiết bị nhớ thứ cấp được sử dụng để lưu trữ CSDL.

# CÁC HỆ CSDL TRUYỀN THÔNG

- ❖ Hệ CSDL thương mại đầu tiên xuất hiện vào những năm 1960. Đó là các hệ thống lưu trữ theo kiểu tệp truyền thống.
- ❖ Xét theo 4 đặc tính của hệ quản trị CSDL, các hệ thống tệp:
  - Cung cấp đặc tính (3), tuy nhiên, không hoặc cung cấp rất ít đặc tính (4).
  - Không hỗ trợ trực tiếp đặc tính (2), ví dụ: không hỗ trợ ngôn ngữ truy vấn.
  - Không hỗ trợ đặc tính (1). Chỉ tạo cấu trúc thư mục cho các tệp. Việc hỗ trợ cho các lược đồ rất hạn chế.
- ❖ Một số hệ CSDL truyền thống quan trọng hơn, trong đó dữ liệu được chia nhỏ thành các mục, các truy vấn và sửa đổi có thể được thực hiện. Ví dụ: hệ thống bán vé máy bay hoặc hệ thống ngân hàng.

# CÁC HỆ CSDL TRUYỀN THÔNG (cont.)

- ❖ Phát triển vượt bậc của các hệ CSDL được đề xuất bởi Codd vào năm 1970.
  - ❖ CSDL được biểu diễn dưới dạng các bảng (các quan hệ)
  - ❖ Cấu trúc dữ liệu phức tạp cho phép đáp ứng nhanh các truy vấn. Người dùng không cần biết đến cấu trúc lưu trữ dữ liệu.
  - ❖ Các truy vấn có thể được thể hiện bởi một ngôn ngữ bậc cao, làm tăng hiệu suất cho những người lập trình CSDL.

# HỆ THỐNG NGÀY CÀNG NHỎ

## ❖ Trước đây:

- Hệ quản trị CSDL là hệ thống rất lớn, có giá thành cao và chạy trên các máy tính mainframe.
- Kích cỡ lưu trữ dữ liệu rất lớn nên cần các bộ lưu trữ lớn.

## ❖ Ngày nay:

- Do công nghệ phát triển, một gigabyte có thể được lưu trữ trên một đĩa đơn. Và các hệ quản trị CSDL có thể chạy trên một máy tính cá nhân.

*=> Hệ thống ngày càng nhỏ dần theo thời gian do công nghệ điện tử ngày càng phát triển.*

*=> Hệ quản trị CSDL dựa trên mô hình quan hệ bắt đầu xuất hiện như một công cụ chung cho các ứng dụng máy tính.*

# DỮ LIỆU NGÀY CÀNG LỚN

- ❖ Ngày nay, một gigabyte không còn được coi là dữ liệu có kích cỡ lớn nữa. Các hệ cơ sở dữ liệu lớn phải chứa hàng Terabytes hoặc nhiều hơn.
- ❖ Khi bộ nhớ lưu trữ trở nên rẻ và sẵn hơn, con người thường có các lý do mới để lưu trữ nhiều dữ liệu hơn.  
*Ví dụ*, các cửa hàng bán lẻ thường lưu trữ tới terabytes (1 terabyte = 1000 gigabytes hoặc  $10^{12}$  bytes) thông tin về lịch sử giao dịch mua bán trong một khoảng thời gian rất dài.
- ❖ Ngoài dạng văn bản và số, dữ liệu còn có nhiều dạng khác như âm thanh, hình ảnh thường chiếm không gian lưu trữ rất lớn.

*Ví dụ*: một giờ của video sẽ chiếm một gigabyte; hay CSDL lưu trữ các hình ảnh vệ tinh sẽ chiếm nhiều petabytes dữ liệu (1 petabyte=1000 Terabytes hay  $10^{15}$  bytes).

**=> Xu hướng hiện nay là dữ liệu ngày càng lớn.**

# DỮ LIỆU NGÀY CÀNG LỚN (cont.)

- ❖ Để xử lý được các CSDL lớn đòi hỏi nhiều công nghệ tiên tiến.
  - Các CSDL hầu như không bao giờ cho rằng “dữ liệu” sẽ vừa với bộ nhớ trong. Các hệ thống cũ thường chỉ có các thiết bị lưu trữ thứ cấp dưới dạng các đĩa từ (công nghệ tương tự).
  - CSDL hiện đại được lưu trữ trên một mảng các ổ cứng (các thiết bị lưu trữ thứ cấp).
- ❖ Hai xu hướng cho phép các hệ CSDL có thể xử lý được khối lượng dữ liệu lớn một cách nhanh hơn là: **Lưu trữ mức độ cấp 3** và **Tính toán song song**.

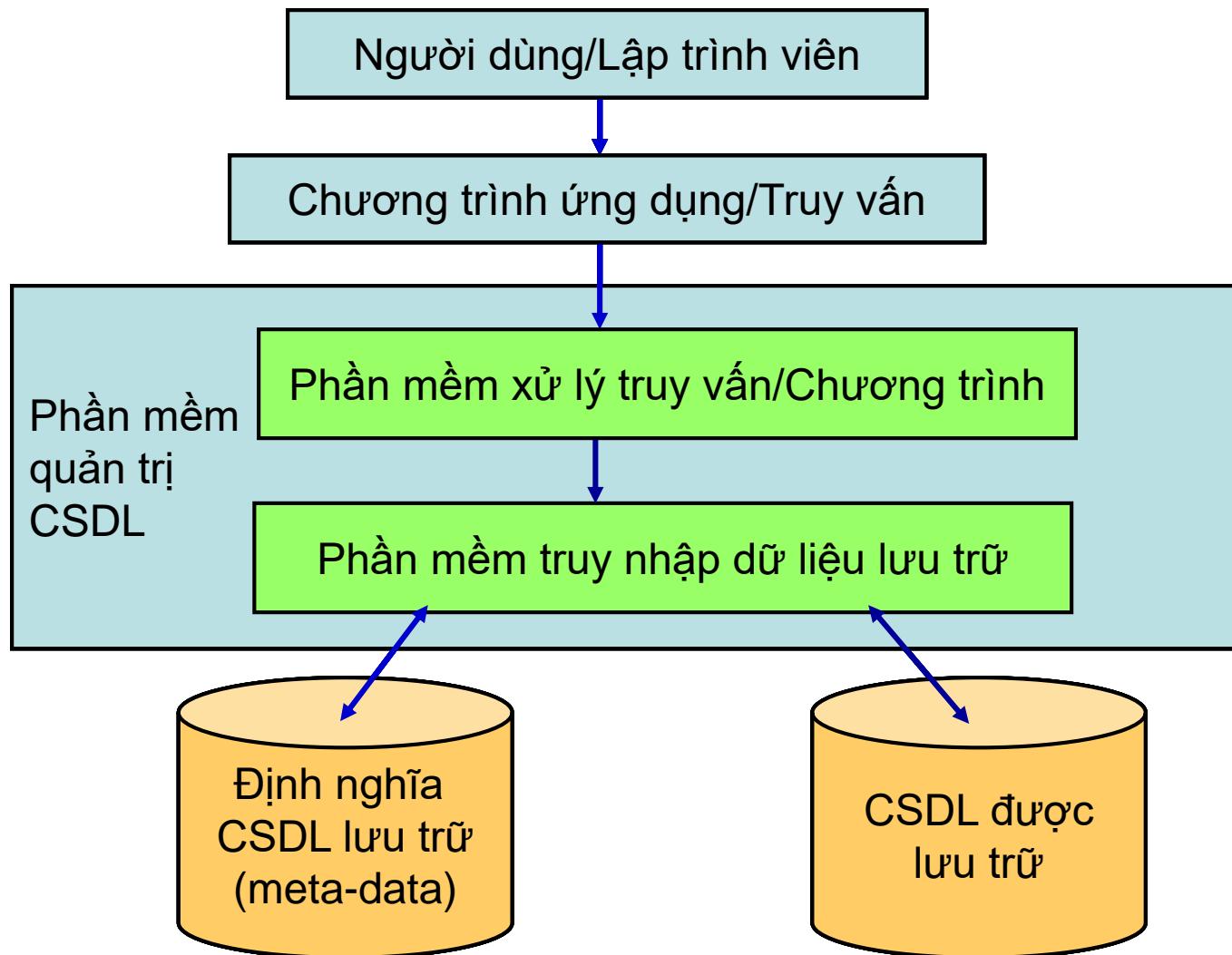
# LƯU TRỮ MỨC ĐỘ CẤP 3

- ❖ Các CSDL lớn hiện nay đòi hỏi nhiều hơn việc chỉ lưu trữ trên các ổ đĩa (cấp 2). **Các thiết bị cấp 3** có xu hướng lưu trữ theo đơn vị terabyte và có thời gian truy nhập dài hơn các ổ đĩa truyền thống.
  - Thời gian truy nhập của một đĩa truyền thống là khoảng 10-20 msec. Trong khi đó của thiết bị cấp 3 là vài giây.
  - Các thiết bị cấp 3 liên quan tới việc chuyển một đối tượng mà trên đó dữ liệu được lưu trữ, tới một thiết bị đọc nào đó.
  - **Ví dụ:** Đĩa CDs là một phương tiện lưu trữ mức độ cấp 3.

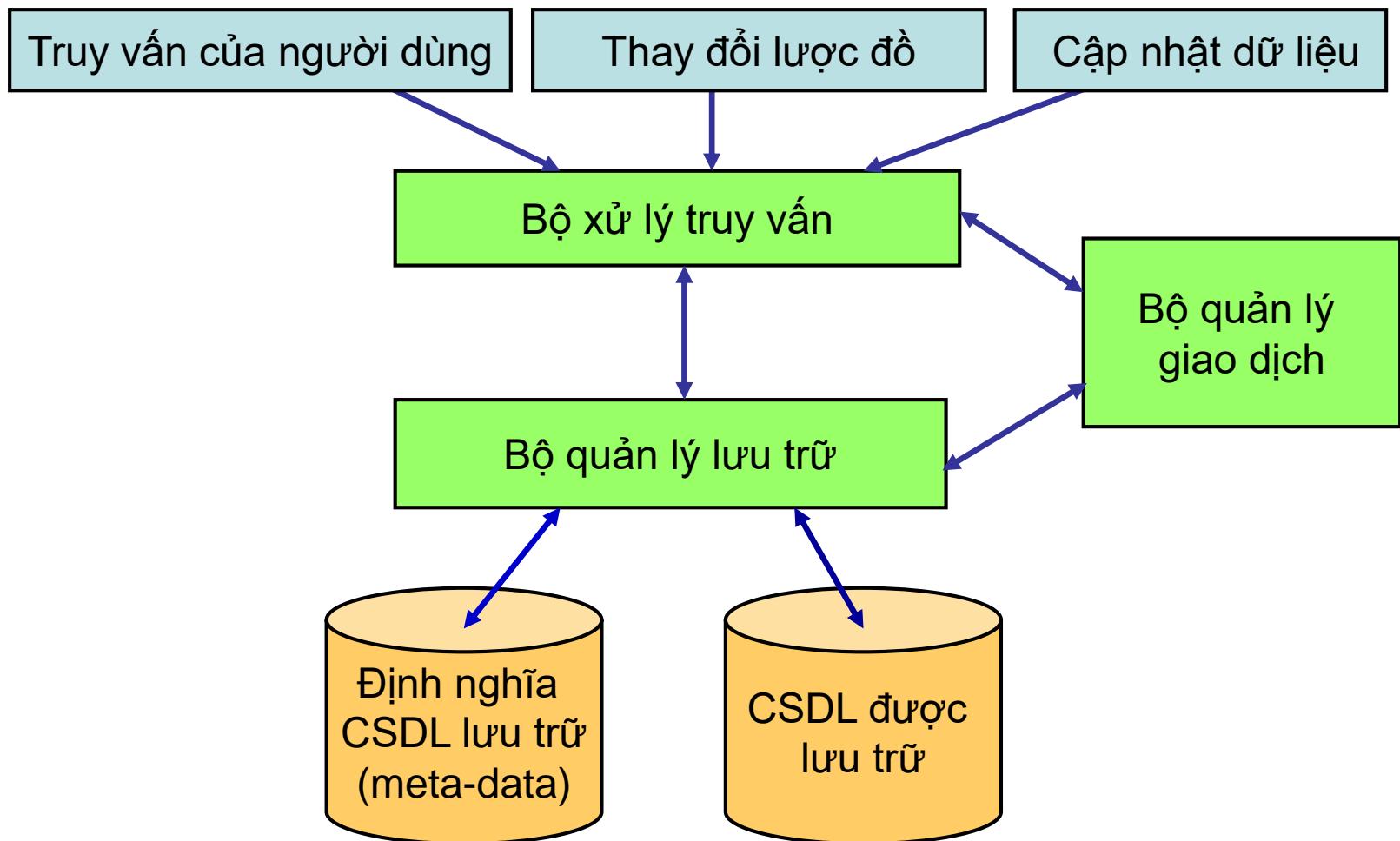
# TÍNH TOÁN SONG SONG

- ❖ Khả năng lưu trữ khối lượng dữ liệu khổng lồ là rất quan trọng nhưng nó sẽ ít được sử dụng nếu như không thể truy nhập vào khối dữ liệu đó một cách nhanh chóng => **Yêu cầu cải thiện tốc độ.**
- ❖ Trong CSDL hiện đại, việc cải thiện tốc độ được thực hiện bằng:
  - **Các cấu trúc chỉ mục**
  - **Cơ chế song song hóa** - liên quan tới cả song song hóa bộ vi xử lý và song song hóa bản thân dữ liệu.

# TỔNG QUAN CÁC THÀNH PHẦN CỦA MỘT HỆ CSDL



# CÁC THÀNH PHẦN CỦA MỘT DBMS (cont.)



*Kiến trúc của một hệ quản trị CSDL*

# CÁC THÀNH PHẦN CỦA MỘT DBMS (cont.)

## ❖ *CSDL lưu trữ và meta-data:*

- CSDL được lưu trữ tại thiết bị nhớ thứ cấp hoặc cấp 3.
- Meta-data (**siêu dữ liệu**) là dữ liệu về dữ liệu: Mô tả các thành phần dữ liệu của CSDL (vị trí tương đối của các trường trong bản ghi, thông tin về lược đồ, thông tin về chỉ mục, ...).
- Với mỗi CSDL, hệ quản trị CSDL có thể duy trì nhiều **chỉ mục** khác nhau được thiết kế để cung cấp truy nhập nhanh tới dữ liệu ngẫu nhiên.
- Trong các CSDL hiện đại, hầu hết các chỉ mục được biểu diễn dưới dạng **B-tree (cây tìm kiếm nhị phân)**. Các B-tree có xu hướng “ngắn và béo” giúp truy nhập nhanh từ gốc đến lá.

# CÁC THÀNH PHẦN CỦA MỘT DBMS (cont.)

- ❖ **Bộ quản lý lưu trữ:** Trong các hệ CSDL đơn giản, bộ quản lý lưu trữ chỉ như là hệ thống tệp trong hệ điều hành. Với các hệ thống lớn hơn, để hiệu quả, hệ quản trị CSDL thường quản lý việc lưu trữ trực tiếp trên ổ đĩa.

Bộ quản lý lưu trữ có 2 thành phần cơ bản:

- **Bộ quản lý tệp:** Lưu vị trí các tệp trên ổ đĩa và lấy ra được khôi hoặc các khối chứa tệp theo yêu cầu từ bộ quản lý vùng đệm.
- **Bộ quản lý vùng đệm:** Quản lý bộ nhớ chính. Lấy các khối dữ liệu từ ổ đĩa, qua bộ quản lý tệp, và chọn một trang trong bộ nhớ chính để lưu trữ. Thuật toán tạo trang sẽ xác định trang sẽ tồn tại bao lâu trong bộ nhớ chính.

# CÁC THÀNH PHẦN CỦA MỘT DBMS (cont.)

- ❖ **Bộ xử lý truy vấn:** Biến đổi một câu truy vấn hoặc một thao tác CSDL, đang được biểu diễn tại một mức rất cao (ví dụ, ngôn ngữ SQL), thành một chuỗi các yêu cầu đối với dữ liệu được lưu trữ trong CSDL.

Phần phức tạp nhất của bộ xử lý truy vấn là **tối ưu hóa truy vấn**, nghĩa là chọn ra được chiến lược tốt nhất để thực thi truy vấn.

# CÁC THÀNH PHẦN CỦA MỘT DBMS (cont.)

- ❖ **Bộ quản lý giao dịch:** Giao dịch là một tập các thao tác được xử lý như một đơn vị không chia cắt được. Để đảm bảo được tính chất này, bộ quản lý giao dịch phải đảm bảo 4 tính chất (được gọi là **thuộc tính ACID**):
  - **Tính nguyên tố (Atomicity):** tất cả các thao tác của giao dịch được thực hiện hoặc không thao tác nào được thực hiện.
  - **Tính nhất quán (Consistency):** các thao tác phải đảm bảo tính nhất quán của CSDL.
  - **Tính biệt lập (Isolation):** các giao dịch đồng thời phải được tách riêng biệt nhau.
  - **Tính duy trì (Durability):** những thay đổi tới CSDL bởi một giao dịch sẽ không bị mất đi ngay cả khi hệ thống có lỗi ngay sau khi giao dịch hoàn thành.

# CÁC THÀNH PHẦN CỦA MỘT DBMS (cont.)

## ❖ *Ba kiểu thao tác:*

- **Truy vấn của người dùng:** là các thao tác hỏi đáp về dữ liệu được lưu trữ trong CSDL. Chúng được sinh ra theo 2 cách: (1) Thông qua giao diện truy vấn chung, (2) Thông qua giao diện chương trình ứng dụng.
- **Cập nhật dữ liệu:** là các thao tác thay đổi dữ liệu, như thêm, sửa, xóa dữ liệu trong CSDL. Chúng cũng được sinh ra theo 2 cách (1) và (2) như trên.
- **Thay đổi lược đồ:** là các lệnh được sinh ra bởi người dùng được cấp phép, thường là người quản trị CSDL.

# DỮ LIỆU VÀ THÔNG TIN

- ❖ Xét ví dụ về dữ liệu gồm các con số như sau:

**Data:** 0    11,500

      5    12,300

      10    12,800

      15    10,455

      20    12,200

      25    13,900

      30    14,220

- ❖ Thể hiện dưới dạng “thô” như trên, dữ liệu có rất ít ý nghĩa. Nó đơn giản là một cặp danh sách các số nguyên. Không có ngữ cảnh làm nền cho dữ liệu.

# DỮ LIỆU VÀ THÔNG TIN (cont.)

- ❖ **Chuyển dữ liệu thành một dạng có ý nghĩa hơn:**
  - Quá trình xử lý cơ bản là đặt dữ liệu vào trong ngũ cành (*thường được thực hiện bằng cách thêm dữ liệu vào. Mặc dù những dữ liệu thêm vào thực sự là siêu dữ liệu*).
- ❖ **Dữ liệu bắt đầu có ý nghĩa hơn như sau:**

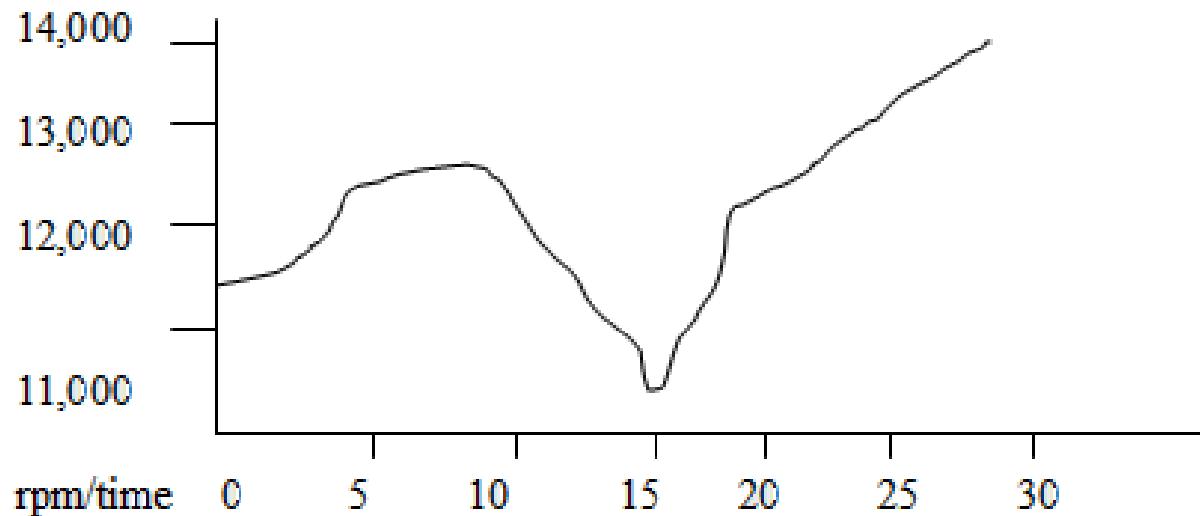
Thông tin: Dữ liệu Engine RPM: Roebling Road 10/4/2003 – Yamaha Heavy

Lap 12: time rpm

0	11,500
5	12,300
10	12,800
15	10,455
20	12,200
25	13,900
30	14,220

# DỮ LIỆU VÀ THÔNG TIN (cont.)

- ❖ *Cùng với dữ liệu trên, xem xét quá trình xử lý chuyển dữ liệu thành dạng đồ thị:*



Đồ thị: Một phần Lap 12 - Roebling Road 10/4/2003 – Yamaha Heavy

# DỮ LIỆU DẪN XUẤT VÀ DỮ LIỆU VẬT LÝ

- ❖ **Dữ liệu vật lý:** là những dữ liệu có thực, được nhập vào trong CSDL.
- ❖ **Dữ liệu dẫn xuất:** Là những dữ liệu được tính toán từ những dữ liệu nằm trong CSDL.
- ❖ Phụ thuộc vào mức độ phức tạp của chương trình ứng dụng và hệ quản trị CSDL, khối dữ liệu dẫn xuất có thể lớn hơn rất nhiều khối dữ liệu vật lý.
- ❖ Cần nhắc khi nào dữ liệu dẫn xuất trở thành dữ liệu vật lý?  
**Ví dụ:** CSDL lưu thông tin về Sinh viên, trong đó lưu điểm thi của SV. Giá trị trung bình điểm thi của SV có cần lưu trong CSDL hay sẽ được tính toán khi cần?

# **CHƯƠNG 1.**

# **KHÁI NIỆM CƠ BẢN VỀ**

# **CƠ SỞ DỮ LIỆU**

**(Phần 2)\*\***

# KHÁI NIỆM CƠ BẢN VỀ CSDL

- ❖ Các khái niệm cơ bản: Cơ sở dữ liệu, Hệ quản trị CSDL, Hệ cơ sở dữ liệu
- ❖ Các hệ CSDL truyền thống
- ❖ Các thành phần của một hệ quản trị CSDL
- ❖ **Sự cần thiết của việc thiết kế CSDL**
- ❖ Các vai trò trong môi trường CSDL
- ❖ Mô hình trừu tượng 3 lớp
- ❖ Các ngôn ngữ cơ sở dữ liệu
- ❖ Phân loại các hệ CSDL

# SỰ CẦN THIẾT CỦA VIỆC THIẾT KẾ CSDL

- ❖ Đối với các hệ thống phục vụ người dùng, hoạt động thiết kế CSDL là cần thiết.
- ❖ Một CSDL được thiết kế không cẩn thận có thể tạo ra nhiều lỗi, dẫn đến những quyết định không đúng đắn, làm ảnh hưởng nghiêm trọng đến tổ chức.
- ❖ Mặt khác, một CSDL được thiết kế tốt, theo một cách hiệu quả, sẽ cung cấp những thông tin chính xác hỗ trợ cho quá trình tạo quyết định đúng đắn, dẫn tới thành công.

# CÁC VAI TRÒ TRONG MÔI TRƯỜNG CSDL (VAI TRÒ TÁC NHÂN THAM GIA VÀO MÔI TRƯỜNG CSDL)

- ❖ ***Người quản trị dữ liệu*** (DA – Data Administrator): có trách nhiệm quản lý tài nguyên dữ liệu, bao gồm lập kế hoạch cho CSDL, phát triển và duy trì các chuẩn, chính sách và thủ tục, và thiết kế CSDL mức khái niệm/logic.
- ❖ ***Người quản trị CSDL*** (DBA – Database Administrator): có trách nhiệm với việc lưu trữ vật lý CSDL, bao gồm thiết kế và cài đặt CSDL vật lý, kiểm soát bảo mật và toàn vẹn dữ liệu, duy trì hệ điều hành, và đảm bảo thỏa mãn hiệu năng cho các ứng dụng người dùng.

=> Vai trò của DBA liên quan đến nhiều đặc tính kỹ thuật hơn DA.

# CÁC VAI TRÒ TRONG MÔI TRƯỜNG CSDL (cont.)

- ❖ **Người thiết kế CSDL:** trong các dự án thiết kế CSDL lớn, cần phân biệt 2 loại thiết kế:
  - **Người thiết kế CSDL logic** liên quan tới việc xác định CSDL (các thực thể và thuộc tính), các mối quan hệ giữa dữ liệu, và các ràng buộc đối với dữ liệu sẽ được lưu trữ trong CSDL.
  - **Người thiết kế CSDL vật lý** phụ thuộc nhiều vào hệ quản trị CSDL đích, và có thể có nhiều cách để cài đặt CSDL. Người thiết kế CSDL vật lý phải có hiểu biết đầy đủ về tính năng của hệ quản trị CSDL đích.
- ❖ **Người phát triển ứng dụng:** Có trách nhiệm xây dựng chương trình ứng dụng cung cấp các chức năng cần thiết cho người dùng cuối, sau khi CSDL đã được cài đặt.

# CÁC VAI TRÒ TRONG MÔI TRƯỜNG CSDL (cont.)

- ❖ ***Người dùng cuối:*** là “khách hàng” của CSDL, và có thể được phân thành 2 nhóm dựa theo cách mà họ sử dụng hệ thống:
  - Nhóm người dùng không biết đến khái niệm CSDL hoặc hệ quản trị CSDL: Truy nhập CSDL thông qua chương trình ứng dụng được viết riêng biệt, giúp cho các thao tác của người dùng đơn giản nhất có thể.
  - Nhóm người dùng nhận biết được cấu trúc CSDL và các phương tiện được cung cấp bởi hệ quản trị CSDL: Thường dùng các ngôn ngữ truy vấn bậc cao như SQL để thực hiện những thao tác được yêu cầu và thậm chí có thể viết những chương trình ứng dụng để phục vụ cho mục đích riêng.

# **ƯU ĐIỂM CỦA CÁC HỆ QUẢN TRỊ CSDL**

Kiểm soát dư thừa dữ liệu	Kinh tế khi tăng số lượng dữ liệu
Đảm bảo tính nhất quán dữ liệu	Cân bằng những yêu cầu bị xung đột
Thêm thông tin từ cùng dữ liệu	Cải tiến việc truy nhập dữ liệu
Hỗ trợ tính sẵn dùng của khối lượng lớn dữ liệu	Tăng hiệu suất của hệ thống
Chia sẻ dữ liệu	Cải tiến việc bảo trì
Cải tiến sự toàn vẹn dữ liệu	Tăng xử lý đồng thời
Cải tiến tính bảo mật dữ liệu	Cải tiến sao lưu và khôi phục dữ liệu
Ép buộc chuẩn hóa dữ liệu	Cải tiến đáp ứng truy vấn

# NHƯỢC ĐIỂM CỦA CÁC HỆ QUẢN TRỊ CSDL

Phức tạp

Kích thước lớn

Chi phí mua và bảo trì

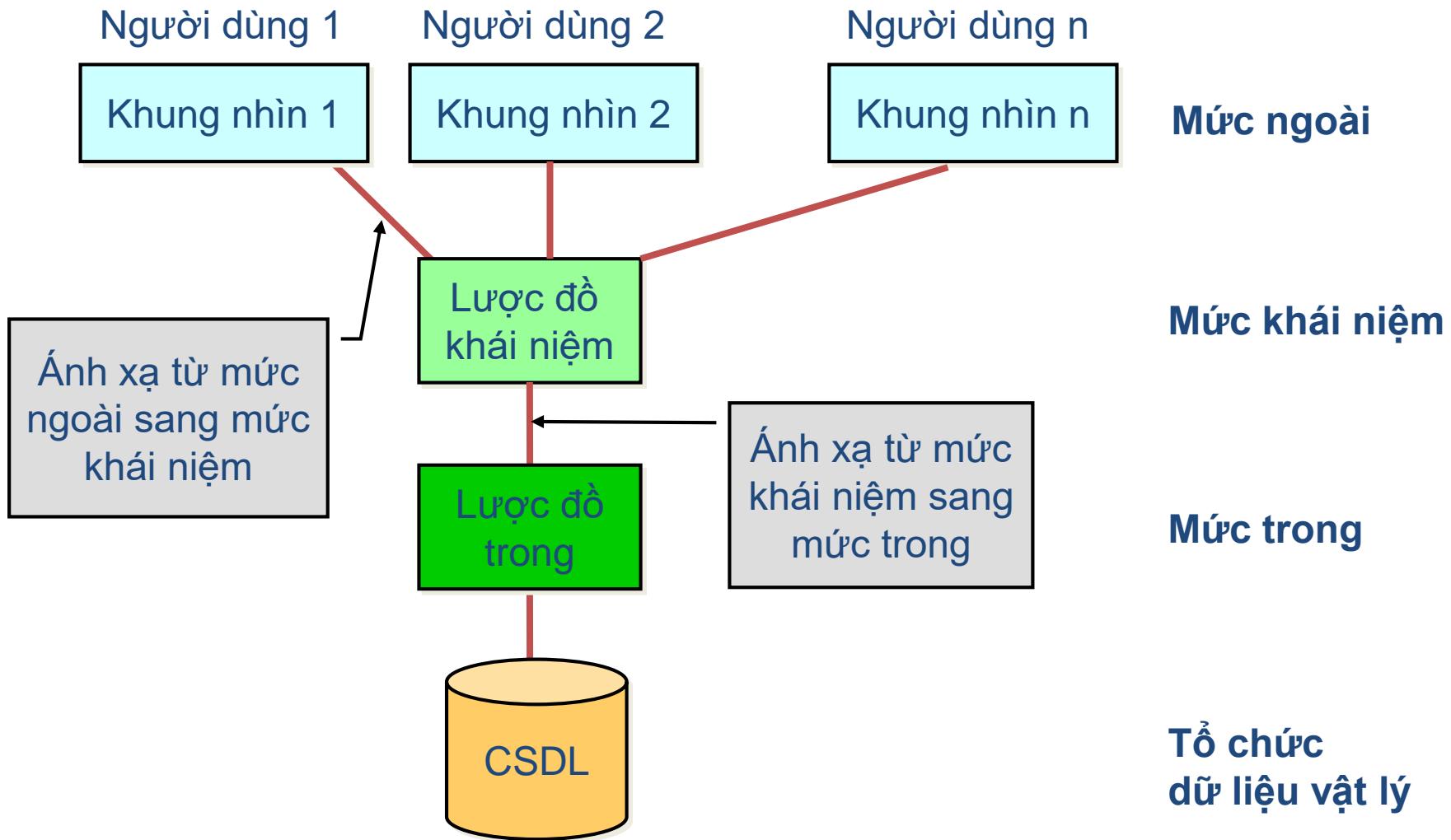
Thêm giá thành cho các phần cứng hỗ trợ

Chi phí chuyển đổi hệ thống

Hạn chế hiệu năng  
(trong một số trường hợp cụ thể)

Ảnh hưởng lớn khi có lỗi

# MÔ HÌNH TRƯU TƯỢNG 3 LỚP



# MỨC NGOÀI

- ❖ **Mức ngoài** là khung nhìn của người sử dụng CSDL, mô tả phần CSDL tương ứng với người dùng đó.
- ❖ Mỗi người dùng có một khung nhìn khác nhau về thế giới thực, được biểu diễn theo cách thân thiện với họ trong CSDL.  
=> **Mức ngoài bao gồm một số khung nhìn khác nhau về CSDL.**
- ❖ Mức ngoài chỉ bao gồm các thực thể, thuộc tính, quan hệ trong thế giới thực mà người dùng quan tâm. (Những thực thể, thuộc tính, quan hệ khác có thể tồn tại, nhưng người dùng sẽ không cần biết đến sự tồn tại của chúng.)

## MỨC NGOÀI (cont.)

- ❖ Thường xảy ra trường hợp các khung nhìn ngoài khác nhau sẽ có những biểu diễn khác nhau đối với cùng dữ liệu.

**Ví dụ:** Một khung nhìn có thể thể hiện ngày tháng dưới dạng (ngày, tháng, năm), trong khi một khung nhìn khác lại thể hiện dưới dạng (tháng, ngày, năm).

- ❖ Một số khung nhìn có thể bao gồm các dữ liệu dẫn xuất. Các dữ liệu này sẽ không được lưu thực trong CSDL mà chúng sẽ được tạo ra khi cần.

**Ví dụ:** Một khung nhìn cần biết tuổi của một người. Tuy nhiên, dữ liệu về tuổi không cần thiết phải được lưu trong CSDL vì nó được cập nhật hàng ngày: Tuổi sẽ được tính dựa theo dữ liệu ngày sinh của người đó và ngày hiện tại trong hệ thống.

# MỨC KHÁI NIỆM

- ❖ **Mức khái niệm** là khung nhìn của người thiết kế CSDL, mô tả dữ liệu nào được lưu trong CSDL và mối quan hệ giữa chúng.
- ❖ Người quản trị CSDL nhìn thấy toàn bộ cấu trúc logic của CSDL. Cấu trúc này thể hiện khung nhìn hoàn chỉnh về những yêu cầu dữ liệu của tổ chức mà không liên quan tới bất kỳ phương thức lưu trữ nào.

## MỨC KHÁI NIỆM (cont.)

- ❖ Mức khái niệm hỗ trợ từng khung nhìn ngoài, trong đó bất kỳ dữ liệu nào chuyển tới người dùng đều phải được lưu lại hoặc được sinh ra từ mức khái niệm.
- ❖ Mức khái niệm không liên quan tới bất kỳ thông tin nào về việc lưu trữ dữ liệu.

**Ví dụ:** Một thực thể được định nghĩa ở dạng số nguyên tại mức khái niệm, nhưng số byte chứa số nguyên đó sẽ không được chỉ ra ở đây.

# MỨC TRONG

- ❖ ***Mức trong*** thể hiện biểu diễn về mặt vật lý của CSDL trong máy tính, mô tả cách thức lưu trữ dữ liệu trong CSDL.
- ❖ Mô tả cài đặt vật lý cần thiết để đạt được tối ưu về thời gian thực thi và việc sử dụng không gian lưu trữ.
- ❖ Bao gồm các cấu trúc dữ liệu và tổ chức tệp lưu trữ dữ liệu trong các thiết bị nhớ.
- ❖ Có giao diện với các phương thức truy nhập của hệ điều hành (các kỹ thuật quản lý tệp để lưu trữ và lấy các bản ghi dữ liệu) để đưa dữ liệu vào các thiết bị nhớ, xây dựng chỉ mục, lấy dữ liệu, ...

# MỨC VẬT LÝ

- ❖ **Mức vật lý** nằm dưới mức trung, được quản lý bởi hệ điều hành dưới chỉ dẫn của hệ quản trị CSDL.
- ❖ Chức năng của hệ quản trị CSDL và hệ điều hành tại mức vật lý là không có ranh giới rõ ràng và thay đổi từ hệ thống này sang hệ thống khác.
- ❖ Một số hệ quản trị CSDL tận dụng ưu điểm của nhiều phương thức truy nhập của hệ điều hành, trong khi một số hệ quản trị CSDL khác lại chỉ sử dụng những phương thức cơ bản và tự tạo ra tổ chức tệp của riêng chúng.
- ❖ Mức vật lý dưới hệ quản trị CSDL gồm các mục chỉ được biết đến bởi hệ điều hành. **Ví dụ:** làm thế nào để tạo một chuỗi các thực thi? Liệu các trường của các bản ghi trong CSDL có được lưu trữ bởi các byte liền nhau trong ổ đĩa hay không?

# LƯỢC ĐỒ, ÁNH XẠ VÀ THỂ HIỆN

- ❖ **Lược đồ CSDL** là thiết kế tổng thể của CSDL.

Có 3 loại lược đồ khác nhau được định nghĩa theo các mức trừu tượng của mô hình 3 lớp:

- Trên cùng có rất nhiều **lược đồ ngoài** (hay gọi là **lược đồ con**) tương ứng với các khung nhìn dữ liệu khác nhau.
- Tại mức khái niệm có một **lược đồ khái niệm**, mô tả tất cả các thực thể, thuộc tính và các quan hệ cùng với những ràng buộc toàn vẹn.
- Mức trừu tượng thấp nhất có một **lược đồ trong**, mô tả toàn bộ mô hình trong, gồm định nghĩa các bản ghi được lưu trữ, phương thức biểu diễn, ...

## LƯỢC ĐỒ, ÁNH XẠ VÀ THỂ HIỆN (cont.)

=> Hệ quản trị CSDL có trách nhiệm ánh xạ các loại lược đồ với nhau.

Hệ quản trị CSDL phải kiểm tra được tính nhất quán của các lược đồ; nghĩa là phải kiểm tra xem mỗi lược đồ ngoài có được suy ra từ lược đồ khái niệm hay không, và sử dụng các thông tin trong lược đồ khái niệm để ánh xạ giữa các lược đồ ngoài và lược đồ trong.

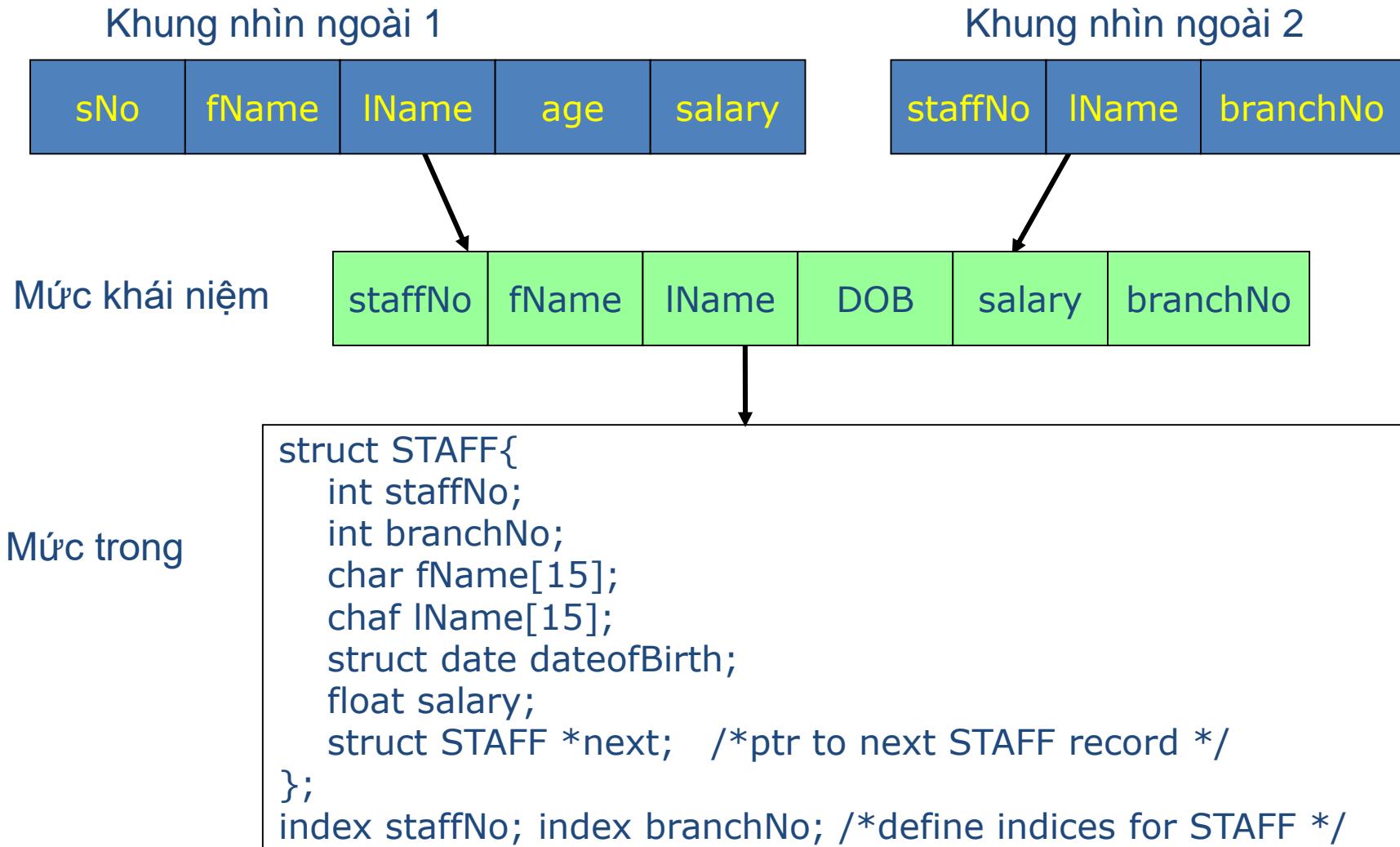
## LƯỢC ĐỒ, ÁNH XẠ VÀ THỂ HIỆN (cont.)

- ❖ Lược đồ khái niệm liên kết với lược đồ thông qua ánh xạ mức khái niệm/mức trong.
  - => Ánh xạ này giúp cho hệ quản trị CSDL tìm ra được bản ghi thực tế hoặc kết nối các bản ghi trong bộ lưu trữ vật lý để tạo bản ghi logic trong lược đồ khái niệm, cùng với một số ràng buộc được gắn với các thao tác cho bản ghi logic đó.
- ❖ Lược đồ ngoài liên kết với lược đồ khái niệm thông qua ánh xạ mức ngoài/mức khái niệm.
  - => Ánh xạ này giúp cho hệ quản trị CSDL ánh xạ các tên trong khung nhìn của người dùng thành các phần có liên quan trong lược đồ khái niệm.

## LƯỢC ĐỒ, ÁNH XẠ VÀ THỂ HIỆN (cont.)

- ❖ Các cơ sở dữ liệu thay đổi theo thời gian khi thông tin được thêm vào hay bị xóa đi.  
=> *Tập hợp các thông tin được lưu trữ trong CSDL tại một thời điểm đặc biệt được gọi là một **thể hiện** của cơ sở dữ liệu đó.*
- Ví dụ: Giá trị của một bản ghi tại một thời điểm được gọi là một thể hiện của CSDL.

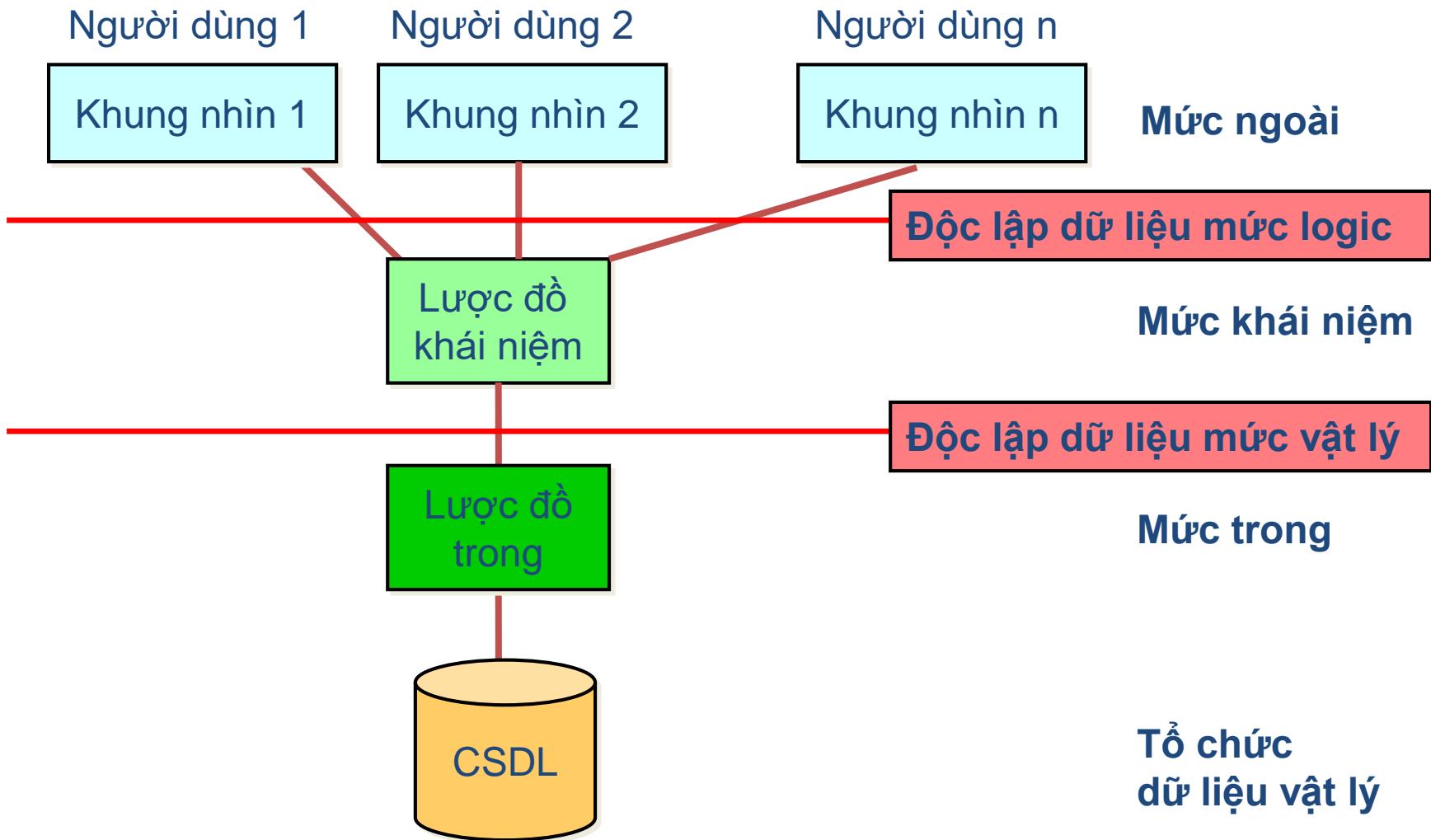
# VÍ DỤ CHUYỂN ĐỔI LƯỢC ĐỒ GIỮA 3 MỨC TRÙU TƯỢNG



# TÍNH ĐỘC LẬP DỮ LIỆU

- ❖ Một trong những mục tiêu chính của kiến trúc 3 lớp là cung cấp **tính độc lập dữ liệu**, nghĩa là các mức cao hơn không bị ảnh hưởng bởi bất kỳ sự thay đổi nào từ các mức thấp hơn.
- ❖ Có 2 loại độc lập dữ liệu:
  - **Độc lập dữ liệu mức logic (mức khái niệm)**: các lược đồ ngoài không bị ảnh hưởng bởi sự thay đổi của lược đồ khái niệm.
  - **Độc lập dữ liệu mức vật lý**: lược đồ khái niệm không bị ảnh hưởng bởi sự thay đổi của lược đồ trong.

# TÍNH ĐỘC LẬP DỮ LIỆU (cont.)



# CÁC NGÔN NGỮ CSDL

- ❖ Một ngôn ngữ con dữ liệu bao gồm 2 phần: Một **ngôn ngữ định nghĩa dữ liệu (DDL - Data Definition Language)** và một **ngôn ngữ thao tác dữ liệu (DML - Data Manipulation Language)**.
- ❖ DDL được dùng để xác định lược đồ CSDL, và DML dùng để đọc và cập nhật CSDL.
- ❖ Các ngôn ngữ này được gọi là *ngôn ngữ con dữ liệu* vì chúng không bao gồm các cấu trúc lập trình cần thiết cho việc tính toán như là các cấu trúc điều khiển hoặc câu lệnh lặp (được cung cấp bởi các ngôn ngữ lập trình bậc cao).

## CÁC NGÔN NGỮ CSDL (cont.)

- ❖ Hầu hết các hệ quản trị CSDL đều có một môi trường cho phép nhúng các ngôn ngữ con dữ liệu vào trong một ngôn ngữ lập trình bậc cao như COBOL, Pascal, C, C++, Java hay Visual Basic (được gọi là các *ngôn ngữ chủ*).
- ❖ Nhiều ngôn ngữ con dữ liệu cũng cung cấp một phiên bản tương tác hoặc không nhúng vào ngôn ngữ mà có thể được đưa vào trực tiếp từ một thiết bị đầu cuối.

# NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU (DDL)

- ❖ DDL là ngôn ngữ cho phép người quản trị CSDL hoặc người dùng mô tả và đặt tên các thực thể, thuộc tính và các quan hệ cần thiết cho ứng dụng, cùng với những ràng buộc về bảo mật và toàn vẹn liên quan.
- ❖ Kết quả của việc thực thi/biên dịch câu lệnh DDL là một tập các bảng được lưu trong các tệp đặc biệt, được gọi là **danh mục hệ thống** (system catalog) (hay **từ điển dữ liệu**/ **thư mục dữ liệu**).

# NGÔN NGỮ THAO TÁC DỮ LIỆU (DML)

- ❖ DML là ngôn ngữ cung cấp một tập các thao tác hỗ trợ cho các phép toán thao tác dữ liệu cơ bản trên dữ liệu được lưu trong CSDL.
- ❖ Các thao tác của DML bao gồm:
  - Chèn dữ liệu mới vào CSDL
  - Sửa đổi dữ liệu đã được lưu trữ trong CSDL
  - Lấy dữ liệu từ CSDL
  - Xóa dữ liệu trong CSDL
- ❖ Phần thao tác lấy dữ liệu ra được gọi là **ngôn ngữ truy vấn**.

# NGÔN NGỮ THAO TÁC DỮ LIỆU (cont.)

- ❖ Các DML được phân biệt bởi cấu trúc lấy dữ liệu bên trong nó, gồm hai loại chính: **có thủ tục** và **không thủ tục**.
  - **DMLs có thủ tục** là các ngôn ngữ trong đó người dùng có thông báo với hệ thống những dữ liệu nào cần và cách thức chính xác để lấy nó ra.
  - **DMLs không có thủ tục** là các ngôn ngữ trong đó người dùng chỉ thông báo cho hệ thống dữ liệu nào cần và hệ thống sẽ tự xác định cách thức lấy dữ liệu đó ra.
- ❖ DMLs có thủ tục thường được nhúng vào các ngôn ngữ lập trình bậc cao.
- ❖ DMLs có thủ tục có xu hướng tập trung vào từng bản ghi đơn còn DMLs không thủ tục có xu hướng thực hiện trên một tập các bản ghi.

# NGÔN NGỮ THẾ HỆ THỨ TƯ

- ❖ **Ngôn ngữ thế hệ thứ tư** là một ngôn ngữ lập trình rất nhanh. Những yêu cầu được thực hiện với hàng trăm dòng lệnh trong ngôn ngữ thế hệ thứ ba sẽ được thể hiện chỉ trong một vài dòng lệnh của ngôn ngữ thế hệ thứ tư.
- ❖ Ngôn ngữ thế hệ thứ ba là loại có thủ tục, còn ngôn ngữ thế hệ thứ tư là loại không có thủ tục.
- ❖ Ngôn ngữ thế hệ thứ tư gồm các ngôn ngữ làm việc trên bảng tính và trên cơ sở dữ liệu.

**Ví dụ:** Ngôn ngữ SQL.

# PHÂN LOẠI CÁC HỆ CSDL

*Có 2 loại kiến trúc CSDL:*

- ❖ Hệ cơ sở dữ liệu tập trung:
  - Hệ CSDL cá nhân
  - Hệ CSDL trung tâm
  - Hệ CSDL khách/chủ (client/server)
- ❖ Hệ cơ sở dữ liệu phân tán
  - Hệ CSDL phân tán thuần nhất
  - Hệ CSDL phân tán không thuần nhất

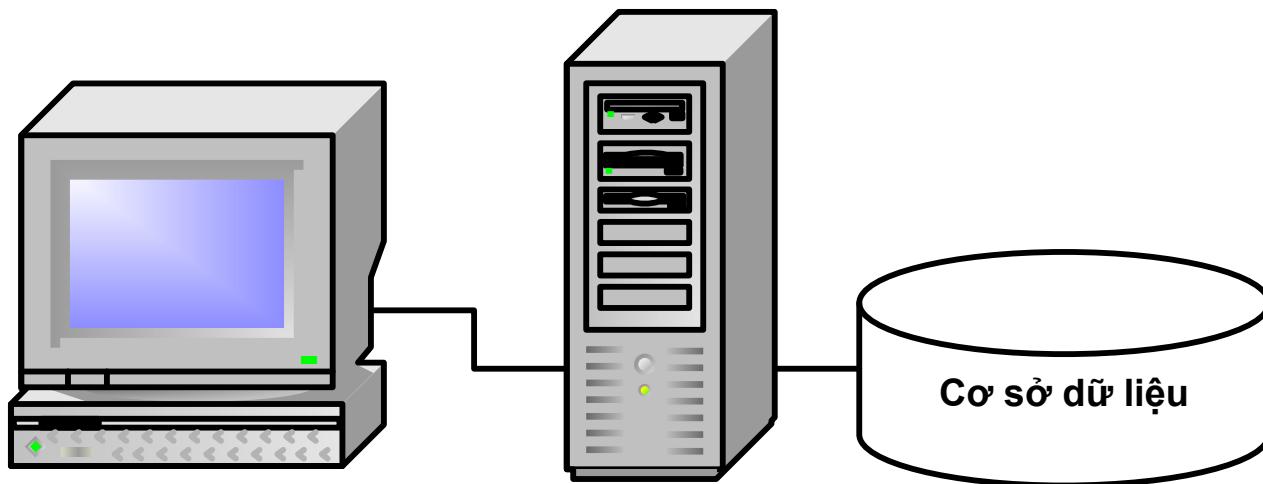
# HỆ CSDL TẬP TRUNG

- ❖ Là các hệ CSDL mà trong đó phần CSDL được lưu trữ tại một vị trí nhất định.
- ❖ Người dùng tại các trạm từ xa nói chung có thể truy nhập CSDL thông qua các công cụ truyền thông dữ liệu.
- ❖ Cung cấp một sự kiểm soát lớn đối với việc truy nhập và cập nhật dữ liệu.
- ❖ Dễ bị lỗi do phụ thuộc vào tính sẵn sàng của các tài nguyên.

# HỆ CSDL CÁ NHÂN

- ❖ Là hệ CSDL nhỏ, trong đó người quản trị CSDL chính là người viết chương trình ứng dụng, đồng thời cũng là người dùng cuối.
- ❖ Ứng dụng: trong các tổ chức nhỏ và vừa, [ví dụ](#): quản lý nhân sự ở một đơn vị hành chính.
- ❖ Việc phát triển và sử dụng các hệ CSDL cá nhân là khá đơn giản và dễ dàng.
- ❖ Có nguy cơ phải chịu rủi ro, vì CSDL chỉ được lưu trữ tại một trạm đơn lẻ.
- ❖ Dữ liệu khó được chia sẻ cho nhiều ứng dụng khác nhau.

# HỆ CSDL CÁ NHÂN (Cont.)



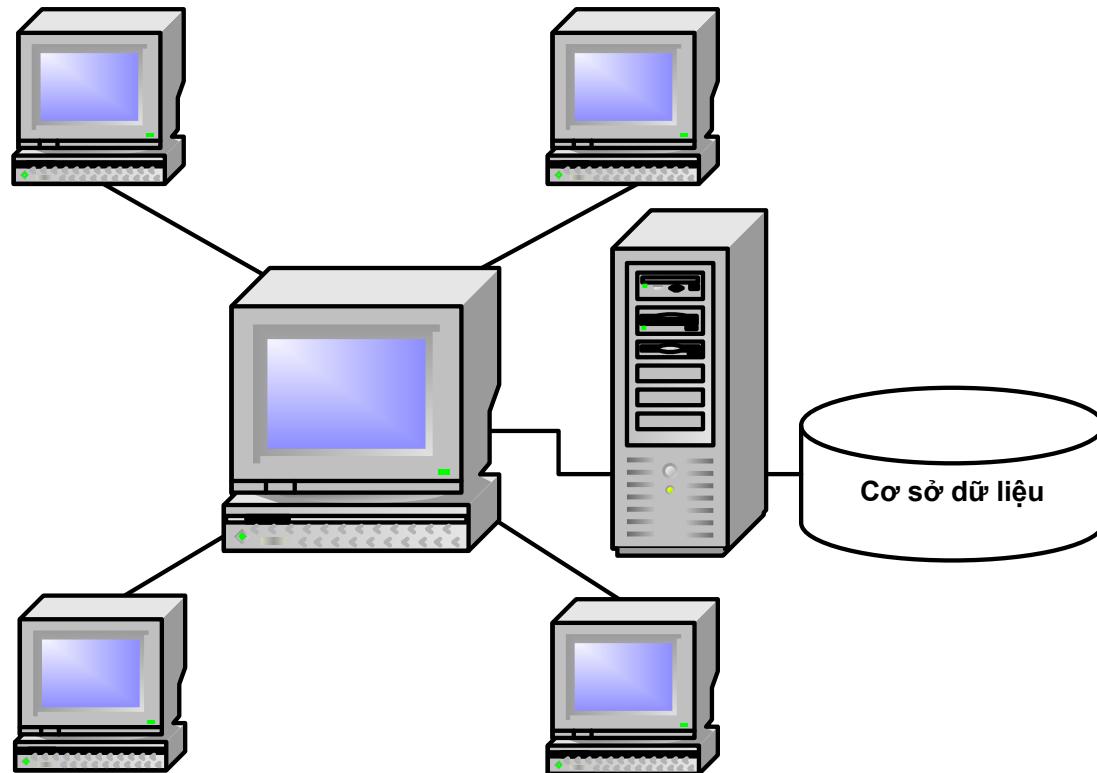
*Hệ cơ sở dữ liệu cá nhân*

# HỆ CSDL TRUNG TÂM

- ❖ Trong các tổ chức lớn, dữ liệu mà hầu hết các ứng dụng có thể truy nhập được lưu trữ trên một máy tính trung tâm.
- ❖ Người dùng từ xa có thể truy nhập CSDL này thông qua các thiết bị đầu cuối và các kết nối truyền thông dữ liệu.
- ❖ CSDL trung tâm thường lưu trữ các CSDL tích hợp rất lớn và được nhiều người dùng truy nhập.
- ❖ Việc sử dụng thường có cường độ lớn với hàng trăm đến hàng nghìn giao dịch trong một giây.

**Ví dụ:** hệ thống đặt vé máy bay, hoặc các hệ thống ngân hàng.

# HỆ CSDL TRUNG TÂM (cont.)

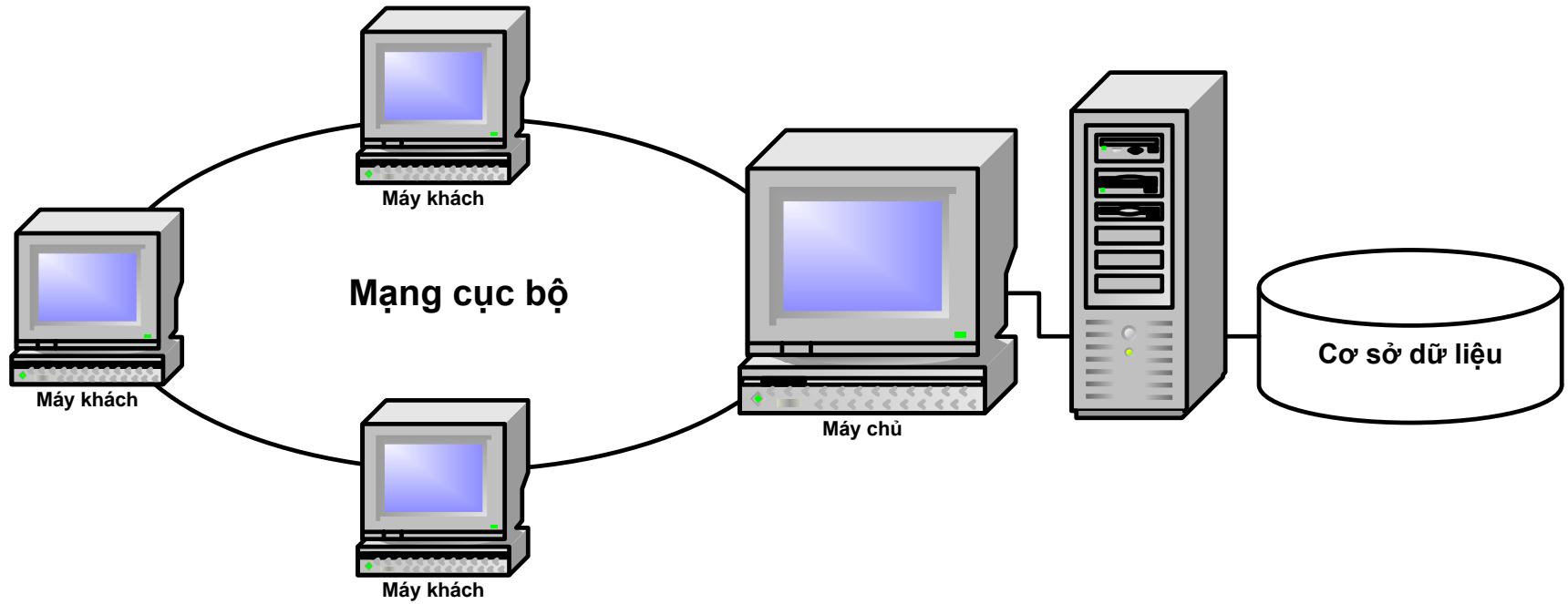


*Hệ cơ sở dữ liệu trung tâm*

## HỆ CSDL KHÁCH/CHỦ

- ❖ Mục đích chính của kiến trúc khách/chủ là cho phép các ứng dụng máy khách truy nhập dữ liệu được quản lý bởi máy chủ. Giao diện người dùng và logic của chương trình ứng dụng được xử lý bên máy khách, trong khi xử lý CSDL được thực hiện bên máy chủ.
- ❖ Máy chủ không cần có cấu hình quá mạnh như trong các hệ CSDL trung tâm.

# HỆ CSDL KHÁCH/CHỦ (cont.)

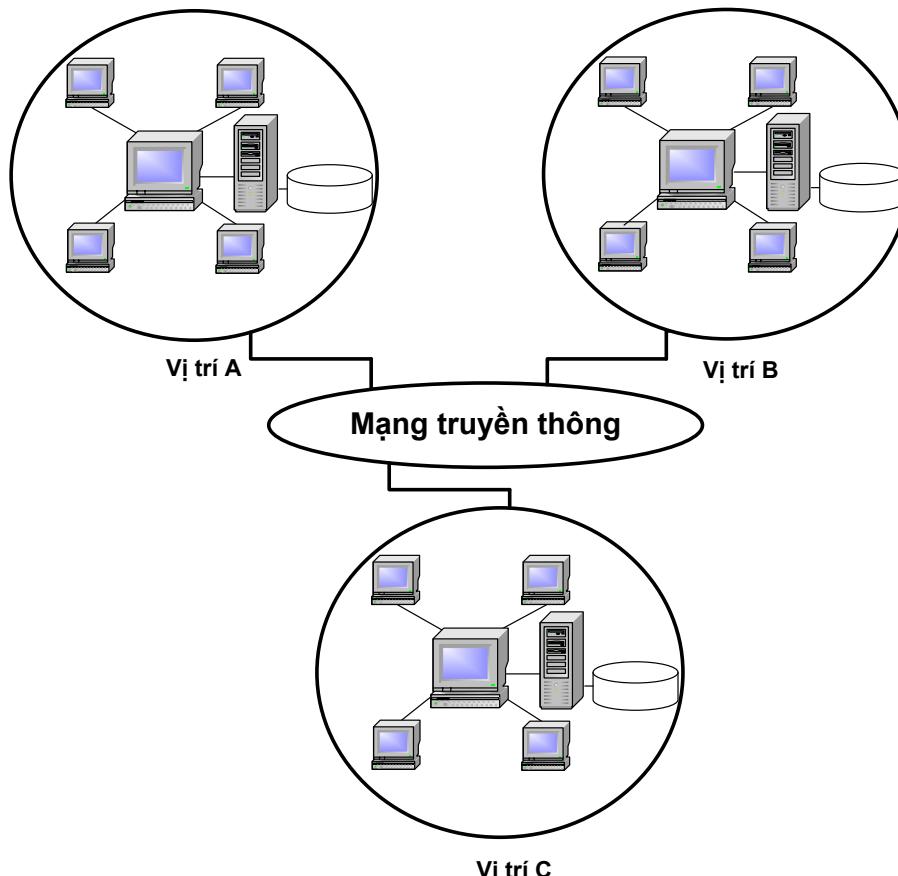


*Hệ cơ sở dữ liệu khách/chủ*

# HỆ CSDL PHÂN TÁN

- ❖ Hiện nay, nhiều tổ chức phân bố trên nhiều vị trí địa lý khác nhau (các thành phố hay các quốc gia khác nhau).  
=> Việc xây dựng các hệ CSDL tập trung là không thực tế và không kinh tế.
- ❖ CSDL phân tán là một CSDL logic đơn lẻ được trải ra về mặt vật lý trên nhiều máy tính ở nhiều vị trí địa lý khác nhau.  
**Ví dụ:** CSDL của một ngân hàng sẽ được phân bổ theo các chi nhánh tại từng địa phương.

# HỆ CSDL PHÂN TÁN (cont.)



*Hệ cơ sở dữ liệu phân tán*

## HỆ CSDL PHÂN TÁN THUẦN NHẤT

- ❖ Các hệ điều hành máy tính tại mỗi vị trí địa lý là như nhau hoặc có khả năng tương thích cao.
- ❖ Các mô hình dữ liệu được sử dụng tại mỗi vị trí địa lý là như nhau. Mô hình quan hệ được sử dụng chung nhất đối với các hệ CSDL phân tán ngày nay.
- ❖ Các hệ quản trị CSDL được sử dụng tại mỗi vị trí địa lý là như nhau hoặc có khả năng tương thích cao.
- ❖ Dữ liệu tại các vị trí khác nhau có định nghĩa và khuôn dạng chung.

# HỆ CSDL PHÂN TÁN KHÔNG THUẦN NHẤT

- ❖ Các máy tính khác nhau và các hệ điều hành khác nhau có thể được sử dụng tại các vị trí địa lý khác nhau.
- ❖ Các mô hình dữ liệu khác nhau và các hệ quản trị CSDL khác nhau cũng có thể được sử dụng.
- ❖ Dữ liệu trên các vị trí thường không tương thích. Có khác biệt về cú pháp và ngữ nghĩa.

=> Khi có nhu cầu chia sẻ dữ liệu, giải pháp là phát triển một hệ CSDL mới hoàn toàn, hợp nhất tất cả các hệ CSDL đang tồn tại. Tuy nhiên, giải pháp này rất khó khăn về mặt kỹ thuật và kinh tế.

# **CHƯƠNG 2.**

# **CÁC MÔ HÌNH DỮ LIỆU**

**(Phần 1)**

# CÁC MÔ HÌNH DỮ LIỆU

- ❖ Giới thiệu
- ❖ Quá trình thiết kế một CSDL
- ❖ Lược đồ thực thể liên kết E-R
- ❖ Một số vấn đề cần quan tâm khi thiết kế lược đồ E-R
- ❖ Lược đồ dữ liệu quan hệ
- ❖ Ánh xạ lược đồ thực thể liên kết sang lược đồ quan hệ

# GIỚI THIỆU

- ❖ **Mô hình dữ liệu** là một tập hợp các khái niệm dùng cho việc mô tả và thao tác dữ liệu, các mối quan hệ và các ràng buộc trên dữ liệu của tổ chức.
- ❖ Mô hình là một biểu diễn đối tượng và sự kiện trong thế giới thực, và mối liên hệ của chúng. Mô hình là một khái niệm trừu tượng tập trung vào các khía cạnh bản chất của một tổ chức và bỏ qua những thuộc tính ngẫu nhiên.
- ❖ Mô hình dữ liệu phải cung cấp các khái niệm và ký hiệu cơ bản, cho phép người thiết kế CSDL và người dùng trao đổi với nhau những hiểu biết về dữ liệu của tổ chức một cách chính xác và không đa nghĩa.

# GIỚI THIỆU (cont.)

- ❖ Một mô hình dữ liệu có thể bao gồm 3 thành phần:
  1. **Phần cấu trúc** gồm tập các luật mà theo đó CSDL được xây dựng.
  2. **Phần thao tác, định nghĩa các thao tác** được phép trên dữ liệu (gồm cả các thao tác cập nhật, lấy dữ liệu từ CSDL, và thay đổi cấu trúc của CSDL).
  3. Có thể có một **tập các luật về tính toàn vẹn**, nhằm đảm bảo CSDL luôn chính xác.

# GIỚI THIỆU (cont.)

- ❖ Dựa vào mô hình kiến trúc 3 lớp, có thể xác định 3 loại mô hình dữ liệu khác nhau:
  1. **Mô hình dữ liệu ngoài** biểu diễn từng khung nhìn của người dùng trong tổ chức.
  2. **Mô hình dữ liệu khái niệm** biểu diễn khung nhìn mức logic, độc lập với hệ quản trị CSDL.
  3. **Mô hình dữ liệu trong** biểu diễn lược đồ khái niệm theo cách mà hệ quản trị CSDL hiểu được.

# GIỚI THIỆU (cont.)

- ❖ Có nhiều hệ CSDL khác nhau đã được lý thuyết hóa, sử dụng, phát triển và cài đặt qua nhiều năm, được chia thành 3 loại chính: **hướng đối tượng**, **hướng bản ghi** và **vật lý**.
- ❖ Có 3 mô hình hướng bản ghi cơ bản: **mô hình dữ liệu quan hệ**, **mô hình dữ liệu mạng** và **mô hình dữ liệu phân cấp**.  
=> *Trong chương trình này tập trung vào mô hình dữ liệu quan hệ.*

# GIỚI THIỆU (cont.)

- ❖ **Mô hình dữ liệu ngũ nghĩa** cố gắng nắm bắt được ý nghĩa của CSDL => cung cấp một cách tiếp cận cho việc mô hình hóa dữ liệu mức khái niệm.
- ❖ **Mô hình dữ liệu quan hệ** (Relational Data Model) là mô hình dữ liệu ngũ nghĩa được sử dụng phổ biến nhất.
- ❖ RDM thường được dùng như một phương tiện trao đổi giữa những người thiết kế CSDL và người dùng trong các quá trình phát triển một CSDL.

# QUÁ TRÌNH THIẾT KẾ MỘT CSDL

**Gồm 6 bước cơ bản:** (3 bước đầu liên quan đến mô hình dữ liệu ngũ nghĩa.)

## 1. *Phân tích yêu cầu:* Phải xác định được:

- Dữ liệu nào được lưu trữ trong CSDL,
- Ứng dụng nào sẽ được xây dựng trên CSDL này,
- Các thao tác nào được sử dụng thường xuyên và các yêu cầu về hiệu năng của hệ thống.

*=> Quá trình này liên quan đến những trao đổi của các nhóm người dùng và nhóm nghiên cứu môi trường hiện tại. Tìm hiểu các ứng dụng đang có xem có cần thay thế hoặc bổ trợ cho hệ CSDL không.*

# QUÁ TRÌNH THIẾT KẾ MỘT CSDL (cont.)

2. ***Thiết kế CSDL mức khái niệm:*** Những thông tin có được từ bước phân tích yêu cầu sẽ được dùng để phát triển một mô tả mức tổng quát dữ liệu được lưu trong CSDL, cùng với các ràng buộc cần thiết trên dữ liệu này.
3. ***Thiết kế CSDL mức logic:*** Một hệ quản trị CSDL sẽ được chọn để cài đặt CSDL và chuyển thiết kế CSDL mức khái niệm thành một lược đồ CSDL với mô hình dữ liệu của hệ quản trị CSDL đã chọn.

# QUÁ TRÌNH THIẾT KẾ MỘT CSDL (cont.)

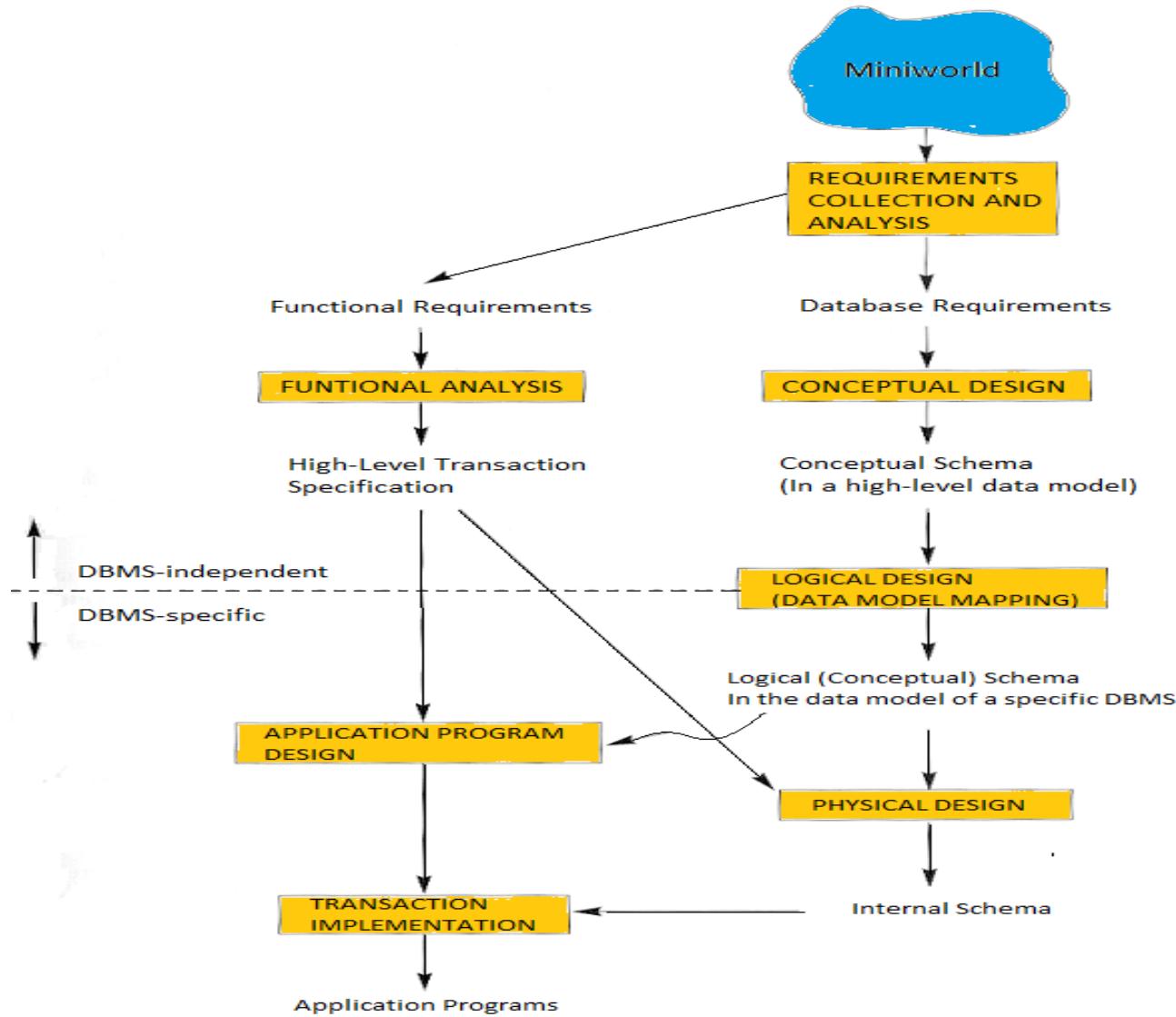
4. **Cải tiến lược đồ:** Các lược đồ được phát triển ở bước 3 sẽ được phân tích các vấn đề tiềm ẩn. Tại đây, CSDL sẽ được **chuẩn hóa**, dựa trên lý thuyết toán học.
5. **Thiết kế CSDL mức vật lý:** Khối lượng công việc tiềm ẩn và các phương pháp truy nhập được mô phỏng để xác định các điểm yếu tiềm ẩn trong CSDL mức khái niệm. Quá trình này thường là nguyên nhân tạo ra các tệp chỉ mục hoặc/và các quan hệ nhóm. Trong trường hợp đặc biệt, toàn bộ mô hình khái niệm sẽ được xây dựng lại.

# QUÁ TRÌNH THIẾT KẾ MỘT CSDL (cont.)

6. **Thiết kế an toàn bảo mật:** Xác định các nhóm người dùng và phân tích vai trò của họ để định nghĩa các phương pháp truy nhập dữ liệu.

Trong quá trình phát triển, thường có bước cuối cùng (bước thứ 7), gọi là **pha điều chỉnh (tuning phase)**, trong đó CSDL sẽ được thực hiện (mặc dù nó có thể chỉ được chạy mô phỏng) và sẽ được cải tiến, chỉnh sửa để đáp ứng nhu cầu thực thi trong môi trường mong đợi.

# QUÁ TRÌNH THIẾT KẾ MỘT CSDL (cont.)

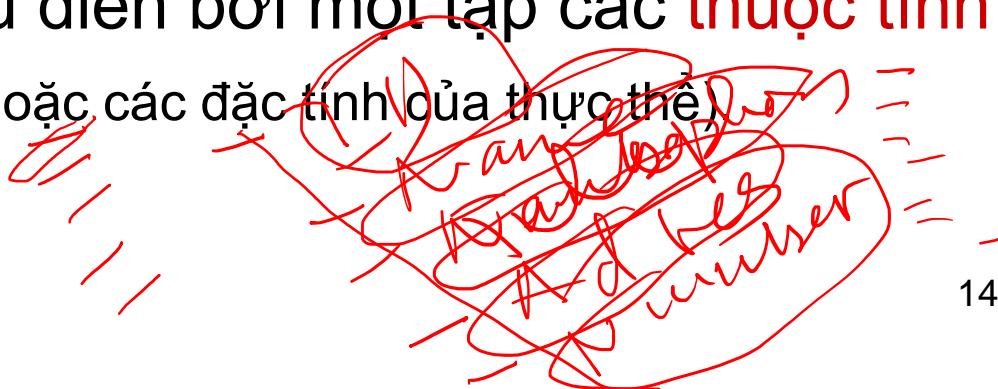


Tóm tắt các bước chính trong quá trình thiết kế CSDL

# LƯỢC ĐỒ THỰC THẾ LIÊN KẾT (The Entity-Relationship Diagram)

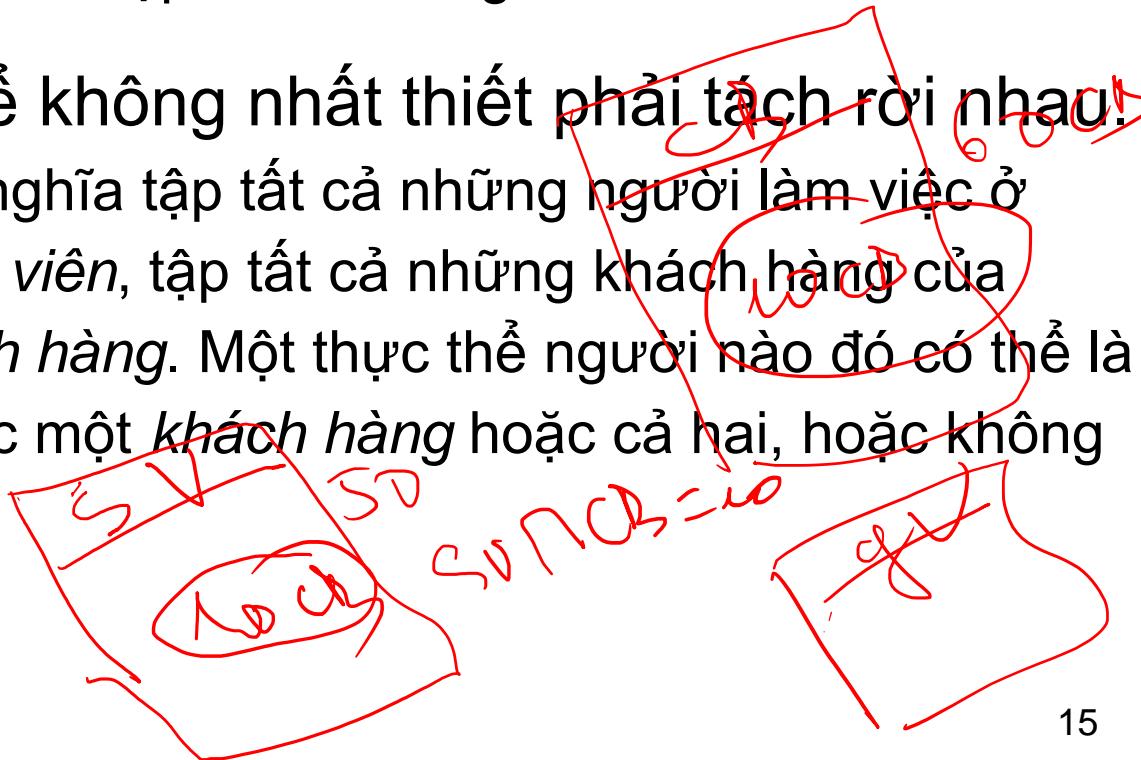
# LƯỢC ĐỒ THỰC THỂ LIÊN KẾT

- ❖ Lược đồ thực thể liên kết (The Entity-Relationship Diagram) gồm 3 khái niệm cơ bản: tập thực thể, tập quan hệ và thuộc tính.
- ❖ Thực thể là một đối tượng trong thế giới thực và có thể phân biệt được với các đối tượng khác. Thực thể có thể cụ thể (một người, một quyển sách, ...) hoặc cũng có thể trừu tượng (một khoản vay ngân hàng, một khái niệm, ...).
- ❖ Thực thể được biểu diễn bởi một tập các thuộc tính (là các thuộc tính mô tả hoặc các đặc tính của thực thể)



# LƯỢC ĐỒ THỰC THỂ LIÊN KẾT (cont.)

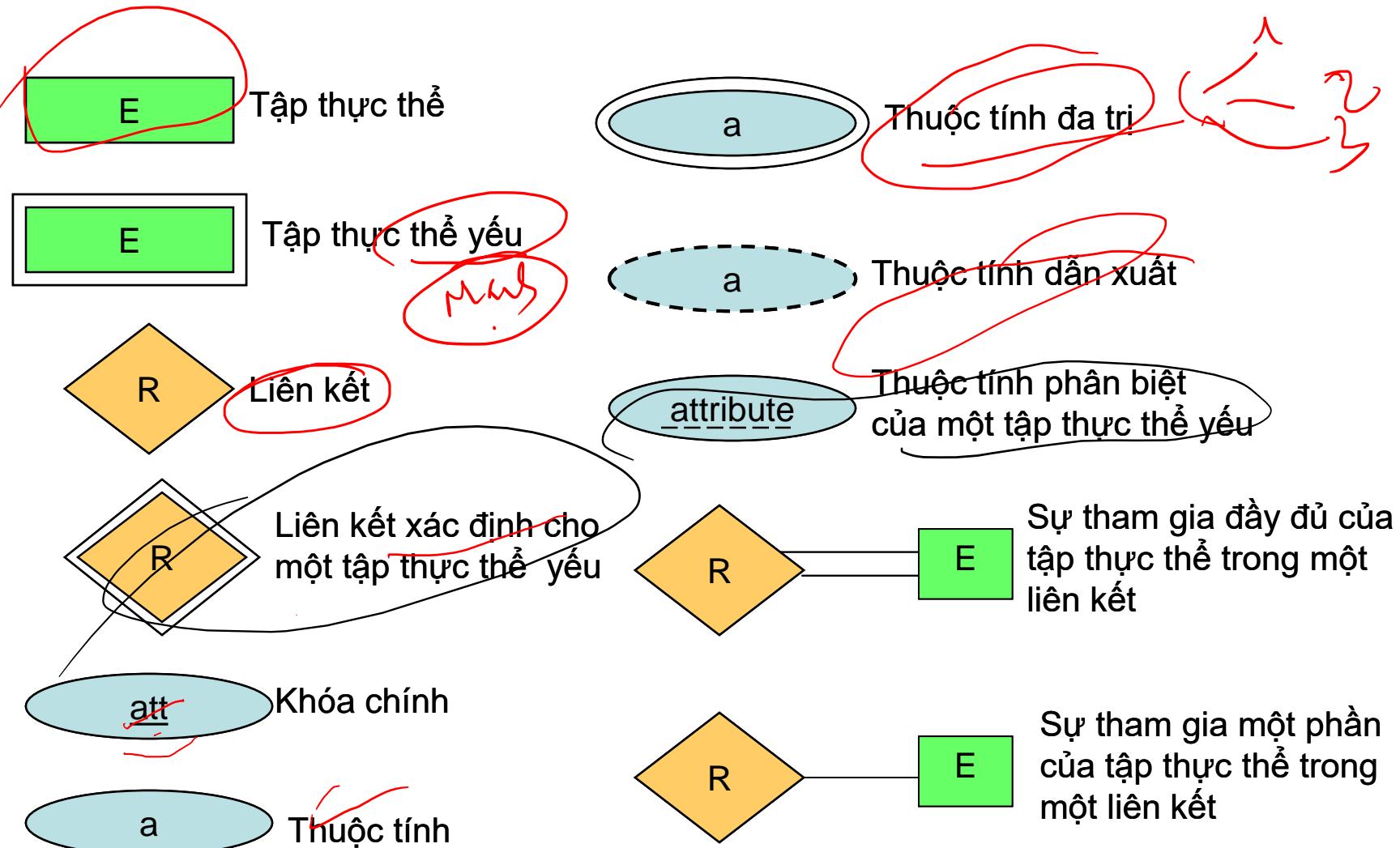
- ❖ *Tập thực thể* là một nhóm các thực thể có cùng thuộc tính. **Ví dụ:** tập tất cả khách hàng của ngân hàng có thể được định nghĩa là tập khách hàng.
- ❖ Các tập thực thể không nhất thiết phải tách rời nhau.  
**Ví dụ:** có thể định nghĩa tập tất cả những người làm việc ở ngân hàng là *nhan vien*, tập tất cả những khách hàng của ngân hàng là *khach hang*. Một thực thể người nào đó có thể là một *nhan vien* hoặc một *khach hang* hoặc cả hai, hoặc không phải cả hai.



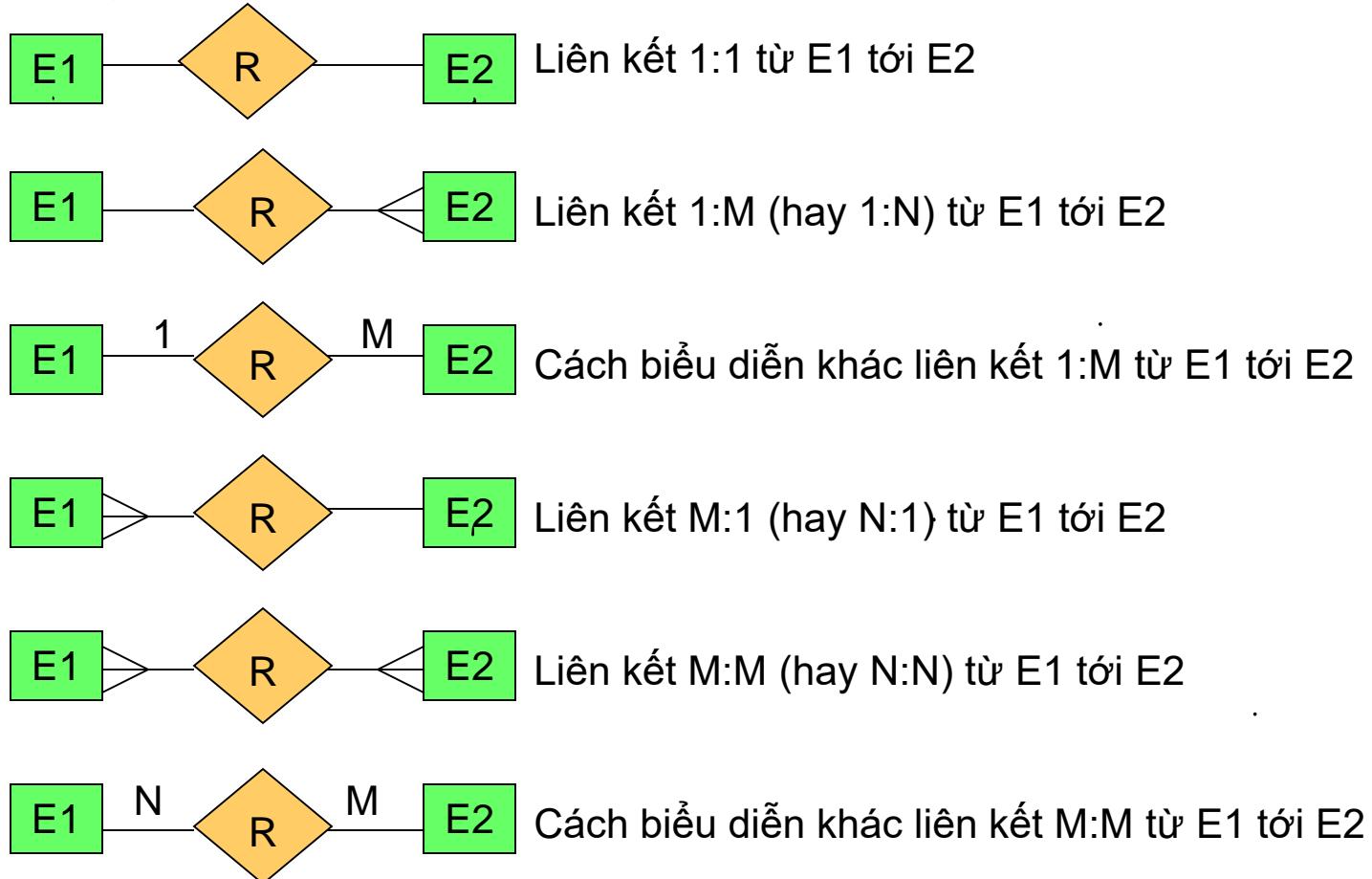
# LƯỢC ĐỒ THỰC THỂ LIÊN KẾT<sub>(cont.)</sub>

- ❖ Thuộc tính: Là 1 đặc trưng mà trị của nó tham gia vào việc mô tả một thực thể
- ❖ Mỗi thuộc tính có một tập giá trị cho phép, được gọi là **miền** (hay tập giá trị) của thuộc tính đó.
  - Một thuộc tính của một tập thực thể là một hàm ánh xạ từ một tập thực thể vào một miền giá trị.
  - Một tập thực thể có thể có nhiều thuộc tính.  
=> Mỗi thực thể trong tập có thể được mô tả bởi một tập các cặp **< thuộc tính, giá trị>**, ứng với từng thuộc tính trong tập thực thể.
- ❖ Một CSDL bao gồm một tập các thực thể

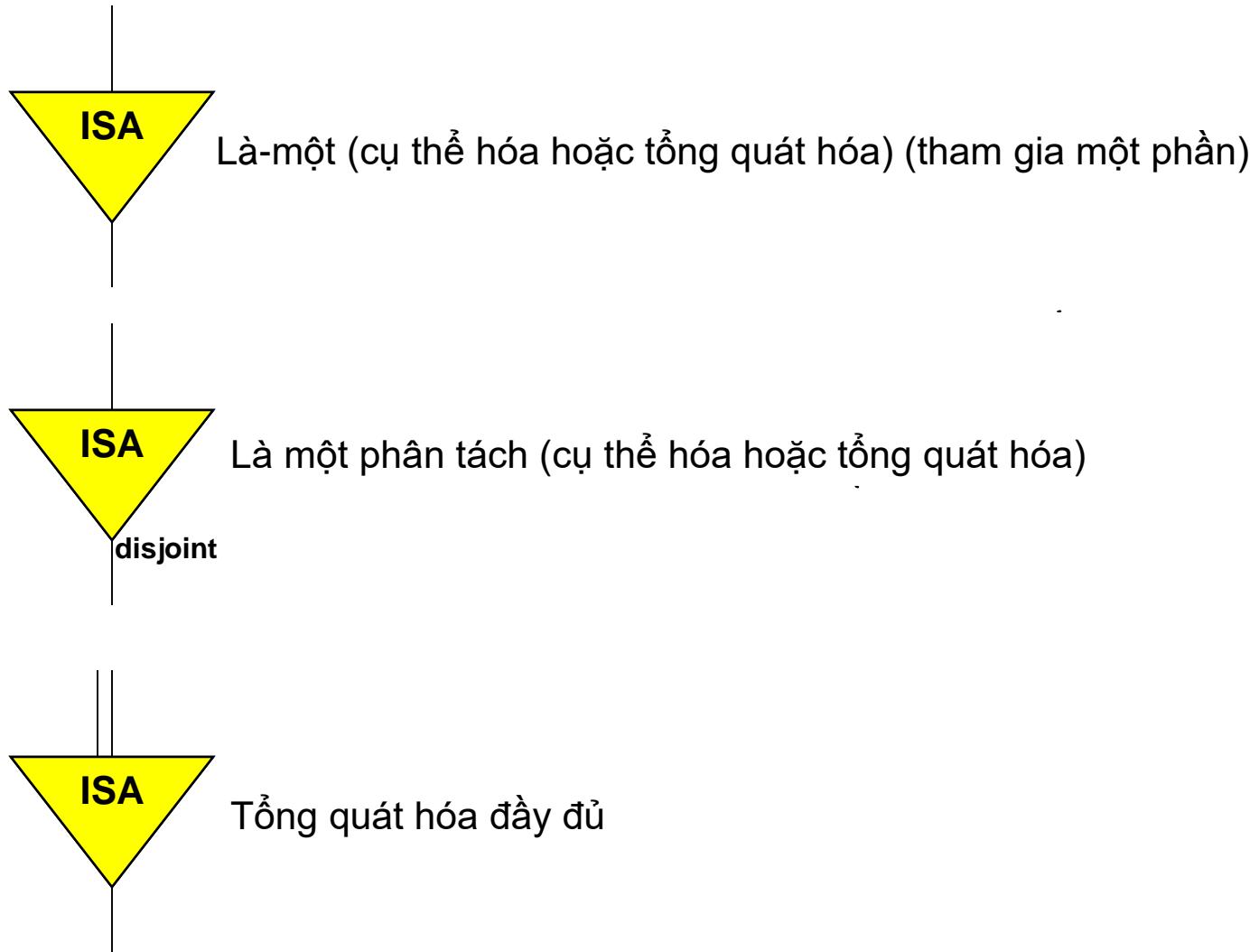
# KÝ HIỆU TRONG LƯỢC ĐỒ E-R



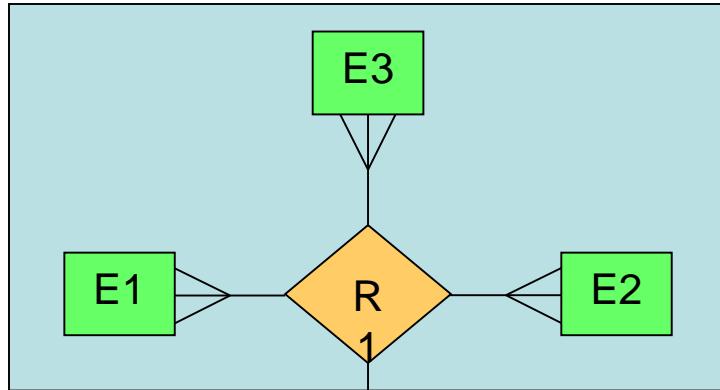
# KÝ HIỆU TRONG LƯỢC ĐỒ E-R (cont.)



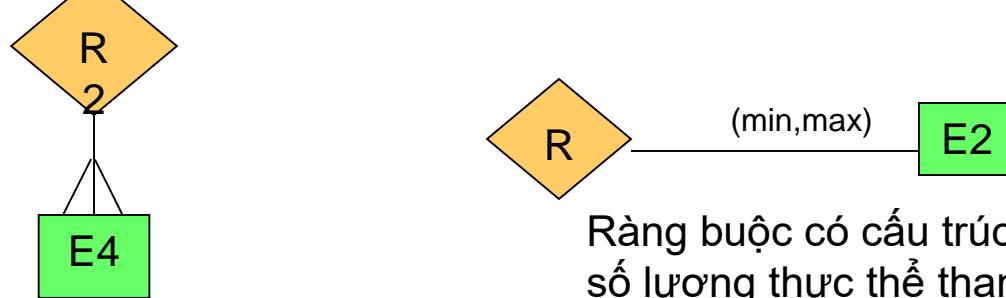
# KÝ HIỆU TRONG LƯỢC ĐỒ E-R (cont.)



# KÝ HIỆU TRONG LƯỢC ĐỒ E-R (cont.)

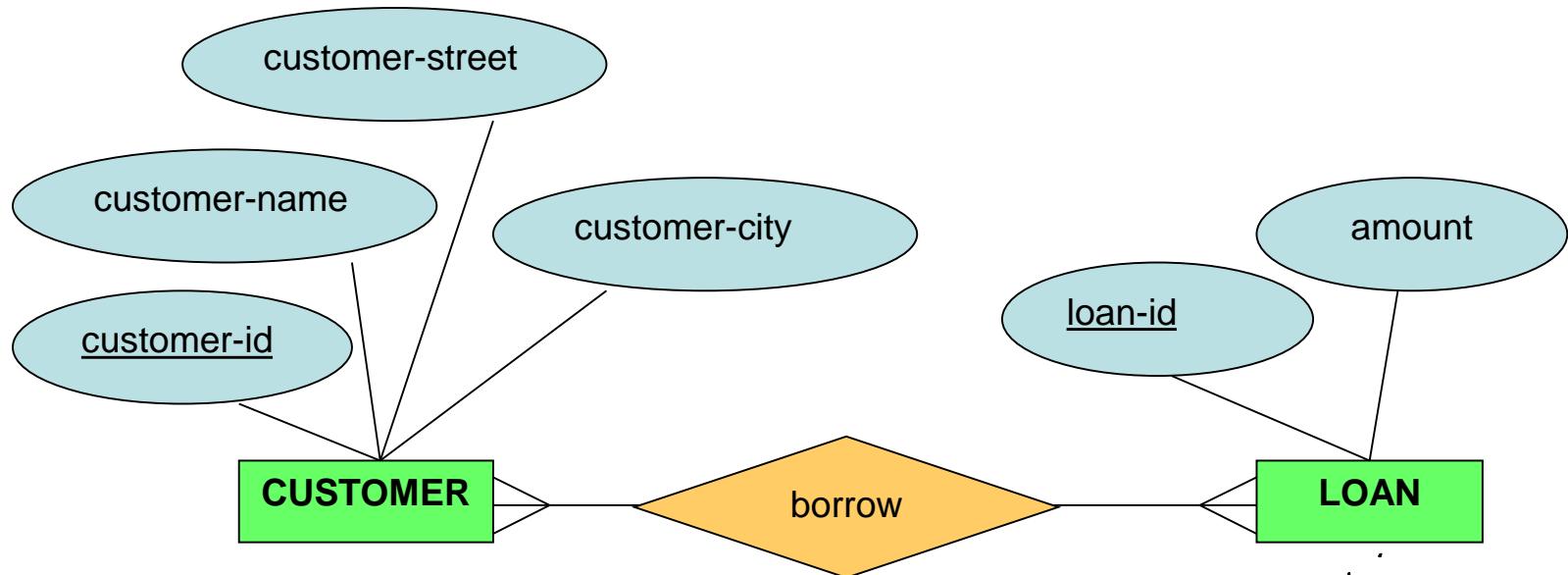


Kết hợp: hộp chữ nhật bao quanh liên kết  
được coi như một thực thể

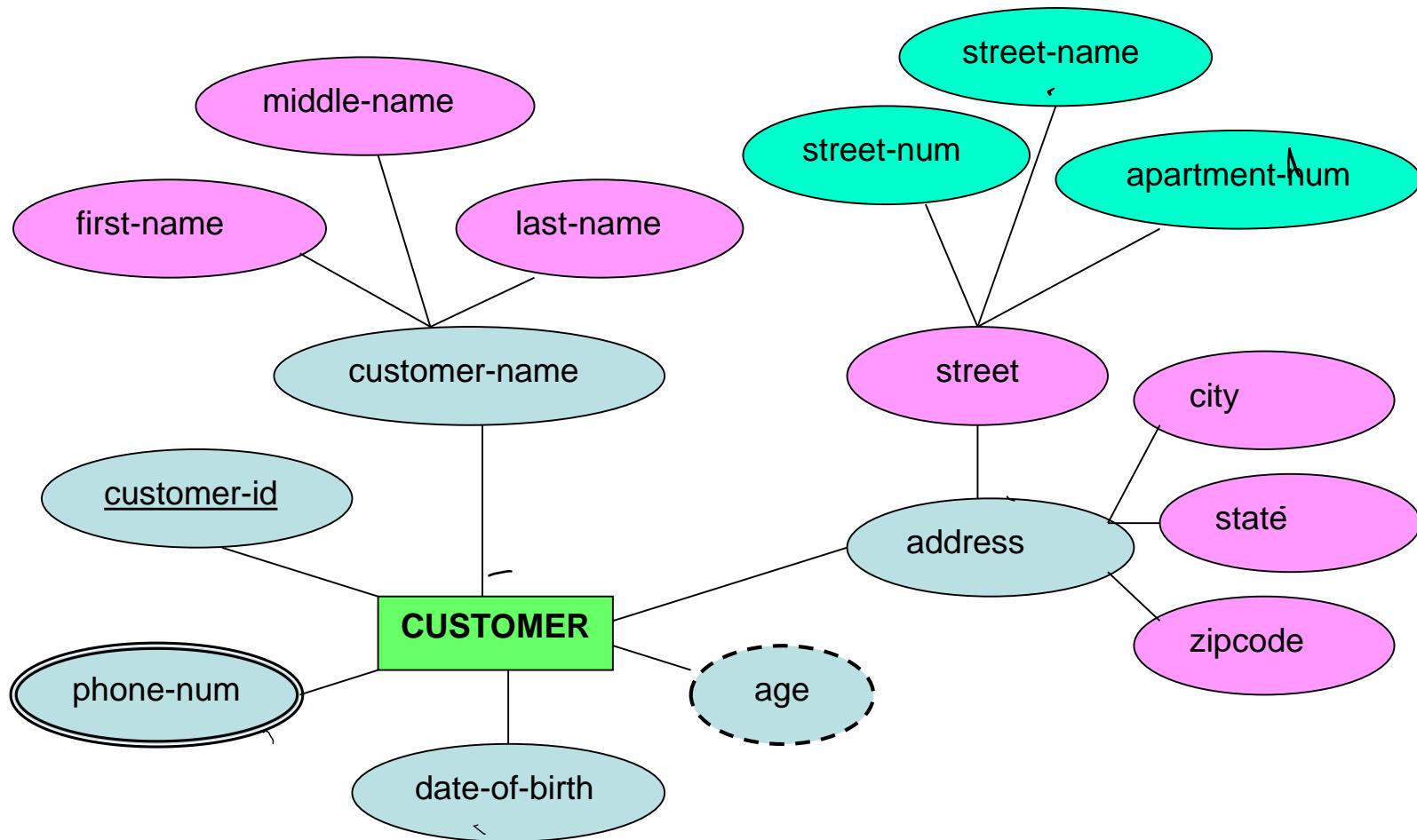


Ràng buộc có cấu trúc: (min,max) của  
số lượng thực thể tham gia liên kết

# VÍ DỤ VỀ LƯỢC ĐỒ E-R (ERD)



# VÍ DỤ KHÁC VỀ LƯỢC ĐỒ E-R



# CÁC THUỘC TÍNH TRONG LƯỢC ĐỒ E-R

Thuộc tính có thể được phân chia thành các loại sau:

- ❖ **Thuộc tính đơn** hoặc **thuộc tính kép**: Thuộc tính đơn không bao gồm các thành phần cấu thành, trong khi thuộc tính kép bao gồm các thành phần con cấu thành.

**Ví dụ:** thuộc tính **tên**: Nếu **tên** biểu diễn một thuộc tính đơn thì có thể coi bộ ba cấu thành **tên** là **họ**, **tên đệm** và **tên gọi** là một thuộc tính nguyên tố, không phân chia được nữa. Còn nếu coi **tên** là một thuộc tính kép thì có thể lựa chọn thao tác với thuộc tính này là một tên đầy đủ hoặc có thể thao tác với từng thành phần cấu thành tên.

# CÁC THUỘC TÍNH TRONG LƯỢC ĐỒ E-R (cont.)

- ❖ **Thuộc tính đơn trị** hoặc **thuộc tính đa trị**: Thuộc tính đơn trị có nhiều nhất một giá trị tại một thời điểm cụ thể. Thuộc tính đa trị có thể có nhiều giá trị khác nhau tại một thời điểm.

**Ví dụ:** Tại một trường học sinh viên được đăng ký học theo tín chỉ. Tại một kỳ học nào đó, số tín chỉ một sinh viên đăng ký là đơn trị, ví dụ là 7 (tín chỉ) => số tín chỉ không thể nhận giá trị đa trị. Thuộc tính số điện thoại của sinh viên có thể chứa nhiều giá trị cùng lúc do tại một thời điểm, một sinh viên có thể có một vài số điện thoại khác nhau. => thuộc tính số điện thoại là đa trị.

## CÁC THUỘC TÍNH TRONG LƯỢC ĐỒ E-R (cont.)

- ❖ **Thuộc tính dẫn xuất:** là thuộc tính mà giá trị của nó được dẫn xuất (hoặc được tính toán) từ những giá trị của các thuộc tính hoặc các thực thể có liên quan.

**Ví dụ:** Giả sử thực thể **KHÁCH HÀNG** của một ngân hàng có một thuộc tính tên là ***loans-held***, chứa số lượng các khoản vay của một khách hàng tại ngân hàng. Giá trị của thuộc tính này có thể được tính bằng cách đếm số lượng thực thể các khoản vay liên quan tới khách hàng.

- ❖ **Thuộc tính rỗng (Null):** thuộc tính nhận giá trị rỗng khi một thực thể không có giá trị cho nó.

# CÁC LIÊN KẾT TRONG LƯỢC ĐỒ E-R

- ❖ Một **liên kết** (hay **quan hệ**) là một mối liên hệ giữa một vài thực thể.

**Ví dụ**, có thể định nghĩa một mối liên kết thể hiện sinh viên của một lớp học. Mối quan hệ này xác nhận rằng sinh viên đã đăng ký học lớp đó.

Một cách hình thức, một *tập quan hệ* là một tập các liên hệ cùng loại. Nó là một quan hệ toán học của  $n$  tập thực thể (có thể giao nhau,  $n \geq 2$ ).

Nếu  $E_1, E_2, \dots, E_n$  là các tập thực thể, thì một tập quan hệ  $R$  là một tập con của:

$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

Với  $(e_1, e_2, \dots, e_n)$  là một quan hệ.

## CÁC LIÊN KẾT TRONG LUỢC ĐỒ E-R (cont.)

- ❖ Mỗi liên hệ giữa các tập thực thể được gọi là **sự tham gia**; nghĩa là các tập thực thể  $E_1, E_2, \dots, E_n$  tham gia vào quan hệ  $R$ .
- ❖ Một **thể hiện quan hệ** trong lược đồ E-R biểu diễn mối liên hệ giữa các thực thể xác định trong thế giới thực đang được mô hình hóa.
- ❖ Một mối quan hệ cũng có thể có các thuộc tính được gọi là các **thuộc tính mô tả**.

**Ví dụ:** Xét ngữ cảnh ngân hàng, giả sử có tập các mối quan hệ **depositor** và các tập thực thể **CUSTOMER** và **ACCOUNT** => có thể có thuộc tính mô tả **access-date** cho mối quan hệ **depositor** để mô tả ngày gần nhất mà khách hàng truy nhập vào tài khoản của họ.

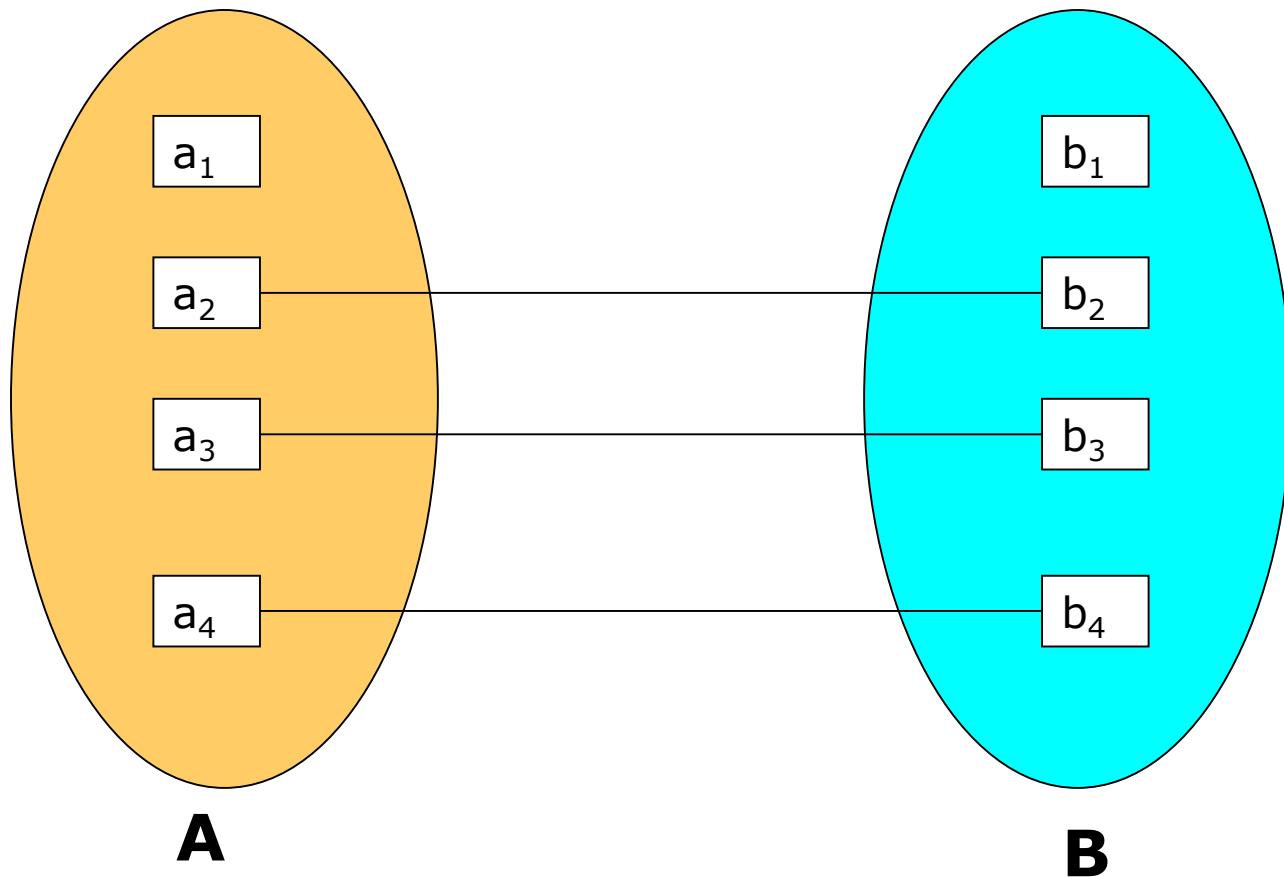
## CÁC RÀNG BUỘC TRONG LƯỢC ĐỒ E-R

- ❖ Các giá trị được lưu trữ trong CSDL thường có những ràng buộc để đảm bảo rằng chúng mô hình hóa một cách chính xác toàn bộ thế giới thực của tổ chức đang được thể hiện trong CSDL.
- ❖ Lược đồ E-R có khả năng mô hình hóa các loại ràng buộc này.
- ❖ Tập trung vào hai loại ràng buộc quan trọng: **ánh xạ lực lượng liên kết** và **các ràng buộc tham gia**.

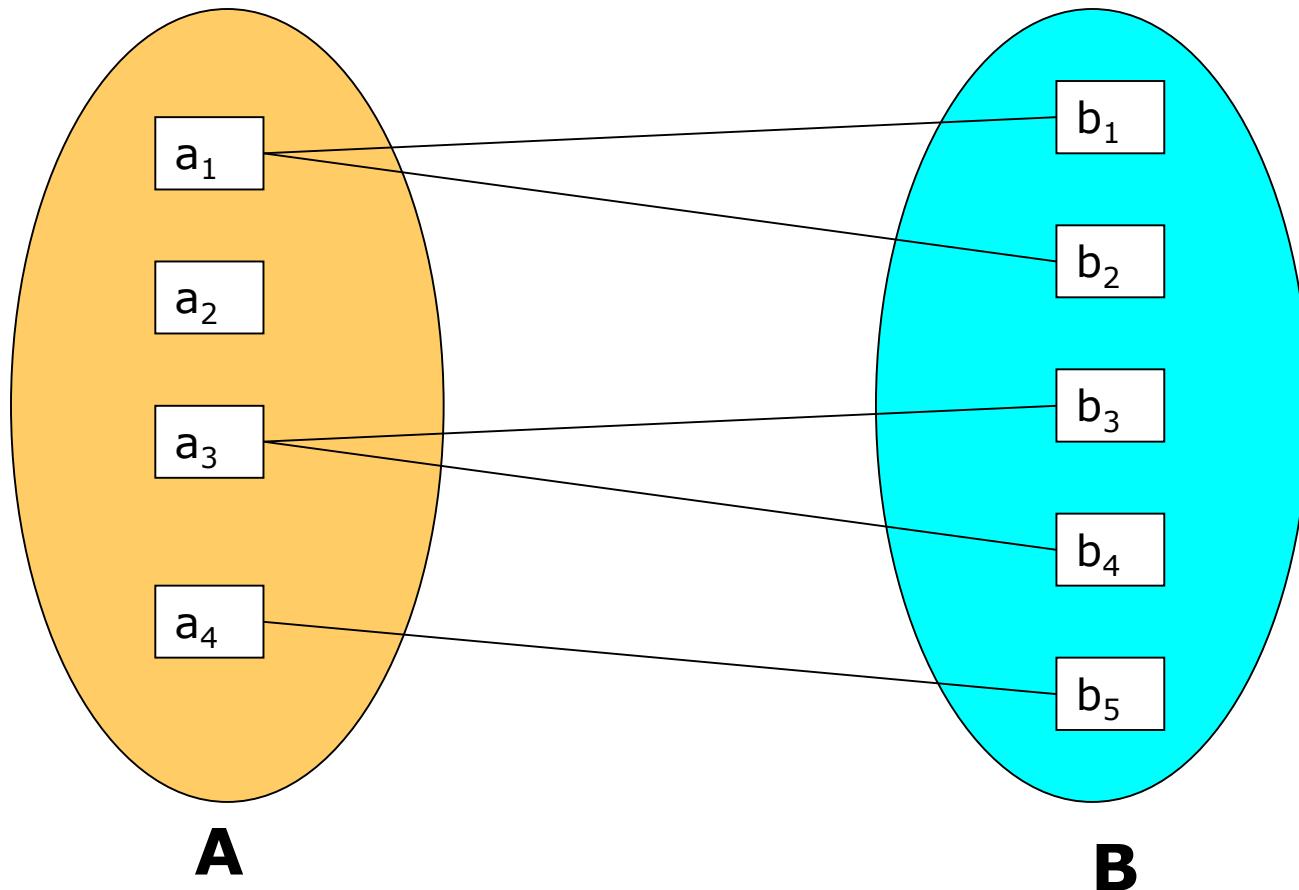
# ÁNH XẠ LỰC LƯỢNG LIÊN KẾT

- ❖ Ánh xạ lực lượng liên kết thể hiện số lượng các thực thể mà một thực thể khác có thể liên hệ thông qua một tập quan hệ.
- ❖ Ràng buộc ánh xạ lực lượng liên kết có ích nhất khi mô tả các quan hệ hai ngôi.
- ❖ Với một tập quan hệ hai ngôi  $R$  giữa tập thực thể  $A$  và  $B$ , ánh xạ lực lượng liên kết gồm các loại sau:
  - $(1:1)$  một tới một từ  $A$  đến  $B$
  - $(1:M)$  một tới nhiều từ  $A$  đến  $B$
  - $(M:1)$  nhiều tới một từ  $A$  đến  $B$
  - $(M:M)$  nhiều tới nhiều từ  $A$  đến  $B$

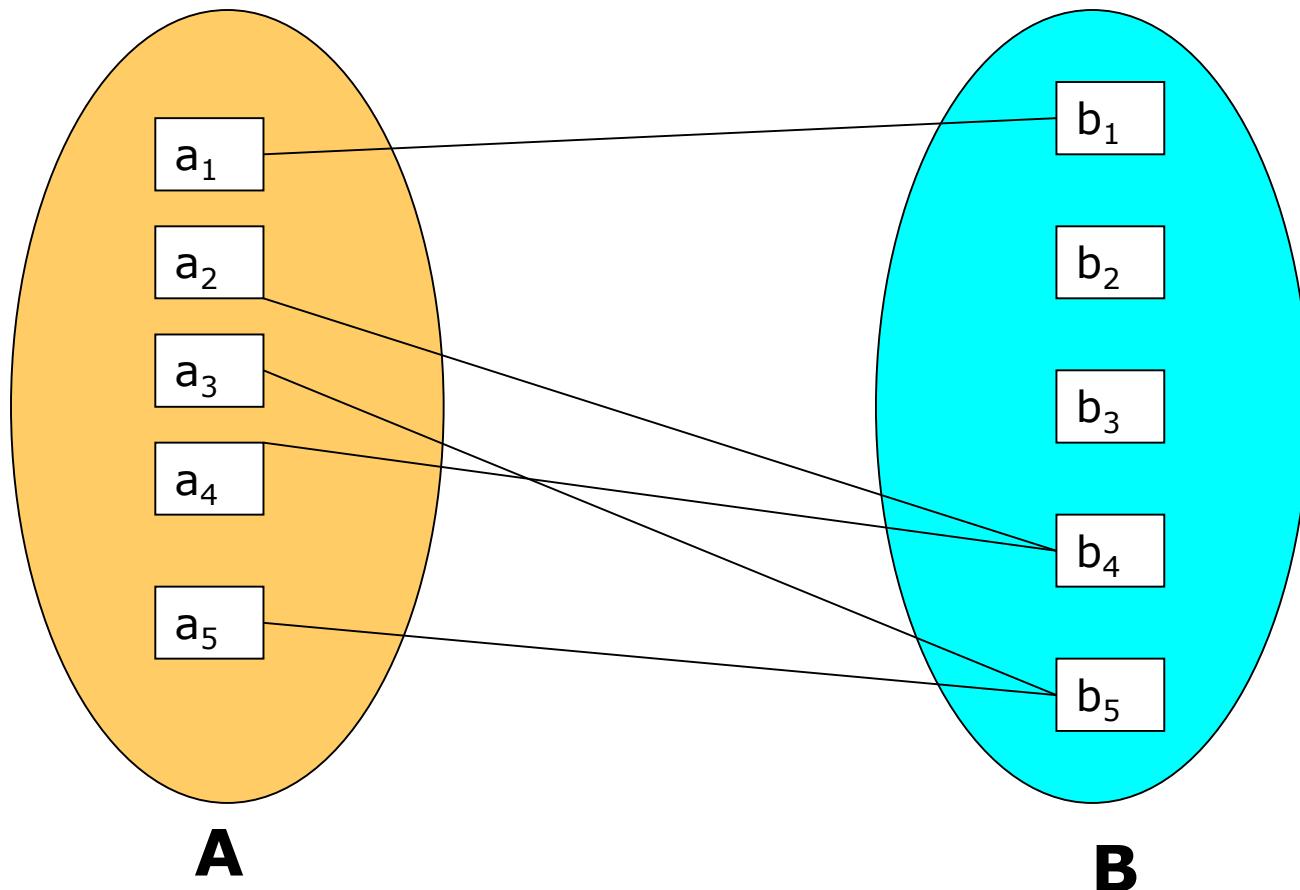
# ÁNH XẠ LỰC LƯỢNG LIÊN KẾT: 1:1 TỪ A ĐẾN B



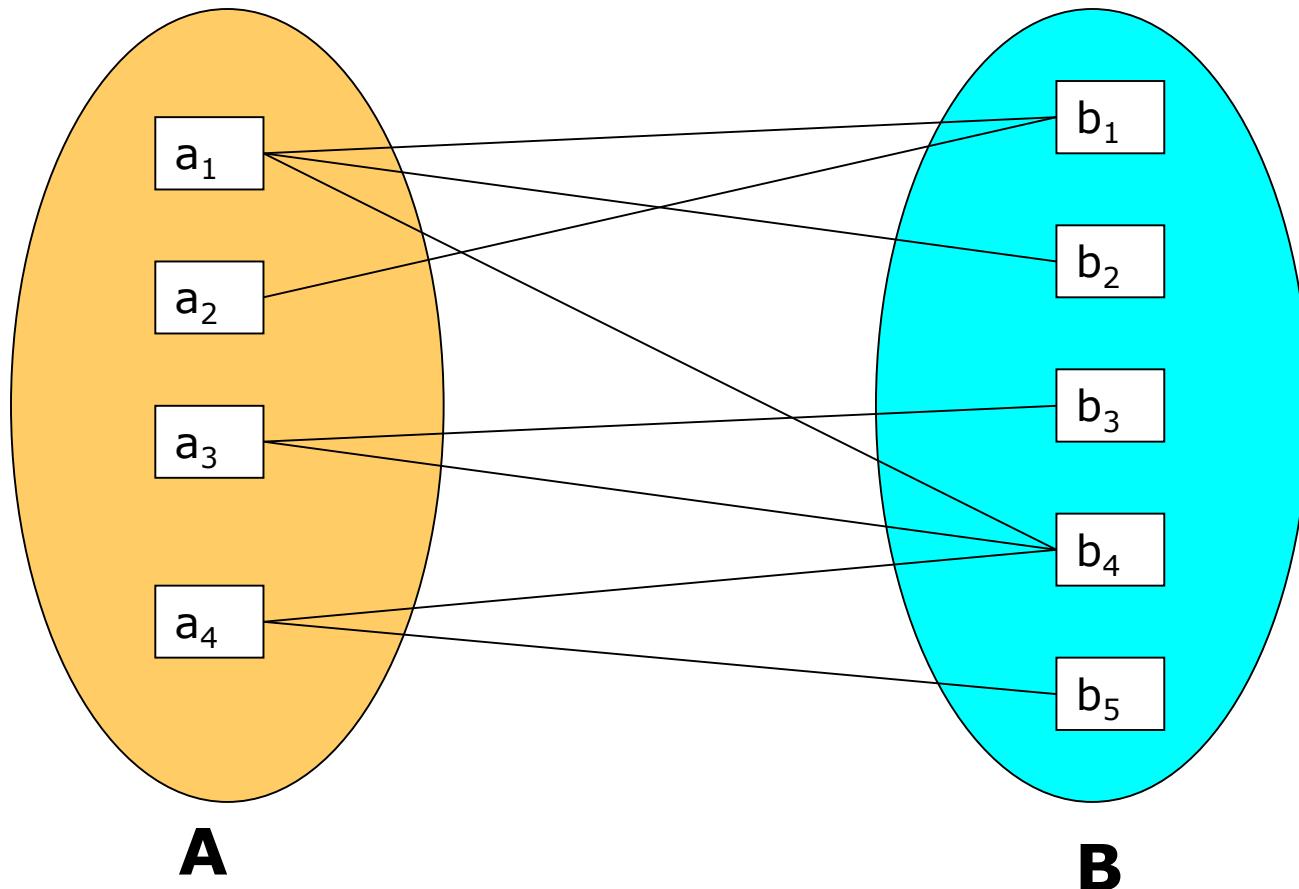
# ÁNH XẠ LỰC LƯỢNG LIÊN KẾT: 1:M TỪ A ĐẾN B



# ÁNH XẠ LỰC LƯỢNG LIÊN KẾT: M:1 TỪ A ĐẾN B



# ÁNH XẠ LỰC LƯỢNG LIÊN KẾT: M:M TỪ A ĐẾN B



## CÁC RÀNG BUỘC THAM GIA TRONG LƯỢC ĐỒ E-R

- ❖ Sự tham gia của một tập thực thể  $E$  trong một tập các quan hệ  $R$  được gọi là **đầy đủ (total)** nếu mọi thực thể của  $E$  tham gia vào ít nhất một quan hệ trong  $R$ .
- ❖ Nếu chỉ có một vài thực thể của  $E$  tham gia vào một quan hệ trong  $R$ , thì sự tham gia của tập thực thể  $E$  trong tập các quan hệ  $R$  được gọi là **một phần (partial)**.

# CÁC RÀNG BUỘC THAM GIA TRONG LƯỢC ĐỒ E-R (Cont.)

## ❖ Ví dụ, xem xét hệ thống ngân hàng.

- Mỗi thực thể **LOAN** (khoản vay) có liên hệ tới ít nhất một **CUSTOMER** (khách hàng) thông qua mối quan hệ **borrower** (vay mượn) => sự tham gia của thực thể LOAN trong tập quan hệ **borrower** là **đầy đủ**.
- Ngược lại, có những khách hàng không liên quan tới các khoản vay mượn ngân hàng. Như vậy, có thể nói rằng chỉ một số các thực thể **CUSTOMER** có liên hệ tới một thực thể **LOAN** thông qua quan hệ **borrower** => sự tham gia của thực thể **CUSTOMER** trong tập quan hệ **borrower** là **một phần**.

# KHÓA CỦA MỘT TẬP THỰC THỂ

- ❖ Cần có một cơ chế để phân biệt các thực thể trong một tập thực thể.
- ❖ Theo khía cạnh CSDL, sự khác nhau giữa các thực thể phải được thể hiện thông qua các thuộc tính.  
=> Các giá trị của thuộc tính của một thực thể phải được xác định sao cho chúng có thể **xác định duy nhất** thực thể đó. Nghĩa là, không có hai thực thể nào trong một tập thực thể được phép có giá trị trùng nhau trên tất cả các thuộc tính.
- ❖ **Khóa (key)** cho phép xác định một tập các thuộc tính đủ để phân biệt các thực thể với nhau. Khóa cũng giúp cho việc xác định duy nhất các mối quan hệ, và vì vậy, phân biệt các mối quan hệ với nhau.

# KHÓA CHÍNH, SIÊU KHÓA VÀ KHÓA DỰ BỊ

- ❖ Siêu khóa (superkey) là một tập gồm một hoặc nhiều thuộc tính được lựa chọn cho phép xác định duy nhất một thực thể trong tập thực thể.

Giả sử có một tập thực thể sinh viên trong một lớp học với lược đồ sau:

*STUDENTS(SS#, name, address, age, major, minor, gpa, spring-sch)*

- ❖ Siêu khóa cho tập thực thể STUDENTS có thể là: (ID#, name, major, minor) hoặc (ID#, name)  
=> Khái niệm siêu khóa là một định nghĩa không đầy đủ của khóa vì siêu khóa còn chứa nhiều thuộc tính dư thừa.

## KHÓA CHÍNH, SIÊU KHÓA VÀ KHÓA DỰ BỊ (Cont.)

- ❖ Nếu tập  $K$  là một siêu khóa của tập thực thể  $E$  thì mọi tập cha của  $K$  cũng là siêu khóa.
- ❖ **Khóa dự bị (candidate key)** là những siêu khóa mà không có tập con nào của nó là siêu khóa.  
=> Với mỗi một tập thực thể  $E$  cho trước, tồn tại một hoặc nhiều khóa dự bị.
- ❖ Người thiết kế CSDL chỉ chọn một khóa dự bị làm **khóa chính**, hay còn gọi là **khóa** của tập thực thể.

## KHÓA CHÍNH, SIÊU KHÓA VÀ KHÓA DỰ BỊ (Cont.)

- ❖ Hai thực thể khác nhau sẽ không có cùng giá trị trên tất cả các thuộc tính khóa (khóa chính, khóa dự bị hay siêu khóa) tại cùng một thời điểm.  
=> ràng buộc khóa
- ❖ Người thiết kế CSDL phải cẩn thận khi lựa chọn tập các thuộc tính cấu thành khóa của một tập thực thể để đảm bảo:
  1. Chắn chắn rằng tập các thuộc tính xác định duy nhất thực thể.
  2. Tập thuộc tính khóa sẽ không bao giờ hoặc rất hiếm khi bị thay đổi.

# TẬP CÁC QUAN HỆ

- ❖ Khóa chính của một tập thực thể cho phép phân biệt các thực thể khác nhau trong tập => phải có một cơ chế tương tự cho phép phân biệt các mối quan hệ trong một tập các quan hệ.
- ❖ Cho  $R$  là một tập quan hệ liên quan tới các tập thực thể  $E_1, E_2, \dots, E_n$ . Gọi  $K_i$  là tập các thuộc tính cấu thành khóa chính của tập thực thể  $E_i$ .

Giả thiết rằng:

1. Tên của các thuộc tính trong tất cả các khóa chính là duy nhất.
2. Mỗi tập thực thể sẽ chỉ tham dự một lần duy nhất trong quan hệ.

# TẬP CÁC QUAN HỆ (Cont.)

- ❖ Việc cấu thành khóa chính cho một tập các quan hệ  $R$  phụ thuộc vào tập các thuộc tính liên quan theo các cách sau:
  1. Nếu tập quan hệ  $R$  không có thuộc tính nào liên quan thì tập các thuộc tính:  $K_1 \cup K_2 \cup \dots \cup K_n$  mô tả một quan hệ riêng biệt trong tập  $R$ .
  2. Nếu tập quan hệ  $R$  có các thuộc tính liên quan  $a_1, a_2, \dots, a_m$  thì tập các thuộc tính  $K_1 \cup K_2 \cup \dots \cup K_n \cup \{a_1, a_2, \dots, a_m\}$  sẽ mô tả một quan hệ riêng trong tập  $R$ .
- => Trong cả hai trường hợp, tập các thuộc tính  $K_1 \cup K_2 \cup \dots \cup K_n$  đều hình thành một siêu khóa cho tập các quan hệ.

# **CHƯƠNG 2.**

# **CÁC MÔ HÌNH DỮ LIỆU**

**(Phần 2)**

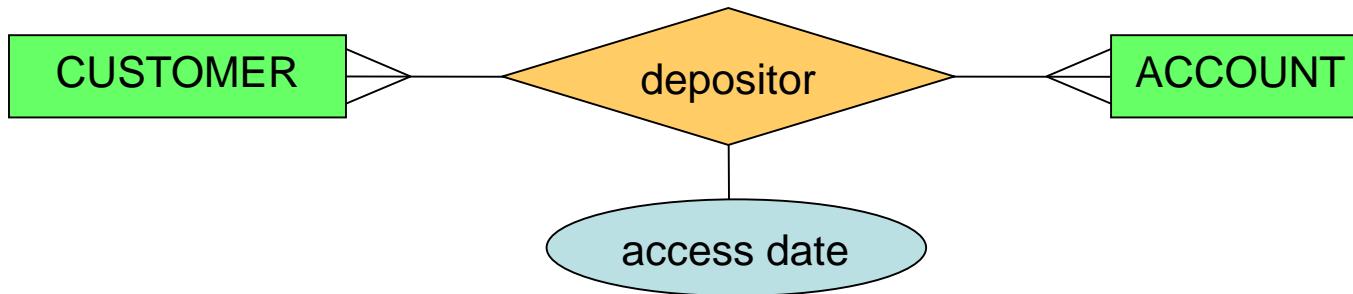
# MÔ HÌNH DỮ LIỆU QUAN HỆ

- ❖ Giới thiệu
- ❖ Quá trình thiết kế một CSDL
- ❖ Lược đồ thực thể liên kết E-R
- ❖ Một số vấn đề cần quan tâm khi thiết kế lược đồ E-R
- ❖ Lược đồ dữ liệu quan hệ
- ❖ Ánh xạ lược đồ thực thể liên kết sang lược đồ quan hệ

# **MỘT SỐ VẤN ĐỀ CẦN QUAN TÂM KHI THIẾT KẾ LƯỢC ĐỒ E-R**

# ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA

- ❖ Cấu trúc của khóa chính cho tập các quan hệ phụ thuộc vào việc ánh xạ lực lượng liên kết. Xét lược đồ E-R sau:



=> Lược đồ này thể hiện một quan hệ M-M cho quan hệ **depositor** với thuộc tính **access-date** liên quan tới tập quan hệ giữa hai thực thể **CUSTOMER** và **ACCOUNT**.

Khóa chính của quan hệ này sẽ bao gồm hợp của các khóa chính của hai tập thực thể **CUSTOMER** và **ACCOUNT**.

# ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

❖ Để rõ hơn, xét lược đồ dữ liệu của hai tập thực thể:

CUSTOMER (customer-id, *customer-name*, *address*, *city*)

ACCOUNT (account-number, *balance*)

- Quan hệ M-M giữa hai tập thực thể **CUSTOMER** và **ACCOUNT** nghĩa là: một khách hàng có thể có nhiều tài khoản, và tương tự một tài khoản có thể được quản lý bởi nhiều khách hàng.
- Phép hợp các khóa chính của cả hai tập thực thể **CUSTOMER** và **ACCOUNT** sẽ xác định duy nhất một quan hệ giữa hai thực thể trong **CUSTOMER** và **ACCOUNT**.

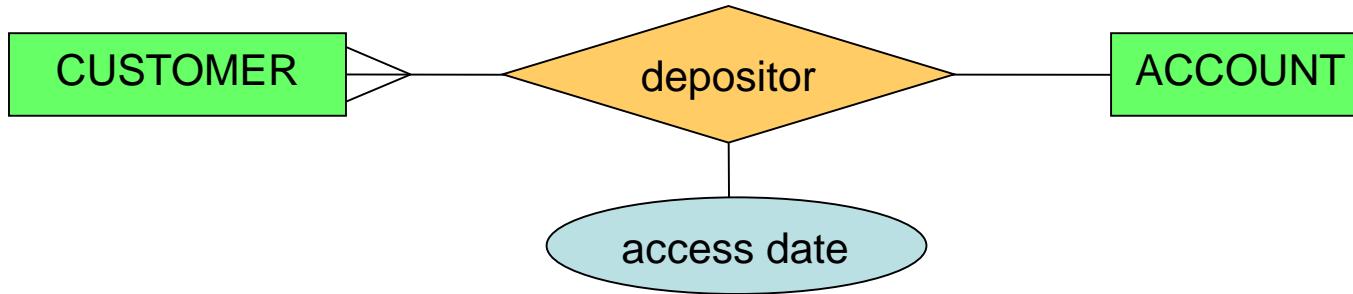
## ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- ❖ Để nhìn thấy lần nộp tiền cuối cùng (*last deposit*) vào một tài khoản cụ thể nào đó, cần xác định người nộp tiền vì với mỗi tài khoản có thể có một số người nộp tiền vào.
- ❖ Lược đồ cho mối quan hệ **depositor** như sau:  
Depositor (customer-id, account-number, access-date) .

# ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- Xét trường hợp một khách hàng chỉ được phép nộp tiền vào một tài khoản duy nhất.

=> quan hệ ***depositor*** là M:1 từ **CUSTOMER** tới **ACCOUNT**:



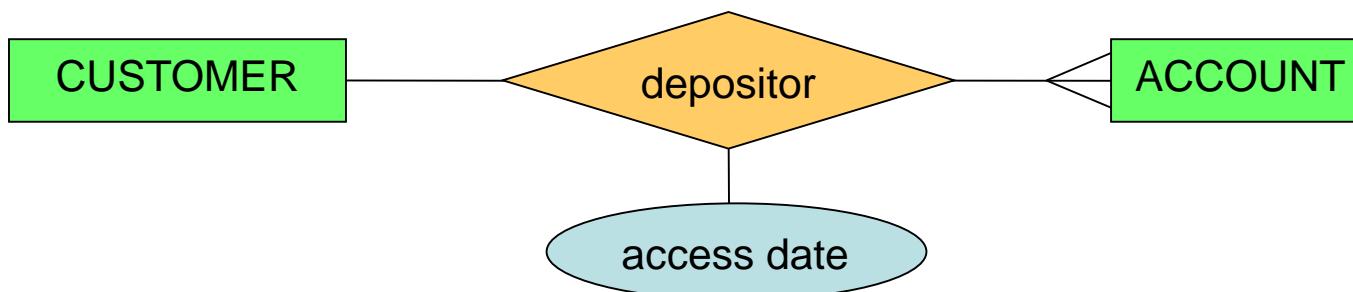
=> Khóa chính của quan hệ ***depositor*** sẽ chỉ bao gồm khóa chính của tập thực thể **CUSTOMER**.

# ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- ❖ Để rõ hơn, xét lược đồ dữ liệu của hai tập thực thể:  
**CUSTOMER** (*customer-id*, *customer-name*, *address*, *city*)  
**ACCOUNT** (*account-number*, *balance*)
  - Quan hệ M:1 nghĩa là một khách hàng chỉ có duy nhất một tài khoản. Như vậy, việc chỉ ra số tài khoản đó là không cần thiết.
  - Khóa chính của quan hệ **depositor** chỉ đơn giản là khóa chính của thực thể **CUSTOMER**. Một khách hàng xác định chỉ có thể thực hiện một lần nộp tiền gần nhất tới tài khoản duy nhất họ có thẻ truy nhập.
- ❖ Lược đồ của tập các quan hệ **depositor** như sau:  
**Depositor** (*customer-id*, *access-date*)

# ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

- Xét trường hợp quan hệ **depositor** là M:1 từ **ACCOUNT** tới **CUSTOMER**, nghĩa là: mỗi tài khoản được làm chủ bởi nhiều nhất là một khách hàng nhưng mỗi khách hàng có thể có nhiều hơn một tài khoản.



# ẢNH HƯỞNG CỦA ÁNH XẠ LỰC LƯỢNG LIÊN KẾT LÊN CÁC KHÓA (Cont.)

=> Trong trường hợp này, khóa chính của quan hệ **depositor** chỉ đơn giản bao gồm khóa chính của thực thể **ACCOUNT** vì chỉ có thẻ có nhiều nhất một lần nộp tiền gần nhất tới một tài khoản nào đó xác định, và chỉ có nhiều nhất một khách hàng có thể thực hiện việc nộp tiền.

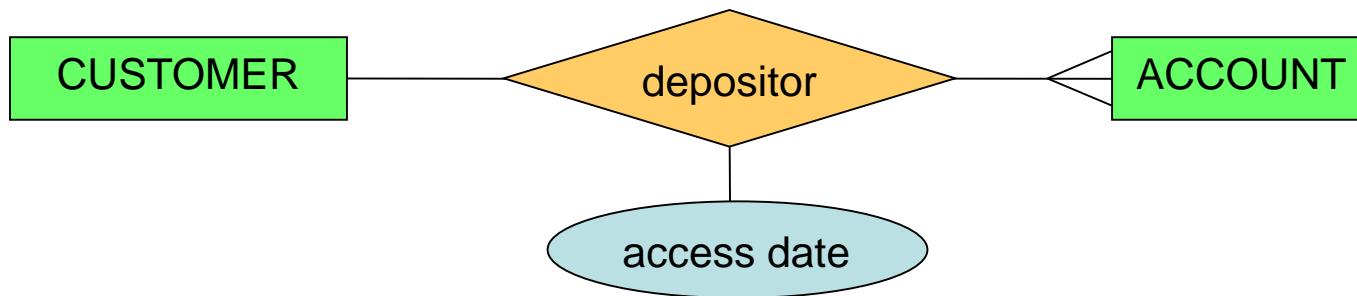
Không cần xác định khách hàng nào đã thực hiện nộp tiền vì chỉ có thẻ có một và chỉ một khách hàng có khả năng nộp tiền vào một tài khoản xác định.

❖ **Lược đồ cho mối quan hệ *depositor* như sau:**

Depositor (account-number, access-date)

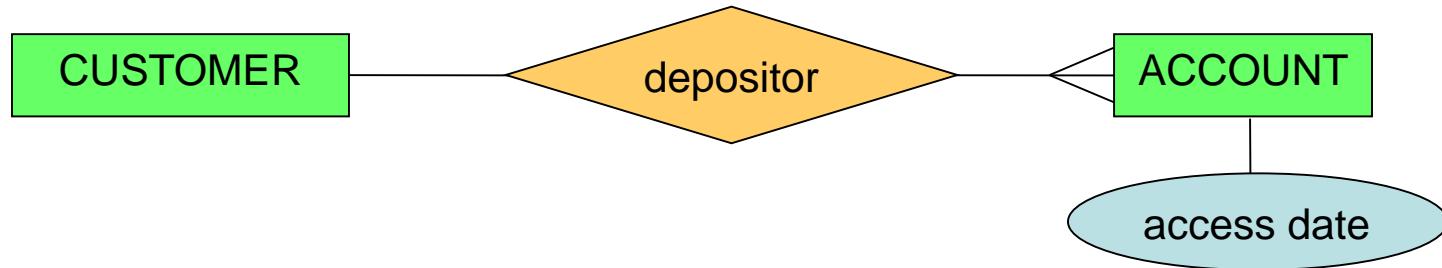
# ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ

- ❖ Các thuộc tính của một tập quan hệ dạng 1:1 hoặc 1:M thường được đặt vào trong các tập thực thể tham gia liên kết, hơn là được đặt vào bản thân tập các mối quan hệ đó.
  - **Ví dụ**, với quan hệ **depositor**:



# ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

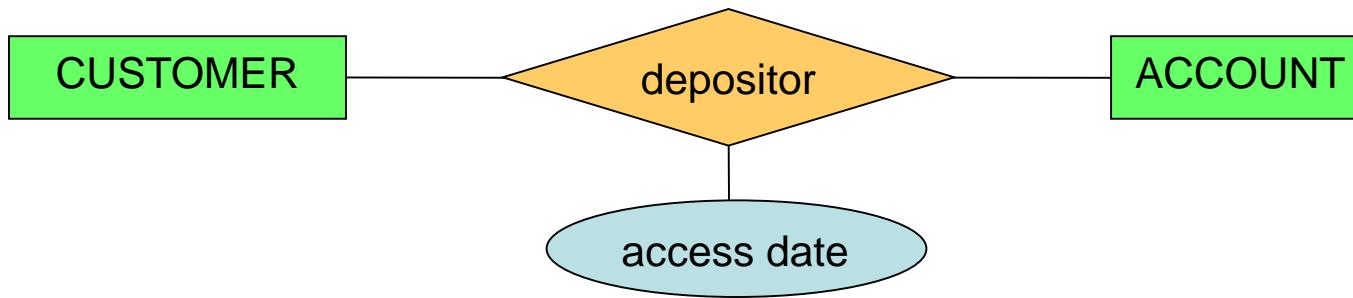
- Thuộc tính **access-date** có thể được đặt liên quan tới thực thể **ACCOUNT** mà không làm tổn thất thông tin:



=> Vì một tài khoản cụ thể thuộc sở hữu bởi nhiều nhất là một khách hàng, và tài khoản đó có thể có nhiều nhất một **access-date**.

# ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

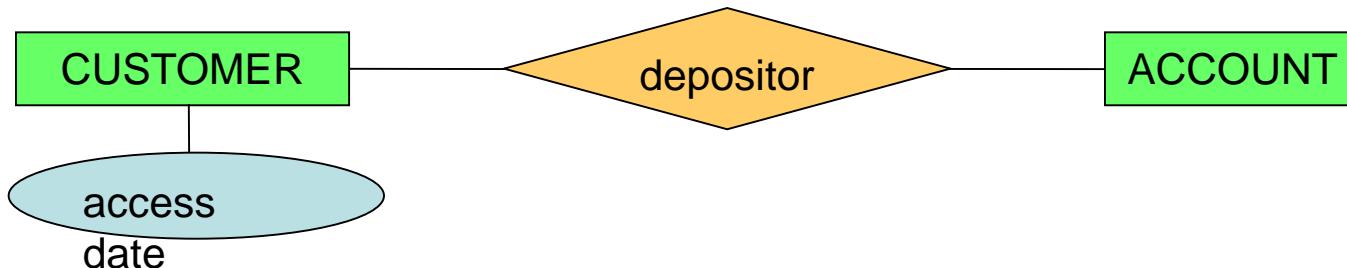
- **Xét trường hợp:** một tài khoản được làm chủ bởi nhiều nhất một khách hàng và một khách hàng chỉ có thể sở hữu duy nhất một tài khoản.



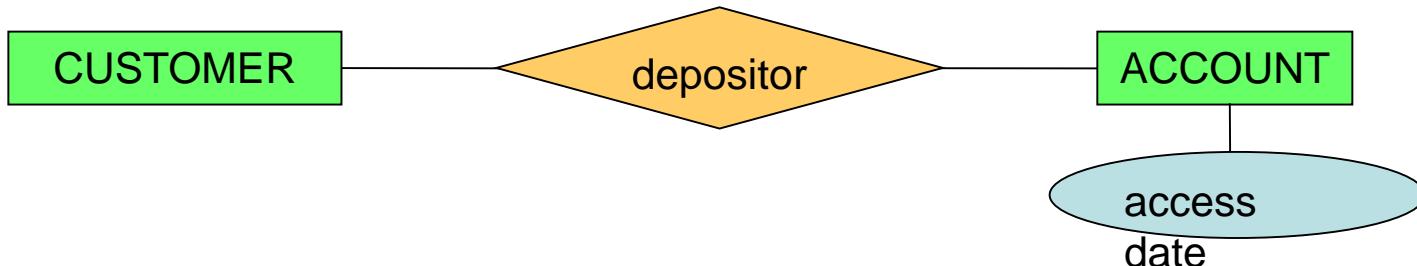
=> Thuộc tính **access-date** có thể gắn vào hoặc thực thể **CUSTOMER** hoặc tập thực thể **ACCOUNT** mà không làm tổn thất thông tin.

# ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

- Nếu thuộc tính **access-date** được lưu trữ với tập **CUSTOMER** thì nó phải tham chiếu tới lần truy nhập cuối cùng của khách hàng tới tài khoản duy nhất mà họ có.

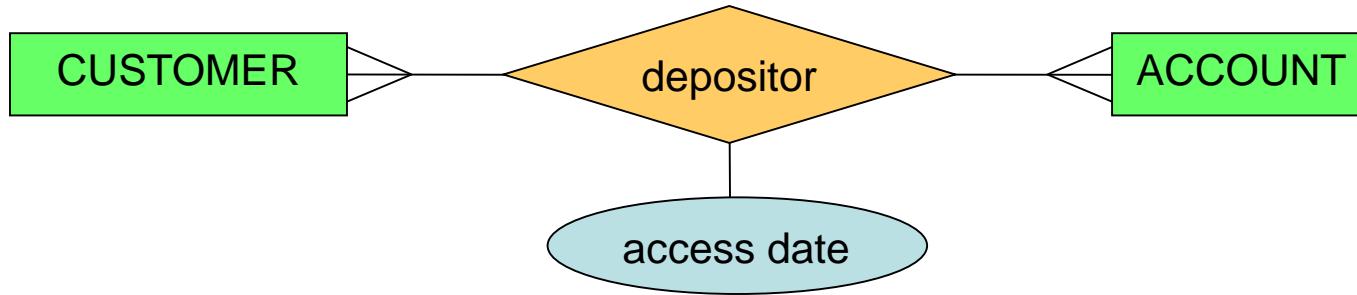


- Nếu thuộc tính **access-date** được lưu trữ trong thực thể **ACCOUNT** thì nó sẽ tham chiếu tới lần truy nhập cuối cùng tới tài khoản bởi người khách hàng duy nhất sở hữu nó.



# ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

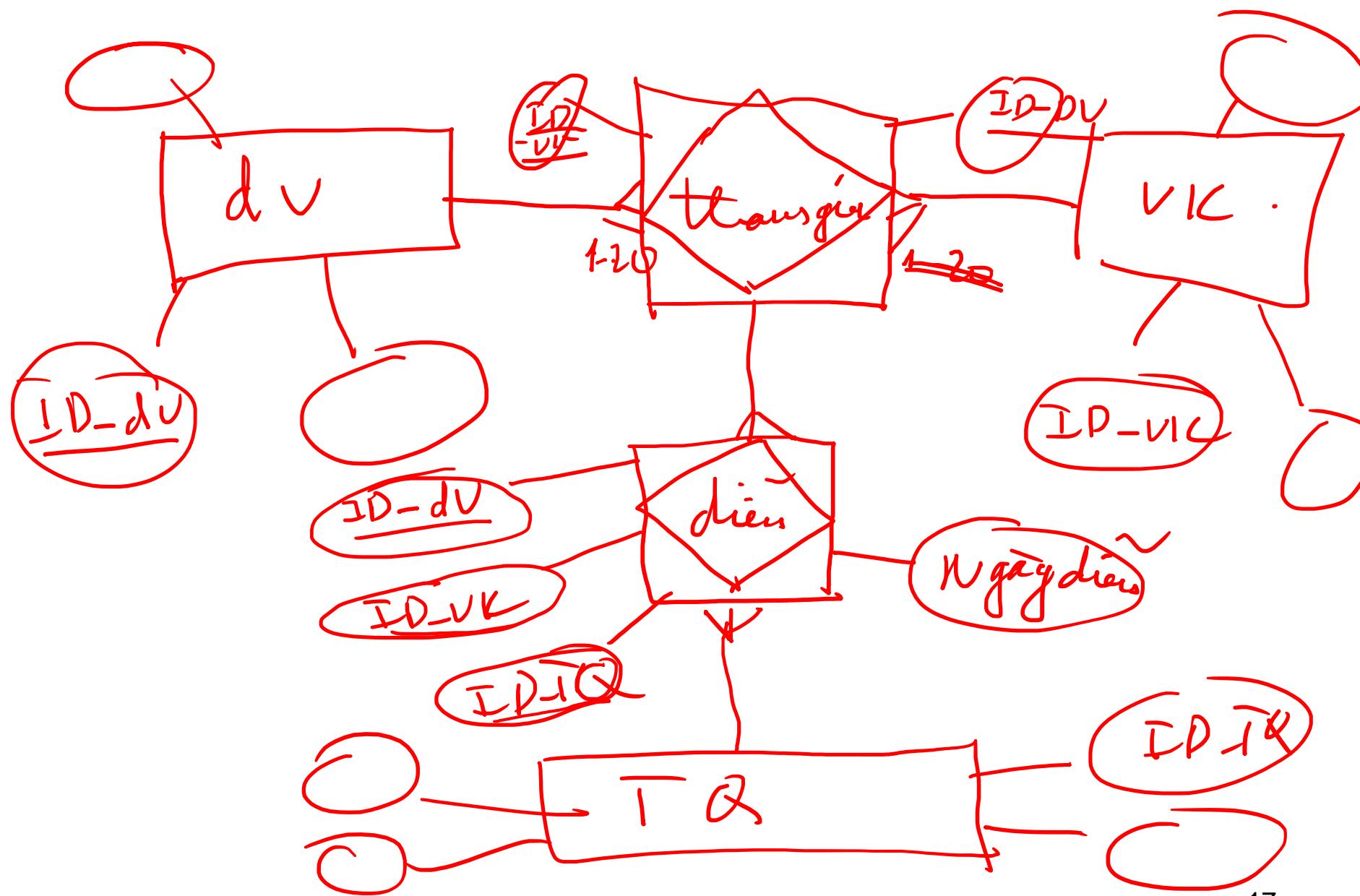
- Xét trường hợp: quan hệ **depositor** có ràng buộc N:N



=> Việc gắn thuộc tính **access-date** với bất kỳ tập thực thể tham gia nào cũng không mô hình hóa được tình huống này mà không làm tổn thất thông tin.

# ĐẶT VỊ TRÍ CHO CÁC THUỘC TÍNH CỦA QUAN HỆ (Cont.)

- ⇒ Nếu cần lưu trữ ngày truy nhập cuối cùng của một khách hàng cụ thể tới một tài khoản cụ thể thì thuộc tính **access-date** nhất thiết phải là một thuộc tính của tập quan hệ **depositor**, chứ không thể là thuộc tính của bất kỳ tập thực thể tham gia nào.
- Nếu **access-date** là một thuộc tính của **ACCOUNT** thì không thể xác định được khách hàng nào đã thực hiện việc chuyển tiền vào tài khoản đó.
  - Còn nếu **access-date** là một thuộc tính của **CUSTOMER**, thì cũng không thể xác định được tài khoản nào khách hàng đã truy nhập vào lần cuối.





# TẬP THỰC THỂ HAY CÁC THUỘC TÍNH

- ❖ Xét một tập thực thể:  
EMPLOYEE(*emp-name*, *telephone-number*, *age*)
- ❖ Nếu máy điện thoại telephone có thể được coi là một thực thể (với các thuộc tính *telephone-number*, *location*, *manufacturer*, *serial-num*, ...).
  1. Thực thể EMPLOYEE phải được định nghĩa lại như sau:  
EMPLOYEE (*emp-name*, *age*).
  2. Sau đó phải tạo ra một tập thực thể mới:  
TELEPHONE (*telephone-number*, *location*, *manufacturer*, *serial-num*, ...)
  3. Và phải tạo ra một tập các mối quan hệ để xác định mối liên hệ giữa các nhân viên và các máy điện thoại mà họ sở hữu:  
EMP-PHONE(*emp-name*, *telephone-number*, *age*, *location*, *manufacturer*, *serial-num*, ...).

# TẬP THỰC THỂ HAY CÁC THUỘC TÍNH (Cont.)

- ❖ Trong trường hợp mỗi nhân viên có một số điện thoại => coi máy điện thoại như một thuộc tính *telephone-number*.
- ❖ Trong trường hợp các nhân viên có thẻ sở hữu nhiều điện thoại => coi máy điện thoại như một thực thể (không xét đến trường hợp thuộc tính là đa trị).  
=> Sự phân biệt 2 vai trò thực thể và thuộc tính phụ thuộc vào cấu trúc ngũ cảnh trong thế giới thực cần mô hình hóa dữ liệu và dựa vào ngũ nghĩa liên quan đến các thuộc tính trong bài toán cụ thể.

# TẬP THỰC THỂ HAY TẬP QUAN HỆ

- ❖ Xét ví dụ ngân hàng: đã mô hình hóa khoản vay như một thực thể (**LOAN**).  
=> Cách khác: có thể mô hình hóa khoản vay như một mối quan hệ giữa khách hàng (**CUSTOMER**) và các chi nhánh (**BRANCHES**) của ngân hàng với **loan-number** và **amount** là các thuộc tính mô tả.  
(Trong trường hợp: một khoản vay chỉ được sở hữu bởi duy nhất một khách hàng và có liên hệ với duy nhất một chi nhánh ngân hàng).

# TẬP THỰC THẾ HAY TẬP QUAN HỆ (Cont.)

- ❖ Việc mô hình hóa theo cách trên không thuận lợi trong tình huống nhiều khách hàng cùng sở hữu chung một khoản vay.
  - => Cần định nghĩa các mối quan hệ riêng cho từng người sở hữu khoản vay chung. Sau đó, dùng lại tất cả các giá trị cho các thuộc tính ***loan-number*** và ***amount*** trong mỗi quan hệ này (hiển nhiên, các quan hệ phải có cùng giá trị cho các thuộc tính mô tả).
- ❖ Có 2 vấn đề phát sinh khi dùng lặp lại các giá trị:
  1. Dữ liệu được lưu trữ ở nhiều nơi.
  2. Việc cập nhật dữ liệu làm tăng khả năng không nhất quán dữ liệu.

# TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU

- ❖ Tập thực thể không đủ các thuộc tính để hình thành một khóa chính gọi là **tập thực thể yếu**. Tập thực thể có khóa chính được gọi là **tập thực thể mạnh**.
- ❖ **Ví dụ:** Xét tập thực thể trả tiền:

PAYMENT(*payment-number*, *payment date*, *payment amount*).

=> Mã số trả tiền (**payment-number**) thường là các số liên tiếp, bắt đầu từ 1 và được sinh ra riêng rẽ cho mỗi khoản nợ. Do đó, mặc dù mỗi thực thể PAYMENT là khác nhau, việc trả tiền cho các khoản nợ khác nhau có thể có cùng mã số **payment-number**.

=> tập PAYMENT không có khóa chính và chỉ là một tập thực thể yếu.

# TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU

(Cont.)

- ❖ Để một tập thực thể yếu có ý nghĩa, nó phải liên hệ với một tập thực thể khác, được gọi là **tập thực thể xác định** hay **tập thực thể sở hữu**.
- ❖ Mọi quan hệ giữa tập thực thể yếu với tập thực thể xác định được gọi là **mối quan hệ xác định**.
- ❖ Mọi quan hệ xác định là quan hệ M:1 từ tập thực thể yếu tới tập xác định và sự tham gia của tập thực thể yếu trong quan hệ là đầy đủ.

# TẬP THỰC THỂ MẠNH HAY TẬP THỰC THỂ YẾU

(Cont.)

- ❖ Mặc dù tập thực thể yếu không có khóa chính, nhưng có một phương thức để phân biệt tất cả các thực thể trong tập thực thể yếu phụ thuộc vào một thực thể mạnh cụ thể.

=> Tập các thuộc tính của một thực thể yếu cho phép phân biệt các thực thể được gọi là **thuộc tính phân biệt** (hay **khóa bán phần**).

**Ví dụ:** Thuộc tính phân biệt của tập thực thể yếu **PAYMENT** là ***payment-number***, vì với mỗi khoản nợ, một mã số trả tiền sẽ xác định duy nhất một lần trả tiền riêng biệt cho khoản nợ này.

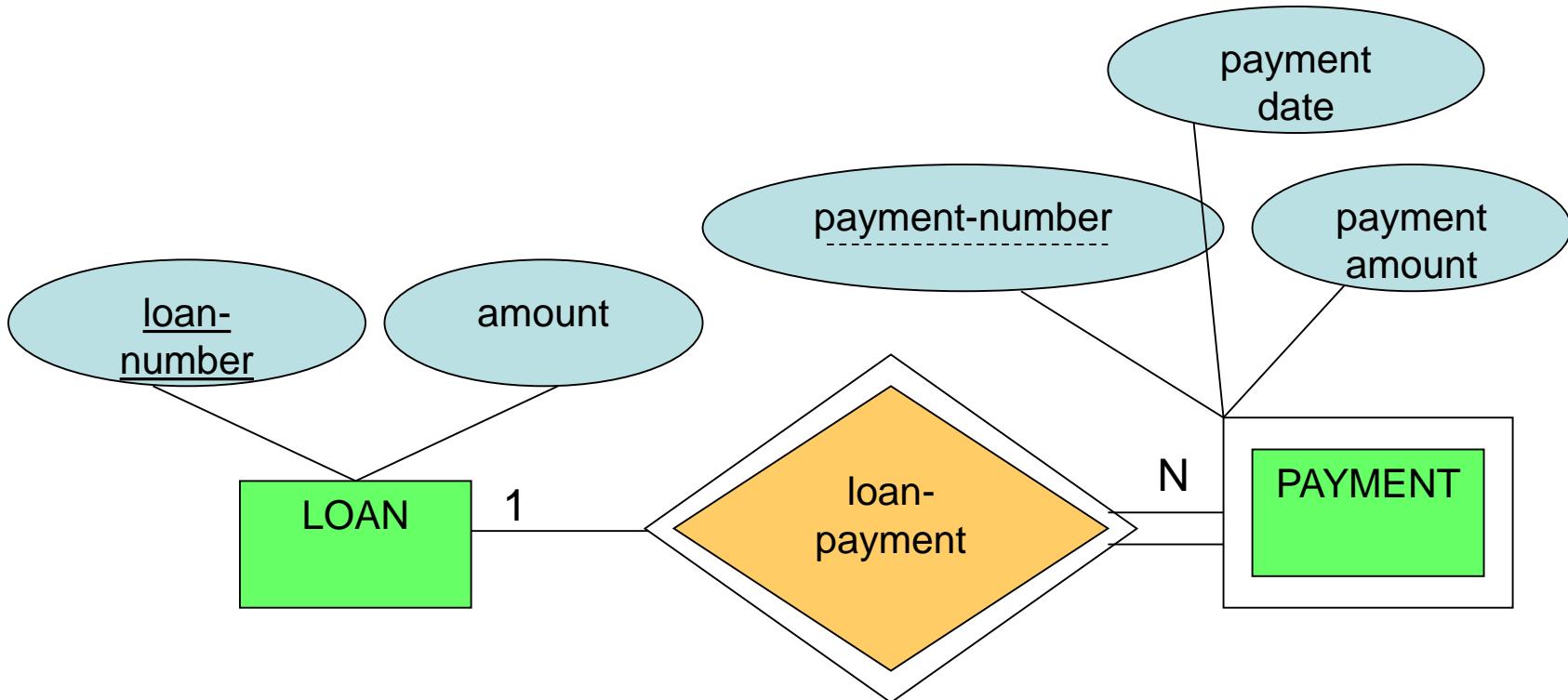
# TẬP THỰC THẺ MẠNH HAY TẬP THỰC THẺ YẾU

(Cont.)

- ❖ Khóa chính của một tập thực thẻ yếu được cấu thành bởi khóa chính của tập thực thẻ xác định và thuộc tính phân biệt của tập thực thẻ yếu.
- ❖ Xét ví dụ trên, khóa chính của tập thực thẻ **PAYMENT** sẽ là (***loan-number***, ***payment-number***), trong đó ***loan-number*** là khóa chính của tập thực thẻ xác định **LOAN** và ***payment-number*** là thuộc tính phân biệt của tập thực thẻ yếu **PAYMENT**.

# TẬP THỰC THẺ MẠNH HAY TẬP THỰC THẺ YẾU

(Cont.)



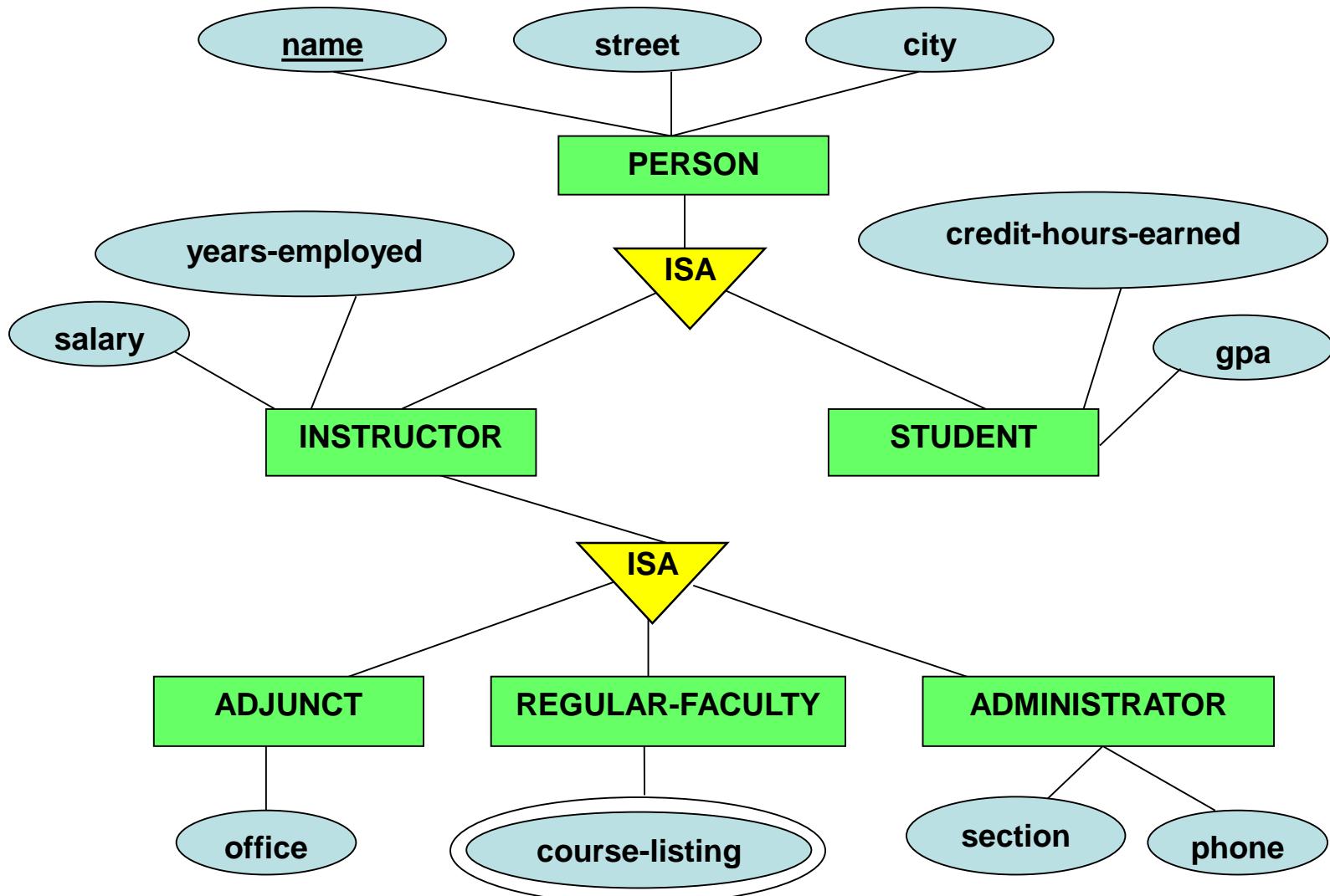
# CỤ THỂ HÓA (Specialization)

- ❖ **Cụ thể hóa** là quá trình thiết kế các phân nhóm trong một tập thực thể. Nghĩa là, phân biệt các tập thực thể con trong một tập thực thể khi chúng có các thuộc tính không giống nhau.
- ❖ **Ví dụ:** Xét tập thực thể người: PERSON(*name, street, city*).

Một **PERSON** có thể được phân chia nhỏ hơn thành **STUDENT** (sinh viên) hoặc **INSTRUCTOR** (giảng viên). Mỗi loại này được mô tả bởi một tập các thuộc tính bao gồm tất cả các thuộc tính của tập thực thể **PERSON**, và một số thuộc tính bổ sung riêng.

- Thực thể **STUDENT** có thể bổ sung thêm các thuộc tính **gpa** và **credit-hours-earned**;
- Thực thể **INSTRUCTOR** bổ sung thêm các thuộc tính **salary** và **years-employed**.

# CỤ THỂ HÓA (Cont.)



# TỔNG QUÁT HÓA (Generalization)

- ❖ Cụ thể hóa thể hiện cách tiếp cận thiết kế từ trên-xuống. Quá trình ngược lại, được tiến hành theo cách tiếp cận từ dưới-lên gọi là **tổng quát hóa**.  
=> Tổng quát hóa là nhiều tập thực thể được đồng bộ vào một tập thực thể ở mức cao hơn trên cơ sở các thuộc tính chung.

# TỔNG QUÁT HÓA (Generalization)

## ❖ Xét ví dụ trước:

PERSON là tập thực thể ở mức cao (gọi là *cha*), và INSTRUCTOR và STUDENT là các tập thực thể ở mức thấp (gọi là *con*).

- Đầu tiên có thể xác định 2 tập thực thể:  
*STUDENT(name, address, city, gpa, credit-hours-earned),*  
*INSTRUCTOR(name, address, city, salary, years-employed).*
- Sau đó, tìm điểm chung giữa các thuộc tính để xây dựng tập thực thể:  
*PERSON (name, address, city).*

## CỤ THỂ HÓA VÀ TỔNG QUÁT HÓA

- ❖ Cụ thể hóa và tổng quát hóa là 2 khái niệm ngược và có thể hoán đổi cho nhau trong việc thiết kế các lược đồ CSDL.
- ❖ Trong sơ đồ E-R, không có sự khác biệt giữa tổng quát hóa và cụ thể hóa mà chỉ khác nhau ở cách xem sơ đồ từ dưới lên hay từ trên xuống.

## CỤ THỂ HÓA VÀ TỔNG QUÁT HÓA (Cont.)

- ❖ Sự khác nhau của hai cách tiếp cận thường được thể hiện bởi điểm bắt đầu và mục đích tổng thể của chúng:
  - **Cụ thể hóa** xuất phát từ tập thực thể riêng biệt; nhấn mạnh sự khác nhau giữa các thực thể trong cùng tập bằng cách tạo ra các tập thực thể phân biệt nhau ở mức thấp hơn.
  - **Tổng quát hóa** xuất phát từ việc nhận ra một số tập thực thể có chung một số đặc tính. Trên cơ sở đó, tổng quát hóa tổng hợp các tập thực thể này thành một tập thực thể ở mức cao hơn. Quá trình tổng quát hóa nhấn mạnh tính tương đồng giữa các tập thực thể ở mức thấp hơn và giàu đi những khác biệt.

# KẾ THỪA THUỘC TÍNH

- ❖ Các tập thực thể ở mức thấp hơn kế thừa các thuộc tính từ tập thực thể ở mức cao hơn.  
**Ví dụ:** **INSTRUCTOR** và **STUDENT** đều kế thừa tất cả các thuộc tính của **PERSON**.
- ❖ Tập thực thể mức thấp hơn cũng kế thừa các mối quan hệ thuộc về tập thực thể mức cao hơn định nghĩa nó.
- ❖ **Cây phân cấp** các tập thực thể:
  - Thực thể ở mức cao nhất nằm trên đỉnh của cây phân cấp.
  - **Kế thừa đơn**: Tập thực thể ở mức thấp hơn có một mối quan hệ ISA.
  - **Đa kế thừa**: Tập thực thể mức thấp có nhiều hơn một mối quan hệ ISA.

# CÁC RÀNG BUỘC TỔNG QUÁT HÓA (CỤ THỂ HÓA)

- ❖ **Ràng buộc thứ nhất:** Xác định thực thể nào có thể là thành viên của tập thực thể mức thấp hơn.  
Thành viên được định nghĩa theo một trong 2 cách:
  - **Mệnh đề xác định:** Thành viên được đánh giá trên cơ sở xác định xem thực thể có thỏa mãn một mệnh đề (điều kiện) tường minh nào đó không.
  - **Người dùng xác định:** Tập các thực thể mức thấp do người dùng xác định không bị ràng buộc bởi các điều kiện thành viên mà người dùng cơ sở dữ liệu gán các thực thể tới tập thực thể nào đó.

# CÁC RÀNG BUỘC TỔNG QUÁT HÓA (CỤ THỂ HÓA) (Cont.)

- ❖ **Ràng buộc thứ hai:** Trong quá trình tổng quát hóa, liệu các thực thể có thuộc vào nhiều hơn một tập thực thể ở mức thấp hơn không.

Các tập thực thể ở mức thấp hơn có thể:

- **Không giao nhau:** Một thực thể không thể thuộc vào nhiều tập thực thể ở mức thấp hơn. (Trong mô hình E-R, ràng buộc không giao nhau được thể hiện bằng từ “disjoint” đặt cạnh biểu tượng hình tam giác).
- **Giao nhau:** Một thực thể có thể thuộc vào nhiều tập thực thể ở mức thấp hơn trong cùng một quá trình tổng quát hóa.

# CÁC RÀNG BUỘC

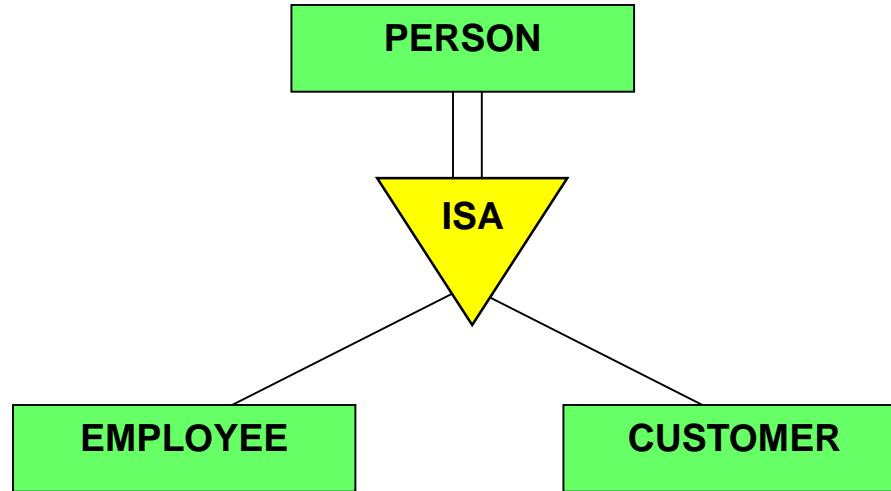
## TỔNG QUÁT HÓA (CỤ THỂ HÓA) (Cont.)

- ❖ **Ràng buộc thứ ba:** Là ràng buộc dựa trên tính toàn bộ, xác định xem một thực thể trong tập thực thể ở mức cao có thuộc vào một trong các tập thực thể ở mức thấp hơn hay không.

Có 2 loại ràng buộc này:

- **Tổng quát hóa/cụ thể hóa toàn bộ:** mỗi thực thể ở mức cao phải thuộc vào một thực thể ở mức thấp hơn.
- **Tổng quát hóa/cụ thể hóa một phần:** Tồn tại thực thể ở mức cao không thuộc vào tập thực thể nào ở mức thấp hơn. (Đây là trường hợp mặc định).

# VÍ DỤ ERD VỚI CÁC RÀNG BUỘC

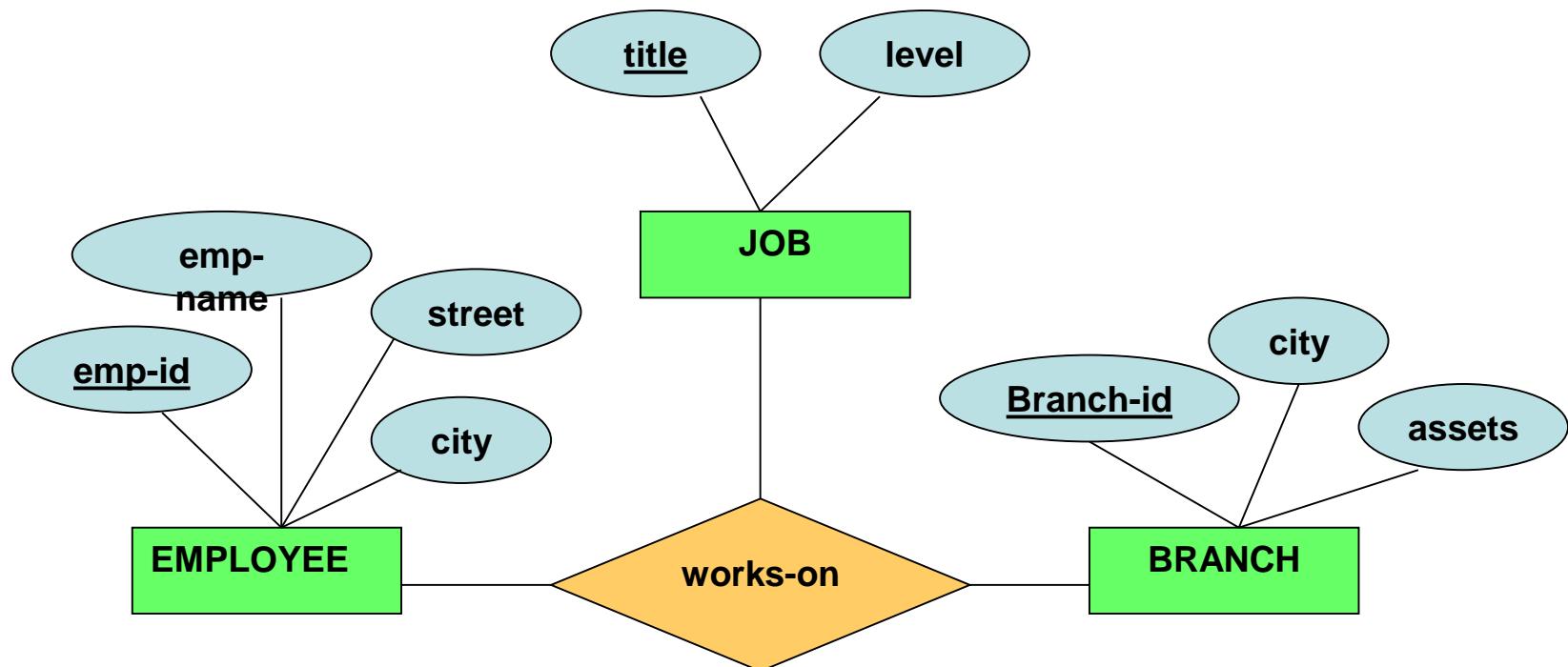


Tổng quát hóa toàn bộ và có giao nhau. Nghĩa là, mỗi người trong công ty xuất hiện như là một nhân viên hoặc là một khách hàng hoặc cũng có thể là cả hai.

# KẾT HỢP

- ❖ Hạn chế của lược đồ E-R: không thể biểu diễn các mối quan hệ trong các mối quan hệ.

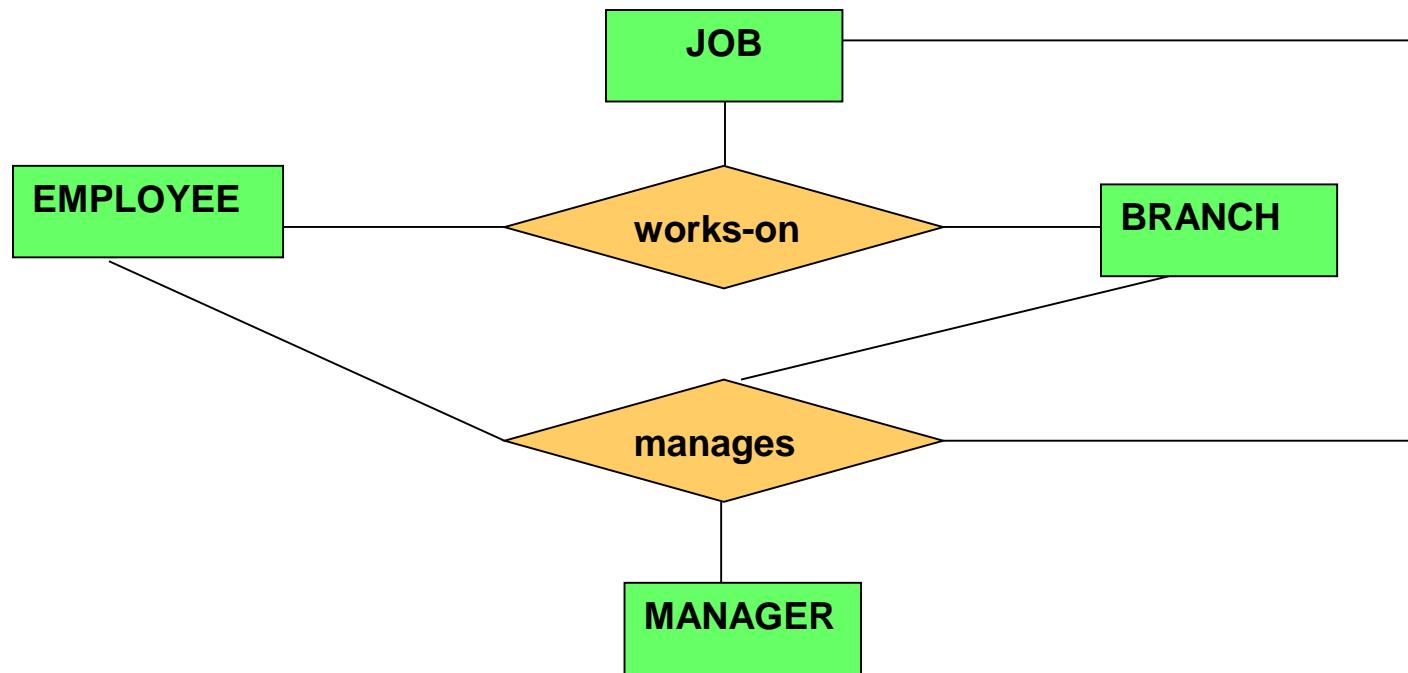
Xét quan hệ ba ngôi **works-on** giữa **EMPLOYEE** (*nhan vien*), **BRANCH** (*chi nhánh ngân hàng*) và **JOB** (*công việc*):



# KẾT HỢP (Cont.)

Giả sử muốn ghi nhận việc quản lý từng công việc được thực hiện bởi một nhân viên tại một chi nhánh; nghĩa là muốn lưu trữ người quản lý cho quan hệ ba ngôi (**EMPLOYEE**, **BRANCH**, **JOB**).

=> Một cách để thực hiện: tạo ra một quan hệ bốn ngôi với một tập thực thể **MANAGER**.



## KẾT HỢP (Cont.)

Câu hỏi đặt ra:

Tại sao không thể hiện mối quan hệ hai ngôi giữa MANAGER và EMPLOYEE ?

=> Câu trả lời:

Một quan hệ hai ngôi sẽ không cho phép thể hiện liên kết (BRANCH, JOB) của một EMPLOYEE (nhân viên) được quản lý bởi người quản lý.

# KẾT HỢP (Cont.)

## Nhận xét:

- Trong lược đồ E-R mô hình hóa tình huống trên, tập các mối quan hệ ***works-on*** và ***manages*** có thể kết hợp thành một tập quan hệ duy nhất.  
=> Tuy nhiên, không thể thực hiện việc kết hợp vì một bộ ba (**EMPLOYEE**, **BRANCH**, **JOB**) có thể không có người quản lý.
- Có thông tin dư thừa trong lược đồ E-R trên vì từng bộ ba (**EMPLOYEE**, **BRANCH**, **JOB**) của quan hệ ***manages*** cũng giống của ***works-on***.

# KẾT HỢP (Cont.)

## Nhận xét (cont.):

- Nếu người quản lý là một giá trị chứ không phải là một thực thể, thì có thể biến **MANAGER** thành một thuộc tính đa trị của quan hệ **works-on**.  
=> Biến đổi này sẽ làm việc tìm kiếm khó khăn hơn (cả về logic lẫn về chi phí thực hiện). (Ví dụ, bộ ba **employee-branch-job** sẽ do người quản lý nào chịu trách nhiệm?).

## KẾT HỢP (Cont.)

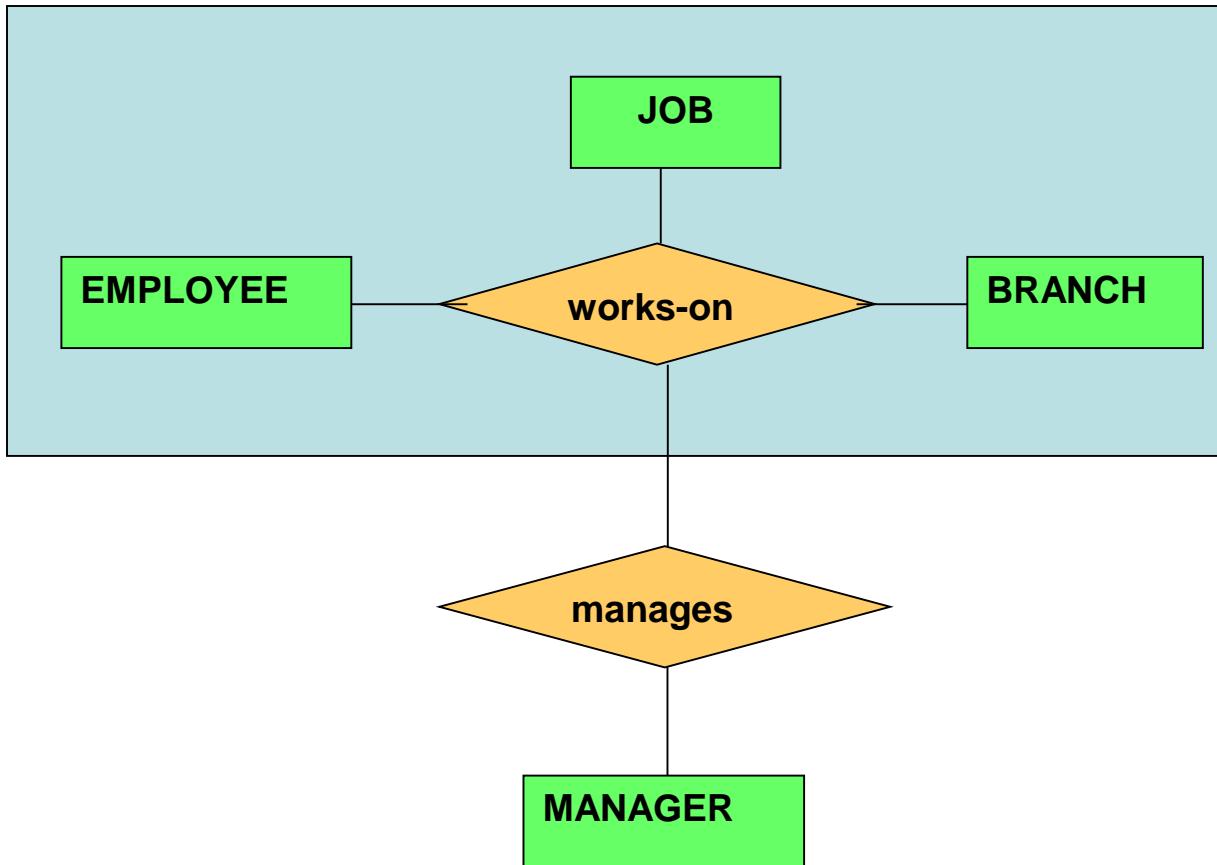
=> Cách tốt nhất để mô hình hóa là sử dụng **kết hợp** (hay **tích hợp**). *Kết hợp là một sự trừu tượng thông qua việc coi các mối quan hệ như là các thực thể ở mức cao.*

=> Xét ví dụ trên:

Có thể coi tập quan hệ **works-on** (liên quan tới các tập thực thể **EMPLOYEE**, **BRANCH** và **JOB**) như một tập thực thể ở mức cao, được đặt tên là **WORKS-ON**.

Tạo một quan hệ hai ngôi **manages** giữa **WORKS-ON** và **MANAGER** để thể hiện ai quản lý các công việc này.

# KẾT HỢP (Cont.)



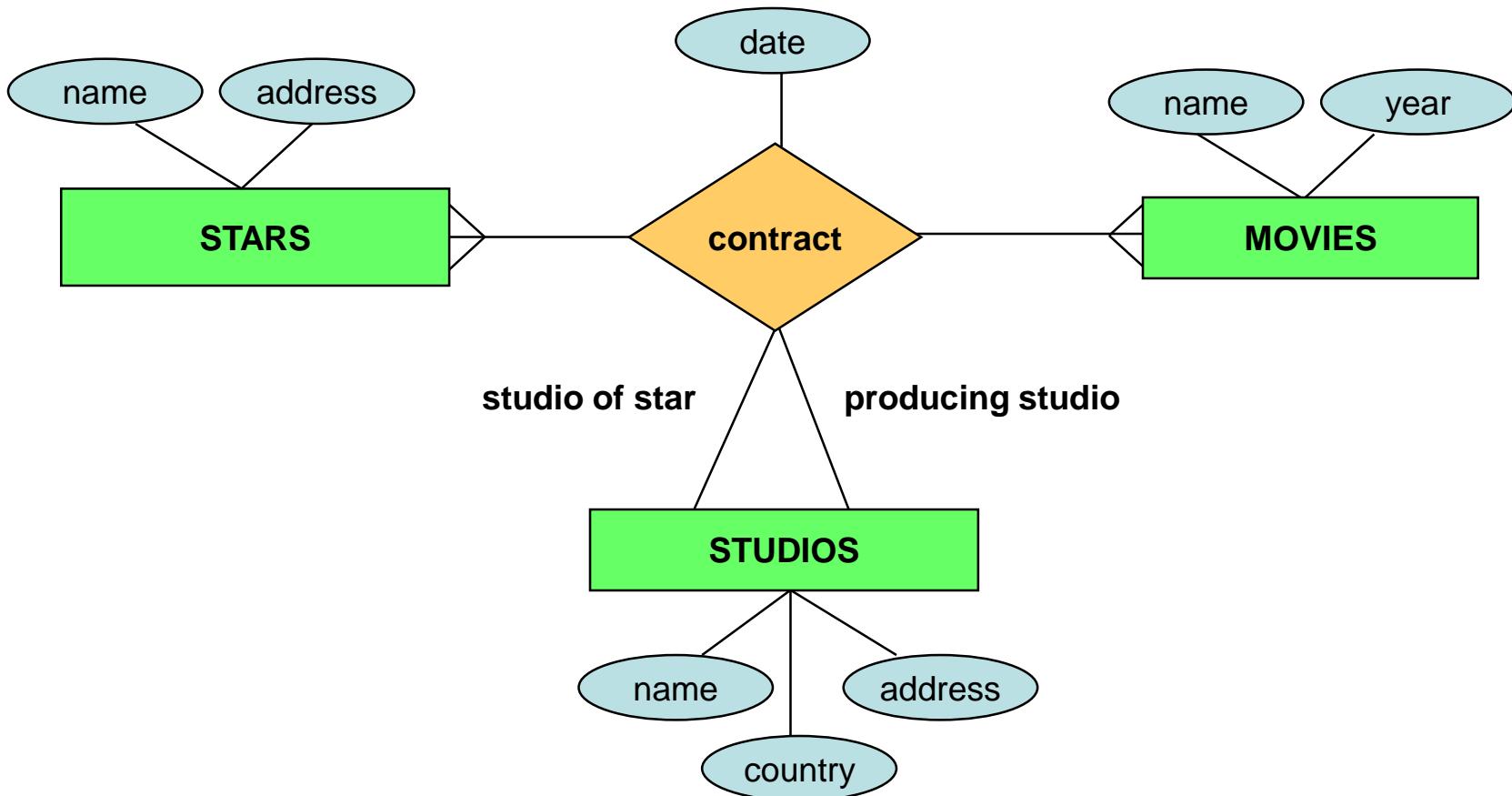
Sự kết hợp trong mô hình E-R

# QUAN HỆ NHIỀU NGÔI

- ❖ Hầu hết các quan hệ đã xét đều là quan hệ hai ngôi, liên quan tới hai tập thực thể.
- ❖ Các quan hệ liên quan tới nhiều hơn hai tập thực thể có thể được chuyển đổi thành một tập các quan hệ hai ngôi, dạng nhiều-một.  
=> Điều này rất có ích vì trong mô hình E-R không hạn chế các quan hệ tới quan hệ hai ngôi nhưng các mô hình dữ liệu khác có hạn chế này.

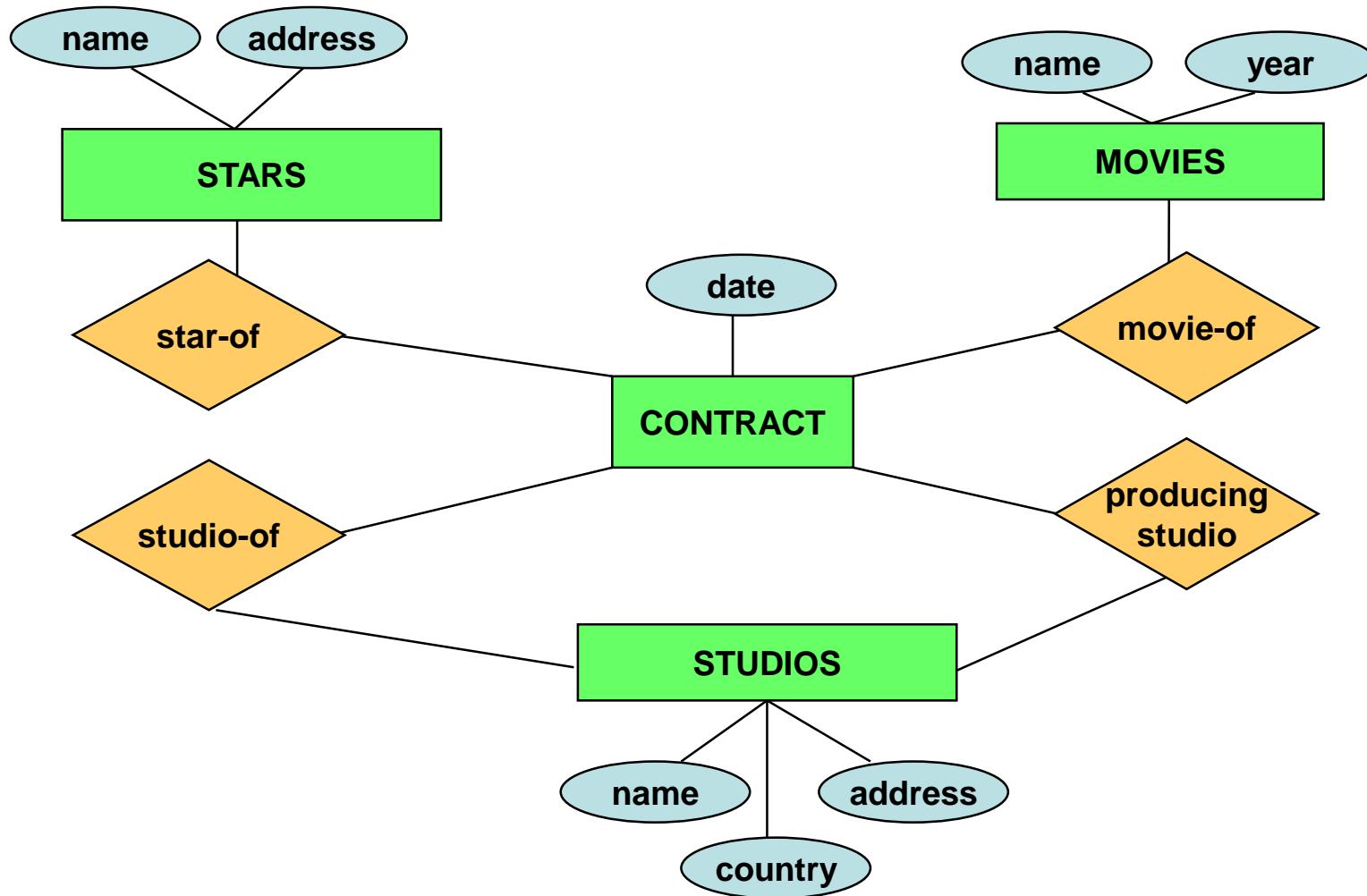
# QUAN HỆ NHIỀU NGÔI (Cont.)

Xét ví dụ mô hình E-R với quan hệ nhiều ngôi:



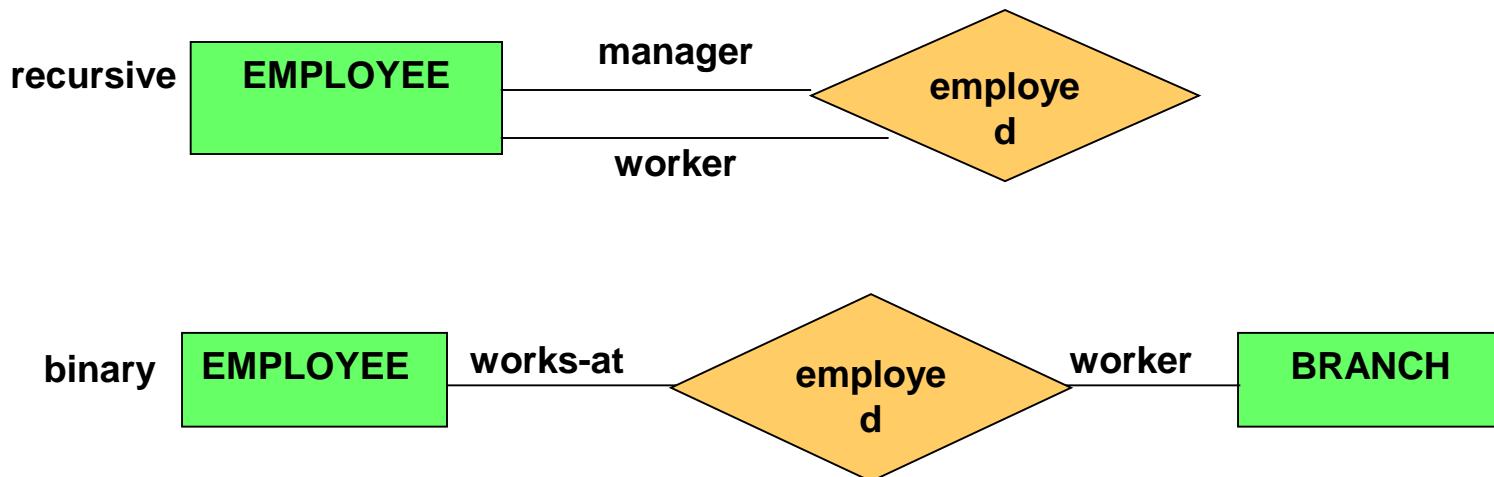
# QUAN HỆ NHIỀU NGÔI (Cont.)

Chuyển quan hệ nhiều ngôi thành một tập các quan hệ 2 ngôi:



# SƠ ĐỒ E-R VỚI CÁC CHỈ THỊ VAI TRÒ

- ❖ Các vai trò trong lược đồ E-R được biểu diễn bằng việc gán nhãn lên các đường kết nối giữa các tập thực thể và các tập quan hệ.
- ❖ Các vai trò này có thể được xác định cho các mối quan hệ đệ quy, nhị phân, và không nhị phân.



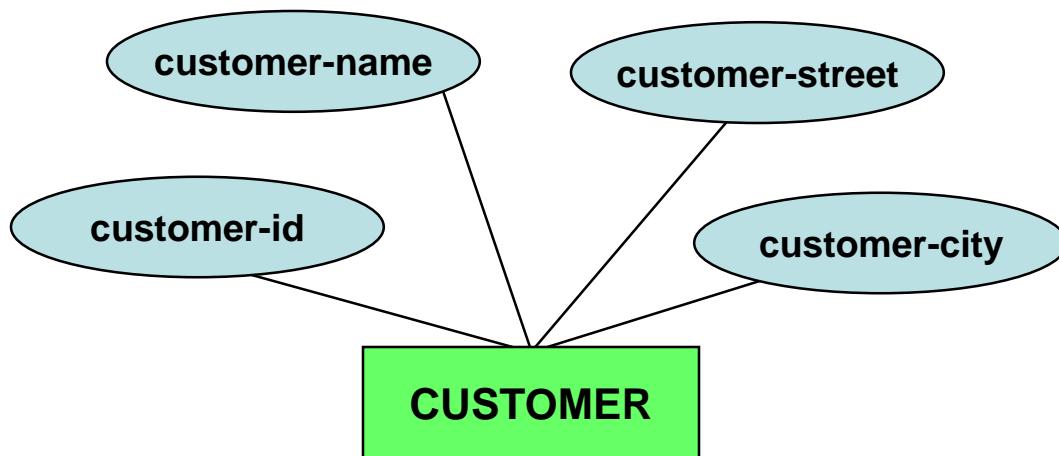
# NGÔN NGỮ UML (Unified Modeling Language)

Một số các thành phần của UML:

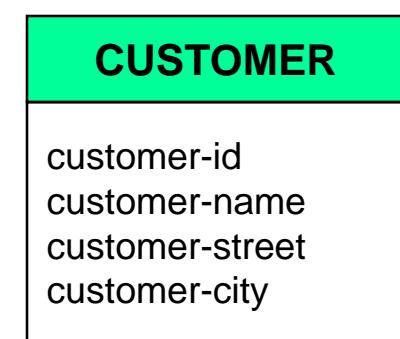
1. **Sơ đồ lớp**: Sơ đồ lớp tương tự như sơ đồ E-R.
2. **Sơ đồ use case**: được dùng để thể hiện sự tương tác giữa người dùng và hệ thống, các bước của công việc mà người dùng cần thực hiện ([ví dụ](#), việc rút tiền từ một tài khoản ngân hàng hoặc việc đăng ký một khóa học).
3. **Sơ đồ hoạt động**: thể hiện luồng công việc giữa các thành phần trong hệ thống.
4. **Sơ đồ cài đặt**: thể hiện các thành phần của hệ thống và sự kết nối giữa chúng ở cả hai khía cạnh phần mềm và phần cứng.

# SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML

Các tập thực thể và thuộc tính



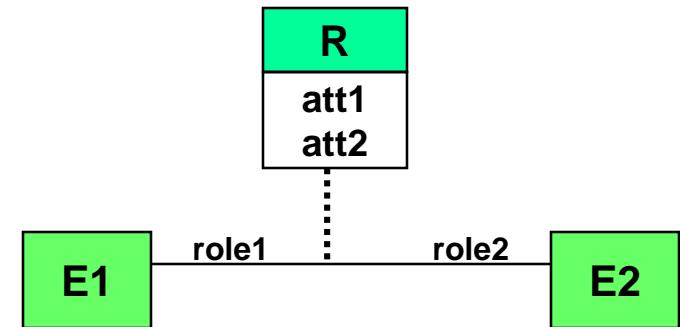
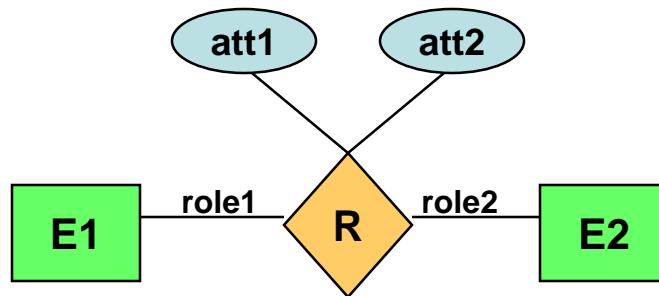
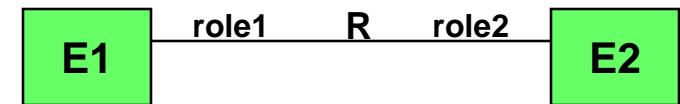
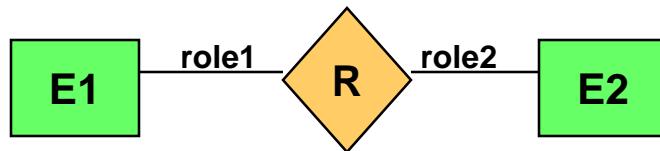
Sơ đồ E-R



Sơ đồ lớp UML

# SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Các quan hệ

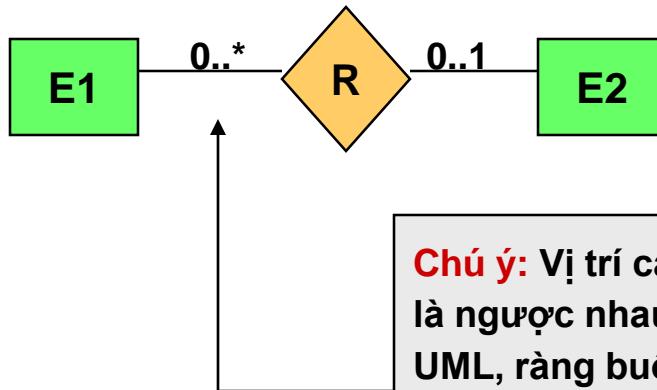


Sơ đồ E-R

Sơ đồ lớp UML

# SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Các ràng buộc về lực lượng liên kết



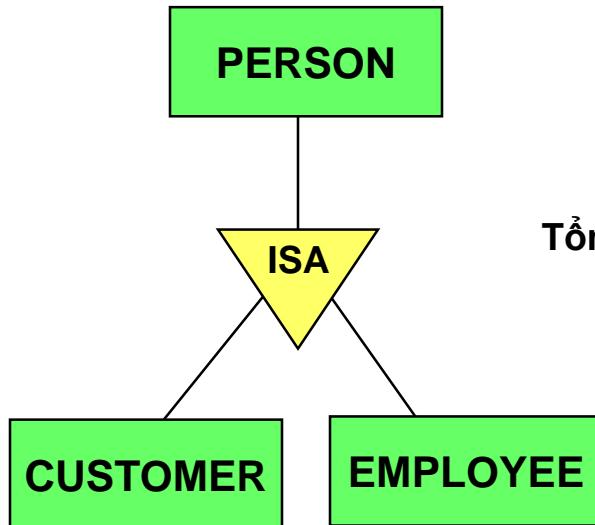
**Chú ý:** Vị trí các ràng buộc lực lượng liên kết là ngược nhau giữa 2 mô hình. Trong mô hình UML, ràng buộc 0..1 ở bên trái nghĩa là thực thể E2 có thể tham gia vào nhiều nhất một mối quan hệ, trong khi mỗi thực thể E1 có thể tham gia vào nhiều quan hệ; nói cách khác, quan hệ này là nhiều-một từ E2 tới E1.

Sơ đồ E-R

Sơ đồ lớp UML

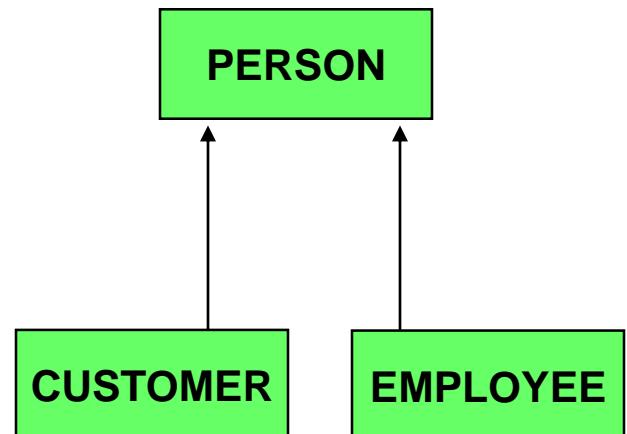
# SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Tổng quát hóa và cụ thể hóa



Tổng quát hóa giao nhau

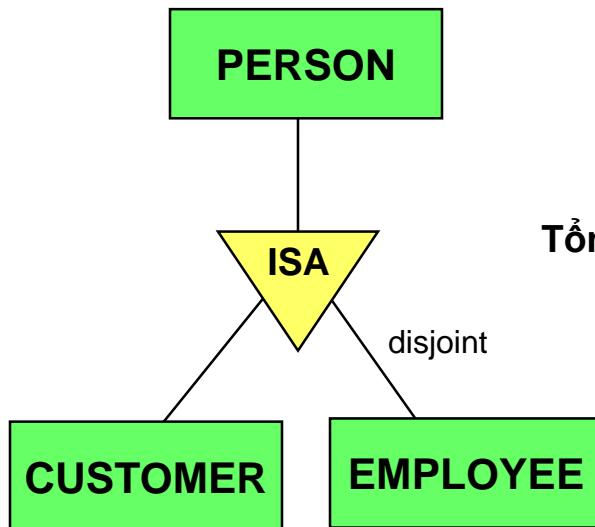
Sơ đồ E-R



Sơ đồ lớp UML

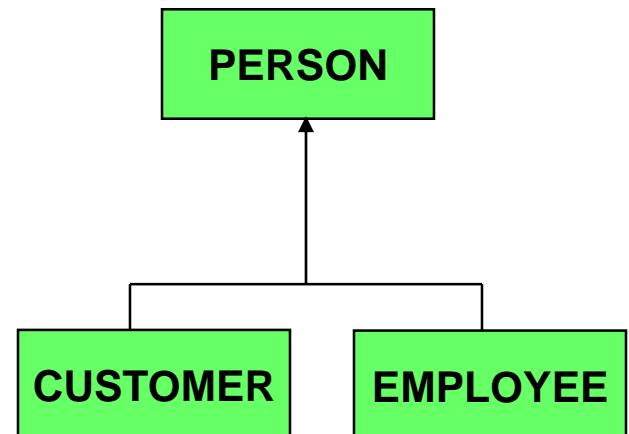
# SỰ TƯƠNG ỨNG GIỮA SƠ ĐỒ E-R VÀ SƠ ĐỒ LỚP UML (Cont.)

Tổng quát hóa và cụ thể hóa



Sơ đồ E-R

Tổng quát hóa phân tách



Sơ đồ lớp UML

# RÀNG BUỘC TOÀN VẸN THAM CHIẾU

- ❖ Ràng buộc toàn vẹn tham chiếu có thể được hiểu đơn giản là việc đảm bảo một thuộc tính nào đó có một giá trị khác rỗng. Tuy nhiên, các ràng buộc toàn vẹn tham chiếu thường liên quan tới các quan hệ giữa các tập thực thể.
- ❖ Ràng buộc toàn vẹn tham chiếu yêu cầu mỗi thực thể “được tham chiếu tới” bởi một quan hệ phải tồn tại trong cơ sở dữ liệu.

## RÀNG BUỘC TOÀN VẸN THAM CHIẾU (Cont.)

- ❖ Các phương pháp được sử dụng để đảm bảo tính ràng buộc toàn vẹn tham chiếu:
  - Không được phép xóa bỏ một thực thể được tham chiếu đến.
  - Nếu một thực thể được tham chiếu bị xóa bỏ thì tất cả các bản ghi tham chiếu tới thực thể đó cũng bị xóa.

# **CHƯƠNG 2.**

# **CÁC MÔ HÌNH DỮ LIỆU**

**(Phần 3)**

# CÁC MÔ HÌNH DỮ LIỆU

- ❖ Giới thiệu
- ❖ Quá trình thiết kế một CSDL
- ❖ Mô hình thực thể liên kết E-R
- ❖ Một số vấn đề cần quan tâm khi thiết kế mô hình E-R
- ❖ Mô hình dữ liệu quan hệ
- ❖ Ánh xạ mô hình thực thể liên kết sang mô hình quan hệ

# MÔ HÌNH DỮ LIỆU QUAN HỆ

(The Relational Data Model)

# MÔ HÌNH DỮ LIỆU QUAN HỆ

- ❖ Mô hình dữ liệu quan hệ được phát triển dựa trên khái niệm về quan hệ toán học.
- ❖ Nhà khoa học đề xuất ra mô hình quan hệ tên là Codd, là một nhà toán học. Mô hình này liên quan chủ yếu đến lý thuyết tập hợp và logic mệnh đề.

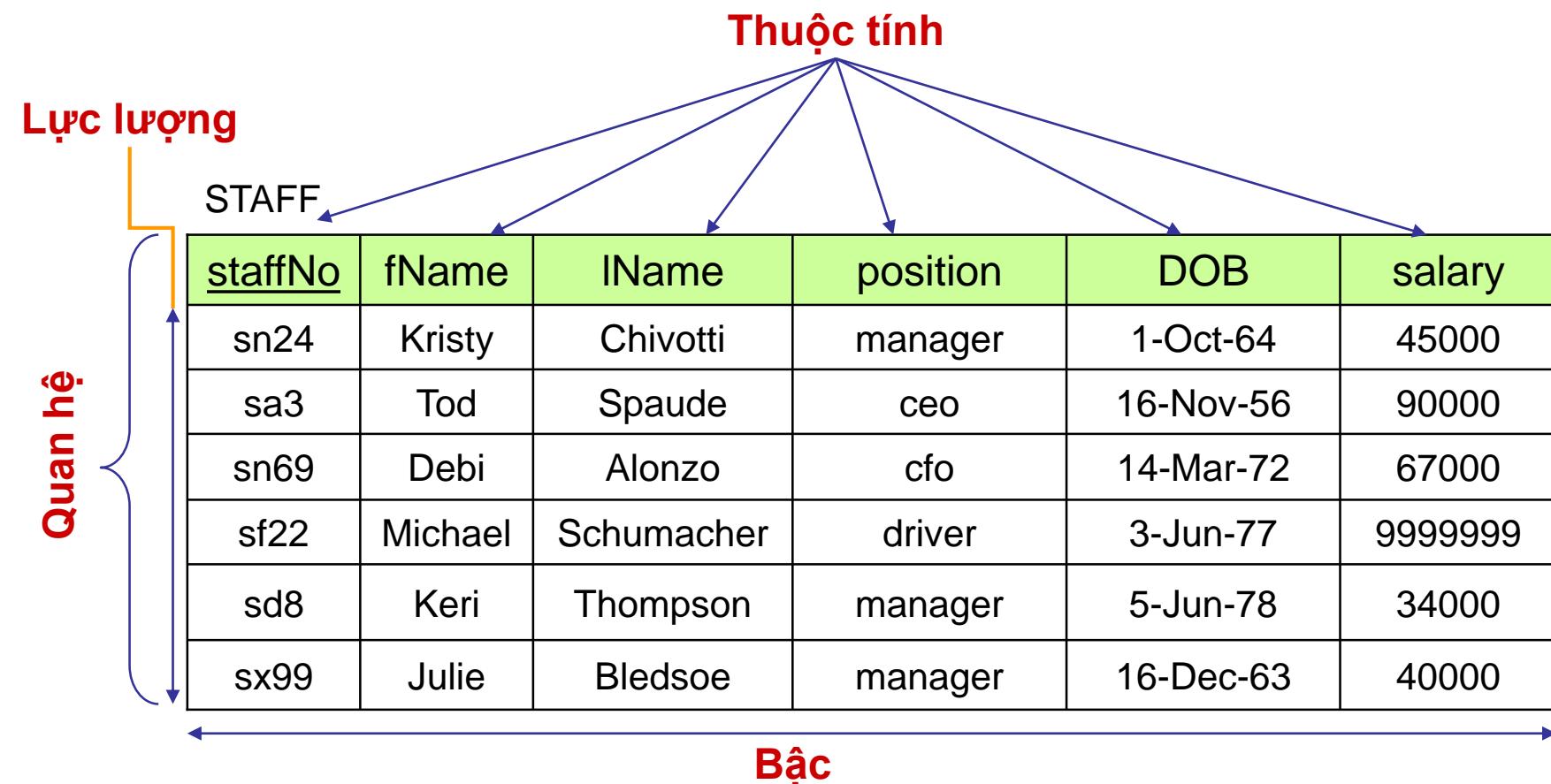
# CÁC KHÁI NIỆM CƠ BẢN

- ❖ **Quan hệ:** là một bảng (ma trận) với các hàng và các cột, lưu giữ thông tin về các đối tượng được mô hình hóa trong CSDL.
- ❖ **Thuộc tính:** là các cột được đặt tên trong một quan hệ. Mỗi thuộc tính là một đặc tính của một thực thể (hay một quan hệ) được mô hình hóa trong CSDL. Các thuộc tính có thể xuất hiện theo bất kỳ thứ tự nào trong quan hệ.
- ❖ **Miền giá trị:** là một tập các giá trị có thể có của một hoặc nhiều thuộc tính. Mỗi thuộc tính được xác định trên một miền giá trị.

# CÁC KHÁI NIỆM CƠ BẢN (Cont.)

- ❖ **Bộ:** là một hàng của một quan hệ. Các bộ có thể xuất hiện theo bất kỳ thứ tự nào trong quan hệ.
- ❖ **Bậc (cấp):** của một quan hệ là số lượng các thuộc tính mà nó có
- ❖ **Lực lượng:** là số lượng các bộ mà một quan hệ có.
- ❖ **Cơ sở dữ liệu quan hệ:** là một tập hợp các quan hệ được chuẩn hóa với các tên phân biệt nhau.

# VÍ DỤ VỀ QUAN HỆ



# MIỀN GIÁ TRỊ CHO VÍ DỤ TRÊN

Thuộc tính	Tên miền	Ý nghĩa	Định nghĩa miền
<b>staffNo</b>	staffnumbers	Tập của tất cả các số hiệu có thể có của nhân viên	Kiểu ký tự: kích cỡ 4, phải bắt đầu bằng chữ s.
<b>fName, lName</b>	name	Tập tất cả các tên có thể có của một người	Kiểu ký tự: kích cỡ 20
<b>DOB</b>	date	Ngày sinh của một người	Kiểu ngày tháng: trong khoảng từ 1-Jan-20, Khuôn dạng: dd-mm-yy
<b>salary</b>	salaries	Các giá trị có thể có của lương nhân viên	Dạng tiền tệ: 7 ký tự, trong khoảng 10,000-9,999,999
<b>position</b>	alljobs	Tập tất cả các vị trí có thể có của một nhân viên trong công ty	Chọn một trong các tập: {ceo, cfo, coo, manager, asst. manager, driver, secretary}

# CÁC THUẬT NGỮ CÓ THỂ DÙNG TƯƠNG ĐƯƠNG TRONG MÔ HÌNH QUAN HỆ

Thuật ngữ chuẩn	Lựa chọn 1	Lựa chọn 2
Quan hệ	Bảng	Tập
Bộ	Hàng	Bản ghi
Thuộc tính	Cột	Trường

# CÁC ĐẶC TÍNH CỦA MỘT QUAN HỆ

1. Quan hệ có một tên gọi phân biệt với tên của các quan hệ khác trong lược đồ quan hệ.
2. Mỗi thuộc tính có một tên gọi riêng.
3. Mỗi thuộc tính có một miền giá trị.
4. Mỗi thuộc tính chứa một giá trị nguyên tố.
5. Các bộ là phân biệt nhau (không có hai bộ nào giống hệt nhau).
6. Thứ tự của các thuộc tính không quan trọng.
7. Thứ tự của các bộ cũng không quan trọng (về mặt lý thuyết).  
=> Tuy nhiên, trong thực tế, thứ tự này có thể ảnh hưởng đến hiệu quả truy nhập vào các bộ.

# LƯỢC ĐỒ QUAN HỆ VÀ THỂ HIỆN QUAN HỆ

- ❖ **Lược đồ** bao gồm tên và các thuộc tính cho quan hệ và thường không thay đổi.
- ❖ Một **thể hiện** của quan hệ là một tập các bộ của quan hệ và có thể thay đổi thường xuyên.

Hầu hết các quá trình cập nhật, chèn thêm hay xóa các bộ sẽ làm thay đổi thể hiện của quan hệ.

Một **CSDL hiện thời** (snapshot database) thể hiện trạng thái hiện tại của thế giới thực tại một thời điểm. Nếu thế giới thực thay đổi, CSDL cũng sẽ thay đổi theo để duy trì biểu diễn đó.

# CÁC QUAN HỆ TƯƠNG ĐƯƠNG

A	B	C
1	2	3
3	2	1
4	4	1
2	1	3

Một thể hiện của quan hệ

B	C	A
2	3	1
2	1	3
4	1	4
1	3	2

Một thể hiện của quan hệ

A	B	C
4	4	1
3	2	1
1	2	3
2	1	3

Một thể hiện của quan hệ

Các thể hiện của  
quan hệ tương  
đương nhau

A	B	C
4	4	1
3	2	1
1	2	4
2	1	3

Một thể hiện của quan hệ

Thể hiện này không  
tương đương với  
3 thể hiện trên

# **ÁNH XẠ MÔ HÌNH THỰC THẾ LIÊN KẾT SANG MÔ HÌNH QUAN HỆ**

# THIẾT KẾ LOGIC

- ❖ **Thiết kế logic:** là quá trình chuyển đổi thiết kế mức khái niệm thành các lược đồ CSDL quan hệ.
  - *Đầu vào là các sơ đồ E-R và đầu ra là các lược đồ quan hệ.*
- ❖ Việc ánh xạ sơ đồ E-R thành các quan hệ là một quá trình tương đối đơn giản với việc định nghĩa một tập các luật. Trong thực tế, nhiều **công cụ CASE** (các công cụ trợ giúp cho Công nghệ phần mềm) có thể thực hiện tự động một số bước trong quá trình chuyển đổi.

# THIẾT KẾ LOGIC (Cont.)

- ❖ Cần hiểu rõ các bước chuyển đổi vì 3 lý do sau:
  1. Các công cụ CASE thường không thể mô hình hóa các quan hệ dữ liệu phức tạp, ([ví dụ](#), các quan hệ ba ngôi và các quan hệ giữa các lớp cha/lớp con.) Việc này được thực hiện thủ công bằng tay.
  2. Khi có nhiều lựa chọn hợp lý thì cũng phải lựa chọn thủ công bằng tay.
  3. Khi sử dụng các công cụ CASE, cần được chuẩn bị để thực hiện việc kiểm tra chất lượng đối với các kết quả thu được.

# THIẾT KẾ LOGIC (Cont.)

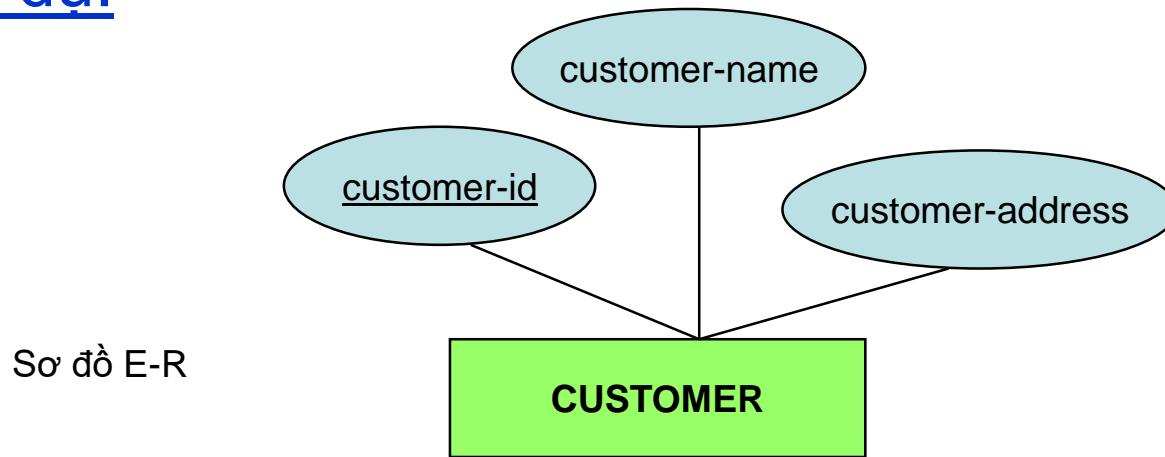
- ❖ Khi ánh xạ các lược đồ E-R sang các lược đồ quan hệ, cần nhớ 3 loại thực thể đã định nghĩa sau:
  - **Thực thể thường (khôe):** là các thực thể có thể tồn tại độc lập và thường thể hiện các đối tượng của thế giới thực, ví dụ, con người hoặc sản phẩm. Trong mô hình E-R, ký hiệu bằng hình chữ nhật với một đường viền đơn.
  - **Thực thể yếu:** là các thực thể không thể tồn tại một mình mà phải đi cùng với một mối quan hệ xác định bởi một loại thực thể xác định nó (thực thể chủ- khôe). Các thực thể yếu được biểu diễn bởi một hình chữ nhật với đường viền kép.
  - **Thực thể kết hợp:** hình thành từ những mối quan hệ nhiều-nhiều giữa các loại thực thể khác nhau, được biểu diễn bởi một hình chữ nhật với đường viền đơn và được bao quanh bởi một biểu tượng quan hệ hình thoi.

## BƯỚC 1: ÁNH XẠ CÁC THỰC THỂ THÔNG THƯỜNG (THỰC THỂ KHỎE)

- ❖ Mỗi thực thể thông thường trong mô hình thực thể liên kết sẽ được chuyển đổi thành một lược đồ quan hệ.
- ❖ Tên của quan hệ thường là tên của thực thể.
- ❖ Mỗi thuộc tính đơn của thực thể là một thuộc tính của lược đồ quan hệ.
- ❖ Thuộc tính xác định thực thể trở thành khóa chính của quan hệ tương ứng.

## BƯỚC 1: ÁNH XẠ CÁC THỰC THỂ THÔNG THƯỜNG (Cont.)

Ví dụ:



**CUSTOMER**

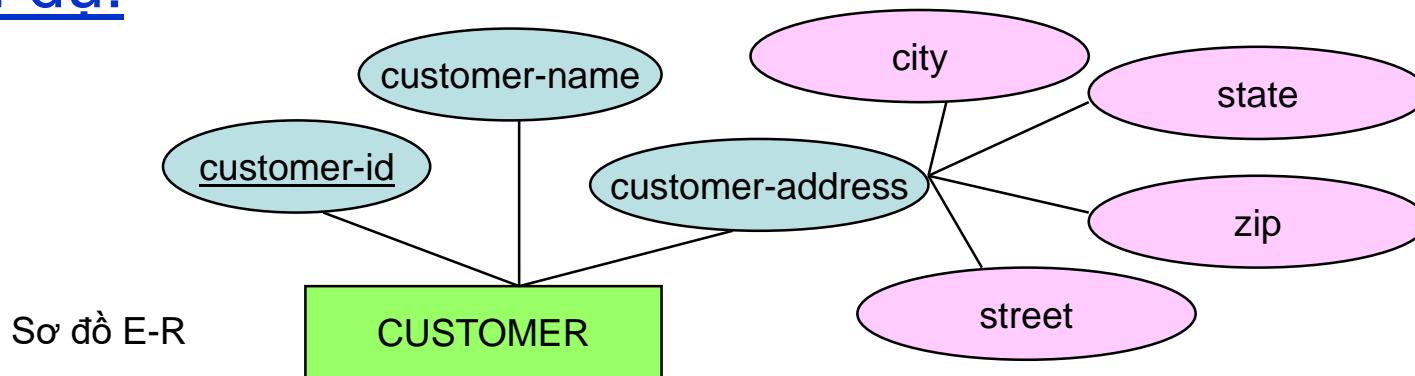
<u>customer-id</u>	customer-name	customer-address
--------------------	---------------	------------------

Quan hệ CUSTOMER

## BƯỚC 1: ÁNH XẠ CÁC THỰC THỂ THÔNG THƯỜNG (Cont.)

Thuộc tính kép: Nếu thực thể có thuộc tính kép thì chỉ những thuộc tính đơn của thuộc tính kép này được đưa vào lược đồ quan hệ mới.

Ví dụ:



**CUSTOMER**

<u>customer-id</u>	customer-name	street	city	state	zip
--------------------	---------------	--------	------	-------	-----

Quan hệ CUSTOMER

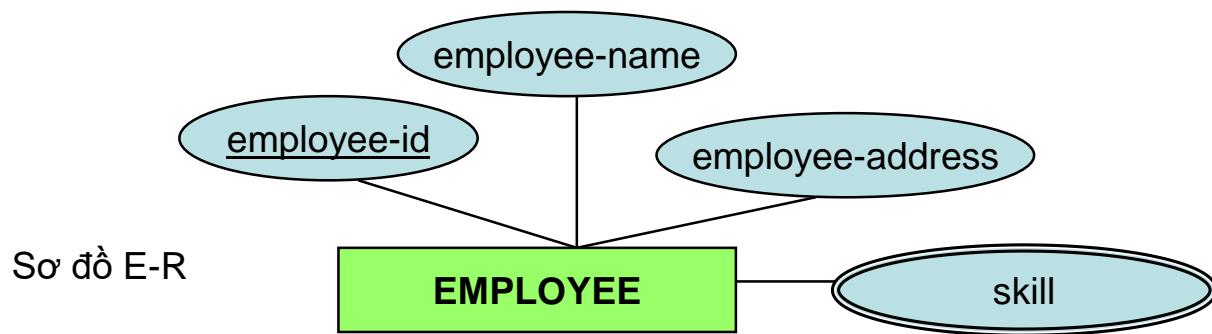
## BƯỚC 1: ÁNH XẠ CÁC THỰC THỂ THÔNG THƯỜNG (Cont.)

### Thuộc tính đa trị:

- ❖ Nếu một thực thể thường có một thuộc tính đa trị thì hai lược đồ quan hệ mới sẽ được tạo ra.
  - Lược đồ quan hệ thứ nhất chứa tất cả các thuộc tính của thực thể trừ thuộc tính đa trị.
  - Lược đồ quan hệ thứ hai sẽ có hai thuộc tính cấu thành khóa chính.
    - Thuộc tính thứ nhất là khoá chính của lược đồ quan hệ thứ nhất  
=> nó sẽ trở thành khoá ngoại trong lược đồ thứ hai.
    - Thuộc tính thứ hai là thuộc tính đa trị.
  - Tên của lược đồ thứ hai nên được đặt để thể hiện ngữ nghĩa của thuộc tính đa trị.

## BƯỚC 1: ÁNH XẠ CÁC THỰC THỂ THÔNG THƯỜNG (Cont.)

Ví dụ cho trường hợp thuộc tính đa trị:



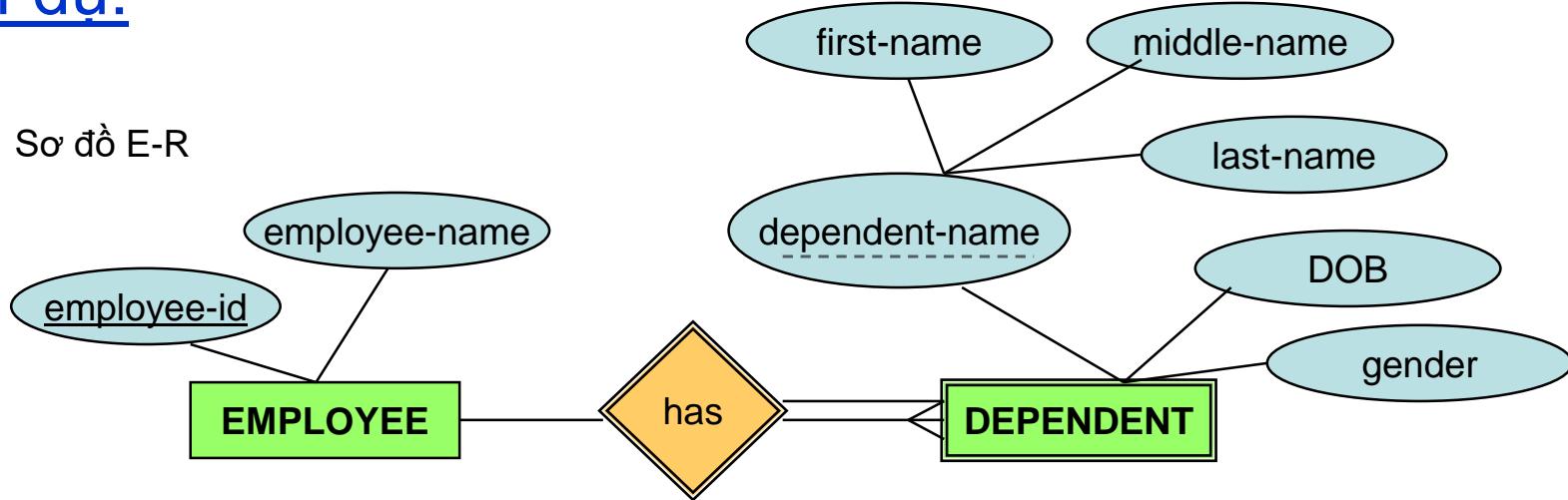
## BƯỚC 2: ÁNH XẠ CÁC THỰC THỂ YẾU

- ❖ Để ánh xạ thực thể yếu thành một lược đồ quan hệ, trước hết giả sử đã tạo một lược đồ quan hệ liên quan tới loại thực thể xác định.
- ❖ Tiếp theo, đối với mỗi thực thể yếu, tạo một lược đồ quan hệ mới và đưa tất cả các thuộc tính đơn (hoặc các thành phần đơn của các thuộc tính kép) vào thành thuộc tính của lược đồ quan hệ này.
- ❖ Sau đó, thêm khóa chính của quan hệ xác định vào thành một thuộc tính khóa ngoài trong lược đồ quan hệ mới.
- ❖ Khóa chính của lược đồ quan hệ mới là sự kết hợp của khoá chính của quan hệ xác định và thuộc tính phân biệt của thực thể yếu.

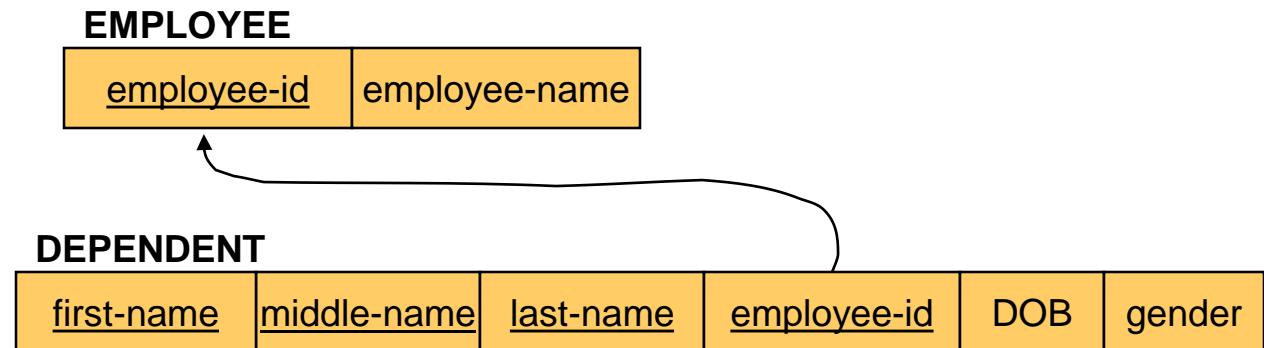
## BƯỚC 2: ÁNH XẠ CÁC THỰC THỂ YÊU (Cont.)

Ví dụ:

Sơ đồ E-R



Các lược đồ quan hệ



## BƯỚC 3: ÁNH XẠ CÁC QUAN HỆ 2 NGÔI

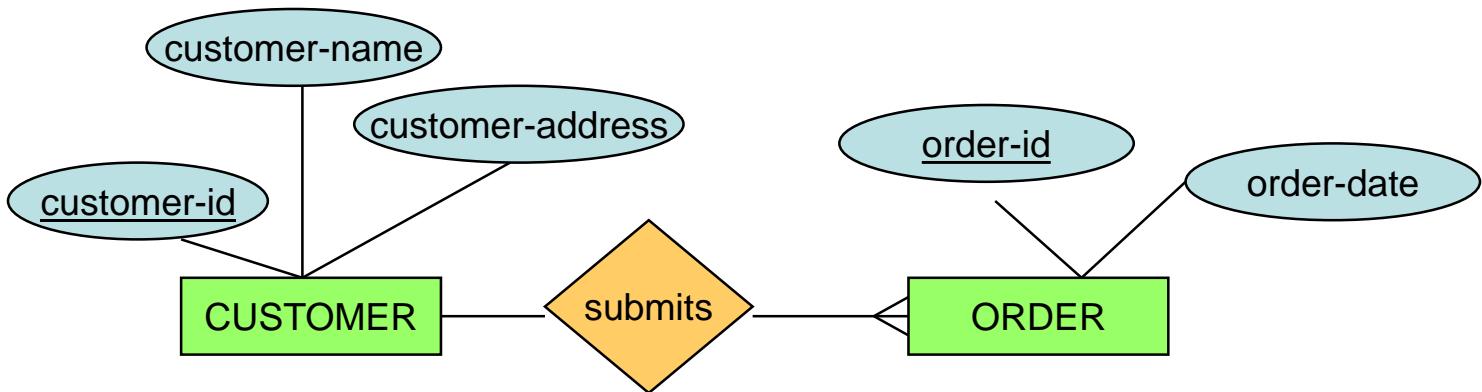
### Quan hệ 1-N hai ngôi:

- ❖ Đầu tiên, tạo lược đồ quan hệ cho mỗi thực thể tham gia vào quan hệ, sử dụng các thủ tục ở bước 1.
- ❖ Sau đó, thêm các thuộc tính khóa chính của thực thể bên phía 1 của mỗi quan hệ thành khóa ngoại cho quan hệ nằm ở bên phía N của mỗi quan hệ (khóa chính lấy từ bên phía N của mỗi quan hệ).
- ❖ **Chú ý:** Quan hệ 1-N và N-1 là đối xứng nhau.

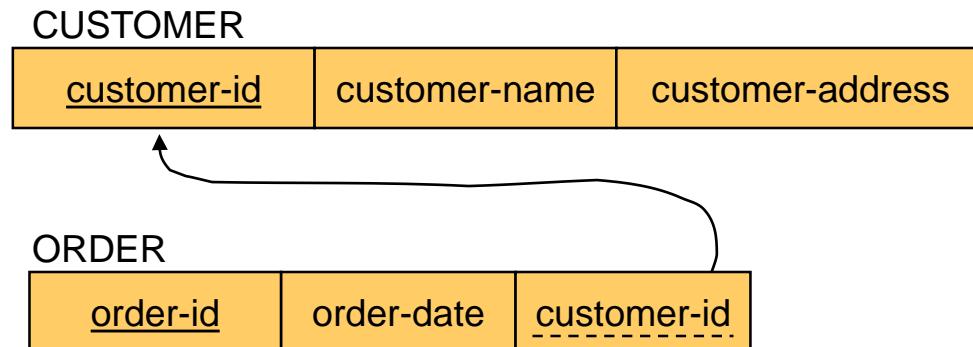
## BƯỚC 3: ÁNH XẠ CÁC QUAN HỆ 2 NGÔI (Cont.)

Ví dụ cho trường hợp quan hệ 1-N hai ngôi:

Sơ đồ E-R



Các lược đồ quan hệ



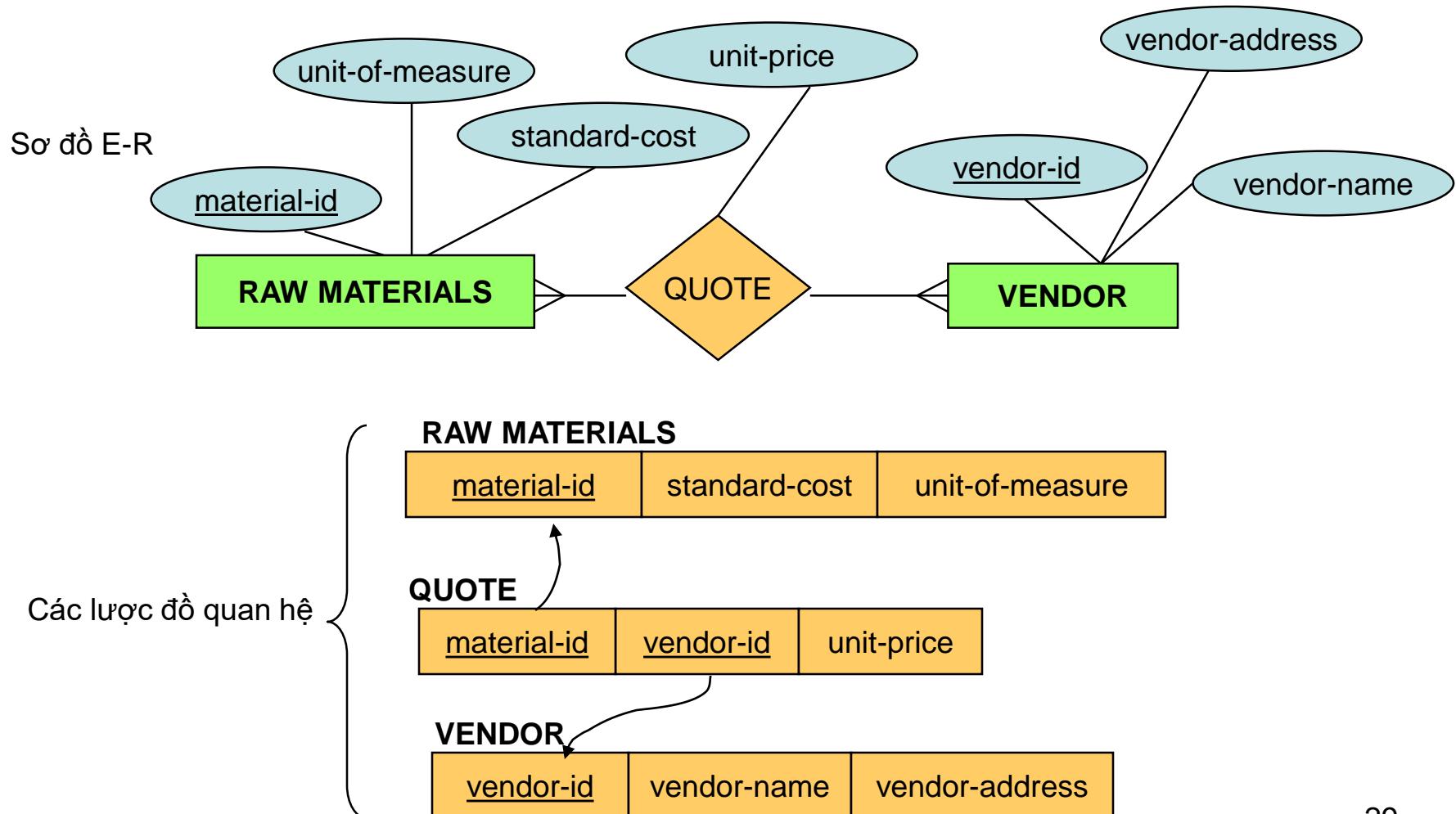
## BƯỚC 3: ÁNH XẠ CÁC QUAN HỆ 2 NGÔI (Cont.)

### Quan hệ N-N hai ngôi:

- ❖ Cho quan hệ hai ngôi N-N giữa hai thực thể A và B.
- ❖ Đầu tiên phải tạo thêm một lược đồ quan hệ mới C. Khóa của lược đồ C là sự kết hợp khóa chính của các tập thực thể tham gia vào quan hệ và các khóa chính này cũng là khóa ngoại của C.
- ❖ Các thuộc tính không phải là khóa mà liên quan tới quan hệ N-N giữa A và B cũng được đưa vào lược đồ quan hệ C.

## BƯỚC 3: ÁNH XẠ CÁC QUAN HỆ 2 NGÔI (Cont.)

Ví dụ cho trường hợp liên kết N-N hai ngôi:



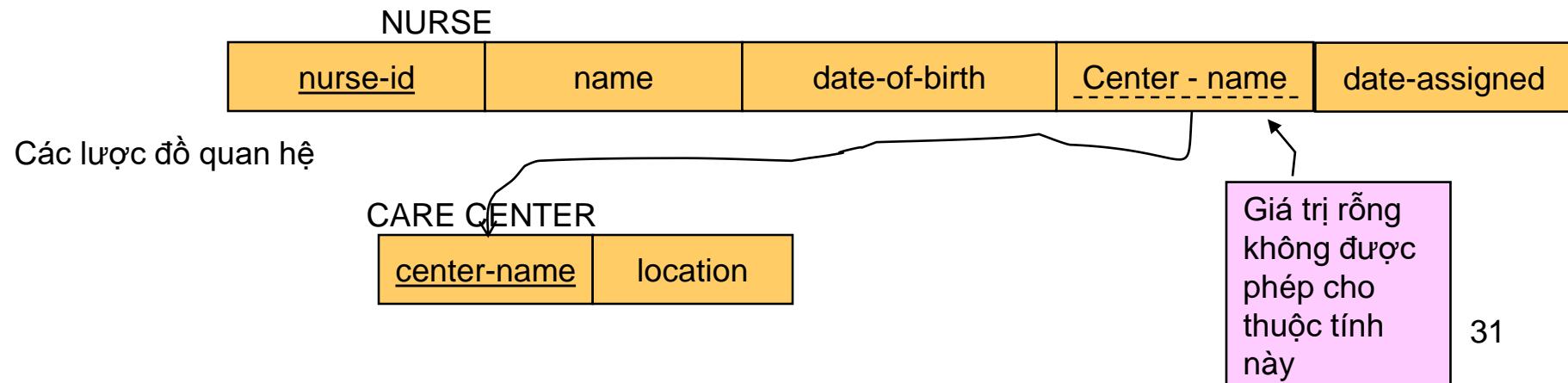
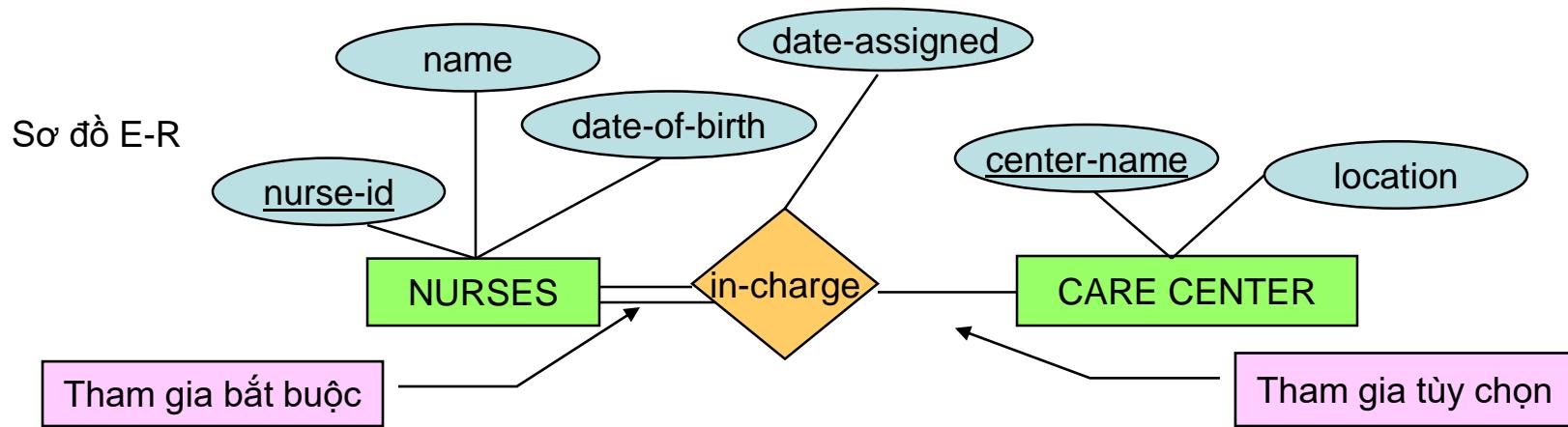
## BƯỚC 3: ÁNH XẠ CÁC LIÊN KẾT 2 NGÔI (Cont.)

### Liên kết 1-1 hai ngôi:

- ❖ Việc ánh xạ gồm 2 bước:
  1. Tạo 2 quan hệ liên quan tới 2 thực thể tham gia vào mỗi quan hệ.
  2. Khóa chính của một quan hệ sẽ thành khóa ngoài trong quan hệ còn lại.
- ❖ Trong liên kết 1-1, việc tham gia vào liên kết trong một bên thường là tùy chọn, và bên kia là bắt buộc.  
=> Nên thêm vào quan hệ của bên tham gia bắt buộc khóa ngoài của tập thực thể tham gia tùy chọn nhằm tránh việc lưu trữ các giá trị rỗng cho thuộc tính khóa ngoài.
- ❖ Các thuộc tính liên quan tới chính mối liên kết cũng được đưa vào quan hệ đó như là khóa ngoài.

## BƯỚC 3: ÁNH XẠ CÁC LIÊN KẾT 2 NGÔI (Cont.)

Ví dụ cho trường hợp liên kết 1-1 hai ngôi:



## BƯỚC 4: ÁNH XẠ CÁC THỰC THỂ KẾT HỢP

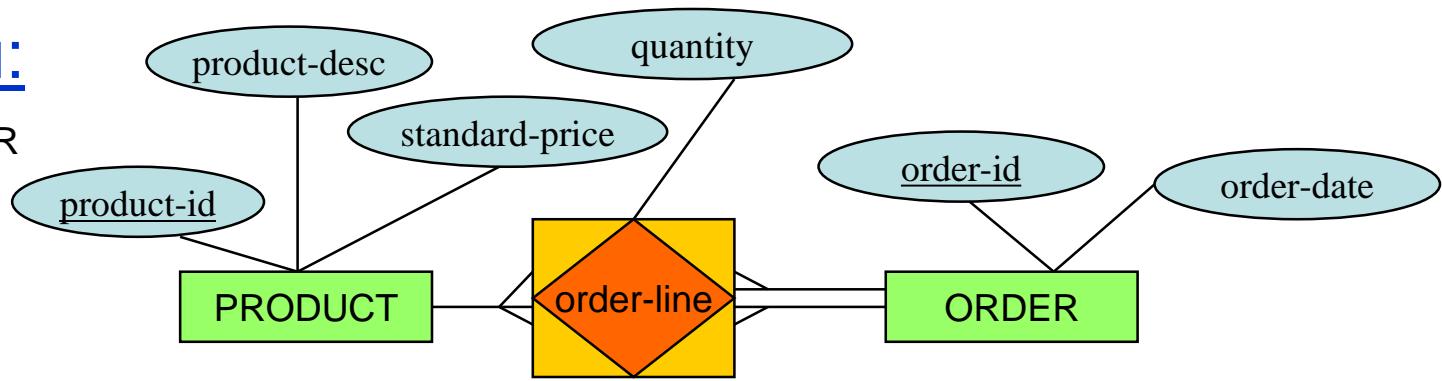
- ❖ Việc ánh xạ thực thể kết hợp sang lược đồ quan hệ tương tự thủ tục chuyển đổi một quan hệ N-N. Thực hiện qua 2 bước:
  1. Có 3 quan hệ được tạo ra: 2 quan hệ liên quan tới các thực thể tham gia vào liên kết, còn quan hệ thứ ba cho thực thể kết hợp, gọi là **quan hệ kết hợp**.
  2. Bước này phụ thuộc vào việc có gán một định danh cho thực thể kết hợp trong lược đồ E-R hay không. Có 2 trường hợp xảy ra:
    - a. Không gán định danh
    - b. Có gán định danh

## BƯỚC 4: ÁNH XẠ CÁC THỰC THỂ KẾT HỢP (Cont.)

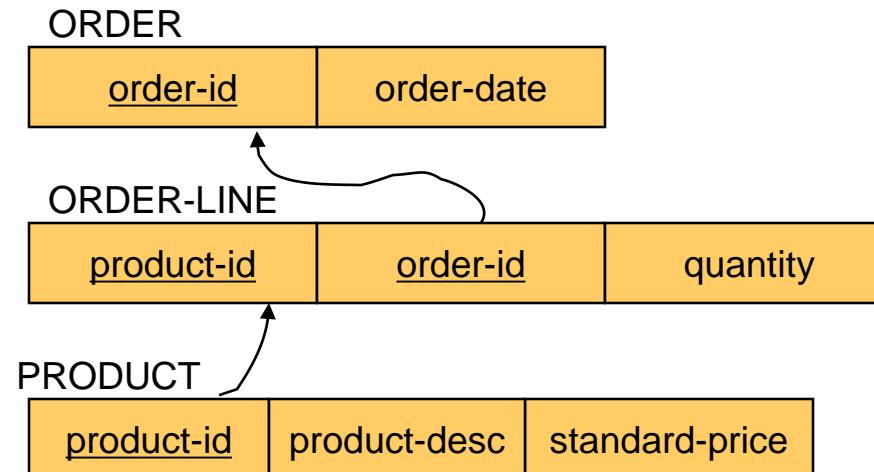
Trường hợp không gán định danh: khóa chính ngầm định cho quan hệ kết hợp gồm các thuộc tính khóa chính từ hai quan hệ còn lại. Các thuộc tính này sẽ là khóa ngoài tham chiếu tới hai quan hệ đó.

Ví dụ:

Sơ đồ E-R



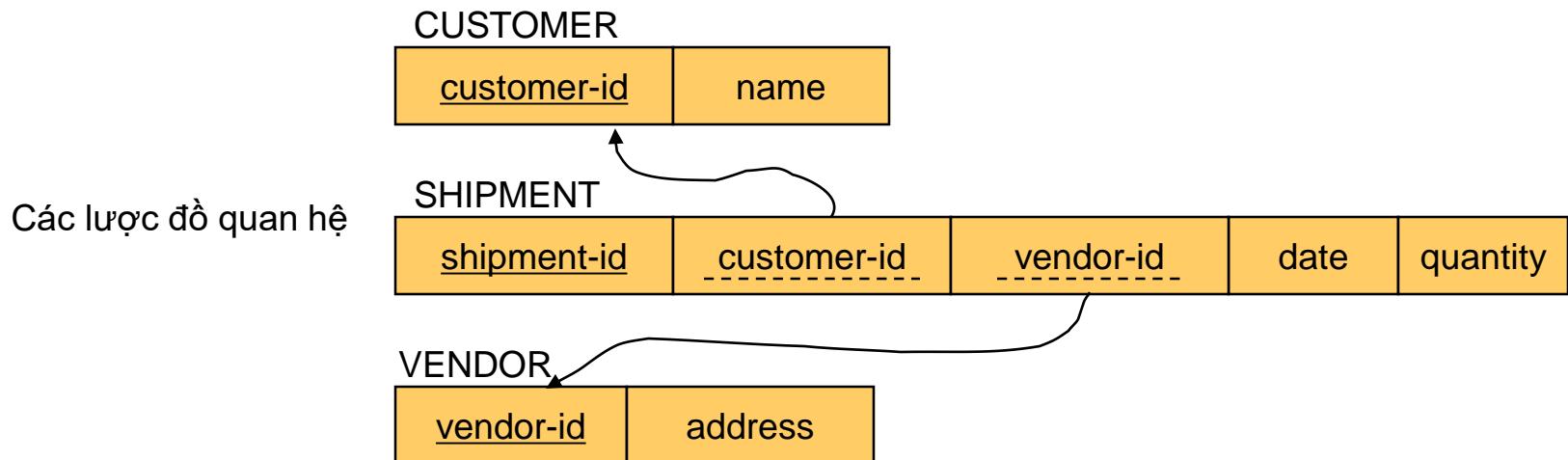
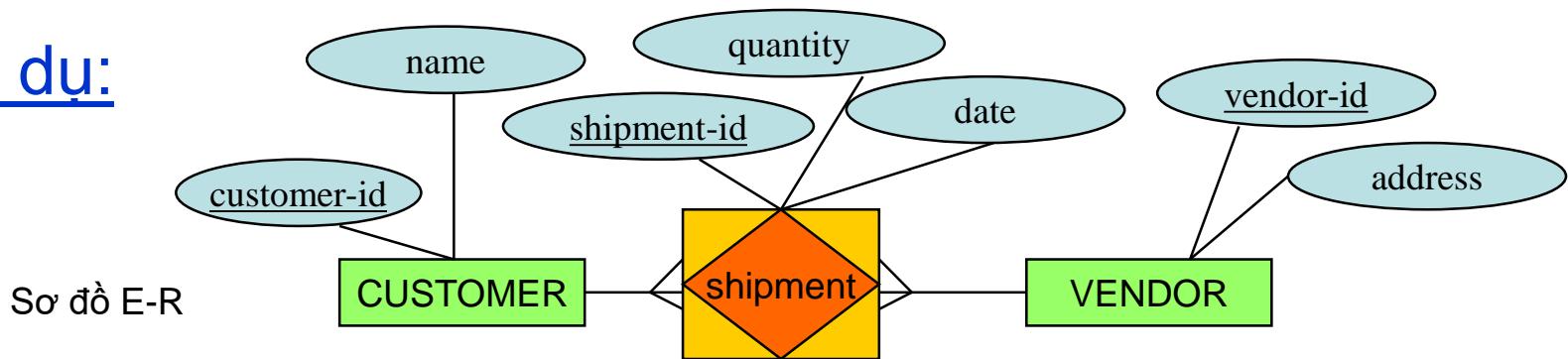
Các lược đồ quan hệ



## BƯỚC 4: ÁNH XẠ CÁC THỰC THỂ KẾT HỢP (Cont.)

Trường hợp có gán định danh: tạo một quan hệ kết hợp mới thể hiện thực thể kết hợp, với khóa chính là định danh được gán trên sơ đồ E-R. Khóa chính của 2 thực thể tham gia sẽ được thêm vào làm khóa ngoài trong quan hệ kết hợp.

Ví dụ:



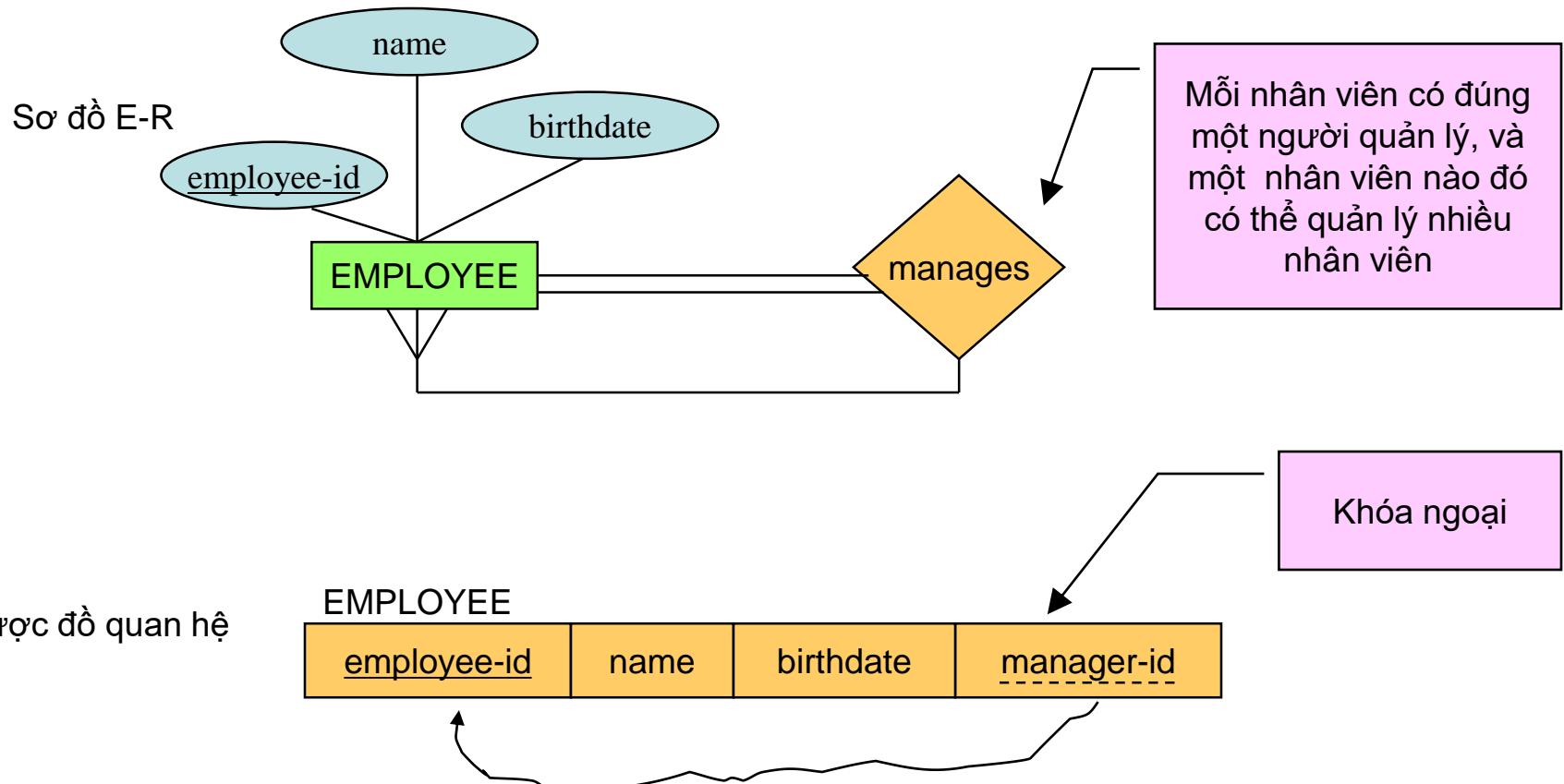
## BƯỚC 5: ÁNH XẠ CÁC QUAN HỆ ĐỆ QUY (MỘT NGÔI)

### Ánh xạ quan hệ đệ quy loại 1-N:

- ❖ Sử dụng bước 1 ánh xạ tập thực thể thành lược đồ quan hệ.
- ❖ Sau đó, thêm thuộc tính khóa ngoại vào quan hệ có tham chiếu tới các giá trị của khóa chính (khóa ngoại này phải có cùng miền giá trị với khóa chính).
- ❖ **Khóa ngoại đệ quy** là khóa ngoại của một quan hệ tham chiếu tới giá trị khóa chính của quan hệ đó.

## BƯỚC 5: ÁNH XẠ CÁC QUAN HỆ ĐỆ QUY (Cont.)

### Ví dụ trường hợp ánh xạ quan hệ đệ quy loại 1-N



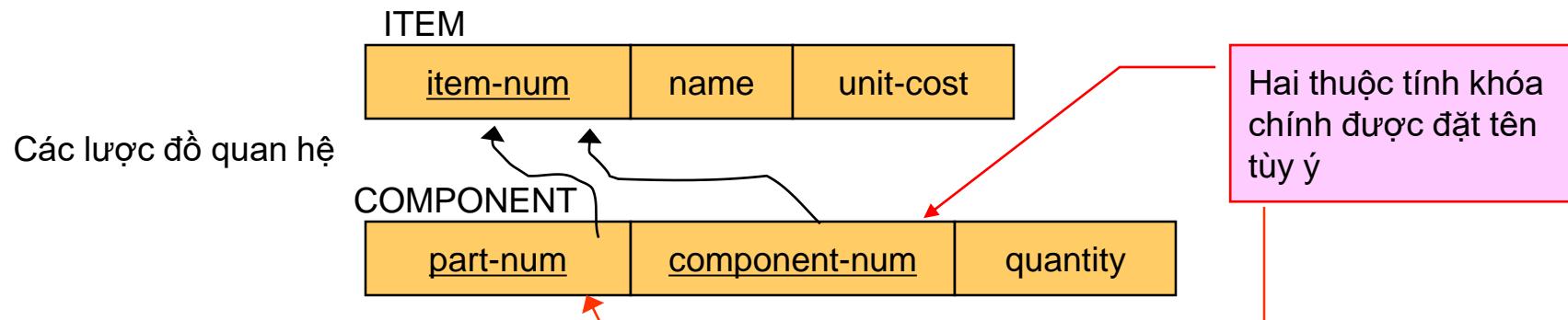
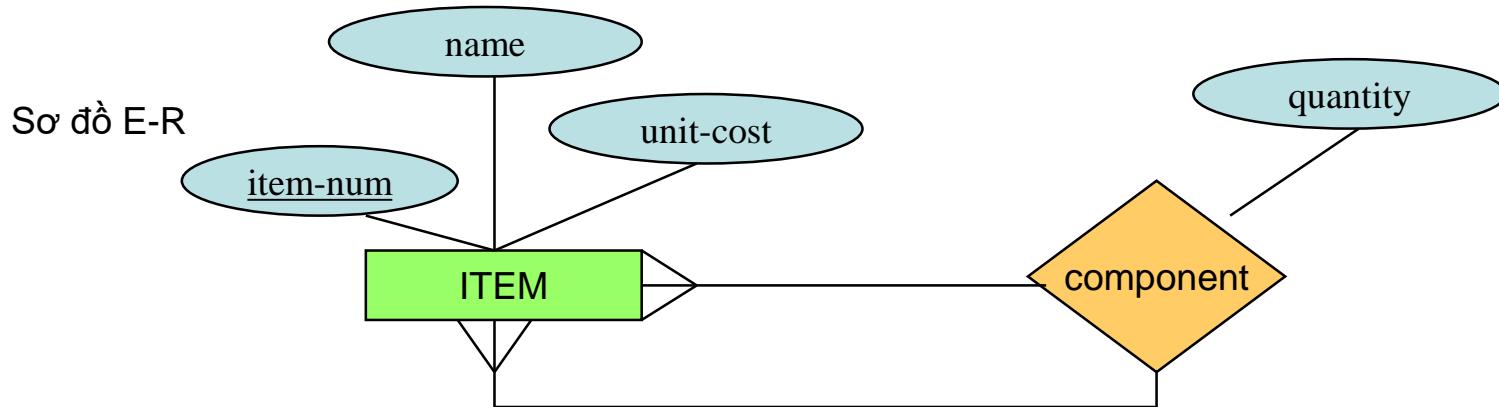
## BƯỚC 5: ÁNH XẠ CÁC QUAN HỆ ĐỀ QUY (Cont.)

### Ánh xạ quan hệ đề quy loại N-N:

- ❖ Hai lược đồ quan hệ được tạo ra: một lược đồ thể hiện tập thực thể và một lược đồ quan hệ liên kết thể hiện mối quan hệ N-N.
- ❖ Khóa chính của quan hệ liên kết gồm hai thuộc tính. Các thuộc tính này (không nhất thiết phải có cùng tên) đều lấy giá trị từ các khóa chính của quan hệ còn lại.
- ❖ Các thuộc tính không khóa của quan hệ được thêm vào quan hệ liên kết.

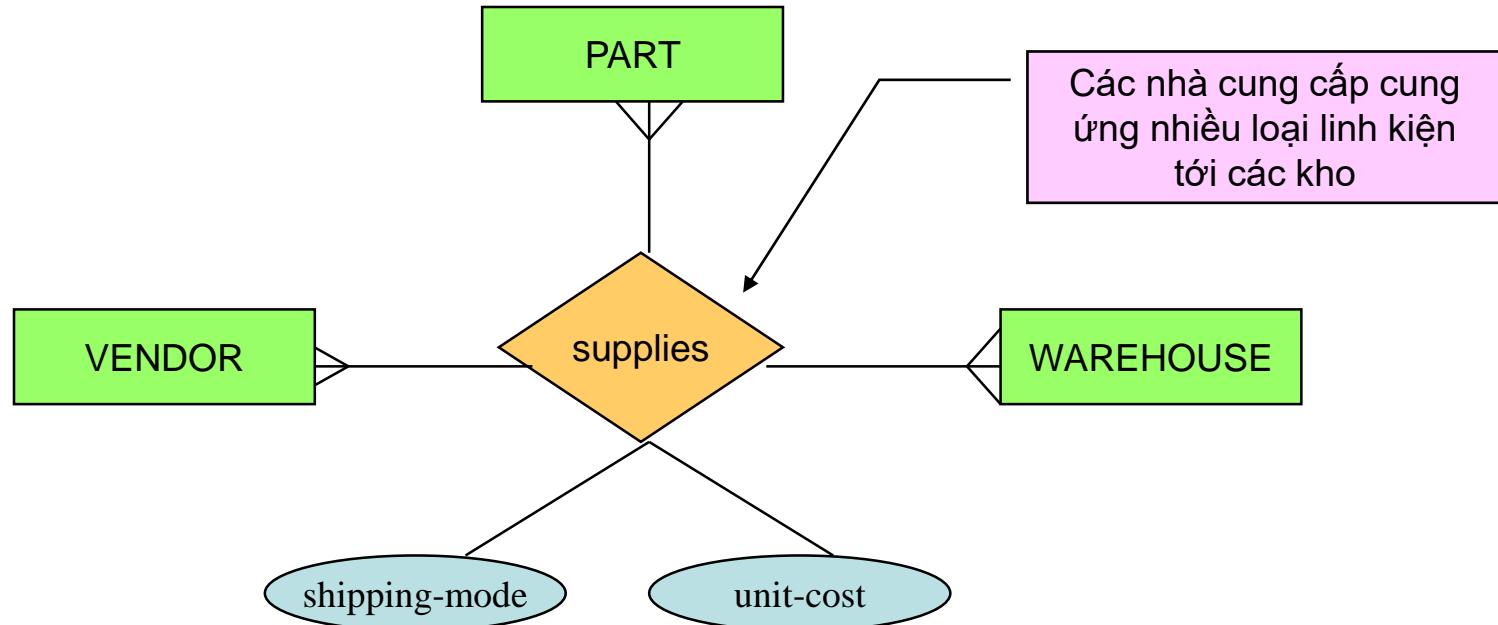
## BƯỚC 5: ÁNH XẠ CÁC QUAN HỆ ĐỀ QUY (Cont.)

### Ví dụ trường hợp ánh xạ quan hệ đề quy loại N-N



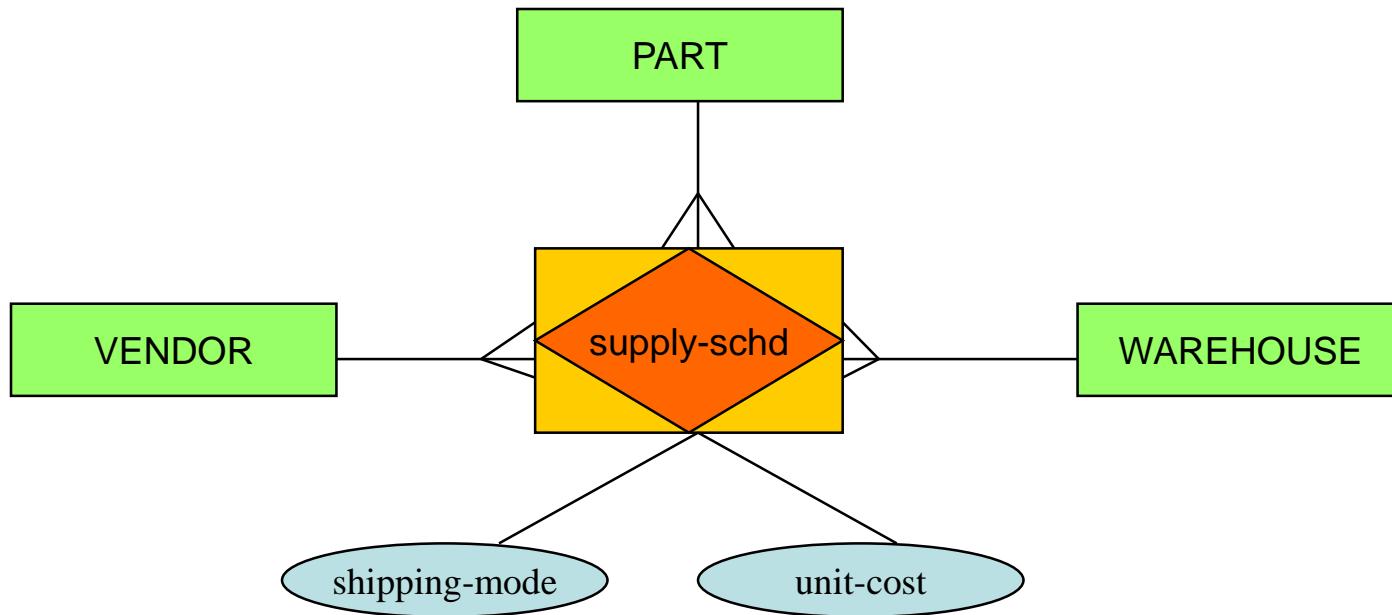
## BƯỚC 6: ÁNH XẠ CÁC QUAN HỆ NHIỀU NGÔI

- ❖ Một quan hệ 3 ngôi được định nghĩa là một quan hệ giữa 3 thực thể, ví dụ như sau:



## BƯỚC 6: ÁNH XẠ CÁC QUAN HỆ NHIỀU NGÔI (Cont.)

- ❖ Quan hệ nhiều ngôi nên được chuyển đổi thành các thực thể kết hợp trước khi được tiếp tục xử lý.  
Chuyển đổi cho ví dụ trên như sau:

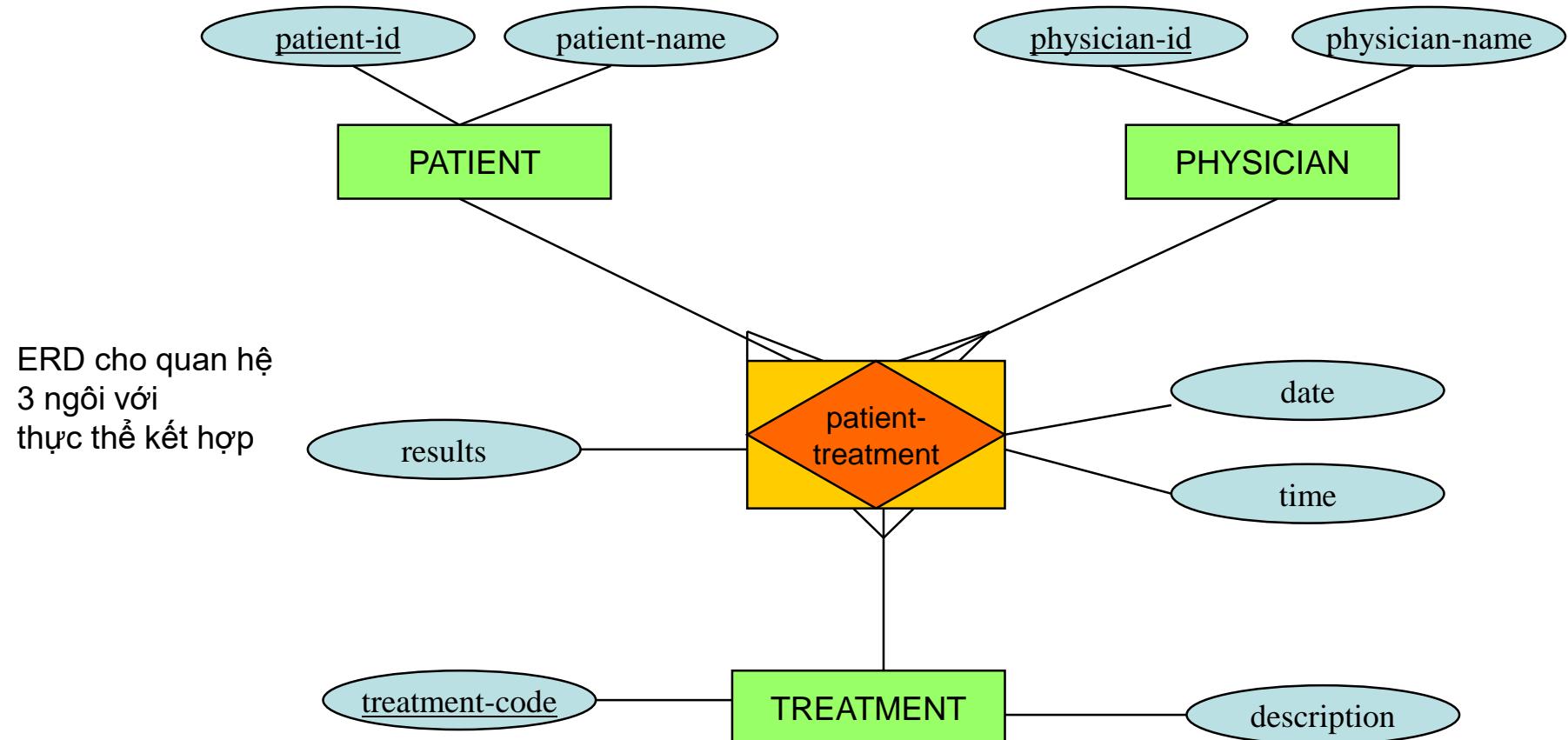


## BƯỚC 6: ÁNH XẠ CÁC QUAN HỆ NHIỀU NGÔI (Cont.)

- ❖ Để ánh xạ một thực thể kết hợp kết nối ba tập thực thể loại thường, cần tạo ra một quan hệ kết hợp mới.
- ❖ Khóa chính ngầm định cho quan hệ kết hợp gồm các thuộc tính khóa chính của các thực thể tham gia liên kết (trong một số trường hợp cần thêm các thuộc tính khác để hình thành một khóa chính duy nhất). Các thuộc tính này được coi là các khóa ngoại tham chiếu tới từng khóa chính của các tập thực thể tham gia liên kết.
- ❖ Mỗi thuộc tính của thực thể kết hợp trở thành thuộc tính trong quan hệ kết hợp mới.

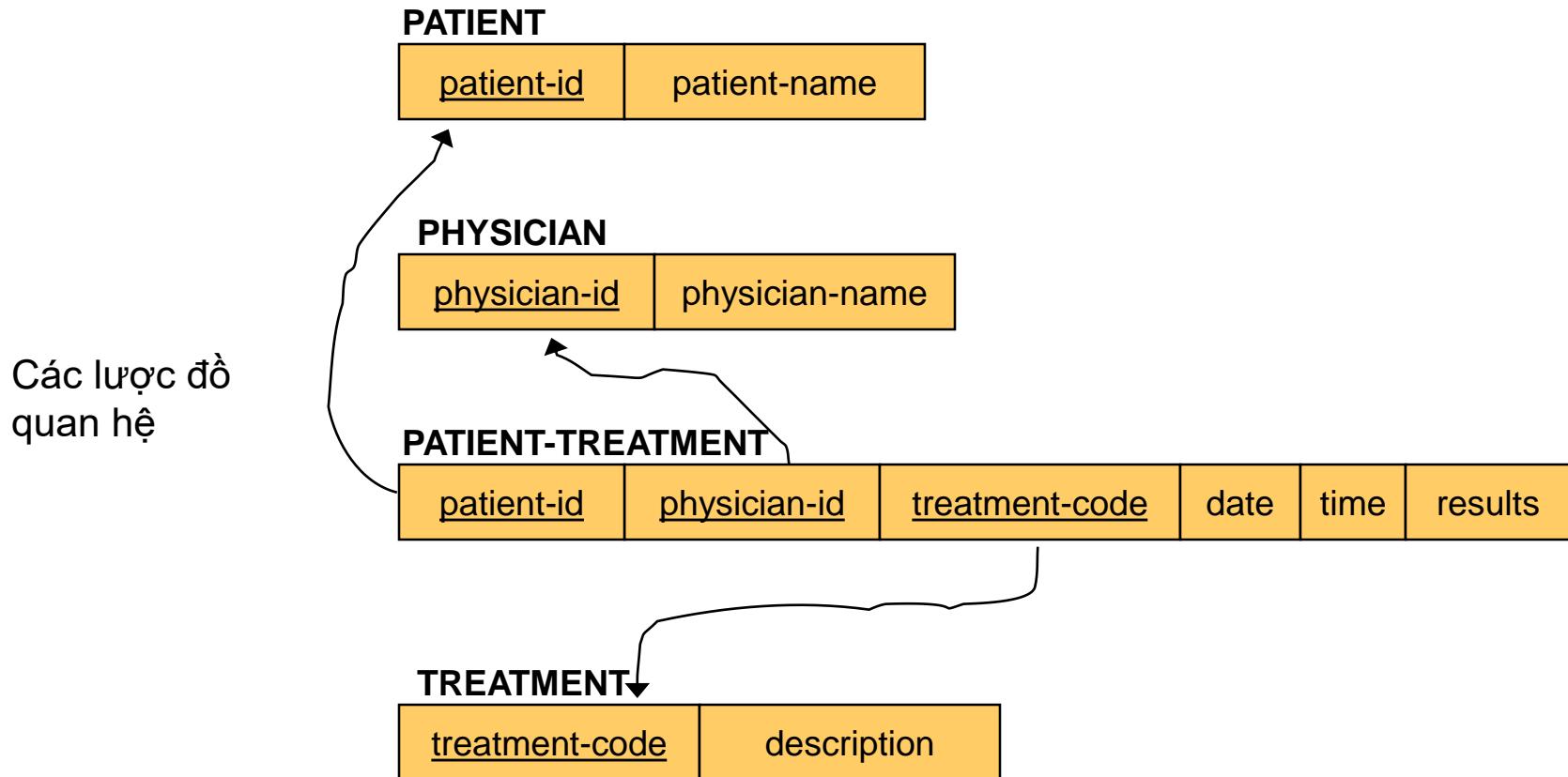
## BƯỚC 6: ÁNH XẠ CÁC QUAN HỆ NHIỀU NGÔI (Cont.)

❖ Ví dụ, cho quan hệ 3 ngôi:



## BƯỚC 6: ÁNH XẠ CÁC QUAN HỆ NHIỀU NGÔI (Cont.)

- ❖ Ánh xạ quan hệ 3 ngôi trên thành các lược đồ quan hệ:



## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON

### ***Thực hiện qua các bước sau:***

1. Tạo ra một lược đồ quan hệ riêng biệt cho tập thực thể lớp cha và cho các tập thực thể con của nó.
2. Gán các thuộc tính chung của tất cả các thành viên lớp con (bao gồm cả khóa chính) cho lược đồ quan hệ tương ứng với lớp cha.
3. Gán khóa chính của lớp cha cho lược đồ quan hệ của từng lớp con và những thuộc tính này là duy nhất cho từng tập thực thể con đó.
4. Gán một (hoặc nhiều) thuộc tính của tập thực thể lớp cha để thực hiện chức năng phân biệt các lớp con.

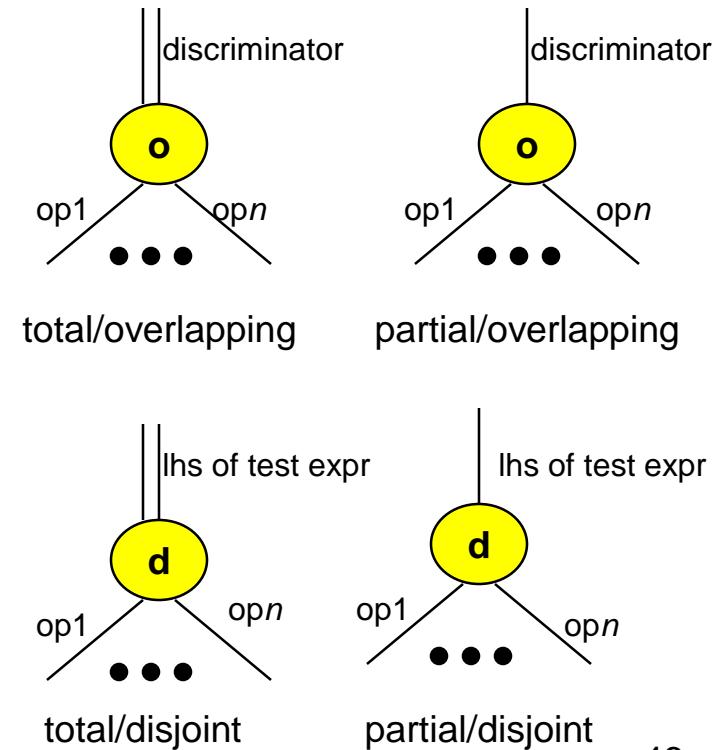
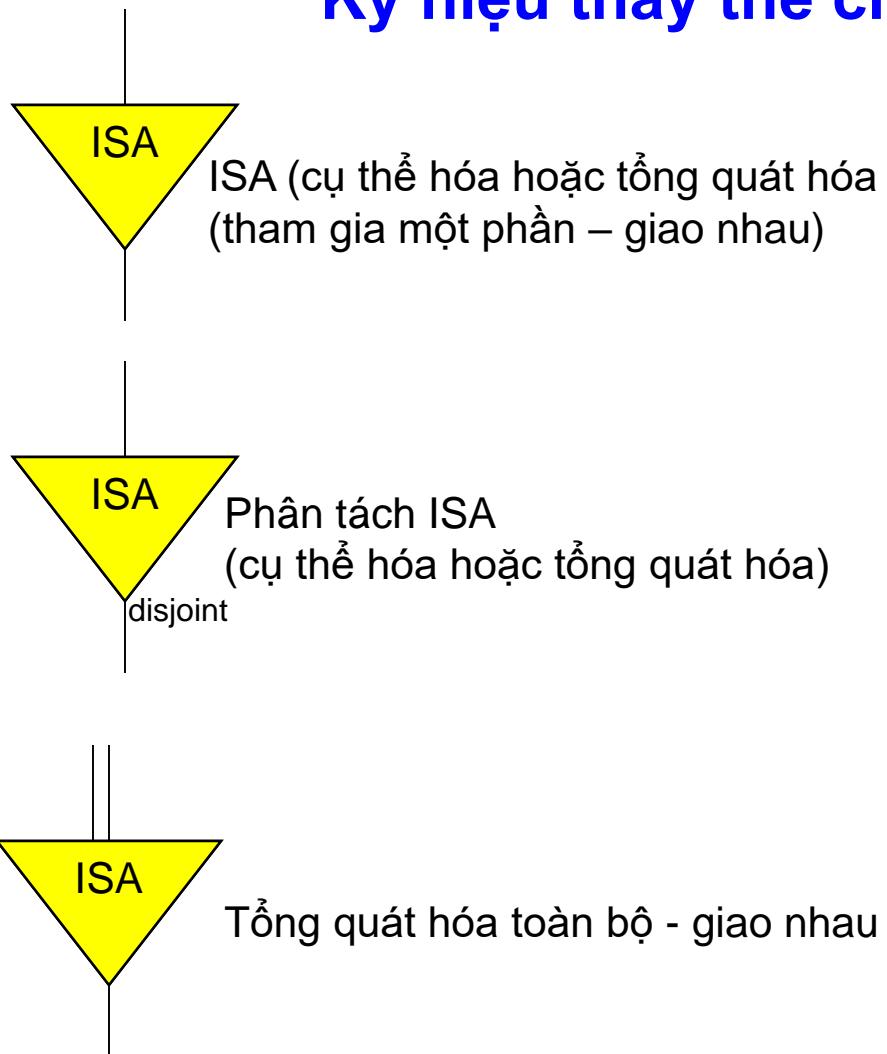
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

### **Định nghĩa các thành phần phân biệt lớp con:**

- Cho một quan hệ lớp cha/lớp con. Nếu muốn thêm một thể hiện của lớp cha vào cơ sở dữ liệu thì thể hiện này sẽ được thêm vào lớp con nào (khi cần)?
- Cách tiếp cận chung là sử dụng một thành phần phân biệt. Một **thành phần phân biệt** các tập thực thể con là một thuộc tính của tập thực thể cha mà giá trị của nó xác định được các tập thực thể con (được sử dụng khi việc xác định các lớp con dựa trên các mệnh đề).
- Có hai trường hợp xảy ra: các tập thực thể con phân tách và các tập thực thể con giao nhau.

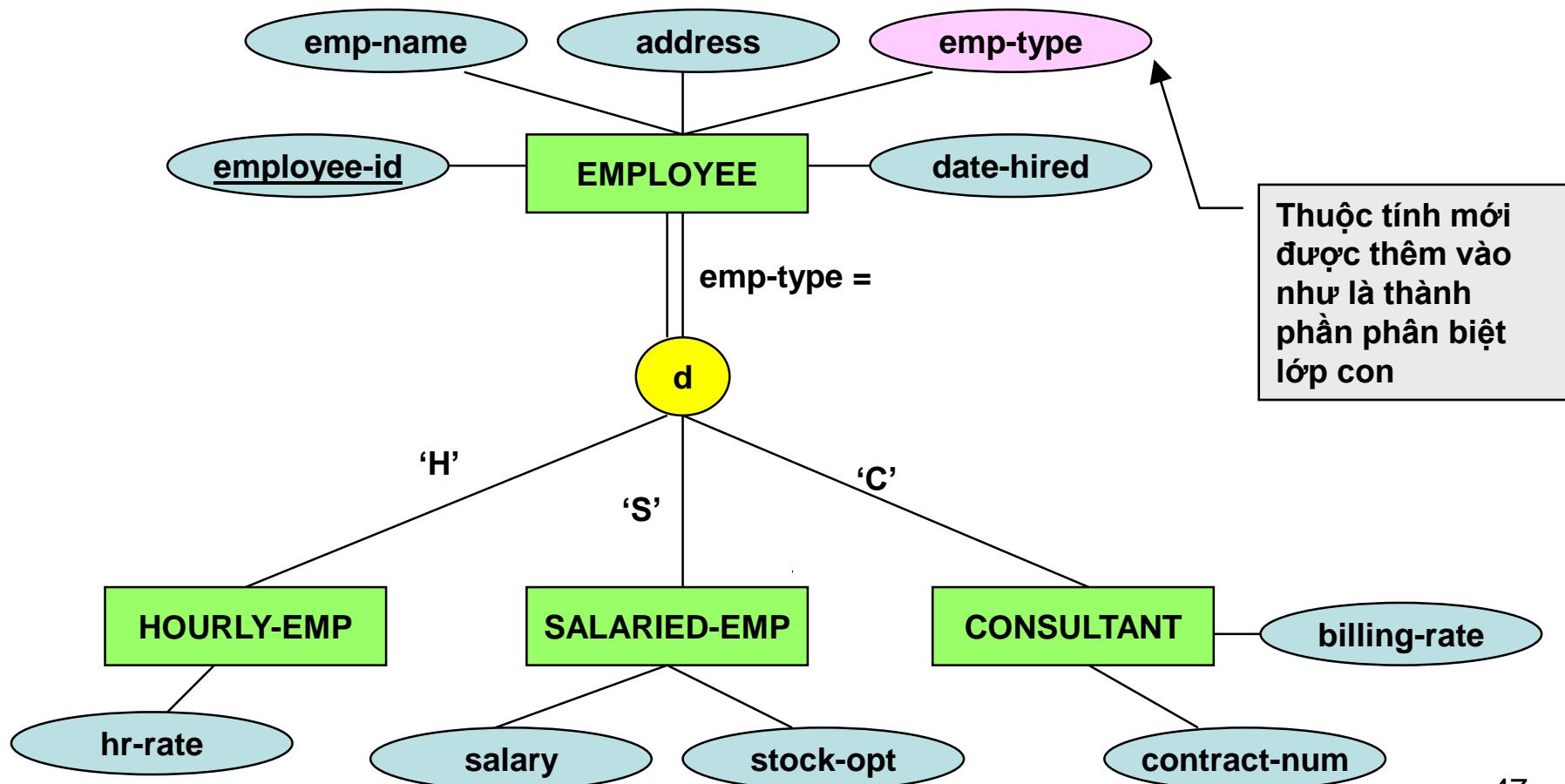
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

### Ký hiệu thay thế cho cụ thể hóa



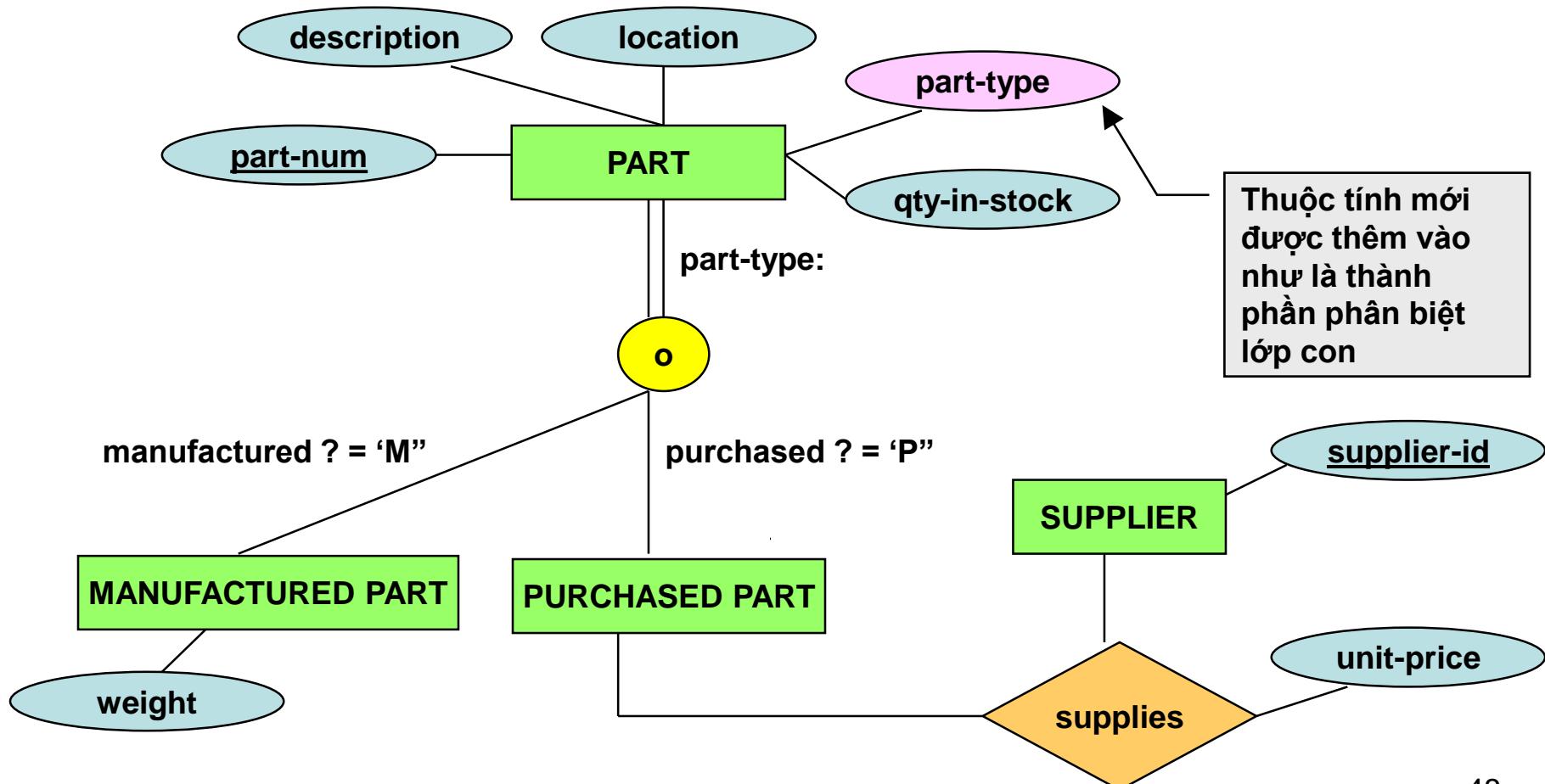
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

Ví dụ: thành phần phân biệt trong ERD cho các thực thể con phân tách



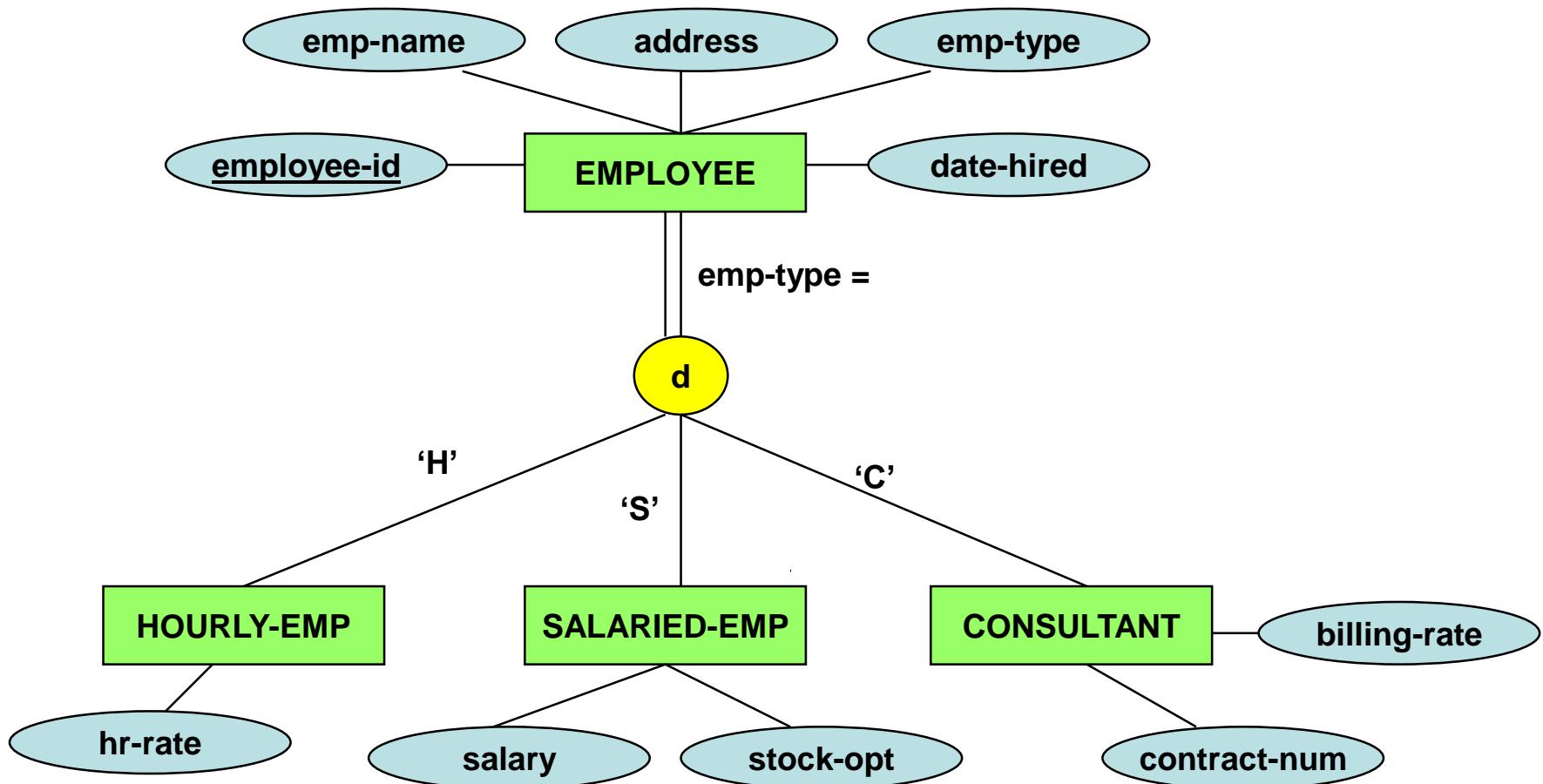
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

Ví dụ: thành phần phân biệt trong ERD cho các thực thể con giao nhau



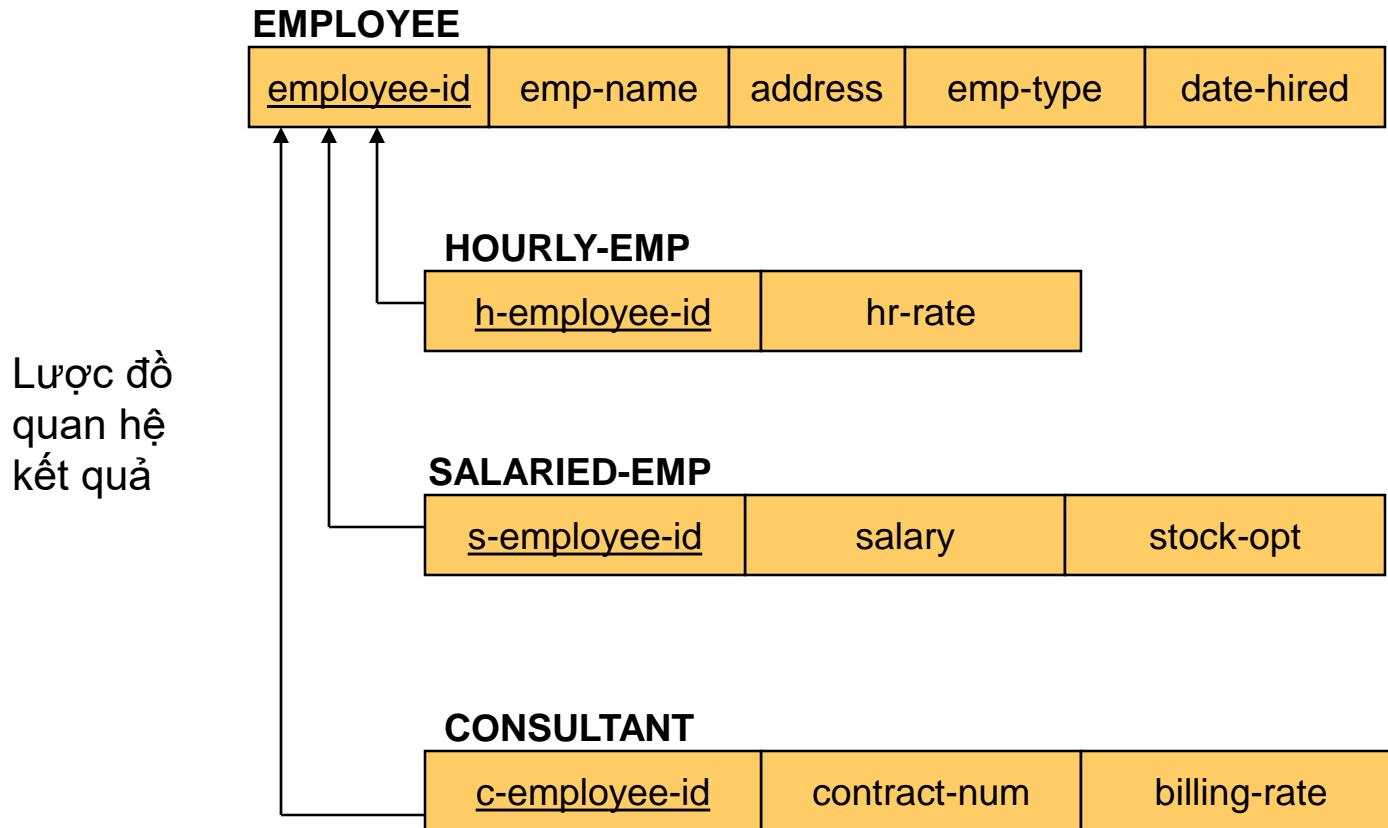
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

Ví dụ: Ánh xạ mối liên kết lớp cha/lớp con sang lược đồ quan hệ



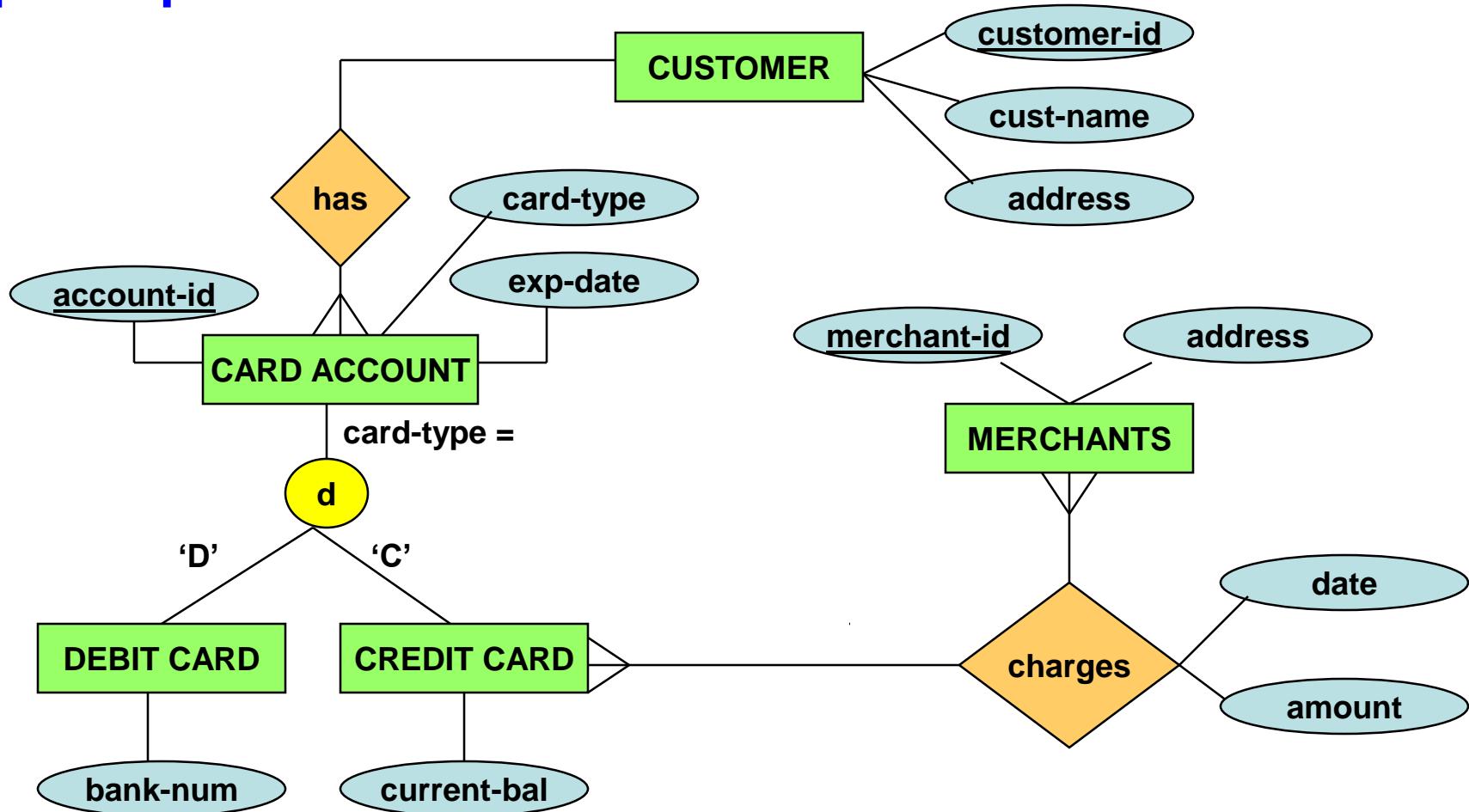
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

Kết quả:



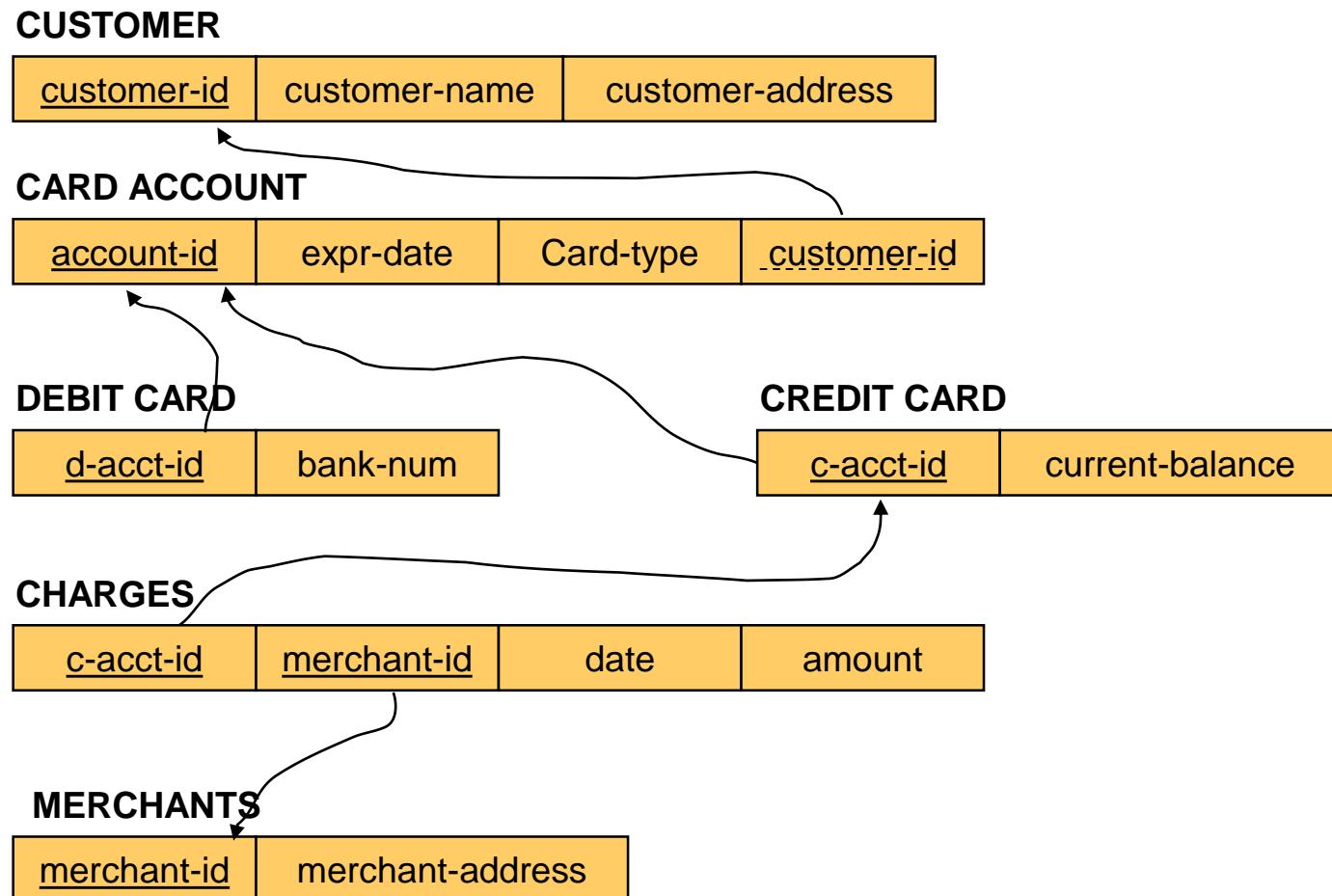
## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

### Bài tập: Ánh xạ mối liên kết lớp cha/lớp con sang lược đồ quan hệ



## BƯỚC 7: ÁNH XẠ CÁC MỐI LIÊN KẾT LỚP CHA/LỚP CON (Cont.)

Kết quả:



# **CHƯƠNG 4.**

# **PHỤ THUỘC HÀM**

# MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM

- ❖ **Chuẩn hóa** là một kỹ thuật tạo ra một tập các quan hệ với các thuộc tính từ các yêu cầu cho trước về dữ liệu cần mô hình hóa của tổ chức.
- ❖ Quá trình chuẩn hóa đầu tiên được phát triển bởi Codd vào năm 1972.
- ❖ Việc chuẩn hóa thường được thực hiện như một chuỗi các kiểm tra trên một quan hệ để xác định xem nó có thỏa mãn hay vi phạm các yêu cầu của một dạng chuẩn cho trước nào đó hay không.

# MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- ❖ Codd định nghĩa 3 dạng chuẩn: dạng chuẩn 1 (1NF), dạng chuẩn 2 (2NF), và dạng chuẩn 3 (3NF).
- ❖ Năm 1974, Boyce và Codd cùng giới thiệu một dạng chuẩn mạnh hơn 3NF được gọi là chuẩn Boyce-Codd (BCNF).
- ❖ Cả 4 dạng chuẩn đều dựa trên *sự phụ thuộc hàm giữa các thuộc tính trong một quan hệ*.
- ❖ **Một phụ thuộc hàm** mô tả mối quan hệ giữa các thuộc tính trong một quan hệ.

Ví dụ: A, B là các thuộc tính hoặc tập các thuộc tính trong quan hệ R. B phụ thuộc hàm vào A (ký hiệu: A->B) nếu mỗi giá trị của A liên hệ tới duy nhất một giá trị của B.

# MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- ❖ Năm 1977, 1979 và các năm tiếp theo, người ta giới thiệu dạng chuẩn 4 (4NF), dạng chuẩn 5 (5NF) và các dạng chuẩn mức cao hơn.  
=> Tuy nhiên, rất hiếm khi gặp.
- ❖ **Quá trình chuẩn hóa** là một phương pháp chính thống để xác định các quan hệ dựa trên khóa chính, khóa dự bị và các phụ thuộc hàm giữa các thuộc tính.
- ❖ **Chuẩn hóa** giúp người thiết kế kiểm tra và chuyển các quan hệ về một dạng chuẩn hóa cụ thể nào đó nhằm ngăn chặn các dị thường khi thực hiện các thao tác cập nhật thông tin.

# MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- ❖ Mục đích của việc thiết kế CSDL quan hệ: nhóm các thuộc tính vào thành các quan hệ sao cho tối thiểu hóa sự dư thừa dữ liệu => giảm không gian lưu trữ và tránh dị thường thông tin khi cập nhật dữ liệu.
- ❖ Xét ví dụ lược đồ quan hệ *staffbranch*.

staffbranch

<u>staff#</u>	<u>sname</u>	<u>position</u>	<u>salary</u>	<u>branch#</u>	<u>baddress</u>
SL21	Kristy	manager	30000	B005	22 Deer Road
SG37	Debi	assistant	12000	B003	163 Main Street
SG14	Alan	supervisor	18000	B003	163 Main Street
SA9	Traci	assistant	12000	B007	375 Fox Avenue
SG5	David	manager	24000	B003	163 Main Street
SL41	Anna	assistant	10000	B005	22 Deer Road

# MỤC ĐÍCH CỦA SỰ CHUẨN HÓA VÀ PHỤ THUỘC HÀM (Cont.)

- ❖ Quan hệ *staffbranch* có dư thừa dữ liệu. Thông tin chi tiết của một chi nhánh ngân hàng (*branch*) sẽ bị lặp lại cho mỗi nhân viên (*staff*) làm việc tại chi nhánh đó.
- ❖ Nếu tách riêng quan hệ *staff* và *branch* thì thông tin về từng chi nhánh ngân hàng chỉ xuất hiện duy nhất một lần.

staff

<b><u>staff#</u></b>	<b>sname</b>	<b>position</b>	<b>salary</b>	<b>branch#</b>
SL21	Kristy	manager	30000	B005
SG37	Debi	assistant	12000	B003
SG14	Alan	supervisor	18000	B003
SA9	Traci	assistant	12000	B007
SG5	David	manager	24000	B003
SL41	Anna	assistant	10000	B005

branch

<b><u>branch#</u></b>	<b>baddress</b>
B005	22 Deer Road
B003	163 Main Street
B007	375 Fox Avenue

# DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT

## ❖ *Khi thực hiện chèn thêm dữ liệu:*

- Để chèn thêm thông tin chi tiết cho các nhân viên mới vào *staffbranch*, cần phải thêm thông tin chi tiết về chi nhánh tương ứng cho mỗi bản ghi nhân viên.  
**Ví du:** Nếu thêm nhân viên mới vào *branch* B007, thì phải thêm cả địa chỉ của B007. Còn đối với lược đồ (*staff, branch*) thì chỉ cần thêm thông tin vào quan hệ *staff* là đủ.
- Để chèn thêm thông tin cho một *branch* mới mà hiện tại chưa có nhân viên nào, cần chèn thêm các giá trị *Null* cho các thuộc tính về nhân viên, như *staff#*, *sname*, ...
  - ⇒ Tuy nhiên, thuộc tính *staff#* là khóa chính nên điều này không thể được (vi phạm tính toàn vẹn của khóa).
  - ⇒ Vậy, không thể nhập thông tin cho một chi nhánh mới mà không có nhân viên nào.

# DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT (Cont.)

## ❖ *Khi thực hiện xóa dữ liệu:*

- Giả sử trong *staffbranch* có một chi nhánh chỉ còn một nhân viên cuối cùng. Nếu xóa thông tin về nhân viên này thì các thông tin về chi nhánh đó cũng sẽ bị xóa khỏi CSDL.

Ví dụ: Nếu xóa thông tin về nhân viên Traci khỏi quan hệ *staffbranch* thì thông tin về chi nhánh B007 cũng sẽ bị xóa.

- ⇒ Điều này không xảy ra đối với lược đồ (*staff, branch*) vì thông tin về nhân viên được lưu trữ tách riêng khỏi thông tin về chi nhánh.

# DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT (Cont.)

## ❖ *Khi thực hiện cập nhật dữ liệu:*

- Giả sử muốn thay đổi giá trị của một trong các thuộc tính của một chi nhánh nào đó trong quan hệ staffbranch, ví dụ: địa chỉ chi nhánh, thì cần phải cập nhật tất cả các bộ của các nhân viên làm việc tại chi nhánh đó.
- Nếu việc cập nhật thay đổi địa chỉ chi nhánh không được thực hiện trên tất cả các bộ liên quan trên quan hệ staffbranch thì CSDL sẽ không nhất quán (một chi nhánh sẽ có 2 địa chỉ khác nhau).

⇒ *Như vậy: trên quan hệ staffbranch, khi thực hiện các thao tác cập nhật dữ liệu thì xuất hiện dị thường thông tin. Việc này sẽ tránh được bằng cách tách lược đồ staffbranch thành 2 quan hệ staff và branch.*

# DƯ THỪA DỮ LIỆU VÀ DỊ THƯỜNG THÔNG TIN KHI CẬP NHẬT (Cont.)

- ❖ **2 tính chất quan trọng khi tách một lược đồ quan hệ thành một tập các lược đồ nhỏ hơn:**
  1. **Kết nối không tồn thất thông tin:** đảm bảo mọi thể hiện của quan hệ ban đầu có thể xác định được từ các thể hiện liên quan trong các quan hệ nhỏ hơn.
  2. **Bảo toàn sự phụ thuộc hàm:** đảm bảo một ràng buộc trên quan hệ ban đầu có thể được duy trì bằng cách sử dụng đơn giản một số ràng buộc trên mỗi quan hệ nhỏ hơn.  
=> Các quan hệ nhỏ hơn không cần kết nối với nhau để kiểm tra xem một ràng buộc trên các quan hệ ban đầu có bị vi phạm hay không.

# KẾT NỐI KHÔNG TỒN THẤT THÔNG TIN

- Xét lược đồ quan hệ SP và sự phân tách nó thành 2 lược đồ S1 và S2.

SP

s#	p#	qty
S1	P1	10
S2	P2	50
S3	P3	10

S1

s#	qty
S1	10
S2	50
S3	10

S2

p#	qty
P1	10
P2	50
P3	10

$S1 * S2$

s#	p#	qty
S1	P1	10
S1	P3	10
S2	P2	50
S3	P1	10
S3	P3	10

Những bộ mới này không xuất hiện trong quan hệ ban đầu. Tuy vậy, không thể nói bộ nào là hợp lệ hay không hợp lệ. Khi sự phân tách xảy ra, quan hệ SP ban đầu có thể bị mất.

# BẢO TOÀN CÁC PHỤ THUỘC HÀM

❖ Xét ví dụ:

$$R = (A, B, C)$$

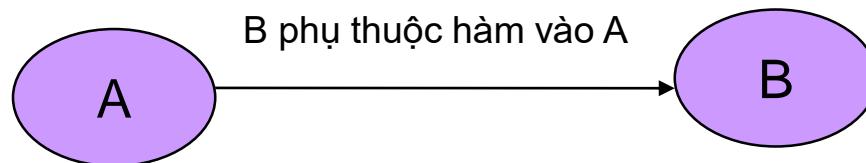
$$F = \{AB \rightarrow C, C \rightarrow A\}$$

$$\gamma = \{(B, C), (A, C)\}$$

- Rõ ràng,  $C \rightarrow A$  là được bảo toàn trong lược đồ quan hệ  $(A, C)$ .
- Làm thế nào để  $AB \rightarrow C$  mà không cần kết nối 2 quan hệ trong lược đồ quan hệ  $\gamma$ ? => Không thể.  
=> Vậy, các phụ thuộc hàm không được bảo toàn trong  $\gamma$ .

# ĐỊNH NGHĨA PHỤ THUỘC HÀM

- ❖ **Một phụ thuộc hàm** thể hiện ngũ nghĩa của các thuộc tính trong một quan hệ: một thuộc tính có quan hệ với thuộc tính khác như thế nào và xác định các phụ thuộc hàm giữa các thuộc tính đó. Phụ thuộc hàm là ràng buộc giữa các thuộc tính.
- ❖ Xét một quan hệ với các thuộc tính A và B, với thuộc tính B là phụ thuộc hàm vào thuộc tính A: Nếu biết giá trị của A sẽ tìm thấy một giá trị duy nhất của B. Nhưng với mỗi giá trị của B cho trước có thể có nhiều giá trị khác nhau tương ứng của A.



# ĐỊNH NGHĨA PHỤ THUỘC HÀM (Cont.)

- ❖ **Đối tượng xác định của một phụ thuộc hàm** là thuộc tính hoặc một nhóm các thuộc tính nằm bên trái của mũi tên trong một phụ thuộc hàm.
- ❖ **Đối tượng (thuộc tính) hệ quả của phụ thuộc hàm** là một thuộc tính hoặc nhóm thuộc tính nằm phía bên phải mũi tên trong phụ thuộc hàm.
- ❖ Ví dụ: A→B:  
A là đối tượng xác định và B là đối tượng hệ quả của A.  
=> Nói: A xác định B hay B phụ thuộc hàm vào A.

# XÁC ĐỊNH CÁC PHỤ THUỘC HÀM

- ❖ Quay lại ví dụ quan hệ staff (trong phần trước):
  - Thể hiện của quan hệ này có một phụ thuộc hàm  $\text{staff}\# \rightarrow \text{position}$ . Không có phụ thuộc theo chiều ngược lại.
  - Mỗi quan hệ giữa  $\text{staff}\#$  và  $\text{position}$  là 1:1, nghĩa là mỗi nhân viên có một vị trí tương ứng duy nhất.
  - Quan hệ giữa  $\text{position}$  và  $\text{staff}\#$  là 1:nhiều, nghĩa là nhiều nhân viên có cùng một vị trí.
  - Position không xác định  $\text{staff}\#$  hay  $\text{staff}\#$  không phụ thuộc hàm vào position.
- ❖ Với mục đích chuẩn hóa: chỉ quan tâm tới việc xác định các phụ thuộc hàm giữa các thuộc tính của quan hệ 1:1.

# XÁC ĐỊNH CÁC PHỤ THUỘC HÀM (Cont.)

- ❖ Khi xác định các phụ thuộc hàm giữa các thuộc tính trong một quan hệ, cần phân biệt rõ ràng giữa các giá trị nhận được bởi một thuộc tính tại một thời điểm và tập tất cả các giá trị có thể nhận được bởi thuộc tính đó tại các thời điểm khác nhau.  
=> **Một phụ thuộc hàm là một đặc điểm chung của một lược đồ quan hệ chứ không phải là một đặc điểm riêng của một thể hiện cụ thể của lược đồ đó.**
- ❖ Cần xác định các phụ thuộc hàm thỏa mãn cho tất cả các giá trị có thể có của các thuộc tính của một quan hệ.  
=> thể hiện các loại ràng buộc toàn vẹn cần xác định.

# VÍ DỤ XÁC ĐỊNH CÁC PHỤ THUỘC HÀM

## ❖ *Xét quan hệ staffbranch:*

Để xác định các phụ thuộc hàm, cần hiểu rõ ngữ nghĩa của các thuộc tính khác nhau trong mỗi lược đồ quan hệ.

Ví dụ: vị trí của mỗi nhân viên và chi nhánh sẽ xác định lương của họ.

=> *Cần hiểu rõ về tổ chức. Đây là nhiệm vụ của giai đoạn phân tích yêu cầu bài toán và giai đoạn thiết kế mức khái niệm.*

# VÍ DỤ XÁC ĐỊNH CÁC PHỤ THUỘC HÀM (Cont.)

## ❖ *Các phụ thuộc hàm được xác định như sau:*

staff# → sname, position, salary, branch#, baddress

branch# → baddress

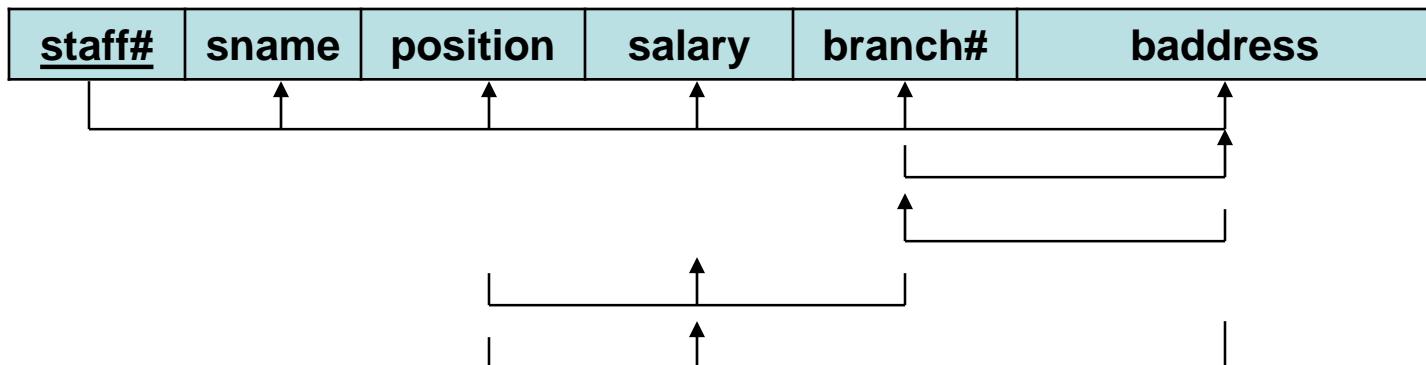
baddress → branch#

branch#, position → salary

baddress, position → salary

## ❖ Có thể dùng một dạng ký pháp lược đồ để biểu diễn như sau:

staffbranch



# PHỤ THUỘC HÀM HIỀN NHIÊN

- ❖ Trong quá trình xác định các phụ thuộc hàm, cần bỏ qua các phụ thuộc hàm hiển nhiên đúng.
- ❖ ***Phụ thuộc hàm hiển nhiên đúng:*** khi và chỉ khi vế phải của phụ thuộc hàm là một tập con của vế trái.  
Ví dụ:      { staff#, sname} → sname  
                { staff#, sname} → staff#
- ❖ Các phụ thuộc hàm này không cung cấp thêm các thông tin cần thiết nào về các ràng buộc toàn vẹn đối với quan hệ.  
***=> Trong phạm vi chuẩn hóa, các phụ thuộc hàm này được bỏ qua.***

# CÁC ĐẶC TÍNH CỦA PHỤ THUỘC HÀM

**Các đặc tính có ích của các phụ thuộc hàm cho việc chuẩn hóa:**

- ❖ Tồn tại mối quan hệ 1:1 giữa các thuộc tính trong đối tượng xác định và đối tượng hệ quả.
- ❖ Phụ thuộc hàm là bất biến theo thời gian, nghĩa là nó thỏa mãn tất cả các thể hiện có thể của quan hệ.
- ❖ Các phụ thuộc hàm là không hiển nhiên. Tất cả các phụ thuộc hàm hiển nhiên đúng đều được bỏ qua.

# CÁC LUẬT SUY DIỄN CHO CÁC PHỤ THUỘC HÀM (Cont.)

- ❖ Một tập các luật suy diễn là cần thiết để suy diễn ra tập các phụ thuộc hàm dựa vào F.
- ❖ Sáu luật suy diễn được biết đến nhiều nhất được áp dụng cho các phụ thuộc hàm như sau:
  - IR1: Luật phản xạ: nếu  $X \supseteq Y$ , thì  $X \rightarrow Y$
  - IR2: Luật tăng trưởng: nếu  $X \rightarrow Y$ , thì  $XZ \rightarrow YZ$
  - IR3: Luật bắc cầu: nếu  $X \rightarrow Y$  và  $Y \rightarrow Z$ , thì  $X \rightarrow Z$
- ⇒ *3 luật suy diễn này là Hệ tiên đề Armstrong: đóng vai trò là một tập luật cần thiết và đầy đủ cho việc tạo ra bao đóng của một tập các phụ thuộc hàm.*

# CÁC LUẬT SUY DIỄN CHO CÁC PHỤ THUỘC HÀM (Cont.)

IR4: Luật chiếu:

Nếu  $X \rightarrow YZ$ , thì  $X \rightarrow Y$  và  $X \rightarrow Z$

IR5: Luật cộng thêm:

Nếu  $X \rightarrow Y$  và  $X \rightarrow Z$ , thì  $X \rightarrow YZ$

IR6: Luật giả bắc cầu:

Nếu  $X \rightarrow Y$  và  $YZ \rightarrow W$ , thì  $XZ \rightarrow W$

# VÍ DỤ

Cho lược đồ quan hệ:  $R = (A, B, C, D, E, F, G, H, I, J)$  và  
 $F = \{AB \rightarrow E, AG \rightarrow J, BE \rightarrow I, E \rightarrow G, GI \rightarrow H\}$   
a. Hỏi  $F \vdash AB \rightarrow GH$ ?      b. Hỏi  $F \vdash BE \rightarrow H$ ?

Chứng minh:

1.  $AB \rightarrow E$ , đã cho trong  $F$
2.  $AB \rightarrow B$ , luật phản xạ IR1
3.  $AB \rightarrow BE$ , luật cộng thêm IR5 từ bước 1 và 2
4.  $BE \rightarrow I$ , đã cho trong  $F$
5.  $AB \rightarrow I$ , luật bắc cầu IR3 từ bước 3 và 4
6.  $E \rightarrow G$ , đã cho trong  $F$
7.  $AB \rightarrow G$ , luật bắc cầu IR3 từ bước 1 và 6
8.  $AB \rightarrow GI$ , luật cộng thêm IR5 từ bước 5 và 7
9.  $GI \rightarrow H$ , đã cho trong  $F$
10.  $AB \rightarrow H$ , luật bắc cầu IR3 từ bước 8 và 9
11.  $AB \rightarrow GH$ , luật cộng thêm IR5 từ bước 7 và 10  
=> kết quả được chứng minh.

# CÁC LUẬT SUY DIỄN CHO CÁC PHỤ THUỘC HÀM

- ❖ Gọi  $F$  là tập các phụ thuộc hàm xác định trên lược đồ quan hệ  $R$ .
- ❖ Ngoài các phụ thuộc hàm hiển nhiên, còn có nhiều các phụ thuộc hàm khác cũng thỏa mãn với các thể hiện của quan hệ mà đã thỏa mãn các phụ thuộc hàm trong  $F$ .
- ❖ *Tập tất cả các phụ thuộc hàm được suy diễn từ một tập phụ thuộc hàm  $F$  được gọi là bao đóng của  $F$  và được ký hiệu là  $F^+$ .*
  - Ký hiệu:  $F \models X \rightarrow Y$  cho biết phụ thuộc hàm  $X \rightarrow Y$  được suy diễn từ tập phụ thuộc hàm  $F$ .
  - $F^+ \equiv \{X \rightarrow Y \mid F \models X \rightarrow Y\}$

# XÁC ĐỊNH BAO ĐÓNG

- ❖  $F^+$  là bao đóng của tập phụ thuộc hàm  $F$ .  $F^+$  là tập (nhỏ nhất) tất cả các phụ thuộc hàm được sinh ra nhờ hệ tiên đề Armstrong.
- ❖  $F^+$  là hữu hạn nhưng có kích thước tăng theo cấp số nhân so với số thuộc tính của  $R$ .

Ví dụ: Quan hệ  $R=(A,B,C)$  và  $F = \{AB \rightarrow C, C \rightarrow B\}$ ,  $F^+$  sẽ bao gồm 29 phụ thuộc hàm (kể cả các phụ thuộc hàm hiển nhiên).

# XÁC ĐỊNH BAO ĐÓNG (Cont.)

- ❖ Để xác định liệu một phụ thuộc hàm  $X \rightarrow Y$  có thỏa mãn lược đồ quan hệ  $R$  với tập phụ thuộc hàm  $F$  hay không thì cần xem  $F \models X \rightarrow Y$  không, hoặc chính xác hơn là xem  $X \rightarrow Y$  có nằm trong  $F^+$  hay không.
- ⇒ **Mong muốn kiểm tra được  $X \rightarrow Y$  có nằm trong  $F^+$  hay không mà không cần sinh ra tất cả các thành phần của bao đóng.**
- ⇒ **Thực hiện bằng cách:** chỉ cần sinh ra bao đóng của tập thuộc tính  $X$ , ký hiệu là  $X^+$ , và kiểm tra xem liệu  $Y$  thuộc  $X^+$  hay không.

# THUẬT TOÁN TÍNH BAO ĐÓNG

- ❖  $X^+$  được gọi là bao đóng của tập thuộc tính  $X$  trên tập phụ thuộc hàm  $F$  nếu mọi thuộc tính trong  $X^+$  đều được sinh ra từ  $X$  nhờ  $F$ .

Thuật toán Closure {trả về  $X^+$  trên  $F$ }

**Đầu vào:** tập thuộc tính  $X$ , và một tập phụ thuộc hàm  $F$

**Đầu ra:** Tính  $X^+$  trên  $F$

Closure ( $X, F$ )

{

$X^+ \leftarrow X;$

repeat

$oldX^+ \leftarrow X^+;$

    for mỗi phụ thuộc hàm  $W \rightarrow Z$  trong  $F$  do

        if  $W \subseteq X^+$  then  $X^+ \leftarrow X^+ \cup Z;$

    until ( $oldX^+ = X^+$ );

}

# VÍ DỤ TÍNH BAO ĐÓNG

Cho  $F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$ .

Tính  $(AE)^+$

## Vòng 1

$$X^+ = \{A, E\}$$

Xét  $A \rightarrow D$ ,  $A \subseteq X^+$ , nên thêm  $D$  vào  $X^+$ ,  $X^+ = \{A, E, D\}$ ;  $F=F-(A \rightarrow D)$

Xét  $AB \rightarrow E$ ,  $AB$  không thuộc  $X^+$

Xét  $BI \rightarrow E$ ,  $BI$  không thuộc  $X^+$

Xét  $CD \rightarrow I$ ,  $CD$  không thuộc  $X^+$

Xét  $E \rightarrow C$ ,  $E \subseteq X^+$ , nên thêm  $C$  vào  $X^+$ ,  $X^+ = \{A, E, D, C\}$ ;  $F=F-(E \rightarrow C)$

Có thay đổi xảy ra với  $X^+$  so với ban đầu vì vậy cần thêm một vòng lặp nữa

## Vòng 2 : $F=\{AB \rightarrow E, BI \rightarrow E, CD \rightarrow I\}$

$$X^+ = \{A, E, D, C\}$$

Xét  $AB \rightarrow E$ ,  $AB$  không thuộc  $X^+$

Xét  $BI \rightarrow E$ ,  $BI$  không thuộc  $X^+$

Xét  $CD \rightarrow I$ ,  $CD \subseteq X^+$ , nên thêm  $I$  vào  $X^+$ ,  $X^+ = \{A, E, D, C, I\}$ ;  $F=F-(CD \rightarrow I)$

Có thay đổi xảy ra với  $X^+$  so với cuối vòng 1 vì vậy cần thêm một vòng lặp nữa

# VÍ DỤ TÍNH BAO ĐÓNG (Cont.)

Vòng 3  $F = \{AB \rightarrow E, BI \rightarrow E\}$

$$X^+ = \{A, E, D, C, I\}$$

Xét  $AB \rightarrow E$ ,  $AB$  không thuộc  $X^+$

Xét  $BI \rightarrow E$ ,  $BI$  không thuộc  $X^+$

Không có thay đổi nào xảy ra với  $X^+$  so với cuối vòng 2, vì vậy thuật toán dừng.

Vậy:  $(AE)^+ = \{A, E, C, D, I\}$

⇒ Điều này có nghĩa là các phụ thuộc hàm sau sẽ thuộc  $F^+$ :

$$AE \rightarrow AECDI$$

-Gs có 1 câu hỏi: Xét phụ thuộc hàm  $AE \rightarrow CDI$  có suy diễn từ  $F$  hay không?

trả lời: Có vì  $CDI$  là tập con của  $(AE)^+ \Rightarrow AE \rightarrow CDI$  suy diễn từ  $F$

# THUẬT TOÁN MEMBER

Thuật toán Member {xác định các thành viên trong  $F^+$ }

**Đầu vào:** một tập các phụ thuộc hàm  $F$ ,  
và một phụ thuộc hàm đơn  $X \rightarrow Y$

**Đầu ra:** Trả lại kết quả đúng (true) nếu  $F \models X \rightarrow Y$ ,  
ngược lại trả lại kết quả sai (false)

Member ( $F, X \rightarrow Y$ )

{

    if  $Y \subseteq \text{Closure}(X, F)$   
        then return true;  
        else return false;

}

# PHỦ VÀ SỰ TƯƠNG ĐƯƠNG CỦA PHỤ THUỘC HÀM

- ❖ Một tập phụ thuộc hàm F được phủ bởi một tập phụ thuộc hàm G (hay nói cách khác G phủ F) nếu mọi phụ thuộc hàm trong F đều nằm trong  $G^+$ .  
⇒ F được phủ nếu mọi phụ thuộc hàm trong F có thể được suy diễn từ G.
- ❖ Hai tập phụ thuộc hàm F và G là tương đương (ký hiệu:  $F \equiv G$ ) nếu  $F^+ = G^+$ .  
⇒ Mọi phụ thuộc hàm trong G có thể được suy diễn từ F và mọi phụ thuộc hàm trong F có thể được suy diễn từ G.
- ❖ Để xác định xem G có phủ F hay không: tính  $X^+$  trên G cho mỗi phụ thuộc hàm  $X \rightarrow Y$  trong F, nếu  $Y \subseteq X^+$  cho mọi  $X \rightarrow Y$  thì G phủ F.

## PHỦ VÀ SỰ TƯƠNG ĐƯƠNG CỦA PHỤ THUỘC HÀM (Cont.)

- ❖ Số phụ thuộc hàm càng ít đòi hỏi càng ít không gian nhớ.  
=> *Chi phí thấp khi thực hiện các thao tác cập nhật.*
- ❖ Có nhiều loại phủ, từ không dư thừa đến các phủ tối thiểu.  
=> *Không xem xét đến tất cả các loại này.*
- ❖ **Ý tưởng:** Sinh ra một tập các phụ thuộc hàm G tương đương với tập F ban đầu nhưng lại có số lượng phụ thuộc hàm càng ít càng tốt và càng tối giản càng tốt (phủ tối thiểu).
- ❖ **Thuật toán Member:** kiểm tra xem một phụ thuộc hàm  $X \rightarrow Y$  có thuộc vào tập F hay không. Thời gian chạy thuật toán phụ thuộc vào kích cỡ của tập phụ thuộc hàm => tập phụ thuộc hàm càng nhỏ thì thời gian chạy thuật toán càng nhanh.

# PHỦ KHÔNG DƯ THỪA

- ❖ Một tập các phụ thuộc hàm F được cho là không dư thừa nếu không có tập con thực sự G nào của F mà G tương đương với F. Nói cách khác, trong F không có phụ thuộc hàm dư thừa.

Ngược lại, nếu tồn tại thì F được gọi là dư thừa.

- ❖ F là một phủ không dư thừa của G nếu F là một phủ của G và F không dư thừa.
- ❖ ***Thuật toán Nonredundant:*** sinh ra một phủ không dư thừa.

# THUẬT TOÁN NONREDUNDANT

Thuật toán Nonredundant {sinh ra một phủ không dư thừa}

**Đầu vào:** một tập phụ thuộc hàm G

**Đầu ra:** một phủ không dư thừa của G

Nonredundant (G)

{

    F  $\leftarrow$  G;

    for mỗi phụ thuộc hàm  $X \rightarrow Y \in G$  do

        if Member(F -  $\{X \rightarrow Y\}$ ,  $X \rightarrow Y$ )

            then F  $\leftarrow$  F -  $\{X \rightarrow Y\}$ ;

    return (F);

}

# VÍ DỤ: TÌM PHỦ KHÔNG DƯ THỪA

Cho  $G = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C\}$ , tìm một phủ không dư thừa của  $G$ .

$F \leftarrow G$

Member( $\{B \rightarrow A, B \rightarrow C, A \rightarrow C\}$ ,  $A \rightarrow B$ )

Closure( $A, \{B \rightarrow A, B \rightarrow C, A \rightarrow C\}$ )

$A^+ = \{A, C\}$ , nên  $A \rightarrow B$  là không dư thừa

Member( $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ ,  $B \rightarrow A$ )

Closure( $B, \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ )

$B^+ = \{B, C\}$ , nên  $B \rightarrow A$  là không dư thừa

Member( $\{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$ ,  $B \rightarrow C$ )

Closure( $B, \{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$ )

$B^+ = \{B, A, C\}$ , nên  $B \rightarrow C$  là dư thừa,  $F = F - \{B \rightarrow C\}$

Member( $\{A \rightarrow B, B \rightarrow A\}$ ,  $A \rightarrow C$ )

Closure( $A, \{A \rightarrow B, B \rightarrow A\}$ )

$A^+ = \{A, B\}$ , nên  $A \rightarrow C$  là không dư thừa

Vậy  $F = \{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$  là một phủ không dư thừa của  $G$

## VÍ DỤ 2: TÌM PHỦ KHÔNG DƯ THỪA

Nếu  $G = \{A \rightarrow B, A \rightarrow C, B \rightarrow A, B \rightarrow C\}$ , giống tập phụ thuộc hàm trước nhưng khác nhau về thứ tự của các phụ thuộc hàm. Kết quả sẽ sinh ra một phủ không dư thừa khác!

$F \leftarrow G$

Member( $\{A \rightarrow C, B \rightarrow A, B \rightarrow C\}$ ,  $A \rightarrow B$ )

Closure( $A, \{A \rightarrow C, B \rightarrow A, B \rightarrow C\}$ )

$A^+ = \{A, C\}$ , nên  $A \rightarrow B$  không dư thừa

Member( $\{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$ ,  $A \rightarrow C$ )

Closure( $A, \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$ )

$A^+ = \{A, B, C\}$ , nên  $A \rightarrow C$  là dư thừa,  $F = F - \{A \rightarrow C\}$

Member( $\{A \rightarrow B, B \rightarrow C\}$ ,  $B \rightarrow A$ )

Closure( $B, \{A \rightarrow B, B \rightarrow C\}$ )

$B^+ = \{B, C\}$ , nên  $B \rightarrow A$  là không dư thừa

Member( $\{A \rightarrow B, B \rightarrow A\}$ ,  $B \rightarrow C$ )

Closure( $B, \{A \rightarrow B, B \rightarrow A\}$ )

$B^+ = \{B, A\}$ , nên  $B \rightarrow C$  là không dư thừa.

Vậy  $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$  là một phủ không dư thừa của  $G$

# PHỦ KHÔNG DƯ THỪA (Cont.)

## Một số lưu ý:

- ❖ Với một tập các phụ thuộc hàm cho trước có thể có nhiều hơn một phủ không dư thừa.
- ❖ Phủ không dư thừa của một tập các phụ thuộc hàm G có thể chứa những phụ thuộc hàm không nằm trong G.

Ví dụ:  $G = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C\}$

thì  $F = \{A \rightarrow B, B \rightarrow A, AB \rightarrow C\}$  là một phủ không dư thừa của G. Tuy nhiên, F chứa phụ thuộc hàm  $AB \rightarrow C$  không thuộc G.

# CÁC THUỘC TÍNH DƯ THỪA

- ❖ Nếu  $F$  là một tập các phụ thuộc hàm không dư thừa thì  $F$  không thể nhỏ hơn bằng cách loại bỏ các phụ thuộc hàm.
  - ❖ Để giảm kích cỡ của  $F$ , thực hiện loại bỏ các thuộc tính dư thừa từ các phụ thuộc hàm trong  $F$ .
  - ❖ Nếu  $F$  là tập các phụ thuộc hàm trên lược đồ quan hệ  $R$  và  $X \rightarrow Y \in F$  thì thuộc tính  $A$  được gọi là dư thừa trong  $X \rightarrow Y \in F$  nếu:
    1.  $X = AZ$ ,  $X \neq Z$  and  $\{F - \{X \rightarrow Y\}\} \cup \{Z \rightarrow Y\} \equiv F$ , hoặc
    2.  $Y = AW$ ,  $Y \neq W$  and  $\{F - \{X \rightarrow Y\}\} \cup \{X \rightarrow W\} \equiv F$
- ⇒ **Một thuộc tính  $A$  dư thừa trong  $X \rightarrow Y$  nếu  $A$  có thể được loại bỏ khỏi vé trái hoặc vé phải của phụ thuộc hàm mà không làm thay đổi  $F^+$ .**

## CÁC THUỘC TÍNH DƯ THỪA (Cont.)

### ❖ **Ví dụ:**

Cho  $F = \{A \rightarrow BC, B \rightarrow C, AB \rightarrow D\}$

thuộc tính C là dư thừa trong vế phải của  $A \rightarrow BC$   
tương tự, A là dư thừa trong vế trái của  $AB \rightarrow D$

# TẬP PHỤ THUỘC HÀM TỐI GIẢN TRÁI VÀ TỐI GIẢN PHẢI

Cho  $F$  là một tập các phụ thuộc hàm trên lược đồ  $R$  và cho  $X \rightarrow Y \in F$ .

- ❖  $X \rightarrow Y$  được gọi là tối giản trái nếu  $X$  không chứa các thuộc tính dư thừa  $A$ .
  - ❖ *Một phụ thuộc hàm tối giản về trái cũng được gọi là một phụ thuộc hàm đầy đủ.*
- ❖  $X \rightarrow Y$  được gọi là tối giản phải nếu  $Y$  không chứa các thuộc tính dư thừa  $A$ .
- ❖  $X \rightarrow Y$  được gọi là tối giản nếu nó tối giản trái, tối giản phải và  $Y$  khác rỗng.

# THUẬT TOÁN TỐI GIẢN TRÁI

Thuật toán tối giản trái sinh ra một tập các phụ thuộc hàm tối giản về trái.

Thuật toán tối giản trái {trả lại tập các phụ thuộc hàm tối giản về trái F}

**Đầu vào:** tập các phụ thuộc hàm G

**Đầu ra:** một phủ tối giản trái của G

Left-Reduce (G)

{

    F  $\leftarrow$  G;

    for mỗi phụ thuộc hàm  $X \rightarrow Y$  trong G do

        for mỗi thuộc tính A trong X do

            if Member(F, (X-A)  $\rightarrow$  Y)

                then loại bỏ A khỏi  $X \rightarrow Y$  của F

    return(F);

}

# THUẬT TOÁN TỐI GIẢN PHẢI

Thuật toán tối giản phải sinh ra một tập các phụ thuộc hàm tối giản về phải.

Thuật toán tối giản phải {trả lại tập các phụ thuộc hàm tối giản về trái F}

**Đầu vào:** tập các phụ thuộc hàm G

**Đầu ra:** một phủ tối giản phải của G

Right-Reduce (G)

{

$F \leftarrow G;$

    for mỗi phụ thuộc hàm  $X \rightarrow Y$  trong G do

        for mỗi thuộc tính A trong Y do

            if Member( $F - \{X \rightarrow Y\} \cup \{X \rightarrow (Y - A)\}$ ,  $X \rightarrow A$ )

                then loại bỏ A khỏi Y trong  $X \rightarrow Y$  của F

    return(F);

}

# THUẬT TOÁN TỐI GIẢN PHỤ THUỘC HÀM

Thuật toán tối giản phụ thuộc hàm sinh ra một tập các phụ thuộc hàm tối giản (tối thiểu).

Thuật toán tối giản phụ thuộc hàm

{trả lại tập các phụ thuộc hàm tối giản F}

**Đầu vào:** tập các phụ thuộc hàm G

**Đầu ra:** một tập các phụ thuộc hàm tối giản G

Reduce (G)

{

    F  $\leftarrow$  Right-Reduce( Left-Reduce(G));

    loại bỏ tất cả các phụ thuộc hàm dạng  $X \rightarrow \text{null}$  từ F

    return(F);

}

Nếu G chứa một phụ thuộc hàm dư thừa  $X \rightarrow Y$ , thì mọi thuộc tính trong Y sẽ là dư thừa, và do vậy sẽ tối giản hóa  $X \rightarrow \text{null}$ , nên sẽ phải loại bỏ các phụ thuộc hàm này.

# THUẬT TOÁN TỐI GIẢN PHỤ THUỘC HÀM (Cont.)

- ❖ Thứ tự thuật toán thực hiện rất quan trọng. **Tập các phụ thuộc hàm phải được tối giản về trái trước rồi mới tối giản về phải.**
- ❖ Ví dụ:

Cho  $G = \{B \rightarrow A, D \rightarrow A, BA \rightarrow D\}$

$G$  là tối giản phải nhưng không phải tối giản trái. Nếu tối giản trái  $G$  để sinh ra  $F = \{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$

# PHỦ TỐI THIỂU

- ❖ **Định nghĩa:** Một tập phụ thuộc hàm F là tối thiểu nếu:
  1. Mọi phụ thuộc hàm đều có về phải là một thuộc tính
  2. F là không dư thừa
  3. Không có phụ thuộc hàm nào dạng  $X \rightarrow A$  có thể được thay thế bởi dạng  $Y \rightarrow A$  với  $Y \subseteq X$  và vẫn là một tập tương đương. Nói cách khác F là một tối giản trái.

- ❖ **Ví dụ:**

$$G = \{A \rightarrow BCE, AB \rightarrow DE, BI \rightarrow J\}$$

Một phủ tối thiểu của G:

$$F = \{A \rightarrow B, A \rightarrow C, A \rightarrow D, A \rightarrow E, BI \rightarrow J\}$$

# THUẬT TOÁN TÌM PHỦ TỐI THIỂU

Thuật toán tìm phủ tối thiểu {trả lại phủ tối thiểu của F}

Đầu vào: tập các phụ thuộc hàm F

Đầu ra: một phủ tối thiểu của F

MinCover (F)

{

    G  $\leftarrow$  F;

    thay thế mỗi phụ thuộc hàm  $X \rightarrow A_1A_2\dots A_n$  của G  
        thành n phụ thuộc hàm  $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_n$

    Left-Reduce(G);

    Nonredundant(G);

    return(G);

}

- Nhóm 2
- <https://www.facebook.com/groups/3944644695584548>
- Nhóm 1. Nguyễn Hải Phú  
<https://www.facebook.com/groups/495163238404603>
- Nhóm 3. Lê tuấn anh
- <https://www.facebook.com/groups/1535005433530860>

- Nguyễn Hải Phú lên trang của Giáo vụ lấy danh sách lớp về rồi điểm danh cho cô nhé

# **CHUẨN HÓA LƯỢC ĐỒ QUAN HỆ**

# CHUẨN HÓA DỰA TRÊN KHÓA CHÍNH

## Xác định khóa cho một lược đồ quan hệ:

- ❖ **Định nghĩa:** Nếu R là một lược đồ quan hệ với các thuộc tính  $A_1, A_2, \dots, A_n$  và một tập các phụ thuộc hàm F trong đó  $X \subseteq \{A_1, A_2, \dots, A_n\}$  thì X là một khóa của R khi:
  1.  $X \rightarrow A_1 A_2 \dots A_n \in F^+$ , và
  2. Không có tập con thực sự nào  $Y \subseteq X$  mà  $Y \rightarrow A_1 A_2 \dots A_n \in F^+$ .
- ⇒ Về cơ bản, định nghĩa này cho thấy phải tạo ra bao đóng của tất cả các tập con có thể có của R và quyết định xem tập nào suy diễn ra được tất cả các thuộc tính của lược đồ.

# VÍ DỤ: XÁC ĐỊNH KHÓA

Cho  $r = (C, T, H, R, S, G)$  với tập phụ thuộc hàm

$$F = \{C \rightarrow T, HR \rightarrow C, HT \rightarrow R, CS \rightarrow G, HS \rightarrow R\}$$

Bước 1: Tính  $(A_i)^+$  với  $1 \leq i \leq n$

$$C^+ = \{CT\}, \quad T^+ = \{T\}, \quad H^+ = \{H\}$$

$$R^+ = \{R\}, \quad S^+ = \{S\}, \quad G^+ = \{G\}$$

⇒ Không có thuộc tính đơn nào là khóa của  $r$

$$\binom{6}{1} = \frac{6!}{1!(6-1)!} = \frac{720}{120} = 6$$

Bước 2: Tính  $X+= (A_i A_j)^+$  với  $1 \leq i \leq n, 1 \leq j \leq n$

$$\binom{6}{2} = \frac{6!}{2!(6-2)!} = \frac{720}{48} = 15$$

$$(CT)^+ = \{C, T\}, \quad (CH)^+ = \{CHTR\}, \quad (CR)^+ = \{CRT\}$$

$$(CS)^+ = \{CSGT\}, \quad (CG)^+ = \{CGT\}, \quad (TH)^+ = \{THRC\}$$

$$(TR)^+ = \{TR\}, \quad (TS)^+ = \{TS\}, \quad (TG)^+ = \{TG\}$$

$$(HR)^+ = \{HRCT\}, \quad (HS)^+ = \{HSRCTG\}, \quad (HG)^+ = \{HG\}$$

$$(RS)^+ = \{RS\}, \quad (RG)^+ = \{RG\}, \quad (SG)^+ = \{SG\}$$

Tập thuộc tính (HS) là một khóa của  $r$

# VÍ DỤ: XÁC ĐỊNH KHÓA (cont.)

Bước 3: Tính  $(A_i A_j A_k)^+$  với  $1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n$

$$(CTH)^+ = \{CTHR\},$$

$$(CTS)^+ = \{CTSG\},$$

$$(CHR)^+ = \{CHRT\},$$

$$(CHG)^+ = \{CHGTR\},$$

$$(CRG)^+ = \{CRGT\},$$

$$(THR)^+ = \{THRC\},$$

$$(THG)^+ = \{THGRC\},$$

$$(TRG)^+ = \{TRG\},$$

$$(HRS)^+ = \{HRSCTG\},$$

$$(HSG)^+ = \{HSGRCT\},$$

$$(CTR)^+ = \{CTR\}$$

$$(CTG)^+ = \{CTG\}$$

$$(CHS)^+ = \{CHSTRG\}$$

$$(CRS)^+ = \{CRSTG\}$$

$$(CSG)^+ = \{CSGT\}$$

$$(THS)^+ = \{THSRCG\}$$

$$(TRS)^+ = \{TRS\}$$

$$(TSG)^+ = \{TSG\}$$

$$(HRG)^+ = \{HRGCT\}$$

$$(RSG)^+ = \{RSG\}$$

$$\binom{6}{3} = \frac{6!}{3! \times (6-3)!} = \frac{720}{36} = 20$$

Các siêu khóa được đánh dấu màu đỏ.

# VÍ DỤ: XÁC ĐỊNH KHÓA (cont.)

Bước 4: Tính  $(A_i A_j A_k A_r)^+$  với  $1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n, 1 \leq r \leq n$

$$(CTHR)^+ = \{CTHR\},$$

$$(CTHS)^+ = \{CTHSRG\}$$

$$\binom{6}{4} = \frac{6!}{4! \times (6-4)!} = \frac{720}{48} = 15$$

$$(CTHG)^+ = \{CTHGR\},$$

$$(CHRS)^+ = \{CHRSTG\}$$

$$(CHRG)^+ = \{CHRG\},$$

$$(CRSG)^+ = \{CRSGT\}$$

$$(THRS)^+ = \{THRSCG\},$$

$$(THRG)^+ = \{THRG\}$$

$$(TRSG)^+ = \{TRSG\},$$

$$(HRSG)^+ = \{HRSGCT\}$$

$$(CTRS)^+ = \{CTRS\},$$

$$(CTSG)^+ = \{CTSG\}$$

$$(CSHG)^+ = \{CSHGTR\},$$

$$(THSG)^+ = \{THSGRC\}$$

$$(CTRG)^+ = \{CTRG\}$$

Các siêu khóa được đánh dấu màu đỏ.

# VÍ DỤ: XÁC ĐỊNH KHÓA (cont.)

Bước 5: Tính  $(A_i A_j A_k A_r A_s)^+$  với  $1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n,$   
 $1 \leq r \leq n, 1 \leq s \leq n$

$$(CTHRS)^+ = \{CTHSRG\}$$

$$(CTHRG)^+ = \{CTHGR\}$$

$$(CTHSG)^+ = \{CTHSGR\}$$

$$(CHRSG)^+ = \{CHRSGT\}$$

$$(CTRSG)^+ = \{CTRSG\}$$

$$(THRSG)^+ = \{THRSGC\}$$

Các siêu khóa được đánh dấu màu đỏ

$$\binom{6}{5} = \frac{6!}{5!(6-5)!} = \frac{720}{120} = 6$$

# VÍ DỤ: XÁC ĐỊNH KHÓA (cont.)

Bước 6: Tính  $(A_i A_j A_k A_r A_s A_t)^+$  với  $1 \leq i \leq n, 1 \leq j \leq n, 1 \leq k \leq n,$   
 $1 \leq r \leq n, 1 \leq s \leq n, 1 \leq t \leq n$

$$(CTHRSG)^+ = \{CTHSRG\}$$

$$\binom{6}{6} = \frac{6!}{6! \times (6-6)!} = \frac{720}{720} = 1$$

Siêu khóa được đánh dấu màu đỏ.

*Với 6 thuộc tính, số trường hợp phải xét là:*

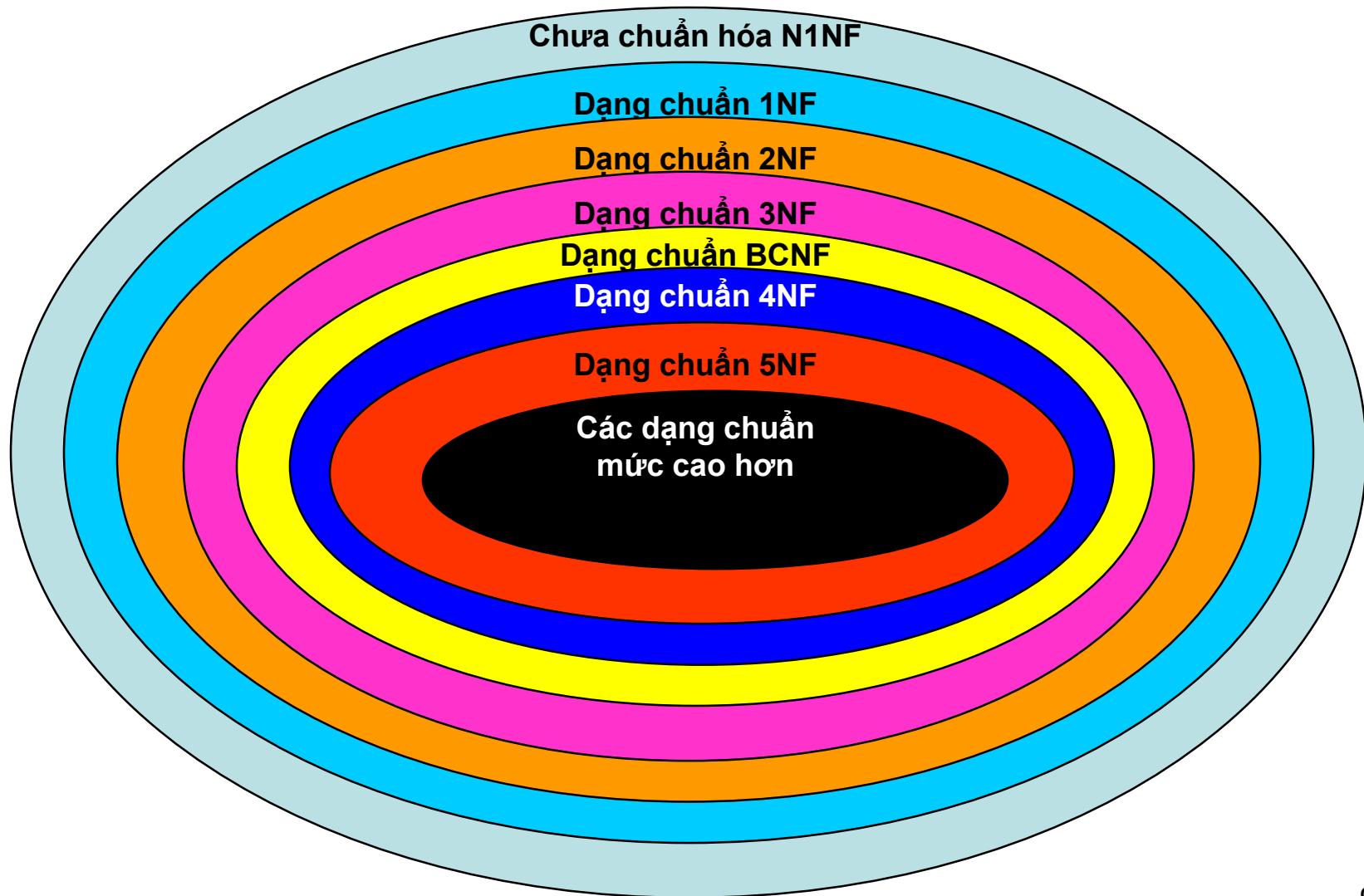
$$\binom{6}{1} + \binom{6}{2} + \binom{6}{3} + \binom{6}{4} + \binom{6}{5} + \binom{6}{6} = 6 + 15 + 20 + 15 + 1 = 63$$

Bài tập: Tìm tất cả các khóa của  $R = (A, B, C, D)$  với tập  
phụ thuộc hàm  $F = \{A \rightarrow B, B \rightarrow C\}$

# CHUẨN HÓA DỰA TRÊN KHÓA CHÍNH (Cont.)

- ❖ **Chuẩn hóa** là một kỹ thuật chính thức dùng cho việc phân tích các quan hệ dựa trên khóa chính (hoặc khóa dự bị), các thuộc tính và các phụ thuộc hàm.
- ❖ Kỹ thuật chuẩn hóa liên quan đến tập các luật được sử dụng để kiểm tra các quan hệ sao cho CSDL có thể đạt chuẩn hóa tới một mức nào đó.
- ❖ Khi quan hệ vi phạm một luật, cần phải tách nó ra thành một số các quan hệ khác nhỏ hơn.
- ❖ Việc chuẩn hóa được thực hiện bao gồm một số các bước, mỗi bước liên quan đến một dạng chuẩn cụ thể với các tính chất rõ ràng.

# QUAN HỆ GIỮA CÁC DẠNG CHUẨN



# CÁC YÊU CẦU CHUẨN HÓA

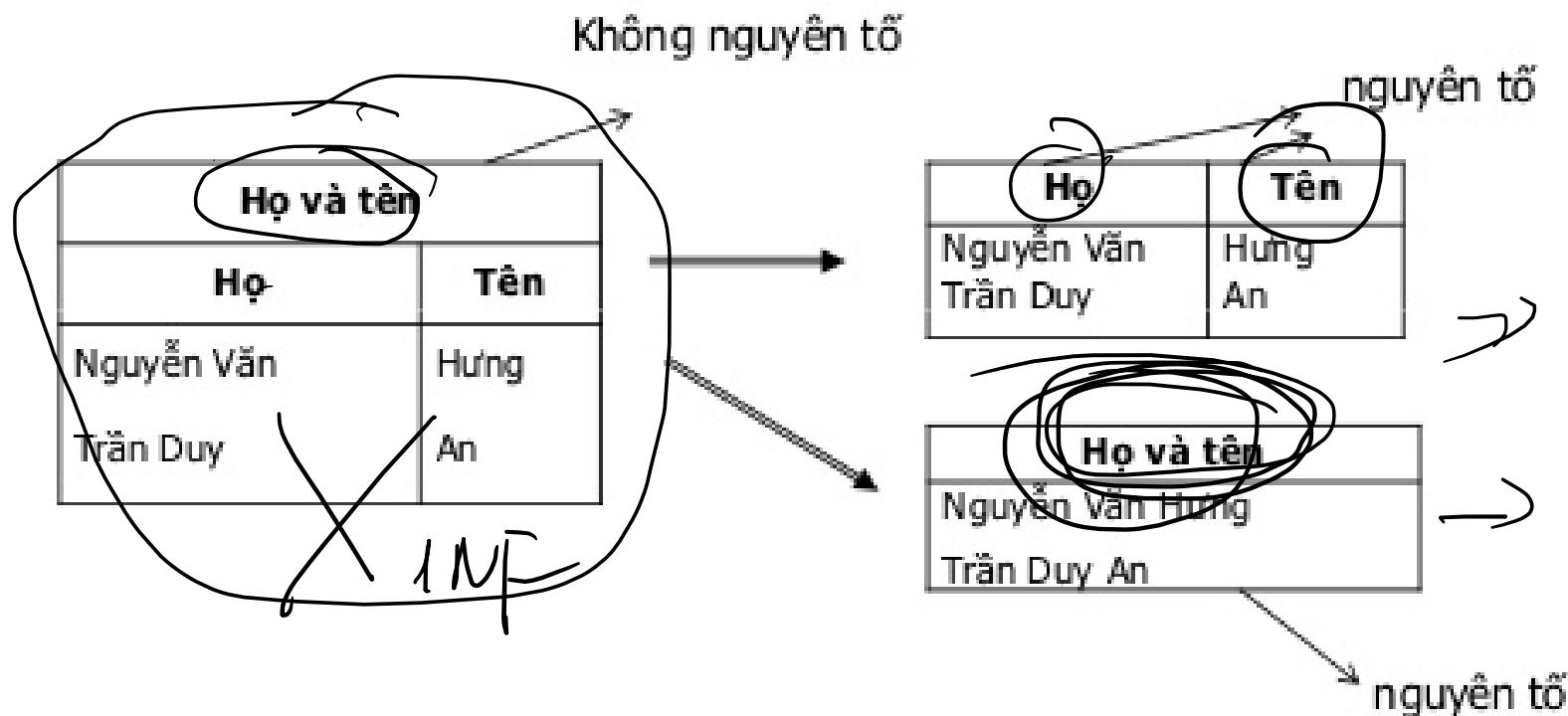
- ❖ Đối với mô hình quan hệ, một vấn đề rất quan trọng và thiết yếu là phải nhận ra được một quan hệ vừa tạo ra đã ở dạng chuẩn 1 (1NF) hay chưa. Tất cả các dạng chuẩn ở mức cao hơn sau đó là tùy theo từng trường hợp, có thể có hoặc không.
- ❖ Tuy nhiên, để tránh trường hợp dữ liệu thông tin khi cập nhật dữ liệu, người thiết kế CSDL thường được khuyến nghị là phải đưa toàn bộ các quan hệ trong CSDL về ít nhất là dạng chuẩn 3 (3NF).
- ❖ Việc chuẩn hóa sẽ được thực hiện từ dạng chưa chuẩn hóa, đưa về dạng chuẩn 1, sau đó đưa về dạng chuẩn 2, dạng chuẩn 3, ... (đến các dạng chuẩn ở các mức cao hơn).

## DẠNG CHƯA CHUẨN HÓA (Non-first Nomal Form - N1NF)

- ❖ Các quan hệ ở dạng chưa chuẩn hóa đồng nghĩa với việc chúng chưa ở dạng chuẩn 1.
- ❖ **Các quan hệ chưa ở dạng chuẩn 1** chứa một hoặc một số thuộc tính không nguyên tố, các thuộc tính lặp, và các thuộc tính dẫn xuất.
- ❖ **Thuộc tính chưa giá trị nguyên tố:** là những thuộc tính chưa giá trị đơn và không thể phân rã được nữa.

# VÍ DỤ

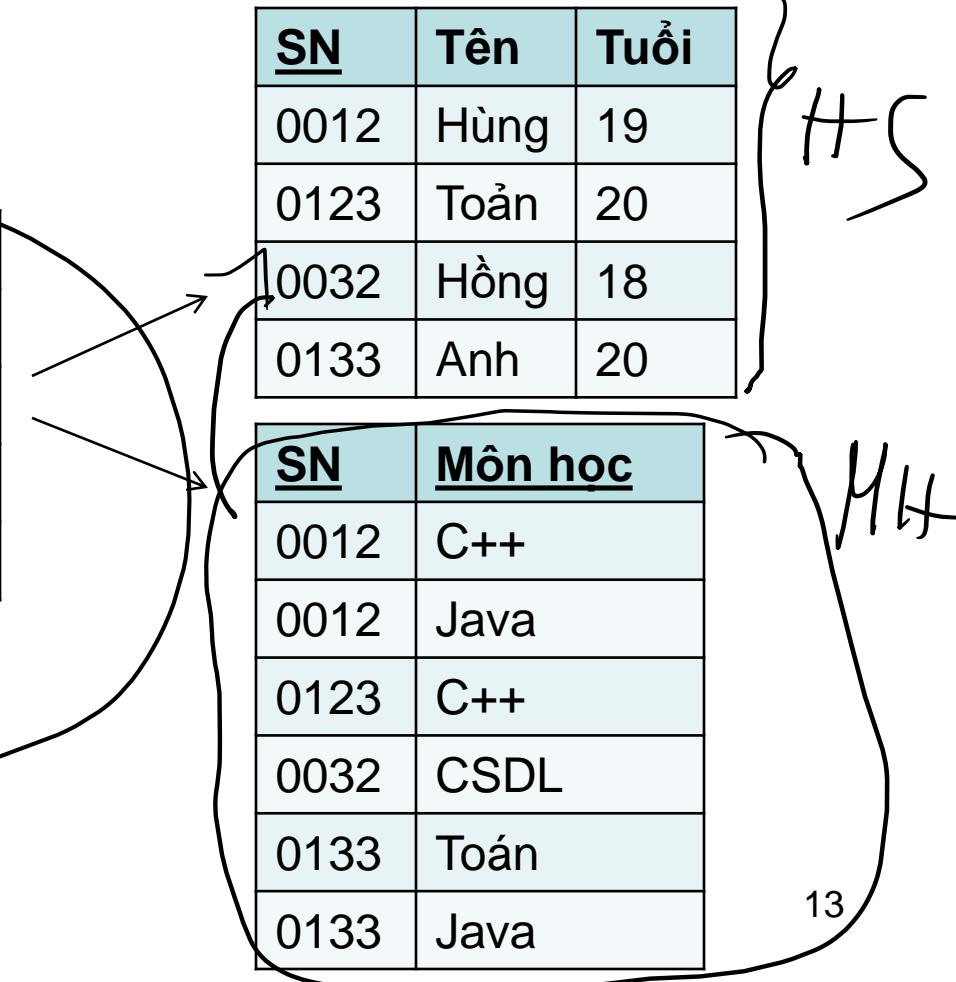
- Chuyển đổi các thuộc tính ghép về các thuộc tính đơn.



# VÍ DỤ

- Chuyển đổi các thuộc tính đa trị thành các thuộc tính đơn trị.

<u>SN</u>	Tên	Tuổi	Môn học
0012	Hùng	19	C++, Java
0123	Toản	20	C++
0032	Hồng	18	CSDL
0133	Anh	20	Toán, Java



# DẠNG CHUẨN 1 (First Nomal Form - 1NF)

## ❖ *Một quan hệ ở dạng chuẩn 1:*

- ❖ Mọi giá trị thuộc tính của quan hệ đều ở dạng nguyên tố.
- ❖ Không có thuộc tính đa trị.
- ❖ Không có thuộc tính dẫn xuất.

# Một số khái niệm về khóa

- **Siêu khóa:** Là một tập các thuộc tính xác định duy nhất thực thể trong quan hệ.
- **Khóa:** là một siêu khóa mà khi loại bỏ bất kỳ thuộc tính nào từ khóa này thì nó không còn là một siêu khóa nữa. Nghĩa là, khóa có số thuộc tính là nhỏ nhất.
- **Khóa dự bị:** là một tập các thuộc tính khóa nhỏ nhất của lược đồ quan hệ.
- **Khóa chính:** là một khóa dự bị được chọn ra. Tất cả các khóa dự bị còn lại trở thành khóa phụ hay khóa thứ cấp.

## Một số khái niệm về khóa (tiếp)

- **Thuộc tính khóa:** là thuộc tính của quan hệ R và là thành viên của một khóa dự bị nào đó.
- **Thuộc tính không khóa:** là thuộc tính của quan hệ R mà không phải là thành viên của một khóa dự bị nào.

## DẠNG CHUẨN 2 (Second Nomal Form - 2NF)

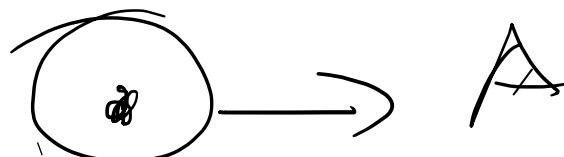
- ❖ **Dạng chuẩn 2** dựa trên khái niệm **phụ thuộc hàm đầy đủ**.
- ❖ **Một phụ thuộc hàm  $X \rightarrow Y$  được gọi là phụ thuộc hàm đầy đủ** nếu loại bỏ bất kỳ thuộc tính A nào của X thì sẽ làm phụ thuộc hàm này không còn đúng nữa. Nói cách khác  $X \rightarrow Y$  tối giản trái.  
=> Không có bất kỳ thuộc tính  $A \in X$  nào mà  $X-\{A\} \rightarrow Y$
- ❖ **Một phụ thuộc hàm  $X \rightarrow Y$  là phụ thuộc hàm không đầy đủ** nếu tồn tại một thuộc tính A nào đó của X bị loại bỏ mà phụ thuộc hàm đó vẫn đúng.  
=> Tồn tại một thuộc tính  $A \in X$  mà  $X-\{A\} \rightarrow Y$

## DẠNG CHUẨN 2 (Second Nomal Form - 2NF) (Cont.)

- ❖ **Định nghĩa dạng chuẩn 2:** Một lược đồ quan hệ R với tập phụ thuộc hàm F ở dạng chuẩn 2NF khi thỏa mãn:
  - Là dạng chuẩn **1NF**
  - Mọi thuộc tính không khóa đều phụ thuộc hàm đầy đủ vào mọi khóa dự bị của R.

## DẠNG CHUẨN 2 (Second Nomal Form - 2NF) (Cont.)

- ❖ Một lược đồ quan hệ R với tập phụ thuộc hàm F ở dạng chuẩn 2NF khi thỏa mãn một trong các điều kiện sau:
  - Tất cả các khóa dự bị đều có một thuộc tính
  - Không có thuộc tính không khóa nào.
  - {Không tồn tại thuộc tính không khóa mà phụ thuộc hàm một phần vào một khóa dự bị của R.}



$R \setminus (P, G)$ 

ví dụ  
=  $\{A, D\}$

 $(AD)^+ = ADPG$ 

Ví dụ: Cho  $R = (A, D, P, G)$ , với tập phụ thuộc hàm

$F = \{AD \rightarrow P, A \rightarrow G\}$  và một khóa  $K = \{AD\}$

$R$  không ở dạng chuẩn 2NF vì  $G$  phụ thuộc một phần vào khóa  $AD$  do  $A \rightarrow G$ .

Phân rã  $R$  thành:

 $R_1 = (A, D, P)$  $K_1 = \{AD\}$  $F_1 = \{AD \rightarrow P\}$  $R_2 = (A, G)$  $K_2 = \{A\}$  $F_2 = \{A \rightarrow G\}$

## CHUẨN HÓA TỪ 1NF SANG 2NF

- ❖ Loại bỏ các thuộc tính không khóa mà phụ thuộc một phần vào khóa chính để tách thành bảng riêng. Khóa chính của bảng riêng là bộ phận khóa mà chúng phụ thuộc vào.
- ❖ Các thuộc tính còn lại lập thành một bảng với khóa chính là khóa ban đầu.

# DẠNG CHUẨN 3

## (Third Nomal Form - 3NF)

- ❖ **Dạng chuẩn 3** dựa trên khái niệm phụ thuộc hàm bắc cầu.
- ❖ **Định nghĩa phụ thuộc hàm bắc cầu:** Cho một lược đồ quan hệ R và một tập các phụ thuộc hàm F của R, một tập con  $X \subseteq R$  và một thuộc tính  $A \in R$ . A được gọi là phụ thuộc hàm bắc cầu vào X nếu tồn tại  $Y \subseteq R$  mà  $X \rightarrow Y$ ,  $Y$  không  $\rightarrow X$  và  $Y \rightarrow A$ , và  $A \notin X \cup Y$ .
- ❖ **Một cách định nghĩa khác cho phụ thuộc hàm bắc cầu:** Một phụ thuộc hàm  $X \rightarrow Y$  trong lược đồ quan hệ R là phụ thuộc bắc cầu nếu có một tập thuộc tính  $Z \subseteq R$  mà Z không phải là tập con của bất kỳ khóa nào của R nhưng  $X \rightarrow Z$  và  $Z \rightarrow Y$ .

## DẠNG CHUẨN 3 (Third Nomal Form - 3NF) (Cont.)

- **Định nghĩa dạng chuẩn 3NF:** Một lược đồ quan hệ R ở dạng 3NF với một tập phụ thuộc hàm F nếu thỏa mãn:
  - Với bất kỳ phụ thuộc hàm  $X \rightarrow A$  trong F thì hoặc X là một siêu khóa của R, hoặc A là một thuộc tính khóa.
- **Một cách định nghĩa khác về dạng chuẩn 3NF:** Một lược đồ quan hệ R ở dạng 3NF với một tập phụ thuộc hàm F nếu thỏa mãn:
  - R ở dạng 2NF
  - Không có thuộc tính không khóa nào phụ thuộc hàm bắc cầu vào khóa của R.

## DẠNG CHUẨN 3 (Third Normal Form - 3NF) (Cont.)

### ❖ *Chuẩn hóa về 3NF (cách 1):*

- Tìm phủ tối thiểu của tập phụ thuộc hàm của lược đồ ban đầu
- Loại bỏ các thuộc tính phụ thuộc bắc cầu ra khỏi quan hệ, lập thành bảng mới với khóa chính là thuộc tính bắc cầu
- Các thuộc tính còn lại lập thành một bảng với khóa chính là khóa ban đầu.

## DẠNG CHUẨN 3 (Third Normal Form - 3NF) (Cont.)

### ❖ *Chuẩn hóa về 3NF (cách 2):*

- Tìm phủ tối thiểu (G) của tập phụ thuộc hàm (F) của lược đồ (R) ban đầu
- Ứng với mỗi phụ thuộc hàm trong G, lập một bảng quan hệ mới
- Nếu trong tất cả các bảng mới tạo ra tại bước 2 không có bảng nào chứa một khóa dự bị nào của R thì tạo ra thêm một bảng mới chứa một khóa dự bị của R.

## VÍ DỤ 1

Cho  $R = (A, B, \overbrace{C, D})$ , khóa  $K = \underline{\{A, B\}}$ , và tập phụ thuộc hàm  $F = \{AB \rightarrow C, C \rightarrow D\}$

R không là dạng chuẩn 3NF vì D là một thuộc tính không khóa mà lại phụ thuộc bắc cầu vào khóa AB.

$$R_1 = \{ \overline{A, B, C} \} \quad R_2 = \{ \overline{C, D} \}$$

$$K_1 = \{ AB \} \quad K_2 = \{ C \}$$

$$F_1 = \{ AB \rightarrow C \} \quad F_2 = \{ C \rightarrow D \}$$

## VÍ DỤ 2

Cho  $R = (A, B, C, D)$ , và tập phụ thuộc hàm

$$F = \{AB \rightarrow C, C \rightarrow B, A \rightarrow D\}$$

Chuẩn hóa  $R$  về dạng chuẩn 3NF bằng 2 cách, so sánh kết quả của 2 cách đó.

Khóa dự bị  $K1=(AB)$ ,  $K2=(AC) \Rightarrow$  thuộc tnhks khóa là  $(A,B,C)$

Xét tập các pth

$AB \rightarrow C$ : Có  $AB$  là siêu khóa  $\Rightarrow$  tm

$C \rightarrow B$ :  $B$  là thuộc tính khóa  $\Rightarrow$  thm

$A \rightarrow D$ :  $A$  Không phải là siêu khóa (vì  $A+=AD!=R$ ) và  $D$  cũng không phải thuộc tính khóa

$\Rightarrow$  Không ở dạng chuẩn 3NF

Kiểm tra có ở 2NF: Ko thm 2NF vì:

$AB \rightarrow C$ : tm

$C \rightarrow B$ : tm;  $A \rightarrow D$ :  $D$  là thuộc tính không khóa phụ thuộc 1 phần (ko đầy đủ) vào khóa

- Đưa về chuẩn 2 sẽ là chuẩn 3

-R2=(A,D)

F2={A → D}

K2=(A)

-R1=(A,B,C)

F1={AB → C, C → B}

K1=(AB)

KT duy thừa PTH:

# VÍ DỤ 3

StaffBranch

staff#	sname	position	salary	branch#	baddress

```
graph TD; staff#[staff#] --> sname[sname]; staff#[staff#] --> position[position]; staff#[staff#] --> salary[salary]; staff#[staff#] --> branch#[branch#]; staff#[staff#] --> baddress[baddress]; branch#[branch#] --> baddress[baddress]
```

Tập các phụ thuộc hàm:

staff# → sname, position, salary, branch#, baddress

branch# → baddress

baddress → branch#

Lược đồ StaffBranch thuộc dạng chuẩn nào? Hãy chuẩn hóa về dạng chuẩn 3NF

# VÍ DỤ 4

StaffBranch

staff#	sname	position	salary	branch#	baddress

```
graph TD; staff#[staff#] --> sname[sname]; staff#[staff#] --> position[position]; staff#[staff#] --> salary[salary]; staff#[staff#] --> branch#[branch#]; staff#[staff#] --> baddress[baddress]; branch#[branch#] --> baddress[baddress]; baddress[baddress] --> branch#[branch#]
```

Tập các phụ thuộc hàm:

$\text{staff\#} \rightarrow \text{sname, position, salary, branch\#, baddress}$

$\text{branch\#} \rightarrow \text{baddress}$

$\text{baddress} \rightarrow \text{branch\#}$

$\text{branch\#, position} \rightarrow \text{salary}$

$\text{baddress, position} \rightarrow \text{salary}$

Lược đồ StaffBranch thuộc dạng chuẩn nào? Hãy chuẩn hóa về dạng chuẩn 3NF

# VÍ DỤ 5

- Cho các lược đồ cùng các phụ thuộc hàm sau:

R1 (A, B, C)

F1( A → BC) (Thm 3NF)

R2 (A, B, C)

F2 (A → BC, C → A) (thm 3NF)

R3 (A, B, C, D, E, F, G)

F3 (C → DE, E → F, ABC → G)

Khóa là (ABC):

R312=(C,D,E)

F312={C->DE}

K312=C

R311=(E,F)

F311=(E->F)

K311=(E)

R32=(A,B,C,G)

F32={ABC->G}

K32=(ABC)

Trong các lược đồ trên, lược đồ nào thuộc chuẩn 2NF? 3NF?  
Hãy chuyển các lược đồ trên sang dạng chuẩn 3NF.

# CHUẨN HÓA

- Chuẩn hóa về dạng 1NF: loại bỏ dữ liệu dư thừa
- Chuẩn hóa về 2NF: loại bỏ các phụ thuộc hàm bộ phận
- Chuẩn hóa về 3NF: loại bỏ các phụ thuộc hàm bắc cầu

## DẠNG CHUẨN BOYCE-CODD (Boyce-Codd Nomal Form - BCNF)

- ❖ Dạng chuẩn Boyce-Codd (BCNF) là một chuẩn nghiêm ngặt hơn chuẩn 3NF.
- ❖ **Định nghĩa:** Một lược đồ quan hệ R được coi là ở dạng chuẩn Boyce-Codd với tập phụ thuộc hàm F nếu với bất kỳ phụ thuộc hàm  $X \rightarrow A$  nào và  $A \not\subseteq X$ , thì X là một siêu khóa của R.
- ❖ **Ví dụ:** Cho lược đồ quan hệ  $R = (A, B, C)$ , tập phụ thuộc hàm  $F = \{AB \rightarrow C, C \rightarrow A\}$  và khóa  $K = \{AB\}$   
 $R$  không ở dạng chuẩn BCNF vì có  $C \rightarrow A$  và  $C$  không phải là một siêu khóa của  $R$ .

## DẠNG CHUẨN BOYCE-CODD (Boyce-Codd Normal Form - BCNF) (Cont.)

- ❖ Sự khác nhau giữa dạng chuẩn 3NF và BCNF là BCNF cho phép bỏ đi luật với phụ thuộc hàm  $X \rightarrow A$  thì  $X$  phải là thuộc tính khóa.
- ❖ Trong thực tế, hầu hết các lược đồ quan hệ ở dạng chuẩn 3NF thì cũng ở dạng chuẩn BCNF. Chỉ khi trong lược đồ có  $X \rightarrow A$  mà  $X$  không phải là một siêu khóa hoặc  $A$  là một thuộc tính khóa thì lược đồ này ở dạng chuẩn 3NF mà không ở dạng chuẩn BCNF.

## VÍ DỤ

- Hãy chỉ ra các phụ thuộc hàm vi phạm chuẩn BCNF trong các lược đồ sau:

R1 (A, B, C, D)

F1( AB → C, C → D, D → A)

R2 (A, B, C, D)

F2 (AB → C, BC → D, CD → A, AD → B)

R3 (A, B, C, D)

F3 (A → B, B → C, C → D, D → A)

R4 (A, B, C, D, E)

F4 (AB → C, DE → C, B → D)

R5 (A, B, C, D, E)

F5 (AB → C, C → D, D → B, D → E)

# PHÂN TÁCH LƯỢC ĐỒ VỀ CÁC DẠNG CHUẨN

- ❖ Mục đích cơ bản của thiết kế CSDL quan hệ là đảm bảo mọi quan hệ trong CSDL đều ở dạng chuẩn 3NF hoặc BCNF. Điều này giúp loại bỏ được hầu hết các dị thường thông tin khi cập nhật.
- ❖ Để đưa một lược đồ CSDL về dạng chuẩn 3NF hoặc BCNF, người thiết kế phải đảm bảo thỏa mãn 2 tính chất:
  1. Kết nối không mất mát thông tin
  2. Các phụ thuộc hàm được bảo toàn sau khi phân tách.

# PHÂN TÁCH LƯỢC ĐỒ VỀ CÁC DẠNG CHUẨN

(Cont.)

- ❖ Hiện tại, có một số thuật toán phân tách về 3NF đảm bảo cả 2 tính chất (1) và (2).
  - ❖ Tuy nhiên, không có thuật toán nào phân tách về BCNF mà đảm bảo cả tính chất (1) và (2).
    - Chỉ có một thuật toán phân tách về BCNF đảm bảo tính chất (1) nhưng không đảm bảo tính chất (2).
- => *Chuẩn 3NF được coi là dạng chuẩn mạnh trong khả năng có thể để phân tách các lược đồ quan hệ. Nếu cố gắng đưa về BCNF thì có thể dẫn tới việc không bảo toàn các phụ thuộc hàm.*

# **NGÔN NGỮ TRUY VĂN ĐẠI SỐ QUAN HỆ**

# NGÔN NGỮ ĐẠI SỐ QUAN HỆ

- ❖ Là một ngôn ngữ truy vấn thủ tục, bao gồm các phép toán tập hợp một ngôi hoặc hai ngôi, nghĩa là các toán hạng của chúng là một quan hệ hoặc hai quan hệ. Kết quả đầu ra là một quan hệ.
- ❖ **Năm phép toán cơ bản của đại số quan hệ** là phép chọn, phép chiếu, phép hợp, phép trừ và phép tích Đè-các.
- ❖ Một số các phép toán mở rộng khác cũng được định nghĩa trong đại số quan hệ bao gồm: phép giao, phép kết nối tự nhiên, phép chia, phép bán kết nối, và phép kết nối ngoài.

# PHÉP CHỌN

Loại: Một ngôi

Ký hiệu: Sigma,  $\sigma$

Khuôn dạng chung:  $\sigma_{(mệnh\ đề)}(\text{thể}\ hiện\ của\ quan\ hệ)$

Lược đồ của quan hệ kết quả: tương tự như quan hệ toán hạng

Kích thước của quan hệ kết quả (số bộ):  $\leq |\text{quan\ hệ\ toán\ hạng}|$

Ví dụ:

$\sigma_{(\text{major} = \text{"CS"})}(\text{STUDENTS})$

$\sigma_{(\text{major} = \text{"CS"} \wedge \text{hair-color} = \text{"brown"})}(\text{STUDENTS})$

$\sigma_{(\text{hours-attempted} > \text{hours-earned})}(\text{STUDENTS})$

- Phép chọn lựa chọn ra các bộ tử thể hiện của quan hệ sao cho thỏa mãn mệnh đề điều kiện cụ thể nào đó.
- Mệnh đề có thể chứa các toán tử so sánh, như  $=, \neq, <, \leq, >, \geq$ . Hoặc kết hợp với các phép toán liên kết và ( $\wedge$ ), hoặc ( $\vee$ ), và phủ định ( $\neg$ ).
- Phép chọn có thể được coi như một lát cắt ngang của quan hệ toán hạng.

# VÍ DỤ PHÉP CHỌN

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$$r = \sigma_{(A = 'a')}(R)$$

A	B	C	D
a	a	yes	1
a	d	no	6
a	c	no	7
a	a	yes	5

$$r = \sigma_{(A = 'a' \wedge C = "yes")}(R)$$

A	B	C	D
a	a	yes	1
a	a	yes	5

Một quan  
hệ rỗng

$$r = \sigma_{(B = 'm')}(R)$$

A	B	C	D

# Bài tập 1

Cho lược đồ quan hệ  
**SanPham (ten, gia, loai)**

Viết các biểu thức đại số quan hệ để:

- Liệt kê tất cả các sản phẩm thuộc loại “bia”
- Liệt kê các sản phẩm thuộc loại “bia” có giá lớn hơn 100.000đ.
- Liệt kê các sản phẩm có giá dưới 100.000đ hoặc thuộc loại “văn phòng phẩm”

# PHÉP CHIẾU

Loại: Một ngôi

Ký hiệu: Pi,  $\pi$

Khuôn dạng chung:  $\pi_{(\text{danh sách các thuộc tính})}(\text{thể hiện của quan hệ})$

Lược đồ của quan hệ kết quả: được xác định bởi <danh sách các thuộc tính>

Kích thước của quan hệ kết quả (số bộ):  $\leq |\text{quan hệ toàn hạng}|$

Ví dụ:

$\pi_{(\text{student-id, name, major})}(\text{STUDENTS})$

$\pi_{(\text{name, advisor})}(\text{STUDENTS})$

$\pi_{(\text{name, gpa, hours-attempted})}(\text{STUDENTS})$

- Phép chiếu có thể được coi như một lát cắt dọc của quan hệ toàn hạng.
- Nếu phép toán sinh ra các bộ giống hệt nhau, thì sẽ chỉ giữ lại một bộ và loại bỏ đi các bộ bị trùng.

# VÍ DỤ PHÉP CHIỀU

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6
a	c	no	7
b	b	no	69
c	a	yes	24
d	d	yes	47
h	d	yes	34
e	c	no	26
a	a	yes	5

$r = \pi_{(A, C)}(R)$

A	C
a	yes
b	no
c	yes
a	no
d	yes
h	yes
e	no

$r = \pi_{(A, D)}(R)$

A	D
a	1
b	7
c	34
a	6
a	7
b	69
c	24
d	47
h	34
e	26
a	5

$r = \pi_{(C)}(R)$

C
yes
no

# Bài tập 2

Cho lược đồ quan hệ  
**SanPham (ten, gia, loai)**

Viết các biểu thức đại số quan hệ để:

- Liệt kê giá của tất cả các sản phẩm.
- Liệt kê giá của các sản phẩm thuộc loại “bia”.
- Liệt kê giá các sản phẩm có giá dưới 100.000đ hoặc thuộc loại “văn phòng phẩm”

# PHÉP HỢP

Loại: hai ngôi

Ký hiệu:  $\cup$

Khuôn dạng chung:  $r \cup s$ , với  $r$  và  $s$  là 2 quan hệ khả hợp

Lược đồ quan hệ kết quả: Lược đồ của các quan hệ toán hạng

Kích thước của quan hệ kết quả (số bộ):  $\leq \max\{ |r| + |s| \}$

Ví dụ:

$$r \cup s$$

$$\pi_{(a, b)}(r) \cup \pi_{(a, b)}(s)$$

- Phép hợp cung cấp một phương tiện để trích lọc thông tin nằm trên hai quan hệ toán hạng khả hợp với nhau.
- **2 quan hệ  $r(R)$  và  $s(S)$  được gọi là khả hợp** khi thỏa mãn 2 điều kiện sau:
  - ✓ Chúng phải có cùng số bậc hay cùng số lượng thuộc tính.
  - ✓ Miền giá trị của thuộc tính thứ (i) của  $r$  và thuộc tính thứ (j) của  $s$  phải giống nhau, cho mọi giá trị của  $I, j$ .

# VÍ DỤ PHÉP HỢP

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

T

A	B
a	a
b	d
c	f
a	d
a	c

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

X

A	B
a	a
b	d
a	c

$$r = R \cup T$$

Không hợp lệ – R và T  
không phải là 2 quan hệ  
khả hợp

$$r = R \cup S$$

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7
a	m	4
b	c	22
a	d	16

$$r = T \cup X$$

A	B
a	a
b	d
c	f
a	d
a	c

# Bài tập 3

Cho lược đồ quan hệ  
**SanPham (ten, gia, loai)**

Viết các biểu thức đại số quan hệ để:

- Liệt kê giá của tất cả các sản phẩm thuộc loại “bia” hoặc thuộc loại “văn phòng phẩm.

# PHÉP TRỪ

Loại: Hai ngôi

Ký hiệu: –

Khuôn dạng chung:  $r - s$ , với  $r$  và  $s$  là hai quan hệ khả hợp

Lược đồ quan hệ kết quả: Lược đồ của quan hệ toán hạng

Kích thước của quan hệ kết quả (số bộ):  $\leq | \text{quan hệ } r |$

Ví dụ:  $r - s$

- Phép trừ cho phép trích lọc thông tin được chứa trong một quan hệ mà nó không được chứa trong quan hệ thứ hai.
- Tương tự như phép hợp, phép trừ yêu cầu 2 quan hệ toán hạng phải là khả hợp.

# VÍ DỤ PHÉP TRỪ

R

A	B	D
a	a	1
b	d	7
c	f	34
a	d	6
a	c	7

T

A	B
a	a
b	d
c	f
a	d
a	c

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

X

A	B
a	a
b	d
a	c

$$r = R - T$$

Không hợp lệ – R và T  
không phải là 2 quan hệ  
khả hợp

$$r = T - X$$

A	B
c	f
a	d

$$r = R - S$$

E	F	G
a	a	1
b	d	7
c	f	34
a	d	6

$$r = S - R$$

E	F	G
a	m	4
b	c	22
a	d	16

$$r = X - T$$

A	B

Quan hệ rỗng

# Bài tập 4

Cho lược đồ quan hệ  
**SanPham (ten, gia, loai)**

Viết các biểu thức đại số quan hệ để:

- **Liệt kê giá** của tất cả các sản phẩm không thuộc loại “bia”.

# PHÉP TÍCH ĐỀ-CÁC

Loại: Hai ngôi

Ký hiệu:  $\times$

Khuôn dạng chung:  $r \times s$  (không giới hạn trên  $r$  và  $s$ )

Lược đồ quan hệ kết quả: lược đồ  $r \times$  lược đồ  $s$  với việc thay đổi tên gọi  
một số thuộc tính

Kích thước của quan hệ kết quả (số bộ):  $> |$  quan hệ  $r$   $|$  và  $> |$  quan hệ  $s$   $|$

Ví dụ:

$r \times s$

- Tích Đề-các cho phép kết nối 2 quan hệ bất kỳ thành một quan hệ đơn.
- Một quan hệ là một tập con của tích Đề-các tập các miền giá trị.

# VÍ DỤ PHÉP TÍCH ĐỀ-CÁC

T

A	B
a	a
b	d

X

A	B
a	a
b	d
a	c
c	a

$$r = T \times X$$

T.A	T.B	X.A	X.B
a	a	a	a
a	a	b	d
a	a	a	c
a	a	c	a
b	d	a	a
b	d	b	d
b	d	a	c
b	d	c	a

# VÍ DỤ PHÉP TÍCH ĐỀ-CÁC

R

A	B	C	D
a	a	1	yes
b	d	7	yes
c	f	34	no

S

X	Y	Z
a	m	4
b	c	22
a	d	16
a	c	7

$$r = R \times S$$

A	B	C	D	X	Y	Z
a	a	1	yes	a	m	4
a	a	1	yes	b	c	22
a	a	1	yes	a	d	16
a	a	1	yes	a	c	7
b	d	7	yes	a	m	4
b	d	7	yes	b	c	22
b	d	7	yes	a	d	16
b	d	7	yes	a	c	7
c	f	34	no	a	m	4
c	f	34	no	b	c	22
c	f	34	no	a	d	16
c	f	34	no	a	c	7

# Bài tập 5

Cho lược đồ quan hệ

SanPham (tenSP, gia, loai)

CungCap (MaCC, tenSP, SoLuong)

Viết các biểu thức đại số quan hệ để:

- Liệt kê mã của tất cả các nhà cung cấp mà cung cấp các sản phẩm thuộc loại “bia”.

# BIỂU DIỄN NGÔN NGỮ ĐẠI SỐ QUAN HỆ

- ❖ Mỗi toán tử đại số quan hệ cơ bản (trong số năm toán tử) có thể được sử dụng riêng rẽ để hình thành một truy vấn. Tuy nhiên, sức mạnh thực sự được cải thiện đáng kể khi chúng được kết hợp với nhau để hình thành các biểu thức truy vấn.
- ❖ Phần sau xem xét việc tạo ra các tổ hợp phức tạp hơn của 5 phép toán cơ bản (trước khi giới thiệu các phép toán mở rộng trong đại số quan hệ => Có được đánh giá cao hơn cho các phép toán mở rộng).

# TOÁN TỬ ĐẶT LẠI TÊN

- ❖ Không giống như các quan hệ trong CSDL, các quan hệ trung gian được sinh ra từ các kết quả của truy vấn không có tên gọi để tham chiếu đến.
- ❖ Nếu các quan hệ này không được lưu trữ một cách tường minh, nó sẽ bị mất sau khi truy vấn được thực thi.
- ❖ Tuy nhiên, trong một số trường hợp cần lưu trữ lại các quan hệ trung gian, ví dụ: sử dụng kết quả cho một truy vấn khác, ...

# TOÁN TỬ ĐẶT LẠI TÊN (Cont.)

❖ *Biểu diễn toán tử đặt lại tên:*

Chữ cái Hy Lạp thường rho ( $\rho$ ).

❖ *Khuôn dạng chung đầu tiên của toán tử đặt lại tên cho các quan hệ:*

$\rho_{tên\ mới\ của\ quan\ hệ}$ (quan hệ)

❖ *Ví dụ:*

$\rho_x(r)$ : đặt lại tên cho quan hệ  $r$  thành  $x$ .

## TOÁN TỬ ĐẶT LẠI TÊN (Cont.)

- ❖ Dạng thứ hai là *toán tử đặt lại tên cho cả quan hệ và thuộc tính*.
- ❖ Giả sử quan hệ toán hạng có cấp  $n$ , thì *dạng thức của toán tử đặt lại tên* là:

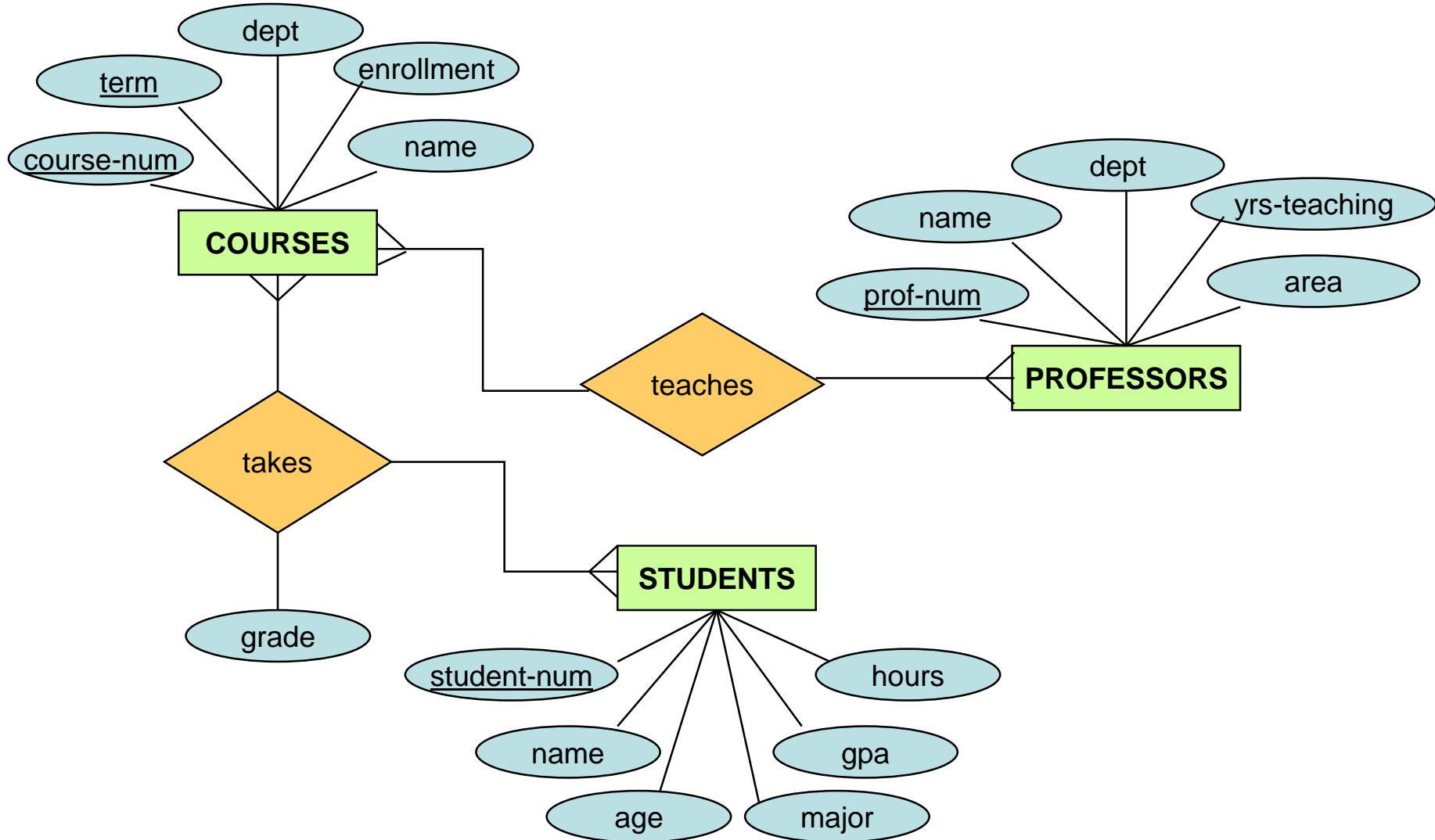
$\rho_{\text{tên mới của quan hệ } (A_1, A_2, \dots, A_N)}(\text{quan hệ})$

- ❖ *Ví dụ:*

$\rho_{x(one, two, \dots, last)}(r)$

đặt lại tên quan hệ  $r$  thành  $x$  và  $n$  thuộc tính của quan hệ  $x$  có tên là *one, two, ..., last*.

# VÍ DỤ: SƠ ĐỒ E-R



# VÍ DỤ: LƯỢC ĐỒ QUAN HỆ

- ❖ Dùng kỹ thuật để chuyển sơ đồ E-R trên thành lược đồ quan hệ như sau:

S = STUDENTS(s#, name, age, major, gpa, hours\_completed)

C = COURSES(c#, term, name, dept, enrollment)

P = PROFESSORS(p#, name, dept, yrs\_teaching, area)

TA = TAKES(s#, c#, term, grade)

TE = TEACH(p#, c#, term)

# VÍ DỤ: THÊM HIỆN QUAN HỆ S

<b>s#</b>	<b>name</b>	<b>age</b>	<b>major</b>	<b>gpa</b>	<b>hrs_completed</b>
S1	Michael Schumacher	19	Computer Science	4.00	45
S5	Jean Alesi	20	Physics	3.46	78
S3	Rubens Barrichello	21	Math	3.82	33
S2	Giancarlo Fisichella	18	Math	2.73	23
S4	Jarno Trulli	18	Computer Science	1.48	99
S7	Bernd Schneider	19	Computer Science	2.29	45
S6	Mika Hakkinen	20	English	2.37	33

# VÍ DỤ: THÈ HIỆN QUAN HỆ C

c#	term	name	dept	enrollment
C1	Fall 2002	CS1	CS	120
C1	Spring 2002	CS1	CS	100
C4	Fall 2002	Architecture	CS	97
C3	Fall 2002	Database	CS	86
C5	Spring 2001	Physics I	Physics	135
C5	Fall 2002	Physics I	Physics	125
C6	Summer 2002	Calculus III	Math	67

# VÍ DỤ: THỂ HIỆN QUAN HỆ TA

<u>s#</u>	<u>c#</u>	<u>term</u>	<u>grade</u>
S1	C3	Fall 03	A
S3	C4	Fall 02	B
S4	C6	Summer 03	C
S5	C5	Spring 01	D
S5	C1	Fall 02	A
S5	C3	Fall 03	C
S5	C6	Summer 02	C
S5	C4	Fall 03	A
S3	C5	Spring 01	C
S3	C1	Fall 03	A
S2	C4	Fall 03	D

# VÍ DỤ: THÊM HIỆN QUAN HỆ P

<u>p#</u>	name	dept	yrs_teaching
P1	Wilson	CS	5
P2	Davis	Math	32
P3	deMoser	CS	17
P4	Roberts	Physics	14

# VÍ DỤ: THÊM HIỆN QUAN HỆ TE

<u>p#</u>	<u>c#</u>	<u>term</u>
P1	C3	Fall 2002
P3	C4	Fall 2002
P4	C6	Summer 2002
P2	C5	Spring 2001
P2	C1	Spring 2002
P1	C4	Fall 2002
P3	C1	Fall 2002

# VÍ DỤ: TRUY VĂN 1

- ❖ **Tìm tên của tất cả các Sinh viên học ngành Công nghệ thông tin (Computer Science).**
- ❖ **Cách tiếp cận:**

- Đầu tiên, chọn ra tất cả những sinh viên học ngành CS:

$$r = \sigma_{(\text{major} = \text{"Computer Science"})}(S)$$

- Tiếp theo, chiểu thuộc tính tên trên kết quả vừa tìm được:

$$\text{result} = \pi_{(\text{name})}(r)$$

- ❖ **Biểu thức truy vấn hoàn chỉnh:**

$$\text{result} = \pi_{(\text{name})}(\sigma_{(\text{major} = \text{"Computer Science"})}(S))$$

# VÍ DỤ: TRUY VĂN 2

- ❖ **Tìm số hiệu sinh viên (s#) và tên của tất cả những sinh viên đã hoàn thành hơn 90 giờ học.**
- ❖ **Cách tiếp cận:**
  - Đầu tiên, chọn ra tất cả những sinh viên đã hoàn thành hơn 90 giờ học:
$$r = \sigma_{(\text{hours\_completed} > 90)}(S)$$
  - Tiếp theo, chiểu thuộc tính số hiệu sinh viên và tên trên kết quả vừa tìm được:
$$\text{result} = \pi_{(\text{s\#, name})}(r)$$
- ❖ **Biểu thức truy vấn hoàn chỉnh:**

$$\text{result} = \pi_{(\text{s\#, name})}(\sigma_{(\text{hours\_completed} > 90)}(S))$$

# VÍ DỤ: TRUY VĂN 3

- ❖ *Tìm tên của tất cả những sinh viên dưới 20 tuổi và đã hoàn thành hơn 80 giờ học.*
- ❖ Cách tiếp cận:

- Đầu tiên, chọn ra tất cả những sinh viên dưới 20 tuổi và đã hoàn thành hơn 80 giờ học:

$$r = \sigma_{((\text{hours\_completed} > 80) \wedge (\text{age} < 20))}(S)$$

- Tiếp theo, chiểu thuộc tính tên trên kết quả vừa tìm được:

$$\text{result} = \pi_{(\text{name})}(r)$$

- ❖ Biểu thức truy vấn hoàn chỉnh:

$$\text{result} = \pi_{(\text{name})}(\sigma_{((\text{hours\_completed} > 80) \wedge (\text{age} < 20))}(S))$$

# VÍ DỤ: TRUY VĂN 4

- ❖ *Tìm tên của tất cả các lớp thuộc Khoa Công nghệ thông tin (CS) hoặc Khoa Vật lý (Physics).*
- ❖ Cách tiếp cận:

- Đầu tiên, chọn ra tất cả các lớp hoặc thuộc Khoa CS hoặc Khoa Physics:

$$r = \sigma_{((dept = Computer\ Science) \vee (dept = Physics))}(C)$$

- Tiếp theo, chiểu thuộc tính tên trên kết quả vừa tìm được:

$$\text{result} = \pi_{(name)}(r)$$

- ❖ Biểu thức truy vấn hoàn chỉnh:

$$\text{result} = \pi_{(name)}(\sigma_{((dept = Computer\ Science) \vee (dept = Physics))}(C))$$

# VÍ DỤ: TRUY VÂN 5

- ❖ *Tìm tên của tất cả các giáo sư đã dạy một lớp trong kỳ mùa thu 2002 (Fall 2002).*
- ❖ Cách tiếp cận:
  - Đầu tiên, đặt toàn bộ thông tin về giáo sư vào cùng với thông tin về lớp học.
  - Tiếp theo, chỉ chọn các giáo sư và lớp học có liên quan đến nhau và thỏa mãn điều kiện (từ kết quả trên).
  - Cuối cùng, chiếu thuộc tính tên (giáo sư) trên kết quả vừa tìm được.
- ❖ Biểu thức truy vấn hoàn chỉnh:

$$\text{result} = \pi_{(\text{P.name})}(\sigma_{((\text{TE.term} = \text{"Fall 2002"}) \wedge (\text{P.p\#} = \text{TE.p\#}))}(\text{P} \times \text{TE}))$$

# VÍ DỤ: TRUY VĂN 6

- ❖ *Tìm tên của tất cả các sinh viên đã học một lớp trong kỳ mùa thu 2003 mà được dạy bởi một giáo sư có hơn 20 năm kinh nghiệm giảng dạy.*
- ❖ Cách tiếp cận:
  - Đầu tiên, đặt toàn bộ thông tin về giáo sư, thông tin về sinh viên, thông tin về việc dạy và thông tin về việc học vào cùng nhau.
  - Tiếp theo, chỉ chọn các sinh viên, giáo sư và lớp học có liên quan đến nhau và thỏa mãn điều kiện (từ kết quả trên).
  - Cuối cùng, chiết thuộc tính tên (sinh viên) trên kết quả vừa tìm được.
- ❖ Biểu thức truy vấn hoàn chỉnh:

$$\begin{aligned} \text{result} = \pi_{(\text{S.name})} & (\sigma_{((\text{TA.term} = \text{Fall 2003}) \wedge (\text{P.yrs\_teaching} > 20) \wedge (\text{S.s\#} = \text{TA.s\#}))} \\ & \wedge (\text{TA.c\#} = \text{TE.c\#}) \wedge (\text{TA.term} = \text{TE.term}) \wedge (\text{P.p\#} = \text{TE.p\#})) (\text{S} \times \text{TA} \times \text{TE} \times \text{P})) \end{aligned}$$

# VÍ DỤ: TRUY VĂN 7

- ❖ *Tìm tên của tất cả các giáo sư hoặc dạy ở Khoa Công nghệ thông tin (CS) hoặc có hơn 20 năm kinh nghiệm giảng dạy.*
- ❖ Biểu thức truy vấn hoàn chỉnh:

```
result = [π(name)(σ(dept = Computer Science)(P))] ∪ [π(name)(σ(yrs_teaching > 20)(P))]
```

hoặc:

```
result = π(name)(σ((dept = Computer Science) ∨ (yrs_teaching > 20))(P))
```

# VÍ DỤ: TRUY VĂN 8

- ❖ *Tìm số hiệu của tất cả các sinh viên mà chỉ đăng ký học vào kỳ mùa xuân 2003.*
- ❖ Biểu thức truy vấn hoàn chỉnh:

```
result = [π(TA.s#)(σ(TA.term = Spring 2003)(TA))] – [π(TA.s#)(σ(TA.term ≠ Spring 2003)(TA))]
```

Chú ý: Biểu thức truy vấn dưới đây sai, vì sao?

```
result = π(TA.s#)(σ(TA.term = Spring 2003))(TA))
```

=> bởi vì ở đây chỉ liệt kê những sinh viên có đăng ký học vào kỳ mùa xuân 2003 và cũng có thể các sinh viên này có đăng ký các kỳ học khác nữa.

# CÁC TOÁN TỬ MỞ RỘNG

- ❖ Năm phép toán cơ bản (được trình bày ở phần trước) là đủ để thể hiện câu truy vấn trong đại số quan hệ (có thể chứng minh được).
- ❖ Trong các câu truy vấn phức tạp cần các biểu thức truy vấn khó và dài dòng. Để đơn giản hóa, trong đại số quan hệ, người ta đề xuất một số phép toán mở rộng để cung cấp sức mạnh thể hiện biểu thức truy vấn.
- ❖ Phần sau sẽ trình bày một số phép toán quan trọng và chung nhất.

# PHÉP GIAO

Loại: Hai ngôi

Ký hiệu:  $\cap$

Khuôn dạng chung:  $r \cap s$  với  $r$  và  $s$  là 2 quan hệ khả hợp

Lược đồ quan hệ kết quả: lược đồ của quan hệ toán hạng

Kích thước của quan hệ kết quả (số bộ):  $\leq \min \{ |r|, |s| \}$

Định nghĩa:  $r \cap s \equiv r - (r - s)$

Ví dụ:

$$(\pi_{(p\#)}(\text{SPJ})) \cap (\pi_{(p\#)}(\text{P}))$$

- Phép giao tạo ra tập các bộ xuất hiện ở cả hai quan hệ toán hạng.

# VÍ DỤ PHÉP GIAO

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6

$r = R \cap S$

A	B	C	D
a	a	yes	1
c	f	yes	34

$r = R \cap T$

A	B	C	D

S

E	F	G	H
a	a	yes	1
b	r	yes	3
c	f	yes	34
m	n	no	56

T

E	F	G	H
a	r	no	31
b	f	yes	30

# PHÉP KẾT NỐI

- ❖ Trong các biểu thức truy vấn có liên quan tới tích Đè-các, cần phải cung cấp thêm các phép chọn để loại bỏ đi những tổ hợp các bộ không liên quan tới nhau trong kết quả.  
=> Phép kết nối (join operation) là sự kết hợp tích Đè-các và các phép chọn.
- ❖ **Các loại phép kết nối:** kết nối theta, kết nối bằng, kết nối tự nhiên, kết nối ngoài và bán kết nối (Semi Join Operator).

# PHÉP KẾT NỐI THETA VÀ KẾT NỐI BẰNG

Loại: Hai ngôi

Ký hiệu/khuôn dạng chung:  $r \triangleright\triangleleft_{(\text{predicate})} s$

Lược đồ quan hệ kết quả: ghép nối các quan hệ toán hạng

Định nghĩa:  $r \triangleright\triangleleft_{(\text{predicate})} s \equiv \sigma_{(\text{predicate})}(r \times s)$

Ví dụ:

$$r \triangleright\triangleleft_{((color='blue') \wedge (size=3))} s$$

$$r \triangleright\triangleleft_{((color='blue') \wedge (size>3))} s$$

Kết nối bằng

Kết nối theta

- Phép kết nối theta là dạng rút gọn của tích Đề-các và sau đó là thực hiện phép chọn.
- Phép kết nối bằng là một trường hợp đặc biệt của kết nối theta mà trong đó tất cả các điều kiện trong mệnh đề đều là điều kiện bằng.
- Cả phép kết nối theta và kết nối bằng đều không loại bỏ các bộ dư thừa, vì vậy việc loại bỏ các bộ dư thừa cần phải được thực hiện một cách tường minh thông qua mệnh đề điều kiện.

# VÍ DỤ PHÉP KẾT NỐI THETA

R

A	B	C	D
a	a	yes	1
b	d	no	7
c	f	yes	34
a	d	no	6

S

E	F	G	H
a	a	yes	1
b	r	yes	3
c	f	yes	34
m	n	no	56

$$r = R \triangleright\triangleleft_{(R.B < S.F)} S$$

A	B	C	D	E	F	G	H
a	a	yes	1	b	r	yes	3
a	a	yes	1	c	f	yes	34
a	a	yes	1	m	n	no	56
b	d	no	7	b	r	yes	3
b	d	no	7	c	f	yes	34
b	d	no	7	m	n	no	56
c	f	yes	34	b	r	yes	3
c	f	yes	34	m	n	no	56
a	d	no	6	b	r	yes	3
a	d	no	6	c	f	yes	34
a	d	no	6	m	n	no	56

# PHÉP KẾT NỐI TỰ NHIÊN

Loại: Hai ngôi

Ký hiệu/khuôn dạng chung:  $r * s$

Lược đồ quan hệ kết quả: ghép nối các quan hệ toán hạng, với các thuộc tính được đặt tên chung và chỉ xuất hiện một lần.

Định nghĩa:  $r * s \equiv r \triangleright\triangleleft_{(r.commonattributes = s.commonattributes)} s$

Ví dụ:  $s * spj * p$

- Phép kết nối tự nhiên thực hiện kết nối bằng trên tất cả các thuộc tính có cùng tên của 2 quan hệ toán hạng.
- Bậc của quan hệ kết quả là tổng số bậc của 2 quan hệ toán hạng trừ đi số các thuộc tính chung của chúng.
- Phép kết nối tự nhiên là phổ biến nhất trong tất cả các phép kết nối. Nó rất có ích trong việc loại bỏ đi các bộ dư thừa. Các thuộc tính chung của 2 quan hệ toán hạng thường được gọi là **các thuộc tính kết nối**.

# VÍ DỤ PHÉP KẾT NỐI TỰ NHIÊN

R

A	B	C	D
a	a	yes	1
b	r	no	7
c	f	yes	34
a	m	no	6

S

B	M	G	H
a	a	yes	1
b	r	yes	3
a	f	yes	34
m	n	no	56

$r = R * S$

A	B	C	D	M	G	H
a	a	yes	1	a	yes	1
a	a	yes	1	f	yes	34
a	m	no	6	n	no	56

$r = R * T$

A	B	C	D	G	H
b	r	no	7	yes	30

T

A	B	G	H
a	f	no	31
b	r	yes	30

# PHÉP KẾT NỐI NGOÀI

Loại: Hai ngôi

Ký hiệu/khuôn dạng chung: Kết nối ngoài trái:  $r \supset\triangleleft s$

Kết nối ngoài phải:  $r \triangleright\subset s$       Kết nối ngoài đầy đủ:  $r \supset\triangleleft\triangleright\subset s$

Lược đồ quan hệ kết quả: ghép nối các quan hệ toán hạng

Định nghĩa:

$r \supset\triangleleft s$   $\equiv$  kết nối tự nhiên r và s với các bộ của r không tương ứng trong s vẫn được giữ lại trong kết quả. Tất cả các giá trị còn khuyết ở thuộc tính của s đều được gán giá trị rỗng (null).

$r \triangleright\subset s$   $\equiv$  kết nối tự nhiên r và s với các bộ của s không tương ứng trong r vẫn được giữ lại trong kết quả. Tất cả các giá trị còn khuyết ở thuộc tính của r đều được gán giá trị rỗng.

$r \supset\triangleleft\triangleright\subset s$   $\equiv$  kết nối tự nhiên r và s với các bộ của r và s không tương ứng vẫn được giữ lại trong kết quả. Tất cả các giá trị còn khuyết sẽ được gán giá trị rỗng.

Ví dụ: Cho  $r(A,B) = \{(a, b), (c, d), (b,c)\}$  và  $s(A,C) = \{(a, d), (s, t), (b, d)\}$

$$r \supset\triangleleft s = (A,B,C) = \{(a,b,d), (b,c,d), (c,d,null)\}$$

$$r \triangleright\subset s = (A,B,C) = \{(a,b,d), (b,c,d), (s,null,t)\}$$

$$r \supset\triangleleft\triangleright\subset s = (A,B,C) = \{(a,b,d), (b,c,d), (s,null,t), (c,d,null)\}$$

# VÍ DỤ PHÉP KẾT NỐI NGOÀI

R

A	B	C
1	2	3
4	5	6
7	8	9

$$r = R \triangleright\triangleleft S$$

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	null
7	8	9	null
null	6	7	12

S

B	C	D
2	3	10
2	3	11
6	7	12

$$r = R \triangleright\triangleleft S$$

A	B	C	D
1	2	3	10
1	2	3	11
4	5	6	null
7	8	9	null

$$r = R \triangleright\subset S$$

B	C	D	A
2	3	10	1
2	3	11	1
6	7	12	null

# PHÉP BÁN KẾT NỐI

Loại: Hai ngôi

Ký hiệu/khuôn dạng chung:  $r \triangleright_{(predicate)} s$

Lược đồ quan hệ kết quả: Lược đồ của  $r$

Định nghĩa:  $r \triangleright_{(predicate)} s \equiv \pi_{(\text{thuộc tính của } r)}(r \triangleright \triangleleft_{(predicate)} s)$

Ví dụ:

$$r \triangleright_{(r.B > s.M)} s$$

- Phép bán kết nối thực hiện một phép kết nối 2 quan hệ toán hạng, sau đó chiếu trên các thuộc tính của quan hệ toán hạng bên trái.
- Ưu điểm chính của phép bán kết nối là làm giảm số các bộ cần xử lý khi thực hiện phép kết nối => rất có ích trong môi trường phân tán.
- Theo một khuôn dạng chung như đã trình bày, phép bán kết nối chính là phép bán kết nối theta. Phép bán kết nối bằng và phép bán kết nối tự nhiên được định nghĩa theo cách tương tự.

# VÍ DỤ PHÉP BÁN KẾT NỐI

R

A	B	C	D
a	a	yes	1
b	r	no	7
c	f	yes	34
a	m	no	6

$$r = R \triangleright_{(R.B > S.M)} S$$

S

B	M	C
a	e	yes
b	r	yes
a	f	no
r	n	no

T

B	G	D
a	4	d
b	7	e
a	4	f
m	2	g

# PHÉP CHIA

Loại: Hai ngôi

Ký hiệu/khuôn dạng chung:  $r \div s$  với  $r(\{X\})$  và  $s(\{Y\})$

Lược đồ quan hệ kết quả:  $Z$  với  $Z = X - Y$

Định nghĩa:  $r \div s \equiv \pi_{(X-Y)}(r) - (\pi_{(X-Y)}((\pi_{(X-Y)}(r) \times s) - r))$

Ví dụ:

Cho  $r(A, B, C) = \{(a, b, c), (a, d, e), (a, b, d), (a, c, c), (a, d, d)\}$

và  $s(C) = \{(c), (d)\}$

thì:  $r \div s = t(A, B) = \{(a, b)\}$

Các yêu cầu đối với phép chia:

- Quan hệ  $r$  được định nghĩa trên tập thuộc tính  $A$  và quan hệ  $s$  được định nghĩa trên tập thuộc tính  $B$  sao cho  $B \subseteq A$ .
- Cho  $C$  là tập các thuộc tính của  $A - B$ .  
=> Phép chia được định nghĩa như sau: một bộ  $t$  là thuộc  $r \div s$  nếu với mọi bộ  $t_s$  của  $s$  có một bộ  $t_r$  của  $r$  thỏa mãn cả 2 điều kiện:  
 $t_r[C] = t_s[C]$  và  $t_r[A-B] = t[A-B]$

# VÍ DỤ PHÉP CHIA

R

A	B	C	D
a	f	yes	1
b	r	no	1
a	f	yes	34
e	g	yes	34
a	m	no	6
b	r	no	34

$$r = R \div S$$

A	B	C
a	f	yes
b	r	no

$$r = R \div T$$

A	B
a	f

$$r = R \div V$$

A
a

$$r = R \div W$$

A
---

S

D
1
34

T

C	D
yes	1
yes	34

U

C	D
no	1
no	34

V

B	C	D
f	yes	1
f	yes	34
m	no	6

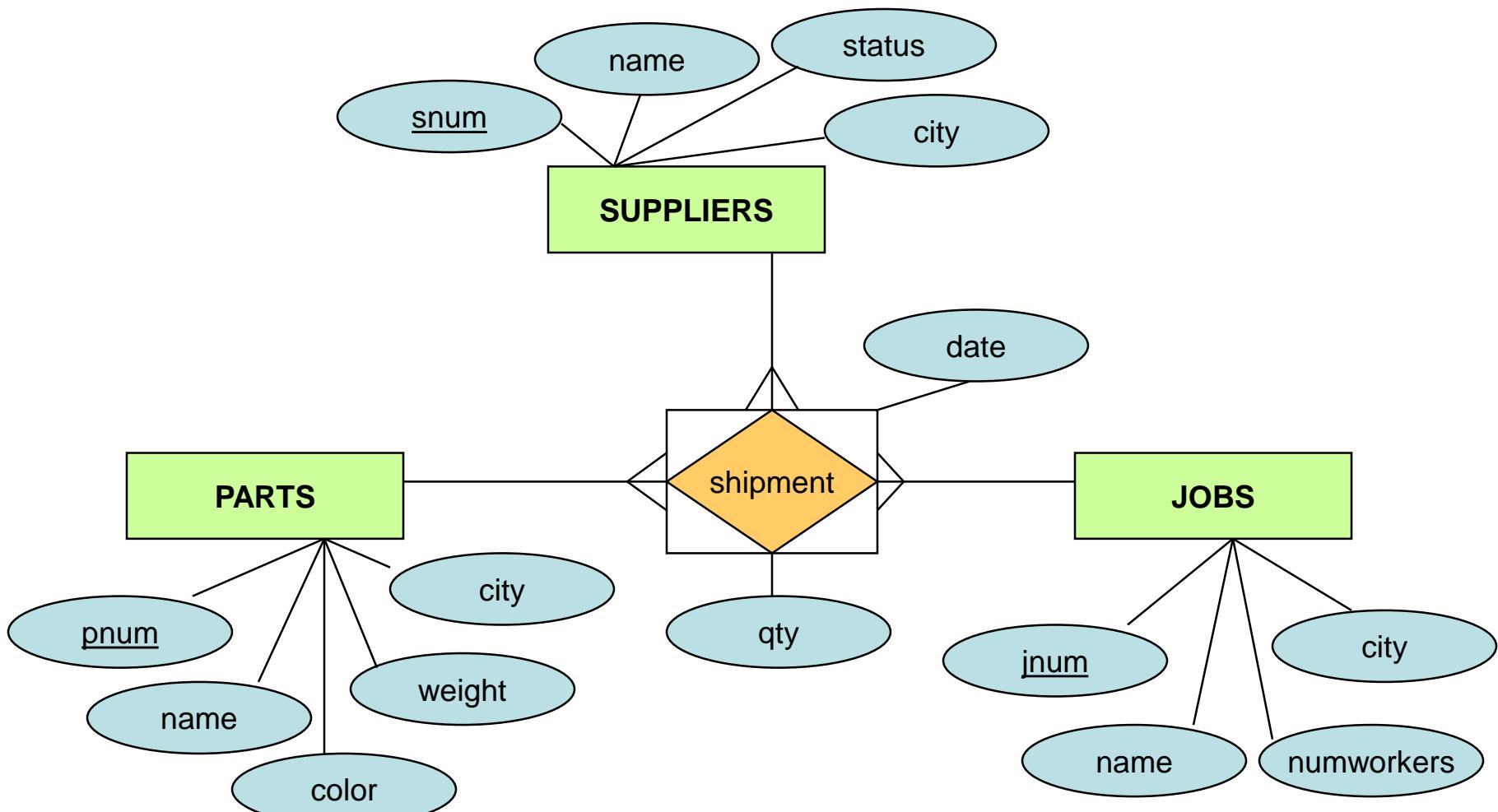
W

B	C	D
f	yes	1
g	yes	69

# TÍNH HỮU ÍCH CỦA CÁC TOÁN TỬ MỞ RỘNG

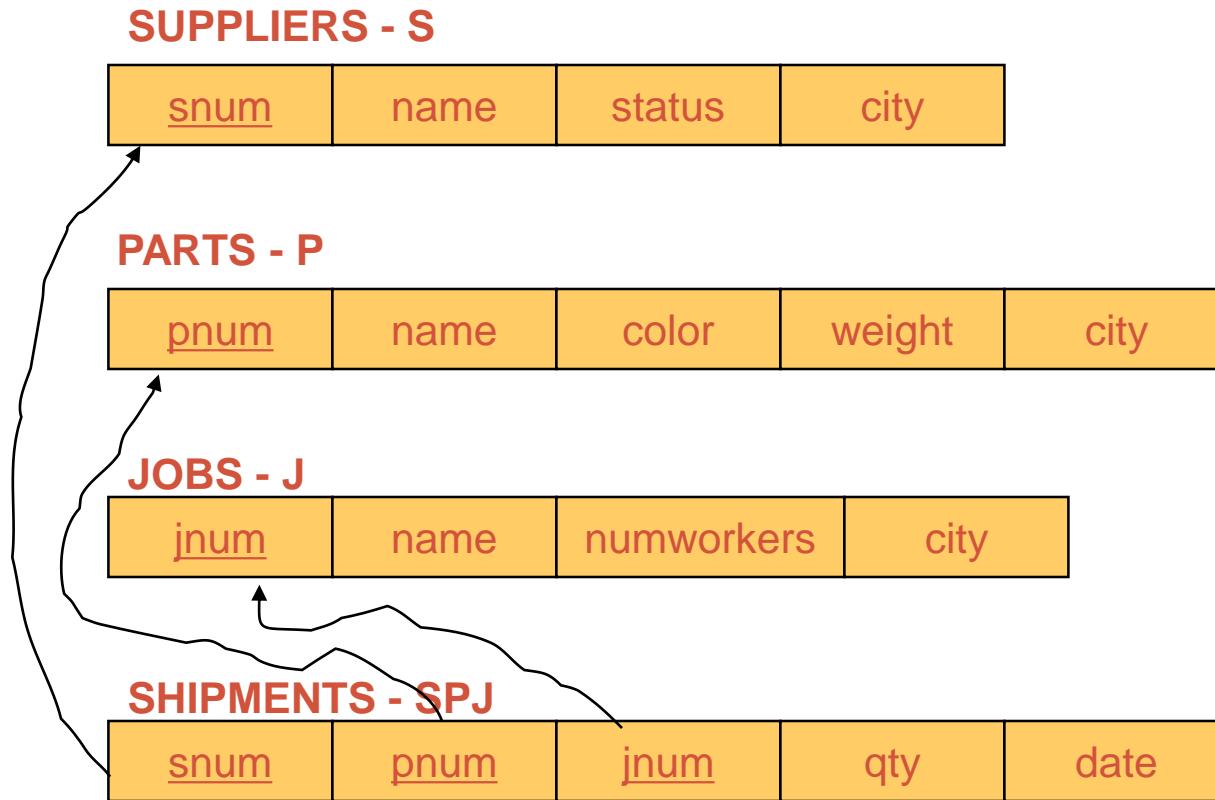
- ❖ Các toán tử đại số quan hệ mở rộng được định nghĩa dựa trên 5 phép toán cơ bản.
- ❖ Tính hữu ích của các phép toán này được thể hiện tốt nhất qua phép chia.
- ❖ Xem xét truy vấn sau dựa trên CSDL Nhà cung cấp-linh kiện-công việc-vận chuyển (suppliers-parts-jobs-shipment):

# LƯỢC ĐỒ CSDL VÍ DỤ



# LƯỢC ĐỒ QUAN HỆ

Câu truy vấn: Tìm số hiệu các nhà cung cấp vận chuyển mọi linh kiện?



# TÍNH HỮU ÍCH CỦA CÁC TOÁN TỬ MỞ RỘNG (Cont.)

## ❖ Giải pháp 1: Chỉ sử dụng 5 phép toán cơ bản:

$T1 = \pi_{(s\#, p\#)}(spj)$  //tất cả các cặp (s#,p#) cho các vận chuyển thực.

$T2 = \pi_{(p\#)}(p)$  //tất cả các linh kiện (p#)

$T3 = \pi_{(s\#)}(T1) \times T2$  //tất cả s# trong T1 được ghép với mỗi bộ trong T2  
{spj.s#, p.p#}

$T4 = T3 - T1$  //tất cả các bộ thuộc T3 và không thuộc T1 – các nhà  
cung cấp vận chuyển (một hoặc một số linh kiện).

$T5 = T1 - T4$  //tất cả các bộ thuộc T1 mà không thuộc T4 – các  
nhà cung cấp vận chuyển tất cả các linh kiện.

$T6 = \pi_{(s\#)}(T5)$  //lời giải

## Kết quả:

$$\pi_{(s\#)}[\pi_{(s\#,p\#)}(spj) - (\pi_{(s\#)}(spj) \times \pi_{(p\#)}(p) - \pi_{(s\#,p\#)}(spj))]$$

# TÍNH HỮU ÍCH CỦA CÁC TOÁN TỬ MỞ RỘNG (Cont.)

## ❖ Giải pháp 2: Sử dụng các phép toán mở rộng:

Kết quả:

$$(\pi_{(s\#, p\#)}(spj)) \div (\pi_{(p\#)}(p))$$

# VIẾT MỘT SỐ TRUY VĂN CHỈ SỬ DỤNG 5 TOÁN TỬ CƠ BẢN

1. *Tìm số hiệu của tất cả các nhà cung cấp hoặc ở tại Milan hoặc chuyển hàng tới bất kỳ công việc nào với số lượng lớn hơn 40.*

$$\pi_{(s\#)}(\sigma_{(\text{city} = \text{Milan})}(S)) \cup [\pi_{(s\#)}(\sigma_{(\text{qty} > 40)}(SPJ))]$$

2. *Tìm tên của tất cả các nhà cung cấp mà chỉ vận chuyển các linh kiện màu đỏ.*

$$\begin{aligned} & \pi_{(S.\text{name})}[\pi_{(S\#, S.\text{name})}(\sigma_{((SPJ.s\#=S.s\#) \wedge (SPJ.p\#=P.p\#) \wedge (\text{color}=red))}(SPJ \times S \times P)) \\ & - \pi_{(S\#, S.\text{name})}(\sigma_{((SPJ.s\#=S.s\#) \wedge (SPJ.p\#=P.p\#) \wedge (\text{color} \neq red))}(SPJ \times S \times P))] \end{aligned}$$

# VIẾT MỘT SỐ TRUY VĂN CHỈ SỬ DỤNG 5 TOÁN TỬ CƠ BẢN (Cont.)

3. *Tìm tên của các nhà cung cấp mà ở cùng thành phố với công việc mà họ vận chuyển linh kiện đến cho.*

$$T1 = (S \times SPJ \times J)$$

$$T2 = \sigma_{(S.s\# = SPJ.s\#)}(T1) \quad // \text{chọn các bộ có cùng mã nhà cung cấp s\#}$$

$$T3 = \sigma_{(J.j\# = SPJ.j\#)}(T2) \quad // \text{chọn các bộ có cùng mã công việc j\#}$$

$$T4 = \sigma_{(J.city = S.city)}(T3) \quad // \text{chọn các bộ có cùng thành phố}$$

$$T5 = \pi_{(S.name)}(T4) \quad // \text{chiếu để lấy tập thuộc tính cuối cùng}$$

*Kết quả:*

$$\pi_{(S.name)}(\sigma_{(S.s\# = SPJ.s\#) \wedge (J.j\# = SPJ.j\#) \wedge (J.city = S.city)}(S \times SPJ \times J))$$

# VIẾT MỘT SỐ TRUY VĂN CHỈ SỬ DỤNG 5 TOÁN TỬ CƠ BẢN (Cont.)

4. *Tìm số hiệu của tất cả các linh kiện mà được vận chuyển bởi cả hai nhà cung cấp “S1” và “S2”.*

Lưu ý: Biểu thức sau đây không đúng, vì sao?

$$\pi_{(p\#)}(\sigma_{((s\# = "S1") \wedge (s\# = "S2"))}(SPJ))$$

→ Vì điều kiện lựa chọn luôn cho kết quả là sai.

*Kết quả đúng:*

$$[\pi_{(p\#)}(\sigma_{(s\#=S1)}(SPJ)] - ([\pi_{(p\#)}(\sigma_{(s\#=S1)}(SPJ)] - [\pi_{(p\#)}(\sigma_{(s\#=S2)}(SPJ)])$$

# VIẾT MỘT SỐ TRUY VĂN CHỈ SỬ DỤNG 5 TOÁN TỬ CƠ BẢN (Cont.)

5. *Tìm số hiệu của tất cả các nhà cung cấp mà vận chuyển cả các linh kiện màu đỏ (“red”) và màu xanh (“blue”).*

Lưu ý: Biểu thức sau đây không đúng, vì sao?

$$\pi_{(s\#)}(\sigma_{((color = "blue") \wedge (SPJ.p\# = P.p\#) \wedge (color = "red"))}(P \times SPJ))$$

→ Vì điều kiện lựa chọn luôn cho kết quả là sai.

*Truy vấn đúng:*

$$T1 = \pi_{(s\#)}(\sigma_{((color = "blue") \wedge (SPJ.p\# = P.p\#))}(P \times SPJ))$$

$$T2 = \pi_{(s\#)}(\sigma_{((color = "red") \wedge (SPJ.p\# = P.p\#))}(P \times SPJ))$$

$$T3 = T2 - T1$$

$$T4 = T2 - T3$$

# VIẾT MỘT SỐ TRUY VĂN CHỈ SỬ DỤNG 5 TOÁN TỬ CƠ BẢN (Cont.)

6. *Tìm tất cả các cặp (s#, j#) của các nhà cung cấp và công việc ở cùng thành phố, nhưng các nhà cung cấp đó không vận chuyển bất kỳ linh kiện nào cho công việc này.*

Biểu thức truy vấn:

$T1 = \pi_{(s\#, j\#)}(\sigma_{(S.\text{city} = J.\text{city})}(S \times J))$  // tất cả các cặp (s#,j#) ở cùng thành phố.

$T2 = \pi_{(s\#, j\#)}(\sigma_{((S.\text{city}=J.\text{city}) \wedge (\text{SPJ}.j\#=J.j\#) \wedge (\text{SPJ}.s\#=S.s\#))}(S \times \text{SPJ} \times J))$   
//T2 chứa tất cả các cặp (s#,j#) trong đó nhà cung cấp vận chuyển các linh kiện tới công việc trong cùng thành phố.

$T3 = T1 - T2$

# THỰC HÀNH TRUY VÂN VỚI TẤT CẢ CÁC TOÁN TỬ ĐẠI SỐ QUAN HỆ

1. *Liệt kê tất cả các cặp số hiệu nhà cung cấp ở cùng một thành phố.*

$$\pi_{(s.s\#,x.s\#)}(s \triangleright\triangleleft_{(s.city = x.city)}(\rho_x(s)))$$

2. *Liệt kê tất cả các vận chuyển có liên quan đến linh kiện màu xanh (“green”).*

$$spj \triangleright\triangleleft_{(spj.pnum = p.pnum)}(\sigma_{(color=green)}(p))$$

# THỰC HÀNH TRUY VÂN VỚI TẤT CẢ CÁC TOÁN TỬ ĐẠI SỐ QUAN HỆ (Cont.)

3. *Liệt kê tất cả số hiệu của các nhà cung cấp mà vận chuyển linh kiện được sản xuất ở cùng thành phố với nhà cung cấp đó.*

$$\pi_{(s.s\#)}(\sigma_{(s.city=p.city)}(s * p * spj))$$

4. *Liệt kê tên của tất cả các nhà cung cấp mà vận chuyển tất cả các linh kiện màu xanh (“blue”).*

$$\pi_{(s.name)}(s * (\pi_{(s\#, p\#)}(spj) \div \pi_{(p\#)}(\sigma_{(color=blue)}(p))))$$

# THỰC HÀNH TRUY VÂN VỚI TẤT CẢ CÁC TOÁN TỬ ĐẠI SỐ QUAN HỆ (Cont.)

5. *Liệt kê số hiệu của các nhà cung cấp mà chỉ vận chuyển các linh kiện màu xanh.*

$$(\pi_{(s\#)}(spj^*(\sigma_{(color=blue)}(p)))) - (\pi_{(s\#)}(spj^*(\sigma_{(color \neq blue)}(p))))$$

# Giới thiệu về ngôn ngữ SQL (phần 1)

Posts and Telecommunications Institute of Technology-PTIT



# Lịch sử của SQL

- SQL là viết tắt của từ tiếng Anh: Structural Query Language. SQL là một ngôn ngữ tuân thủ chuẩn cho việc tạo ra và truy vấn các CSDL quan hệ.
- SQL được chấp nhận bởi Viện tiêu chuẩn quốc gia Hoa Kỳ (ANSI) và tổ chức tiêu chuẩn quốc tế (ISO) cũng như thoả mãn các tiêu chuẩn xử lý thông tin của liên bang (FIPS).
- Giữa những năm 1974 và 1979, các nhân viên làm việc trong phòng nghiên cứu thí nghiệm của công ty IBM tại San Jose, bang California, Hoa Kỳ đã tiến hành phát triển hệ thống có tên là R, ngay sau khi bài báo truyền thống định nghĩa CSDL quan hệ được công bố. Mục tiêu của hệ thống R là chứng minh tính khả thi của việc cài đặt mô hình quan hệ trong một hệ quản trị CSDL. Họ sử dụng một ngôn ngữ có tên là SEQUEL (Structured English Query Language), là một ngôn ngữ nối tiếp của SQUARE (Specifying Queries as Relational Expressions). Cả hai đều được phát triển tại IBM, San Jose.
- Sau đó, SEQUEL được đổi tên thành SQL trong quá trình thực hiện dự án này.



# Lịch sử của SQL (cont.)

- Hệ thống R chưa từng được thương mại hóa nhưng nó trực tiếp dẫn đến sự phát triển của SQL/DS (bản SQL chạy trên hệ điều hành DOS năm 1981, và trên một phiên bản máy ảo VM năm 1982). Đây là hệ quản trị CSDL quan hệ được thương mại hóa đầu tiên của IBM.
- Tuy nhiên, IBM không phải là công ty đưa ra phiên bản cài đặt thương mại đầu tiên cho hệ quản trị CSDL quan hệ mà vinh dự đó thuộc về Oracle với phần mềm quan hệ năm 1979.
- Hiện nay, các hệ thống quản trị CSDL quan hệ đều dựa trên SQL.
- Mỗi nhà cung cấp dịch vụ đề có toàn bộ các đặc tính chuẩn của SQL, kèm theo đó là các tính năng phụ của riêng từng hãng. Các phần mở rộng trong SQL này dẫn đến hiện tượng trùng lặp khi ứng dụng SQL được cài đặt trên các hệ CSDL khác nhau. Và những phần mở rộng này đặc trưng riêng cho từng nhà cung cấp.

# Lịch sử của SQL (cont.)

- SQL-99 (hay còn gọi là SQL3) là phiên bản hiện thời của chuẩn ANSO dành cho SQL. Chuẩn này cũng được chấp thuận bởi ISO.
- Mặc dù có rất nhiều phiên bản của SQL, bản chất bên trong của nó mới là điều cần quan tâm. Dù với Oracle, Microsoft SQL Server, IBM's DB2, Microsoft Access, MySQL, hay bất kỳ hệ quản trị CSDL quan hệ tiên tiến nào khác, những thông tin trong bài này sẽ giúp bạn nhanh chóng tìm ra điểm chính của các hệ thống này.



# SQL

- SQL là một ngôn ngữ CSDL quan hệ đầy đủ. Nó bao gồm cả ngôn ngữ định nghĩa dữ liệu (DDL) và ngôn ngữ thao tác dữ liệu (DML).
- Cả hai ngôn ngữ dữ liệu của SQL đều được đề cập đến trong bài này.
- Nếu dùng Microsoft Access thì bạn không cần biết nhiều về DDL của SQL so với nếu bạn dùng Oracle 9i hay MySQL.

# Ký pháp cho câu lệnh SQL

Ký pháp	Mô tả
VIẾT HOA	từ khoá cần thiết cho câu lệnh SQL
<i>Viết nghiêng</i>	Một tham số do người dùng cung cấp- thường là cần thiết
{a   b   ... }	Một tham số bắt buộc, sử dụng một trong số danh sách lựa chọn
[...]	Một tham số tùy chọn - mọi thứ trong ngoặc vuông đều là tùy chọn
<i>tablename</i>	Tên của bảng
<i>column</i>	Tên của một thuộc tính trong bảng
<i>data type</i>	Một định nghĩa kiểu dữ liệu hợp lệ
<i>constraint</i>	Một định nghĩa ràng buộc hợp lệ
<i>condition</i>	Một biểu thức điều kiện hợp lệ - trả về giá trị đúng hoặc sai
<i>columnlist</i>	Một hoặc nhiều tên cột hoặc biểu thức được phân cách nhau bởi dấu phẩy
<i>tablelist</i>	Một hoặc nhiều tên bảng được phân cách nhau bởi dấu phẩy
<i>conditionlist</i>	Một hoặc nhiều biểu thức điều kiện được phân cách nhau bởi dấu phẩy
<i>expression</i>	Một giá trị đơn (ví dụ 76 or 'married') hoặc một công thức (ví dụ, price-10)

# Ngôn ngữ định nghĩa dữ liệu trong SQL

- Trước khi sử dụng một hệ CSDL quan hệ, bạn phải thực hiện 2 việc: (1) tạo một cấu trúc CSDL, (2) tạo các bảng lưu dữ liệu người sử dụng.
- Công việc thứ nhất liên quan đến việc tạo ra các tệp vật lý để lưu trữ dữ liệu. Hệ CSDL quan hệ tự động tạo ra các bảng định nghĩa dữ liệu và tạo ra hệ quản trị CSDL ngầm định (default database administrator - DBA).
  - Việc tạo ra các tệp vật lý đòi hỏi sự tương tác giữa hệ điều hành và hệ quản trị cơ sở dữ liệu. Vì vậy, tạo ra cấu trúc cơ sở dữ liệu là một đặc tính có sự khác nhau từ một hệ quản trị cơ sở dữ liệu này sang hệ khác.
- Với một ngoại lệ là có thể tạo ra cơ sở dữ liệu, hầu hết các nhà cung cấp hệ thống quản trị cơ sở dữ liệu sử dụng bản SQL có khác một chút với bản SQL chuẩn của ANSI. Mặc dù vậy nhưng cũng chỉ thỉnh thoảng bạn mới gặp những sự khác nhau nhỏ trong cú pháp của các câu lệnh SQL. Ví dụ, hầu hết để yêu cầu mọi câu lệnh SQL được kết thúc bởi dấu chấm phẩy (;) tuy nhiên một số bản cài đặt SQL không sử dụng dấu (;). Hầu hết những sự khác nhau chung về cú pháp sẽ được chỉ ra trong bài giảng này hoặc ít nhất cũng liệt kê những sự khác nhau do người viết nhận thức được.



# Tóm tắt các câu lệnh định nghĩa dữ liệu SQL

Câu lệnh hoặc lựa chọn	Mô tả
CREATE SCHEMA AUTHORIZATION	Tạo một lược đồ CSDL
CREATE TABLE	Tạo một bảng mới trong CSDL người dùng
NOT NULL	Ràng buộc đảm bảo một cột sẽ không có giá trị rỗng
UNIQUE	Ràng buộc đảm bảo một cột sẽ không có giá trị trùng lặp
PRIMARY KEY	Định nghĩa một khóa chính cho một bảng
FOREIGN KEY	Định nghĩa một khóa ngoại cho một bảng
DEFAULT	Định nghĩa một giá trị mặc định cho một cột (khi không nhập giá trị mới nào vào)
CHECK	Ràng buộc dùng để kiểm tra tính đúng đắn của dữ liệu trong cột
CREATE INDEX	Tạo một chỉ mục cho một bảng
CREATE VIEW	Tạo một tập con động cho hàng/cột từ một hoặc nhiều bảng
ALTER TABLE	Thay đổi định nghĩa của một bảng: thêm/xóa/cập nhật các thuộc tính hoặc ràng buộc
DROP TABLE	Xóa vĩnh viễn một bảng (cùng dữ liệu của nó) khỏi lược đồ CSDL
DROP INDEX	Xóa vĩnh viễn một chỉ mục
DROP VIEW	Xóa vĩnh viễn một khung nhìn

# Các câu lệnh định nghĩa dữ liệu

Câu lệnh	Chức năng
CREATE TABLE	Tạo bảng
DROP TABLE	Xóa bảng
ALTER TABLE	Sửa đổi bảng
CREATE VIEW	Tạo khung nhìn
DROP VIEW	Xóa khung nhìn
ALTER VIEW	Sửa đổi khung nhìn
CREATE INDEX	Tạo chỉ mục
DROP INDEX	Xóa chỉ mục
CREATE SCHEMA	Tạo lược đồ cơ sở dữ liệu

# Các câu lệnh định nghĩa dữ liệu

DROP SCHEMA	Xóa lược đồ cơ sở dữ liệu
CREATE PROCEDURE	Tạo thủ tục lưu trữ
DROP PROCEDURE	Xóa thủ tục lưu trữ
ALTER PROCEDURE	Sửa thủ tục lưu trữ
CREATE FUNCTION	Tạo hàm (do người sử dụng định nghĩa)
DROP FUNCTION	Xóa hàm
ALTER FUNCTION	Sửa đổi hàm
CREATE TRIGGER	Tạo Trigger
DROP TRIGGER	Xóa Trigger
ALTER TRIGGER	Sửa đổi Trigger



# Các lệnh thao tác dữ liệu

<b>Câu lệnh</b>	<b>Chức năng</b>
SELECT	Truy xuất dữ liệu
INSERT	Bổ sung dữ liệu
UPDATE	Cập nhật dữ liệu
DELETE	Xóa dữ liệu
TRUNCATE	Xóa toàn bộ dữ liệu trong bảng

# Các kiểu dữ liệu SQL

1. INT or INTEGER.
2. REAL or FLOAT.
3. CHAR( $n$ ) = chuỗi ký tự có độ dài cố định.
4. VARCHAR( $n$ ) = chuỗi ký tự có độ dài thay đổi, và có tối đa  $n$  ký tự.
5. NUMERIC(*precision, decimal*) = kiểu số với độ dài *precision* số, và độ chính xác “*decimal*” đơn vị sau dấu phẩy.  
NUMERIC(10,2) có thể chứa số lớn đến ±99,999,999.99
6. DATE = ngày tháng. SQL có định dạng 'yyyy-mm-dd'
7. TIME = thời gian. SQL có định dạng 'hh:mm:ss[.ss...]'.
8. DATETIME or TIMESTAMP. SQL có định dạng TIMESTAMP 'yyyy-mm-dd hh:mm:ss[.ss...]'



# Các kiểu dữ liệu trong SQL SERVER

Tên kiểu	Mô tả
CHAR (n)	Kiểu chuỗi với độ dài cố định
NCHAR (n)	Kiểu chuỗi với độ dài cố định hỗ trợ UNICODE
VARCHAR (n)	Kiểu chuỗi với độ dài chính xác
NVARCHAR (n)	Kiểu chuỗi với độ dài chính xác hỗ trợ UNICODE
INTEGER	Số nguyên có giá trị từ $-2^{31}$ đến $2^{31} - 1$
INT	Như kiểu Integer
TINYINT	Số nguyên có giá trị từ 0 đến 255.
SMALLINT	Số nguyên có giá trị từ $-2^{15}$ đến $2^{15} - 1$
BIGINT	Số nguyên có giá trị từ $-2^{63}$ đến $2^{63} - 1$
NUMERIC (p,s)	Kiểu số với độ chính xác cố định.
DECIMAL (p,s)	Tương tự kiểu Numeric
FLOAT	Số thực có giá trị từ $-1.79E+308$ đến $1.79E+308$
REAL	Số thực có giá trị từ $-3.40E + 38$ đến $3.40E + 38$
MONEY	Kiểu tiền tệ
BIT	Kiểu bit (có giá trị 0 hoặc 1)
DATETIME	Kiểu ngày giờ (chính xác đến phần trăm của giây)
SMALLDATETIME	Kiểu ngày giờ (chính xác đến phút)
TIMESTAMP	

BINARY	Dữ liệu nhị phân với độ dài cố định (tối đa 8000 bytes)
VARBINARY	Dữ liệu nhị phân với độ dài chính xác (tối đa 8000 bytes)
IMAGE	Dữ liệu nhị phân với độ dài chính xác (tối đa 2,147,483,647 bytes)

# Khai báo khóa

- Có thể dùng PRIMARY KEY hoặc UNIQUE
- SQL chỉ cho phép đánh chỉ số (index) bằng PRIMARY KEY.
- SQL không cho phép thuộc tính PRIMARY KEY chứa giá trị rỗng. Tuy nhiên, nó lại cho phép các thuộc tính UNIQUE mang giá trị rỗng (có thể có nhiều hơn một bản ghi mang giá trị rỗng, tuy nhiên các giá trị khác rỗng của các bản ghi khác nhau phải khác nhau).

VD:

```
CREATE TABLE Bán (
    quán CHAR(20),
    bia VARCHAR(20),
    giá REAL,
    PRIMARY KEY(quán, bia) );
```

```
CREATE TABLE Bán (
    quán CHAR(20),
    bia VARCHAR(20),
    giá REAL,
    UNIQUE(quán, bia) );
```



# Lưu ý khi khai báo khóa bằng UNIQUE

- Các câu lệnh sau là khác nhau.

```
CREATE TABLE Bán (
    quán CHAR(20) UNIQUE,
    bia VARCHAR(20) UNIQUE,
    giá REAL,
);
```

```
CREATE TABLE Bán (
    quán CHAR(20),
    bia VARCHAR(20),
    giá REAL,
    UNIQUE(quán, bia) );
```



# Khóa ngoại

```
CREATE TABLE Bia (
    tên      CHAR(20) PRIMARY KEY,
    Nhà_SX  CHAR(20)
);
```

```
CREATE TABLE Bán (
    quán CHAR(20),
    bia CHAR(20) REFERENCES Bia(tên),
    giá REAL
);
```

Hoặc:

```
CREATE TABLE Bán (
    quán CHAR(20),
    bia CHAR(20),
    giá REAL,
    FOREIGN KEY bia REFERENCES Bia(tên)
);
```



# Thiết lập các quy tắc

- Thêm ON [DELETE, UPDATE] [CASCADE, SET NULL] trong quá trình khai báo khóa ngoại.

Ví dụ:

```
CREATE TABLE Bán (
    quán CHAR(20),
    bia CHAR(20),
    giá REAL,
    FOREIGN KEY bia REFERENCES Bia(tên)
        ON DELETE SET NULL
        ON UPDATE CASCADE
);
```



# Kiểm tra trên từng thuộc tính

- CHECK (*condition*) : kiểm tra giá trị của từng bản ghi tại thuộc tính nào đó theo điều kiện có trong *condition*.
- Các điều kiện trong *condition* chỉ được kiểm tra khi giá trị của các thuộc tính liên quan đến nó bị thay đổi (chèn, cập nhật).

Ví dụ:

```
CREATE TABLE Bán (
    quán CHAR(20) ,
    bia CHAR(20) CHECK(
        bia IN (SELECT tên
                 FROM Bia) ) ,
    giá REAL CHECK(
        giá <= $5.00 )
);
```

# Kiểm tra trên từng bản ghi

- Được khai báo riêng sau khi đã khai báo các thuộc tính,
- Điều kiện *condition* có thể liên quan đến bất kỳ thuộc tính nào trong bảng
- Kiểm tra trong quá trình thêm, sửa bản ghi.

Ví dụ:

```
CREATE TABLE Bán (
    quán CHAR(20) ,
    bia CHAR(20) ,
    giá REAL,
    CHECK(quán = 'Hải Xồm' OR giá <= $5.00)
);
```

Chỉ quán Hải Xồm được phép bán bia đắt hơn \$5.00



# Một số đặc tính khác của thuộc tính

1. NOT NULL = mọi bản ghi phải có một giá trị nào đó tại thuộc tính này.
2. DEFAULT *value* = đặt mặc định một giá trị cho thuộc tính này trong trường hợp không nhập giá trị cho nó.

VD:

```
CREATE TABLE Khách_hàng (
    tên CHAR(30) PRIMARY KEY,
    địa_chỉ CHAR(50) DEFAULT '123 Sesame St',
    Đ_thoại CHAR(16)
);
```

- Khóa chính được mặc định là NOT NULL.

# Chèn giá trị mặc định

1. DEFAULT DATE/TIME/TIMESTAMP.
2. Tạo chuỗi số tự động bằng SEQUENCE

VD:

```
CREATE SEQUENCE Ma_KH;  
CREATE TABLE Khach_hang (  
    maKH INTEGER  
        DEFAULT nextval('Ma_KH'),  
    tên VARCHAR(30)  
);
```

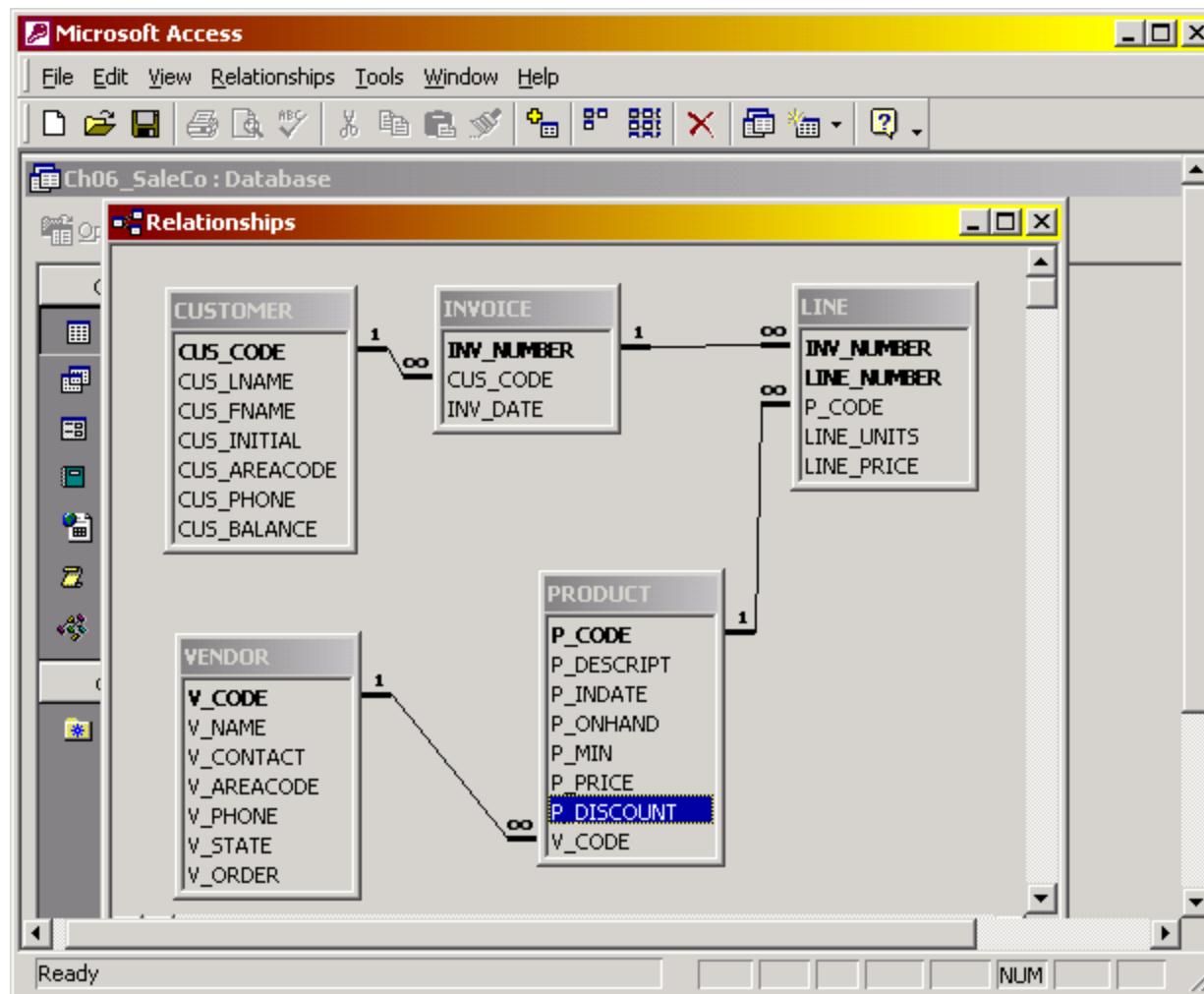


# Các câu lệnh của DDL trong SQL

- Chúng ta sẽ dùng cơ sở dữ liệu dưới đây để mô tả cho cách sử dụng của các lệnh DDL của SQL. Cơ sở dữ liệu này liên quan một chút tới cơ sở dữ liệu về nhà cung cấp - linh kiện - công việc - vận chuyển. Các qui định về nghiệp vụ của hệ thống này như sau:
  1. Một khách hàng có thể có nhiều yêu cầu trả tiền. mỗi bản yêu cầu trả tiền (invoice) chỉ được tạo ra bởi một khách hàng.
  2. Một invoice chứa một hoặc nhiều dòng. mỗi dòng của invoice liên quan tới một invoice.
  3. Mỗi dòng invoice là cho một mặt hàng. Một mặt hàng có thể tìm thấy ở nhiều dòng khác nhau. Một nhà cung cấp có thể cung cấp nhiều mặt hàng. Một vài nhà cung cấp có thể không cung cấp mặt hàng nào cả,
  4. Nếu một mặt hàng được cung cấp bởi một nhà cung cấp, thì mặt hàng đó chỉ được cung cấp bởi duy nhất nhà cung cấp đó.
  5. một số mặt hàng không được cung cấp bởi nhà cung cấp nào cả mà chúng được công ty tự sản xuất (in-house) hoặc được cung cấp qua những cách khác.



# Ví dụ về CSDL



# Tạo các cấu trúc bảng bằng SQL

- Câu lệnh CREATE TABLE có cú pháp như sau:

```
CREATE TABLE tablename (  
    column1 data type [constraint] [,  
    column2 data type [constraint] ] [,  
    PRIMARY KEY (column1 [,column2] )] [,  
    FOREIGN KEY (column1 [,column2] ) REFERENCES tablename ] [,  
    CONSTRAINT constraint ] );
```

# Ví dụ – Tạo bảng

- Ví dụ tạo một bảng VENDOR của cơ sở dữ liệu ví dụ được mô tả ở trên.

```
CREATE TABLE VENDOR (
```

V_CODE	INTEGER	NOT NULL	UNIQUE,
V_NAME	VARCHAR(35)	NOT NULL,	
V_CONTACT	VARCHAR(15)	NOT NULL,	
V_AREACODE	CHAR(3)	NOT NULL,	
V_PHONE	CHAR(8)	NOT NULL,	
V_STATE	CHAR(2)	NOT NULL,	
V_ORDER	CHAR(1)	NOT NULL,	

```
PRIMARY KEY ( V_CODE));
```

# Bảng VENDOR trong Access

The screenshot shows the Microsoft Access application interface. The title bar reads "Microsoft Access". The menu bar includes File, Edit, View, Insert, Tools, Window, and Help. The toolbar contains various icons for database management. The main window displays the "Ch06\_SaleCo : Database" and the "VENDOR : Table" design view. The table structure is shown in a grid with columns: Field Name, Data Type, and Description. The fields listed are V\_CODE (Number, primary key), V\_NAME (Text), V\_CONTACT (Text), V\_AREACODE (Text), V\_PHONE (Text), V\_STATE (Text), and V\_ORDER (Text). Below the table is a "Field Properties" pane. The "General" tab is selected, showing properties for V\_CODE: Field Size (Long Integer), Format (Auto), Decimal Places (0), Input Mask (empty), Caption (empty), Default Value (0), Validation Rule (empty), Validation Text (empty), Required (No), and Indexed (Yes (No Duplicates)). A tooltip in the properties pane states: "A field name can be up to 64 characters long, including spaces. Press F1 for help on field names." At the bottom of the screen, a status bar says "Design view. F6 = Switch panes. F1 = Help.".

Field Name	Data Type	Description
V_CODE	Number	Vendor code (primary key.)
V_NAME	Text	
V_CONTACT	Text	
V_AREACODE	Text	
V_PHONE	Text	
V_STATE	Text	
V_ORDER	Text	

Field Properties

General | Lookup |

Field Size: Long Integer  
Format: Auto  
Decimal Places: 0  
Input Mask:  
Caption:  
Default Value: 0  
Validation Rule:  
Validation Text:  
Required: No  
Indexed: Yes (No Duplicates)

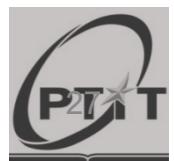
A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Design view. F6 = Switch panes. F1 = Help.

# Ví dụ – Tạo bảng

- Sau đó ta sẽ tạo tiếp đến bảng PRODUCT như sau:

```
CREATE TABLE PRODUCT (
    P_CODE          VARCHAR(10)      NOT NULL        UNIQUE,
    P_DESCRPT       VARCHAR(35)       NOT NULL,
    P_INDATE        DATE            NOT NULL,
    P_ONHAND        SMALLINT        NOT NULL,
    P_MIN           SMALLINT        NOT NULL,
    P_PRICE          NUMERIC(8,2)     NOT NULL,
    P_DISCOUNT      NUMBER(4,2)      NOT NULL,
    V_CODE          INTEGER,
    PRIMARY KEY ( P_CODE),
    FOREIGN KEY (V_CODE) REFERENCES VENDOR ON UPDATE CASCADE);
```



# Bảng PRODUCT trong Access

The screenshot shows the Microsoft Access application interface. The title bar reads "Microsoft Access". The menu bar includes File, Edit, View, Insert, Tools, Window, and Help. The toolbar contains various icons for database management. The main window displays the "Ch06\_SaleCo : Database" and the "PRODUCT : Table" design view. The table structure is shown in a grid:

Field Name	Data Type	Description
P_CODE	Text	Product code: Primary key
P_DESCRPT	Text	Product description
P_INDATE	Date/Time	Product stock date
P_ONHAND	Number	Number of units on hand (i.e., available for sale.)
P_MIN	Number	Minimum number of units on hand
P_PRICE	Currency	Product sales price
P_DISCOUNT	Number	product discount pct.
V_CODE	Number	Vendor code: Foreign key to VENDOR

Below the table grid, there is a "Field Properties" section with tabs for "General" and "Lookup". The "General" tab shows the following properties:

Field Size	10
Format	
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	Yes (No Duplicates)
Unicode Compression	Yes

A tooltip message is displayed in the "General" properties pane: "A field name can be up to 64 characters long, including spaces. Press F1 for help on field names."

At the bottom of the screen, a status bar displays: "Design view. F6 = Switch panes, F1 = Help." The keyboard layout is shown at the very bottom.

# Ví dụ – Tạo bảng

- Tạo ra bảng CUSTOMER như sau:

```
CREATE TABLE CUSTOMER (
```

CUS_CODE	NUMBER	PRIMARY KEY,
CUS_LNAME	VARCHAR(15)	NOT NULL,
CUS_FNAME	VARCHAR(15)	NOT NULL,
CUS_INITIAL	CHAR(1),	
CUS_AREACODE	CHAR(3)	DEFAULT '615' NOT NULL CHECK (CUS_AREACODE IN ('615', '713', '931')),
CUS_PHONE	CHAR(8)	NOT NULL,
CUS_BALANCE	NUMBER(9,2)	DEFAULT 0.00,
CONSTRAINT CUS_UI1 UNIQUE (CUS_LNAME, CUS_FNAME));		

Ràng buộc  
cột



Ràng buộc  
bảng

Tạo một ràng buộc chỉ mục duy nhất là CUS\_UI1 trên  
họ và tên của khách hàng.

# Bảng CUSTOMER trong Access

Microsoft Access

File Edit View Insert Tools Window Help

Ch06\_SaleCo : Database

CUSTOMER : Table

Field Name	Data Type	Description
CUS_CODE	Number	
CUS_LNAME	Text	
CUS_FNAME	Text	
CUS_INITIAL	Text	
CUS_AREACODE	Text	
CUS_PHONE	Text	
CUS_BALANCE	Currency	

Field Properties

General | Lookup |

Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	0
Validation Rule	
Validation Text	
Required	No
Indexed	Yes (No Duplicates)

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

Design view. F6 = Switch panes. F1 = Help.

# Ví dụ – Tạo bảng

- Tạo bảng INVOICE như sau:

```
CREATE TABLE INVOICE (
    INV_NUMBER      NUMBER          PRIMARY KEY,
    CUS_CODE        NUMBER          REFERENCES CUSTOMER(CUS_CODE) NOT NULL,
    INV_DATE        DATE           DEFAULT CONVERT(date, (SYSDATETIME())) NOT
NULL,
CONSTRAINT INV_CK1 CHECK (INV_DATE > '2019-12-03'))
```

Một cách khác để định nghĩa khóa ngoại

Hàm đặc biệt để trả về ngày hiện tại

ràng buộc CHECK được sử dụng để kiểm tra tính hợp lệ của ngày invoice có lớn hơn 1/1/2002 không.  
Hàm TO\_DATE cần hai tham số bao gồm ngày cụ thể và định dạng ngày được sử dụng.

# Bảng INVOICE trong Access

# Ví dụ – Tạo bảng

- Cuối cùng, tạo bảng LINE như sau:

```
CREATE TABLE LINE (
    INV_NUMBER      NUMERIC NOT NULL,
    LINE_NUMBER     NUMERIC(2,0)      NOT NULL,
    P_CODE          VARCHAR(10)      NOT NULL,
    LINE_UNITS      NUMERIC(9,2)      DEFAULT 0.00 NOT NULL,
    LINE_PRICE      NUMERIC(9,2)      DEFAULT 0.00 NOT NULL,
    PRIMARY KEY (INV_NUMBER, LINE_NUMBER),
    FOREIGN KEY (INV_NUMBER) REFERENCES INVOICE   ON DELETE CASCADE,
    FOREIGN KEY (P_CODE) REFERENCES PRODUCT(P_CODE),
    CONSTRAINT LINE_UI1 UNIQUE(INV_NUMBER, P_CODE));
```

ràng buộc trên toàn bảng để  
ngăn chặn việc có hai dòng  
trong một invoice giống nhau

việc sử dụng ON  
DELETE CASCADE  
được khuyến cáo nên  
dùng cho các thực thể  
yếu để đảm bảo rằng  
việc xoá một dòng  
trong thực thể chính sẽ  
gây ra việc xoá tự động  
các dòng tương ứng  
trong thực thể yếu phụ  
thuộc vào thực thể  
chính đó



# Bảng LINE trong Access

The screenshot shows the Microsoft Access application interface. The title bar reads "Microsoft Access". The menu bar includes "File", "Edit", "View", "Insert", "Tools", "Window", and "Help". The toolbar contains various icons for database management. The main window displays the "Ch06\_SaleCo : Database" and the "LINE : Table" design view. The table structure is shown in a grid:

	Field Name	Data Type	Description
1	INV_NUMBER	Number	
2	LINE_NUMBER	Number	
3	P_CODE	Text	
4	LINE_UNITS	Number	
5	LINE_PRICE	Currency	

Below the table grid, there is a "Field Properties" pane. It has tabs for "General" and "Lookup", with "General" selected. The properties listed are:

- Field Size: Long Integer
- Format
- Decimal Places: Auto
- Input Mask
- Caption
- Default Value: 0
- Validation Rule
- Validation Text
- Required: No
- Indexed: No

A note in the pane states: "A field name can be up to 64 characters long, including spaces. Press F1 for help on field names."

At the bottom of the window, a status bar says "Design view. F6 = Switch panes, F1 = Help." There are also numeric keys (0-9, NUM) at the bottom right.

# Một số lưu ý trong việc tạo bảng

- Đối với cơ sở dữ liệu ví dụ trên, bảng PRODUCT chứa một khoá ngoại tham chiếu tới bảng VENDOR. Vì vậy, bảng VENDOR phải được tạo trước. Nói chung, các bảng nằm bên phía lực lượng 1 của một quan hệ 1-nhiều phải được tạo trước khi bảng bên phía lực lượng nhiều có thể được tạo ra.
- Với hệ thống Oracle9i nếu bạn sử dụng cách định nghĩa khoá chính bằng từ khoá PRIMARY KEY bạn không cần đưa yêu cầu NOT NULL và UNIQUE vào câu lệnh tạo bảng nữa. Thực tế, bạn sẽ nhận được một thông báo lỗi nếu bạn làm như vậy.
- Ràng buộc ON UPDATE CASCADE là một phần của chuẩn ANSI nhưng nhiều hệ quản trị cơ sở dữ liệu không hỗ trợ nps. Oracle là một trong số những hệ thống quản trị không hỗ trợ tính năng này.
- Nếu khoá chính là một khoá ghép, tất cả các thuộc tính của khoá được chứa trong một dấu ngoặc đơn và được phân tách nhau bởi dấu phẩy. Ví dụ, bảng LINE có khoá chính được định nghĩa như sau:

PRIMARY KEY (inv\_number, line\_number).



# Một số lưu ý trong việc tạo bảng (cont.)

- Hỗ trợ ràng buộc tham chiếu rất đa dạng, thay đổi từ hệ quản trị này sang hệ quản trị khác.
  - MS Access, SQL Server và Oracle hỗ trợ ON DELETE CASCADE.
  - MS Access, SQL Server hỗ trợ ON UPDATE CASCADE.
  - Oracle không hỗ trợ ON UPDATE CASCADE.
  - Oracle hỗ trợ SET NULL.
  - MS Access, SQL Server không hỗ trợ SET NULL.
- MS Access không hỗ trợ ON DELETE CASCADE hoặc ON UPDATE CASCADE tại mức câu lệnh SQL tuy nhiên nó lại hỗ trợ thông qua giao diện cửa sổ quan hệ.

# Các câu lệnh DDL nâng cao trong SQL

- Các câu lệnh SQL thay đổi cấu trúc của bảng bằng cách thay đổi đặc tính hoặc thêm các thuộc tính.
- Các câu lệnh liên quan đến việc thêm dữ liệu và các cột dữ liệu đã thay đổi cũng được đề cập.
- Cách nhân đôi hoặc xóa cả bảng hoặc từng phần của bảng.



# Câu lệnh ALTER TABLE

- Việc thay đổi cấu trúc của bảng được thực hiện bằng lệnh ALTER TABLE, kèm theo một từ khóa chỉ rõ việc thay đổi là gì.
- Có 3 từ khóa cơ bản được sử dụng: ADD, MODIFY, và DROP.
  - **ADD** cho phép thêm cột vào bảng.
  - **MODIFY** cho phép thay đổi đặc tính của bảng.
  - **DROP** cho phép xóa cột của bảng. Hầu hết các hệ CSDL quan hệ không cho phép xóa cột trừ khi không có giá trị nào trong cột muốn xóa vì nó liên quan mật thiết với các bảng dữ liệu khác.

# Câu lệnh ALTER TABLE (cont.)

- Cú pháp cơ bản của lệnh ALTER TABLE là:

```
ALTER TABLE ten_bang ALTER COLUMN ten_cot kieu_cot;
```

- Lệnh ALTER TABLE cũng được dùng để thêm ràng buộc cho bảng. Trường hợp này có cú pháp như sau:

```
ALTER TABLE tablename
ADD constraint [ ADD constraint];
```

# Câu lệnh ALTER TABLE (cont.)

- Câu lệnh ALTER TABLE còn được dùng để xóa cột hoặc ràng buộc của bảng. Cú pháp cơ bản trong trường hợp này là:

```
ALTER TABLE tablename
    DROP { PRIMARY KEY |
            COLUMN columnname |
            CONSTRAINT constraintname } ;
```

- Lưu ý rằng khi xóa ràng buộc của bảng ta cần xác định tên của ràng buộc đó. Đây là lý do tại sao cần phải đặt tên cho các ràng buộc trong các lệnh CREATE TABLE hoặc ALTER TABLE.

# Thay đổi kiểu dữ liệu cho cột

- Câu lệnh ALTER TABLE cũng được dùng để đổi kiểu dữ liệu cho cột.
- Ví dụ, nếu ta cần phải đổi kiểu dữ liệu của V\_CODE trong bảng PRODUCT từ kiểu integer thành character. Câu lệnh SQL sẽ như sau:

```
• ALTER TABLE PRODUCT  
•     ALTer COLUMN V_CODE Varchar(9,2);
```

- ~~Hầu hết các hệ CSDL quan hệ không cho phép thay đổi kiểu dữ liệu của các thuộc tính trừ khi thuộc tính đó là rỗng.~~ Ví dụ, nếu ta chạy câu lệnh SQL nói trên trong CSDL mà ta đang dùng, dòng báo lỗi sẽ hiện ra bởi vì cột V\_CODE đang chứa dữ liệu. Lý do của lỗi này đơn giản bởi vì thuộc tính V\_CODE trong bảng PRODUCT chỉ đến cột V\_CODE trong bảng VENDOR. Nếu hai cột dữ liệu không cùng kiểu sẽ dẫn đến xung đột giá trị tham vấn. Nếu cột V\_CODE trong bảng PRODUCT là rỗng và khóa ngoại không được xác định trong lúc tạo bảng PRODUCT thì câu lệnh SQL nói trên sẽ được thực hiện đúng.



# Thay đổi đặc tính dữ liệu của cột

- Nếu cột cần thay đổi đang chứa dữ liệu, ta có thể thực hiện bất kỳ thay đổi nào miễn là không làm thay đổi kiểu dữ liệu của nó.
- Ví dụ, nếu cần tăng độ rộng của cột P\_PRICE từ 8 lên 9 ký tự, ta cần phải viết câu lệnh như sau:

```
ALTER TABLE PRODUCT
```

```
alter column P_PRICE DECIMAL(9,2);
```

- Nhiều hệ CSDL quan hệ giới hạn các thay đổi có thể thực hiện lên cột. Ví dụ, Oracle cho phép mở rộng các cột nhưng lại không cho thu hẹp chúng lại.



# Thêm cột vào bảng

- Một bảng CSDL có thể được thay đổi bằng cách thêm hoặc xóa các cột.
- Ví dụ, nếu ta cần thêm cột P\_SALECODE vào bảng PRODUCT, cột này cho phép ta kiểm tra xem những mặt hàng nào đã được kiểm kê về thời gian cần được bày bán. Giả sử các giá trị trong cột P\_SALECODE nằm trong tập {1, 2, 3}, và không có phép tính trong đó thì ta gán kiểu character cho cột này.

```
•ALTER TABLE PRODUCT
  •      ADD P_SALECODE CHAR(1);
```



# Thêm cột vào bảng (cont.)

- Khi thêm cột, không được cho cụm từ NOT NULL vào cột mới. Điều này dẫn đến báo lỗi vì khi ta thêm cột vào bảng, các hàng có sẵn trong bảng sẽ mặc định giá trị là rỗng cho cột đó. Bởi vậy, ta không thể gán cụm từ NOT NULL cho cột này.
- Ta có thể thêm cụm từ NOT NULL vào cấu trúc của bảng sau khi toàn bộ dữ liệu của cột mới đã được nhập vào, và cột đó không chứa giá trị rỗng nào.

# Xóa cột từ một bảng

- Đôi khi việc xóa đi một vài cột trong bảng là cần thiết.
- Giả sử ta cần xóa cột V\_ORDER trong bảng VENDOR. Câu lệnh SQL sau đây có thể được dùng:

```
ALTER TABLE VENDOR  
DROP COLUMN V_ORDER;
```

- Tương tự như phần trước, một vài hệ CSDL quan hệ có các giới hạn trong việc xóa cột khỏi bảng. Ví dụ, hầu hết các hệ CSDL quan hệ không cho phép xóa các cột có quan hệ với khóa ngoại, cũng như xóa các cột chứa khóa đó.

# Thêm các chỉ định khóa chính và khóa ngoại

- Mặc dù ta có thể tạo một bảng mới dựa trên một bảng có sẵn như trong ví dụ vừa rồi, quá trình thực hiện vẫn có thể gặp rắc rối. Về cơ bản, bảng PART được tạo ra mà không cần bảo đảm tính toàn vẹn của CSDL cũ. Cụ thể, ta không cần phải gán khóa chính cho bảng ở slide trước.
- Để xác định khóa chính cho bảng này, ta dùng lệnh ALTER như sau:

```
ALTER TABLE PRODUCT  
ADD PRIMARY KEY (P_CODE);
```

## Thêm các chỉ định khóa chính và khóa ngoại (cont.)

- Thực tế rằng luật toàn vẹn không được tự động thừa kế từ bảng cũ sang bảng mới, do vậy trong nhiều trường hợp chúng ta không có sự toàn vẹn về thực thể cũng như các mối liên kết.
- Ví dụ, ta có thể quên xác định khóa chính và khóa ngoại trong quá trình tạo bảng.
- Các quy tắc toàn vẹn có thể được tái thiết lập sử dụng câu lệnh ALTER như sau:

```
ALTER TABLE PRODUCT  
ADD PRIMARY KEY(P_CODE)  
ADD FOREIGN KEY(V_CODE) REFERENCES VENDOR;
```



# Xóa bảng khỏi CSDL

- Ta có thể xóa bảng PRODUCT bằng lệnh DROP như sau:

```
DROP TABLE PRODUCT
```

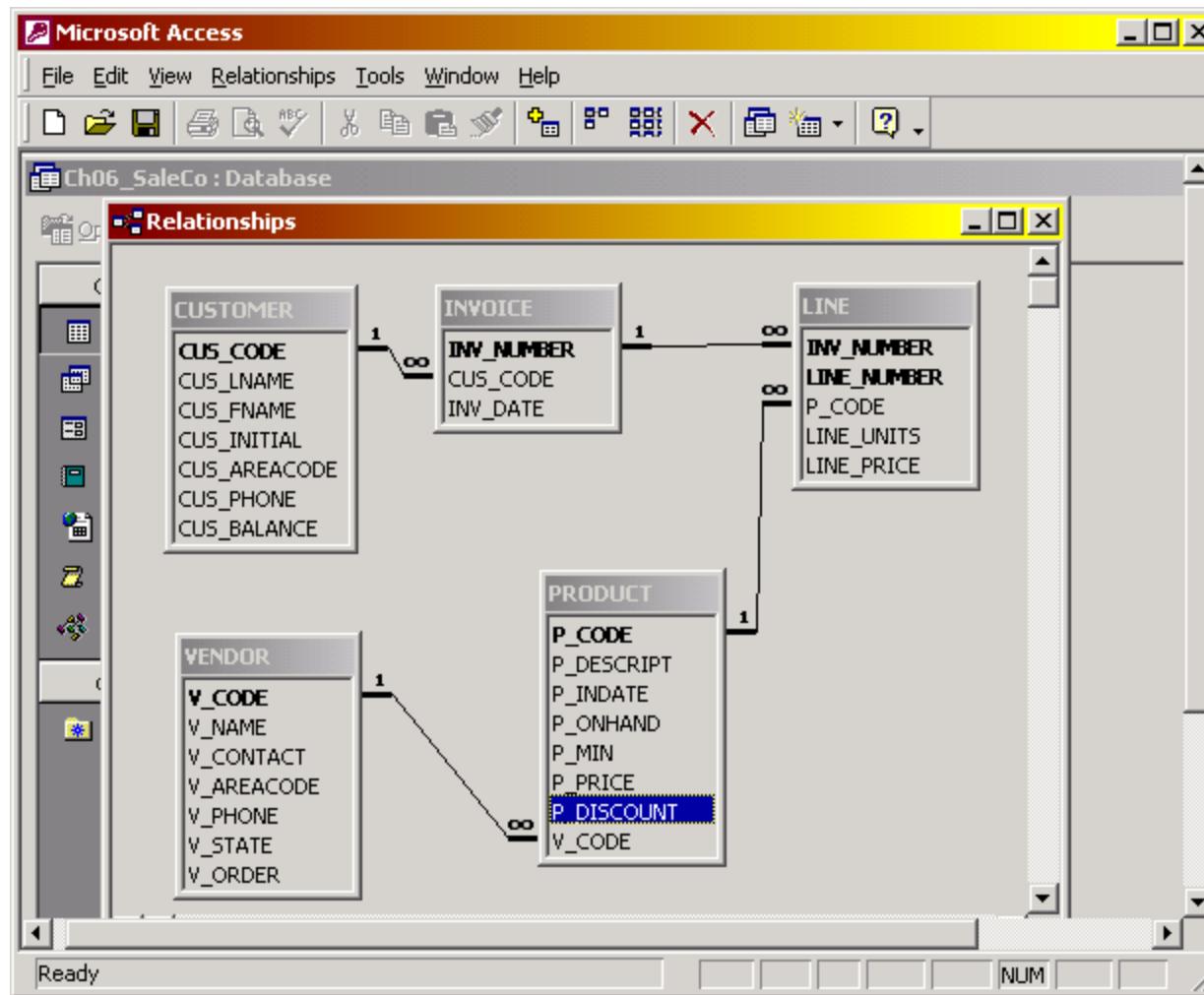
- Một bảng chỉ có thể được xóa khỏi CSDL nếu nó không tham gia vào bất kỳ một mối quan hệ nào. Nếu ta cố xóa một bảng mà vẫn tham gia vào quan hệ thì sẽ có bản tin báo lỗi hiện ra.

# Giới thiệu về ngôn ngữ SQL (phần 2)

Posts and Telecommunications Institute of Technology-PTIT



# Ví dụ về CSDL



# Ngôn ngữ thao tác dữ liệu DML của SQL

- Ngôn ngữ thao tác dữ liệu của SQL có thể được chia ra làm hai phần tách riêng nhưng vẫn chung nhau ở một số phạm vi nào đó. Hai phần này là các câu lệnh DML không truy vấn và các câu lệnh truy vấn dữ liệu.
- Ngôn ngữ thao tác không truy vấn cho phép bạn thêm dữ liệu vào bảng (INSERT), sửa đổi dữ liệu (UPDATE), xoá dữ liệu từ các bảng (DELETE) và thực hiện những thay đổi vĩnh viễn (COMMIT) và huỷ những thay đổi (tới một mức độ nào đó với ROLLBACK).
- Các câu lệnh truy vấn DML chắc chắn bao gồm câu lệnh đơn SELECT với rất nhiều các mệnh đề lựa chọn khác nhau. Chúng ta sẽ xem xét các câu lệnh không truy vấn của DML trước.
- Tóm tắt các câu lệnh DML của SQL được mô tả trong bảng dưới sau.



# Tóm tắt các câu lệnh DML của SQL

Câu lệnh hoặc lựa chọn	Mô tả
INSERT	Chèn thêm một (các) hàng vào trong một bảng
SELECT	Lựa chọn các thuộc tính từ các hàng trong một hoặc nhiều bảng hoặc khung nhìn
WHERE	Hạn chế việc lựa chọn các hàng dựa trên một biểu thức điều kiện
GROUP BY	Gộp nhóm các hàng đã được chọn ra dựa trên một hoặc nhiều thuộc tính
HAVING	Hạn chế sự lựa chọn các hàng để gộp nhóm dựa trên một điều kiện
ORDER BY	Xếp thứ tự các hàng được chọn
UPDATE	Sửa đổi giá trị thuộc tính của một hoặc nhiều hàng của một bảng
DELETE	Xoá một hoặc nhiều hàng từ một bảng
COMMIT	Lưu trữ vĩnh viễn những thay đổi về dữ liệu
ROLLBACK	Phục hồi dữ liệu về những giá trị ban đầu của chúng
<i>Các phép toán so sánh</i>	
=, <, >, <=, >=, <>	Được sử dụng trong các biểu thức điều kiện
<i>Các phép toán logic</i>	
AND, OR, NOT	Được sử dụng trong các biểu thức điều kiện

# Tóm tắt các câu lệnh DML của SQL (cont.)

Câu lệnh hoặc lựa chọn	Mô tả
<b>Các phép toán đặc biệt</b>	<i>được sử dụng trong các biểu thức điều kiện</i>
BETWEEN	Kiểm tra xem các giá trị của một thuộc tính có nằm trong một khoảng xác định
IS NULL	Kiểm tra xem giá trị của một thuộc tính có là trống / hoặc có giá trị không
LIKE	Kiểm tra xem giá trị của một thuộc tính có giống với một kiểu chuỗi ký tự cho trước
IN / NOT IN	Kiểm tra xem giá trị của một thuộc tính có nằm trong / hoặc không nằm trong một danh sách các giá trị nào đó
EXISTS / NOT EXISTS	Kiểm tra xem một truy vấn con có trả về hàng dữ liệu nào không
DISTINCT	Hạn chế các giá trị tới những giá trị duy nhất, hay loại bỏ những giá trị trùng lặp
<b>Các hàm thống kê</b>	<i>được sử dụng với SELECT để trả về những giá trị tổng hợp trên các cột</i>
COUNT	Trả về số lượng các hàng với các giá trị không rỗng cho một cột nào đó
MIN	Trả về giá trị nhỏ nhất của một thuộc tính được tìm thấy trong một cột nào đó
MAX	Trả về giá trị lớn nhất của một thuộc tính được tìm thấy trong một cột nào đó
SUM	Trả về tổng của tất cả các giá trị của một cột nào đó
AVG	Trả về giá trị trung bình của tất cả các giá trị của một cột nào đó

# Thêm các bản ghi vào bảng

- SQL dùng câu lệnh INSERT để thêm dữ liệu mới vào một bảng.
- Cú pháp của câu lệnh này như sau:

```
INSERT INTO tablename
```

```
    VALUES (value1, value 2, ...value n);
```



# Ví dụ - thêm các bản ghi vào bảng

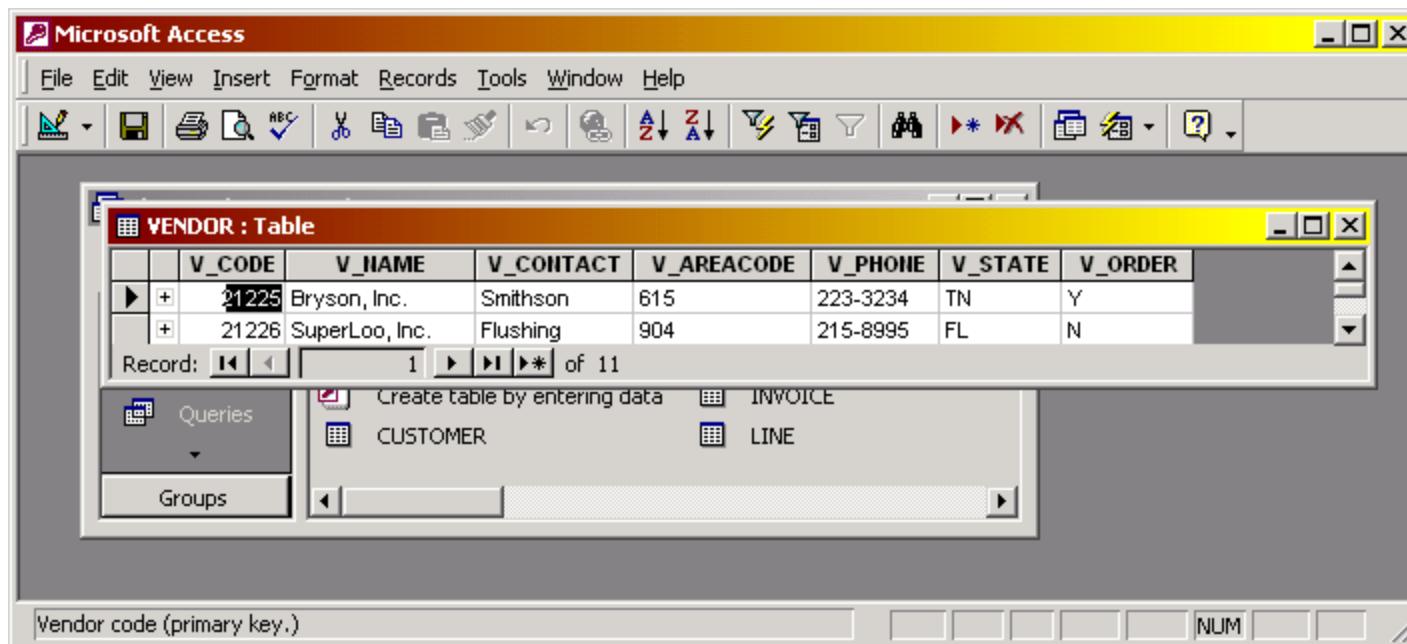
- Thêm hai bản ghi mới vào bảng VENDOR chúng ta cần thực hiện hai câu lệnh SQL dưới đây:

```
INSERT INTO VENDOR
```

```
VALUES (21225, 'Bryson, Inc.', 'Smithson', '615', '223-3234', 'TN', 'Y');
```

```
INSERT INTO VENDOR
```

```
VALUES (21226, 'SuperLoo, Inc.', 'Flushing', '904', '215-8995', 'FL', 'N');
```



# Ví dụ - Thêm bản ghi có thuộc tính rỗng

- Nếu một thuộc tính của một bản ghi không có giá trị (hay có giá trị là null) bạn sẽ sử dụng cú pháp sau đây để thêm một hàng vào bảng:

```
INSERT INTO PRODUCT
```

```
VALUES ('23114-AA', 'Sledge hammer, 12 lb.', '02-Jan-02', 8, 5, 14.40, 0.05, NULL);
```

The screenshot shows the Microsoft Access application interface with the 'PRODUCT : Table' open. The table has columns: P\_CODE, P\_DESCRPT, P\_INDATE, P\_ONHAND, P\_MIN, P\_PRICE, P\_DISCOUNT, and V\_COI. A new row is being inserted at the bottom of the table, highlighted with a black background. The values for the new row are: P\_CODE = '23114-AA', P\_DESCRPT = 'Sledge hammer, 12 lb.', P\_INDATE = '02-Jan-04', P\_ONHAND = 8, P\_MIN = 5, P\_PRICE = '\$14.40', P\_DISCOUNT = '0.05', and V\_COI = null. A green callout box points to the SQL command above, stating: 'Lệnh này chèn bản ghi này vào bảng PRODUCT'. A blue arrow points from the green box down to the newly inserted row in the table.

	P_CODE	P_DESCRPT	P_INDATE	P_ONHAND	P_MIN	P_PRICE	P_DISCOUNT	V_COI
1	+ 11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	8	5	\$109.99	0.00	255
2	+ 13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	32	15	\$14.99	0.05	213
3	+ 14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	18	12	\$17.49	0.00	213
4	+ 1546-QQQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	15	8	\$39.95	0.00	231
5	+ 1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	23	5	\$43.99	0.00	231
6	+ 2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	8	5	\$109.92	0.05	242
7	+ 2232/QME	B&D jigsaw, 8-in. blade	24-Dec-03	6	5	\$99.87	0.05	242
8	+ 2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	12	5	\$38.95	0.05	255
9	+ 23109-HB	Claw hammer	20-Jan-04	23	10	\$9.95	0.10	212
10	+ 23114-AA	Sledge hammer, 12 lb.	02-Jan-04	8	5	\$14.40	0.05	
11	+ 54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	43	20	\$4.99	0.00	213
12	+ 89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	11	5	\$256.99	0.05	242
13	+ PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	188	75	\$5.87	0.00	
14	+ SM-18277	1.25-in. metal screw, 25	01-Mar-04	172	75	\$6.99	0.00	212

# Ví dụ - Thêm bản ghi có thuộc tính rỗng

- Trong các trường hợp có nhiều hơn một thuộc tính nhận giá trị rỗng, thay vì khai báo các thuộc tính là NULL trong lệnh INSERT, ta chỉ cần mô tả các thuộc tính cần có giá trị nhập, không cần quan tâm tới các thuộc tính rỗng.
- Việc này được thực hiện bằng cách liệt kê tên của các thuộc tính mà giá trị của chúng được đưa vào bên trong dấu ngoặc đơn ngay sau tên của bảng.
- Xét ví dụ dưới đây, giả sử rằng chỉ P\_CODE và P\_DESCRIPT cần nhập giá trị vào trong bảng PRODUCT. Hai cách nhập sau đây đều đúng:

```
INSERT INTO PRODUCT
```

```
    VALUES ('23114-AA', 'Sledge hammer, 12 lb.', NULL, NULL, NULL, NULL, NULL, NULL);
```

-or-

```
INSERT INTO PRODUCT(P_CODE, P_DESCRIPT)
```

```
    VALUES('23114-AA', 'Sledge hammer, 12 lb.');
```



# Xóa các bản ghi khỏi bảng

- SQL cho phép dễ dàng xóa một bản ghi ra khỏi bảng bằng lệnh DELETE.
- Cú pháp của lệnh DELETE là:

```
DELETE FROM tablename  
[WHERE conditionlist];
```

- Để xoá một bản ghi từ một bảng dựa trên giá trị của khoá chính, bạn có thể sử dụng câu lệnh như sau:

```
DELETE FROM PRODUCT  
WHERE P_CODE = '23114-AA';
```



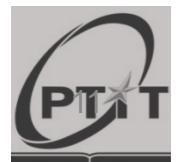
# Xóa các bản ghi khỏi bảng (cont.)

- Lệnh DELETE cũng được dùng để xóa nhiều bản ghi ra khỏi bảng.
  - Ví dụ bạn muốn xoá các sản phẩm từ bảng PRODUCT mà giá trị P\_MIN =5. Để thực hiện điều này bạn có thể dùng câu lệnh:

```
DELETE FROM PRODUCT
```

```
WHERE P_MIN = 5;
```

- **DELETE là một câu lệnh hướng tập hợp. Có nghĩa là điều kiện ở câu lệnh WHERE là có thể có hoặc không, nếu điều kiện đó không được chỉ rõ thì tất cả các hàng của bảng sẽ được xoá!**



# Cập nhật dữ liệu cho các bản ghi trong bảng

- Để thay đổi dữ liệu trong bảng, ta dùng lệnh UPDATE.
- Cú pháp của lệnh UPDATE như sau:

```
UPDATE tablename
```

```
    SET columnname = expression [, columnname = expression ]
```

```
    [ WHERE conditionlist ];
```

- Lưu ý rằng điều kiện WHERE là không bắt buộc trong câu lệnh UPDATE. Nếu không có điều kiện WHERE, thì câu lệnh UPDATE sẽ được thực hiện trên tất cả các bản ghi của bảng đó.



# Cập nhật dữ liệu cho các bản ghi trong bảng (cont.)

- Ví dụ, bạn muốn thay đổi P\_INDATE từ 13/12/2003 thành 18/1/2004 trong hàng thứ hai của bảng PRODUCT. Bạn cần sử dụng giá trị của khoá chính 13-Q2/P2 để xác định đúng hàng trong bảng cần thay đổi, câu lệnh tương ứng như sau:

```
UPDATE PRODUCT  
SET P_INDATE = '18-Jan-2004'  
WHERE P_CODE = '13-Q2/P2';
```

- Nếu có nhiều hơn một thuộc tính cần được thay đổi trong một bản ghi, các câu lệnh UPDATE sẽ được phân cách nhau bởi dấu phẩy:

```
UPDATE PRODUCT  
SET P_INDATE = '18-JAN-2004', P_PRICE = 16.99, P_MIN = 10  
WHERE P_CODE = '13-Q2/P2';
```



# Cập nhật dữ liệu

- Để thêm dữ liệu vào các cột có sẵn, ta sử dụng lệnh UPDATE trong SQL. Lệnh UPDATE chỉ cập nhật dữ liệu trong cột có sẵn.
- Ví dụ, để thêm giá trị ‘2’ vào cột P\_SALECODE tại dòng 4 của bảng PRODUCT\_2, ta dùng lệnh UPDATE kết hợp với giá trị khóa chính của dòng đó. Câu lệnh sau thực hiện việc này.

```
UPDATE PRODUCT_2  
SET P_SALECODE = '2'  
WHERE P_CODE = '1546-QQ2';
```

Kết quả trước và sau câu lệnh được thể hiện ở các slides tiếp theo.



# Cập nhật dữ liệu

Giá trị P\_SALECODE trước khi cập nhật

The screenshot shows a Microsoft Access application window titled "Microsoft Access". The menu bar includes File, Edit, View, Insert, Format, Records, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main area displays a table named "PRODUCT\_2 : Table" with the following columns: P\_CODE, P\_DESCRPT, P\_INDATE, P\_ONHAND, P\_MIN, P\_PRICE, P\_DISCOUNT, V\_CODE, and P\_SALECODE. The table contains 15 rows of product information. An arrow points to the P\_SALECODE column for the row where P\_CODE is 1546-QQ2.

P_CODE	P_DESCRPT	P_INDATE	P_ONHAND	P_MIN	P_PRICE	P_DISCOUNT	V_CODE	P_SALECODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	8	5	\$109.99	0.00	25595	
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	32	15	\$14.99	0.05	21344	
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	18	12	\$17.49	0.00	21344	
► 1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	15	8	\$39.95	0.00	23119	←
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	23	5	\$43.99	0.00	23119	
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	8	5	\$109.92	0.05	24288	1
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-03	6	5	\$99.87	0.05	24288	1
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	12	5	\$38.95	0.05	25595	
23109-HB	Claw hammer	20-Jan-04	23	10	\$9.95	0.10	21225	
23114-AA	Sledge hammer, 12 lb.	02-Jan-04	8	5	\$14.40	0.05		
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	43	20	\$4.99	0.00	21344	
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	11	5	\$256.99	0.05	24288	
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	188	75	\$5.87	0.00		
SM-18277	1.25-in. metal screw, 25	01-Mar-04	172	75	\$6.99	0.00	21225	

# Cập nhật dữ liệu

Giá trị của P\_SALECODE sau cập nhật

Microsoft Access

File Edit View Insert Format Records Tools Window Help

Product\_2 : Table

P_CODE	P_DESCRIP	P_INDATE	P_ONHAND	P_MIN	P_PRICE	P_DISCOUNT	V_CODE	P_SALECODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	8	5	\$109.99	0.00	25595	
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	32	15	\$14.99	0.05	21344	
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	18	12	\$17.49	0.00	21344	
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	15	8	\$39.95	0.00	23119	2
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	23	5	\$43.99	0.00	23119	
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	8	5	\$109.92	0.05	24288	1
2232/QIWE	B&D jigsaw, 8-in. blade	24-Dec-03	6	5	\$99.87	0.05	24288	1
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	12	5	\$38.95	0.05	25595	
23109-HB	Claw hammer	20-Jan-04	23	10	\$9.95	0.10	21225	
23114-AA	Sledge hammer, 12 lb.	02-Jan-04	8	5	\$14.40	0.05		
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	43	20	\$4.99	0.00	21344	
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	11	5	\$256.99	0.05	24288	
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	188	75	\$5.87	0.00		
SM-18277	1.25-in. metal screw, 25	01-Mar-04	172	75	\$6.99	0.00	21225	

# Cập nhật dữ liệu

- Các giá trị tiếp theo có thể được thêm vào theo cùng một cách thức: xác định vị trí thêm vào bằng khóa chính (P\_CODE) và tên cột dữ liệu (P\_SALECODE).
- Bảng dữ liệu kết quả ở trang trước có thể được tạo ra bởi câu lệnh SQL sau:

```
UPDATE PRODUCT_2  
SET P_SALECODE = '1'  
WHERE P_CODE IN ('2232/QWE', '2232/QTY');
```



# Cập nhật dữ liệu

- Mặc dù chuỗi lệnh UPDATE ở phần trước cho phép nhập dữ liệu vào các ô cụ thể trong bảng, quá trình thực hiện về bản chất rất phức tạp. May mắn là có các phương pháp thực hiện tốt hơn.
- Nếu ta có mối liên hệ giữa các giá trị cần nhập với các dữ liệu có sẵn thì có thể dùng đó để gán các giá trị mới vào vị trí thích hợp.
- Ví dụ, nếu ta cần cho mã bán hàng (P\_SALECODE) vào bảng dựa trên các giá trị của P\_INDATE như sau:
  - Nếu P\_INDATE nhỏ hơn 25/12/2003, thì P\_SALECODE = 2
  - Nếu P\_INDATE nằm giữa 16/1/2004 và 10/2/2004, thì P\_SALECODE = 1

# Cập nhật dữ liệu

- Với các điều kiện đã cho ở slide trước, chuỗi câu lệnh sau cho phép cập nhật bảng PRODUCT tại cột P\_SALECODE. Minh họa cho bảng PRODUCT được trình bày trong slide tiếp theo.

```
UPDATE PRODUCT
```

```
SET P_SALECODE = '2'
```

```
WHERE P_INDATE < '25-Dec-2003';
```

```
UPDATE PRODUCT
```

```
SET P_SALECODE = '1'
```

```
WHERE P_INDATE >= '16-Jan-2004'
```

```
AND P_INDATE <= '10-Feb-2004';
```



# Cập nhật dữ liệu

Microsoft Access

File Edit View Insert Format Records Tools Window Help

Ch06\_SaleCo : Database

PRODUCT\_3 : Table

P_CODE	P_DESCRPT	P_INDATE	P_ONHAND	P_MIN	P_PRICE	P_DISCOUNT	V_CODE	P_SALECODE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	8	5	\$109.99	0.00	25595	2
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	32	15	\$14.99	0.05	21344	2
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	18	12	\$17.49	0.00	21344	2
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	15	8	\$39.95	0.00	23119	
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	23	5	\$43.99	0.00	23119	
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	8	5	\$109.92	0.05	24288	
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-03	6	5	\$99.87	0.05	24288	2
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	12	5	\$38.95	0.05	25595	1
23109-HB	Claw hammer	20-Jan-04	23	10	\$9.95	0.10	21225	1
23114-AA	Sledge hammer, 12 lb.	02-Jan-04	8	5	\$14.40	0.05		
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	43	20	\$4.99	0.00	21344	2
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	11	5	\$256.99	0.05	24288	1
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	188	75	\$5.87	0.00		
SM-18277	1.25-in. metal screw, 25	01-Mar-04	172	75	\$6.99	0.00	21225	
SW-23116	2.5-in. wd. screw, 50	24-Feb-04	237	100	\$8.45	0.00	21231	
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	18	5	\$119.95	0.10	25595	1
			0	0	\$0.00	0.00	0	

Record: 17 of 17

Product code: Primary key

# Cập nhật dữ liệu

- Các biểu thức toán học rất có ích trong những trường hợp này.
- Ví dụ, nếu lượng hàng hiện có (P\_ONHAND) trong bảng PRODUCT nhỏ hơn một ngưỡng tối thiểu nào đó, ta cần đặt hàng thêm. Giả sử ta đặt đơn hàng gồm 20 sản phẩm có mã 2232/QWE. Khi hàng về, ta cần nhập kho với câu lệnh sau:

```
UPDATE PRODUCT
```

```
SET P_ONHAND = P_ONHAND + 20
```

```
WHERE P_CODE = '2232/QWE';
```



# Cập nhật dữ liệu

- Giả sử ta cần tăng giá của tất cả sản phẩm có giá dưới \$50 lên thêm 10%, thì câu lệnh sau có thể được sử dụng:

```
UPDATE PRODUCT
```

```
    SET P_PRICE = P_PRICE * 1.10
```

```
    WHERE P_PRICE < 50.00;
```



# Lưu các thay đổi trong bảng

- Mọi thay đổi được thực hiện tới nội dung của bảng sẽ không được lưu trữ một cách vật lý trong một bảng vật lý (là một tệp trong hệ thống) cho tới khi một câu lệnh COMMIT được thực thi.
- Thông thường, nếu hệ thống mất điện trong quá trình cập nhật một bảng (hoặc một cơ sở dữ liệu nói chung) trước khi câu lệnh COMMIT được thực thi, tất cả các thay đổi bạn thực hiện trước đó sẽ bị mất. Những hệ thống phức tạp hơn sẽ có khả năng phục hồi dữ liệu sau những sự cố như vậy. Với các hệ thống máy tính cá nhân thì nên sử dụng bộ lưu điện UPS để hạn chế sự cố này!
- Cú pháp của lệnh COMMIT như sau:

COMMIT [ *tablename* ];

-or-

COMMIT; //saves all changes made in any modified tables



# Phục hồi nội dung của bảng

- Nếu bạn chưa sử dụng câu lệnh COMMIT để lưu trữ vĩnh viễn những thay đổi trong CSDL, bạn có thể phục hồi cơ sở dữ liệu về trạng thái trước đó (kết quả của lần cuối thực hiện câu lệnh COMMIT) bằng việc sử dụng câu lệnh ROLLBACK.
- ROLLBACK phục hồi lại những thay đổi được thực hiện và trả lại dữ liệu những giá trị cũ của nó trước khi những thay đổi này được thực hiện.
- Cú pháp của câu lệnh ROLLBACK như sau:

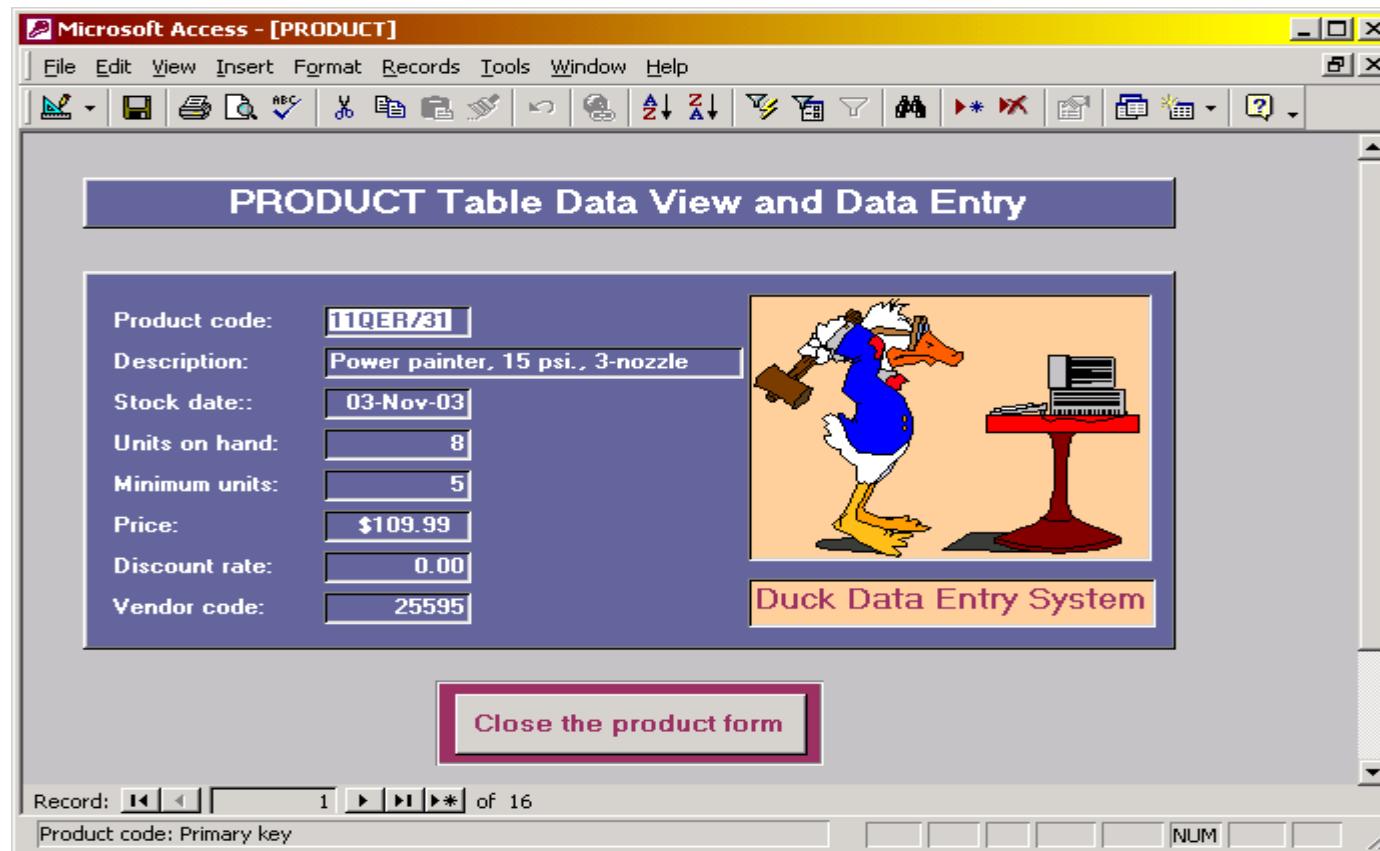
ROLLBACK;

- MS Access không hỗ trợ câu lệnh ROLLBACK. Một số hệ quản trị CSDL như Oracle tự động COMMIT những thay đổi dữ liệu khi thực hiện các câu lệnh DDL, vì vậy ROLLBACK sẽ không có tác dụng đối với những hệ thống loại này.
- ROLLBACK sẽ phục hồi tất cả những thay đổi kể từ lệnh COMMIT cuối cùng. Nghĩa là, thậm chí những thay đổi mà bạn không muốn phục hồi cũng sẽ được khôi phục lại với giá trị cũ nếu chưa một câu lệnh COMMIT nào được thực thi.



# Tóm tắt các câu lệnh không truy vấn DML của SQL

- Có thể thấy rằng, nhập dữ liệu bằng SQL rất rắc rối.
- Tốt nhất, các ứng dụng đầu cuối nên được nhập dữ liệu bằng các tiện ích có các giao diện hấp dẫn và dễ sử dụng. Ví dụ, MS Access xử lý việc nhập liệu tốt hơn nhiều so với SQL thuần tuý.



# Câu lệnh truy vấn DML trong SQL

- Câu lệnh truy vấn của DML bao gồm duy nhất một câu lệnh đơn được gọi là lệnh SELECT.
- Cú pháp của câu lệnh SELECT như sau:

```
SELECT [ ALL | DISTINCT] columnlist
      FROM tablelist
      [ WHERE condition ]
      [GROUP BY columnlist ]
      [HAVING condition ]
      [ORDER BY columnlist ];
```

- Chúng ta sẽ tìm hiểu hầu hết các đặc tính của câu lệnh SELECT, bắt đầu từ các câu truy vấn đơn giản tới những câu truy vấn phức tạp hơn, vẫn sử dụng cùng một cơ sở dữ liệu mà chúng ta đang xây dựng để minh họa.

# Truy vấn lựa chọn đơn giản trong SQL

- Có lẽ câu truy vấn đơn giản nhất là lấy ra tất cả các bản ghi trong một bảng nào đó.
- Ví dụ, ta cần hiển thị tất cả thuộc tính của toàn bộ bản ghi trong bảng PRODUCT. Nói cách khác, hiển thị bảng này. Câu lệnh sau đây cho phép thực hiện yêu cầu đó:

```
SELECT P_CODE, P_DESCRIPT, P_INDATE, P_ONHAND, P_MIN,  
       P_PRICE, P_DISCOUNT, V_CODE  
FROM PRODUCT;
```

-or-

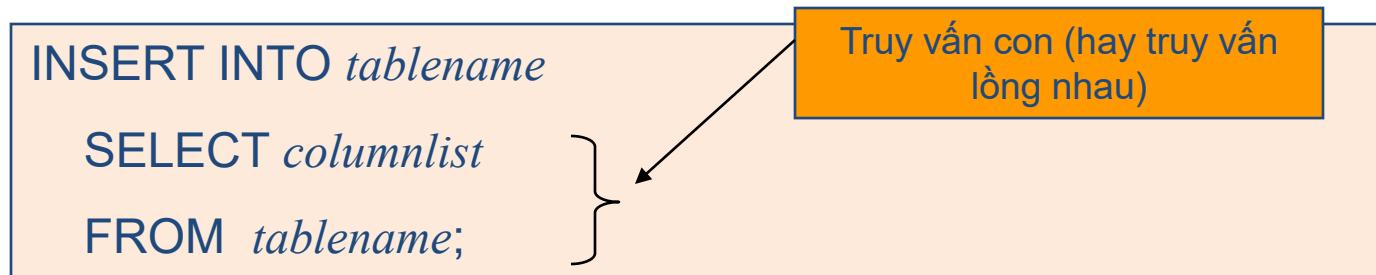
```
SELECT * ←  
FROM PRODUCT;
```

\* Là ký tự thể hiện tất cả các thuộc tính trong bảng



# Thêm các hàng vào bảng bằng câu truy vấn con Select

- Mặc dù lệnh INSERT là một thao tác dữ liệu không truy vấn, nhưng trong nó cũng có thể bao gồm một câu truy vấn. Ví dụ sau sẽ thể hiện điều đó.
- SQL cho phép thêm các hàng của một bảng từ dữ liệu lấy từ một bảng khác. Cú pháp thực hiện loại câu lệnh INSERT này như sau:



- Các câu truy vấn bên trong sẽ được thực hiện trước, kết quả của chúng sẽ được dùng như đầu vào cho câu lệnh bên ngoài sát với câu lệnh đó (ở loại câu lệnh đang xét thì câu lệnh SELECT là loại bên trong, và câu lệnh INSERT là câu lệnh ngoài). Các giá trị được trả về bởi câu lệnh bên trong phải phù hợp với các thuộc tính và kiểu dữ liệu của bảng trong câu lệnh INSERT.

# Các truy vấn lựa chọn có ràng buộc điều kiện

- Có thể chọn một phần nội dung của bảng bằng các giới hạn trên các bản ghi. Điều này được tiến hành với việc dùng mệnh đề WHERE như sau:

`SELECT columnlist`

`FROM tablelist`

`WHERE conditionlist ;`

- Lệnh SELECT sẽ lấy tất cả các hàng trong bảng được chọn thỏa mãn điều kiện được thể hiện ở mệnh đề WHERE..

– Ví dụ: `SELECT P_DESCRIPTOR, P_INDATE, P_PRICE, V_CODE`

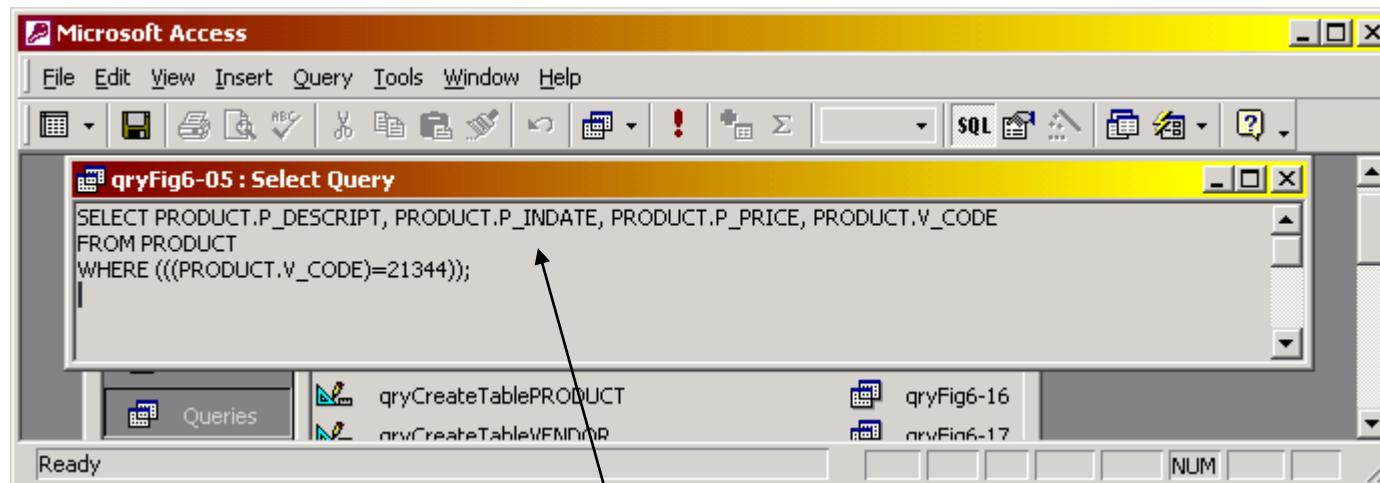
`FROM PRODUCT`

`WHERE V_CODE = 21344;`

	P_DESCRIPTOR	P_INDATE	P_PRICE	V_CODE
▶	7.25-in. pwr. saw blade	13-Dec-03	\$14.99	21344
	9.00-in. pwr. saw blade	13-Nov-03	\$17.49	21344
	Rat-tail file, 1/8-in. fine	15-Dec-03	\$4.99	21344
*			\$0.00	0

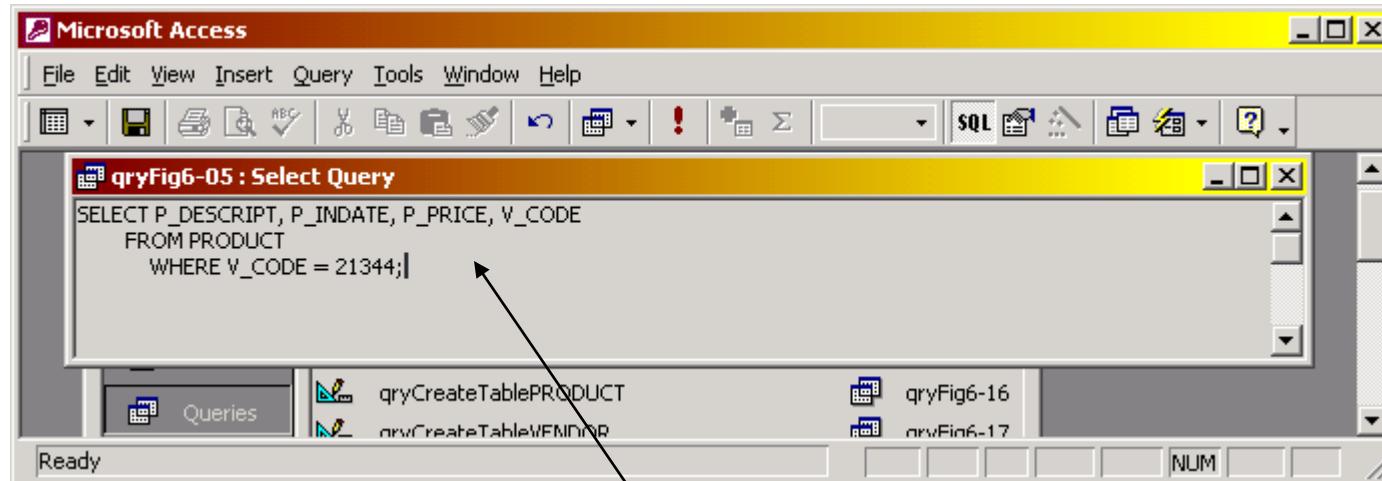
# Lưu ý về giao diện Access QBE cho SQL

- Microsoft Access cung cấp bộ tạo truy vấn Access QBE. Mặc dù Access QBE tạo ra các lệnh SQL của riêng nó, ta vẫn có thể sử dụng các lệnh chuẩn của SQL trong cửa sổ Access SQL như bảng sau.



Access QBE “native” SQL code for the query on the previous page.

# Lưu ý về giao diện Access QBE cho SQL



User generated SQL code for the same query.

A screenshot of Microsoft Access showing the results of the query. The table has four columns: P\_DESCRIP, P\_INDATE, P\_PRICE, and V\_CODE. The data is as follows:

P_DESCRIP	P_INDATE	P_PRICE	V_CODE
7.25-in. pwr. saw blade	13-Dec-03	\$14.99	21344
9.00-in. pwr. saw blade	13-Nov-03	\$17.49	21344
Rat-tail file, 1/8-in. fine	15-Dec-03	\$4.99	21344
*		\$0.00	0

Results of the user generated SQL code showing the same set of tuples as before in the result.

# Các ràng buộc điều kiện trong truy vấn SQL

- Cấu trúc câu lệnh SQL cung cấp không giới hạn sự linh hoạt của câu truy vấn. Có thể áp dụng nhiều điều kiện ràng buộc vào nội dung của bảng.
- Trừ việc kiểm tra giá trị các thuộc tính là NULL, SQL không trả về những hàng mà giá trị thuộc tính được lựa chọn là NULL trong kết quả.
- Xem xét câu truy vấn sau:

```
SELECT P_DESCRIPTOR, P_INDATE, P_PRICE, V_CODE  
      FROM PRODUCT  
     WHERE V_CODE <> 21344;
```

- Bảng PRODUCT được thể hiện trong hình vẽ dưới đây chứa kết quả của truy vấn trên, để ý là hàng thứ 10 và 13 trong bảng PRODUCT không xuất hiện trong các kết quả của truy vấn này.



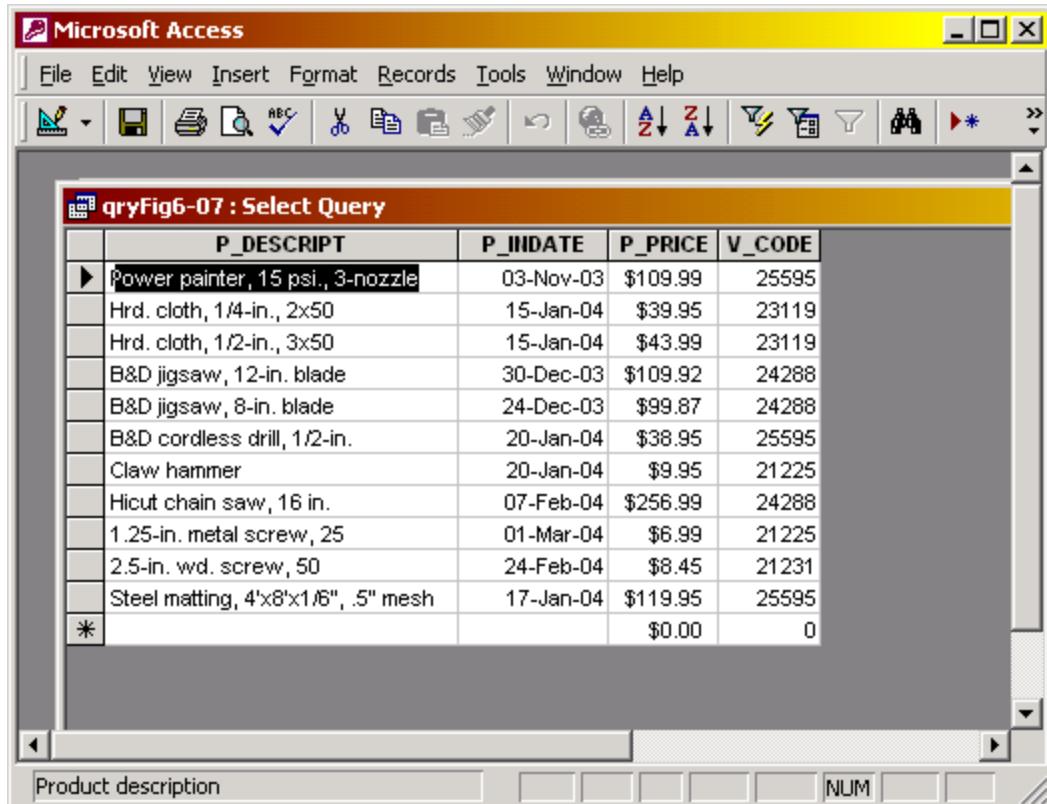
# Các ràng buộc điều kiện trong truy vấn SQL (cont.)

The screenshot shows the Microsoft Access application interface with the 'PRODUCT : Table' open. The table has the following columns: P\_CODE, P\_DESCRPT, P\_INDATE, P\_ONHAND, P\_MINI, P\_PRICE, P\_DISCOUNT, and V\_CODE. There are 18 rows of data. Two specific rows are highlighted with red arrows pointing to them from the right side of the slide:

	P_CODE	P_DESCRPT	P_INDATE	P_ONHAND	P_MINI	P_PRICE	P_DISCOUNT	V_CODE
*	11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	8	5	\$109.99	0.00	25595
*	13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	32	15	\$14.99	0.05	21344
*	14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	18	12	\$17.49	0.00	21344
*	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	15	8	\$39.95	0.00	23119
*	1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	23	5	\$43.99	0.00	23119
*	2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	8	5	\$109.92	0.05	24288
*	2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-03	6	5	\$99.87	0.05	24288
*	2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	12	5	\$38.95	0.05	25595
*	23109-HB	Claw hammer	20-Jan-04	23	10	\$9.95	0.10	21225
*	23114-AA	Sledge hammer, 12 lb.	02-Jan-04	8	5	\$14.40	0.05	
*	54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	43	20	\$4.99	0.00	21344
*	89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	11	5	\$256.99	0.05	24288
*	PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	188	75	\$5.87	0.00	
*	SM-18277	1.25-in. metal screw, 25	01-Mar-04	172	75	\$6.99	0.00	21225
*	SW-23116	2.5-in. wd. screw, 50	24-Feb-04	237	100	\$8.45	0.00	21231
*	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	18	5	\$119.95	0.10	25595
*				0	0	\$0.00	0.00	0

Hai hàng này sẽ không xuất hiện trong phần kết quả tại slide sau.

# Các ràng buộc điều kiện trong truy vấn SQL (cont.)



	P_DESCRPT	P_INDATE	P_PRICE	V_CODE
▶	Power painter, 15 psi., 3-nozzle	03-Nov-03	\$109.99	25595
	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	\$39.95	23119
	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	\$43.99	23119
	B&D jigsaw, 12-in. blade	30-Dec-03	\$109.92	24288
	B&D jigsaw, 8-in. blade	24-Dec-03	\$99.87	24288
	B&D cordless drill, 1/2-in.	20-Jan-04	\$38.95	25595
	Claw hammer	20-Jan-04	\$9.95	21225
	Hicut chain saw, 16 in.	07-Feb-04	\$256.99	24288
	1.25-in. metal screw, 25	01-Mar-04	\$6.99	21225
	2.5-in. wd. screw, 50	24-Feb-04	\$8.45	21231
	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	\$119.95	25595
*			\$0.00	0

Kết quả của câu truy vấn:

```
SELECT P_SDESCRPT,  
P_INDATE, P_PRICE,  
V_CODE  
FROM PRODUCT  
WHERE  
V_CODE <> 21344;
```

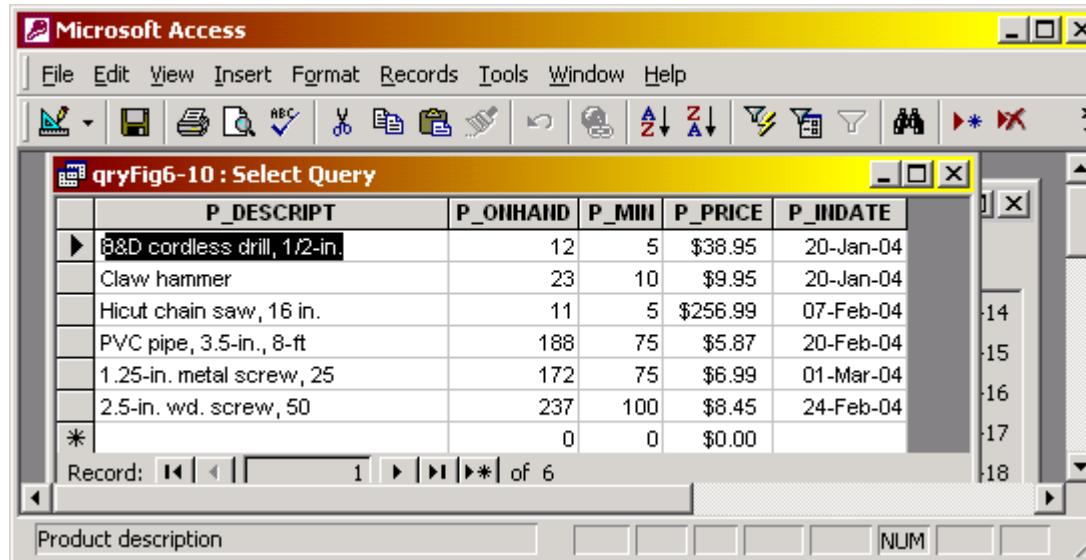
# So sánh về thời gian trong câu truy vấn SQL

- Các thủ tục về ngày tháng thường cụ thể cho từng phần mềm hơn các thủ tục SQL khác. Ví dụ, truy vấn để liệt kê tất cả các hàng trong đó ngày lưu kho từ 20/1/2004, sẽ như sau:

```
SELECT P_DESCRIP, P_ONHAND, P_MIN, P_PRICE, P_INDATE  
FROM PRODUCT  
WHERE P_INDATE >= "20-Jan-2004";
```

- Lưu ý: trong Access, ký tự phân biệt dùng cho ngày tháng là # vì vậy câu truy vấn trên trong Access sẽ như sau:

```
SELECT P_DESCRIP, P_ONHAND, P_MIN, P_PRICE, P_INDATE  
FROM PRODUCT  
WHERE P_INDATE >= #20-Jan-2004#;
```



The screenshot shows the Microsoft Access application interface with a query results grid. The title bar reads "qryFig6-10 : Select Query". The menu bar includes File, Edit, View, Insert, Format, Records, Tools, Window, Help. The toolbar contains various icons for database management. The results grid displays six rows of data with columns: P\_DESCRIP, P\_ONHAND, P\_MIN, P\_PRICE, and P\_INDATE. The data is as follows:

P_DESCRIP	P_ONHAND	P_MIN	P_PRICE	P_INDATE
B&D cordless drill, 1/2-in.	12	5	\$38.95	20-Jan-04
Claw hammer	23	10	\$9.95	20-Jan-04
Hicut chain saw, 16 in.	11	5	\$256.99	07-Feb-04
PVC pipe, 3.5-in., 8-ft	188	75	\$5.87	20-Feb-04
1.25-in. metal screw, 25	172	75	\$6.99	01-Mar-04
2.5-in. wd. screw, 50	237	100	\$8.45	24-Feb-04
*	0	0	\$0.00	

# Sử dụng các cột tính toán và đổi tên các cột

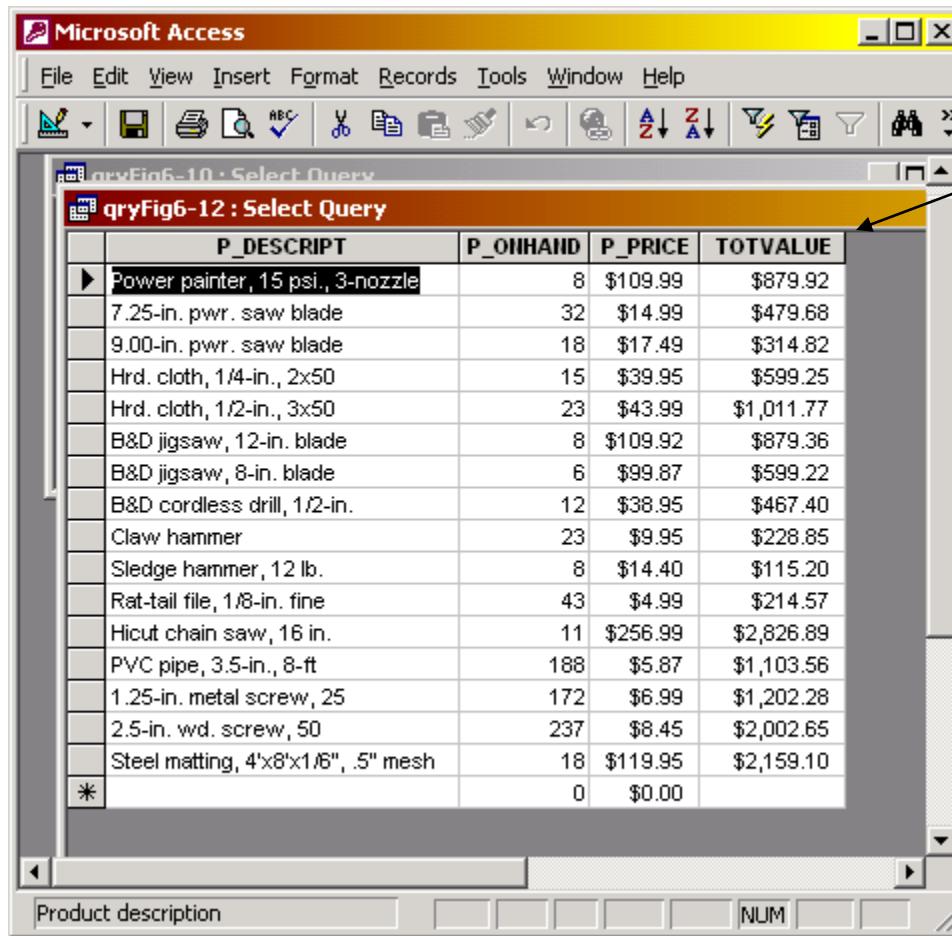
- Giả sử câu truy vấn cần xác định một giá trị không được lưu trữ vật lý mà được tính toán từ các thông số trong CSDL.
- Ví dụ, cần tính tổng giá trị các mặt hàng hiện có trong kho. Về logic, giá trị này bằng tích của số lượng hàng nhân với đơn giá của chúng. Câu truy vấn SQL cho yêu cầu này được mô tả như sau (kết quả được thể hiện trong slide kế tiếp).

```
SELECT P_DESCRIP, P_ONHAND, P_PRICE, P_ONHAND * P_PRICE AS TOTVALUE  
      FROM PRODUCT
```

SQL chấp nhận mọi biểu thức hợp lệ trong các cột tính toán, sử dụng các thuộc tính trong bảng được xác định trong mệnh đề FROM. Lưu ý, Access tự động thêm nhãn Expr vào các cột tính toán. Oracle sử dụng biểu thức thực tế làm nhãn cho các cột này.

SQL thông thường cho phép sử dụng các tên khác nhau cho bất kỳ cột nào trong mệnh đề SELECT. Tên phát sinh cho các cột được xuất hiện sau từ AS.

## Sử dụng các cột tính toán và đổi tên các cột (cont.)



Cột được tính toán kèm tên riêng của nó.

P_DESCRPT	P_OHHAND	P_PRICE	TOTVALUE
Power painter, 15 psi., 3-nozzle	8	\$109.99	\$879.92
7.25-in. pwr. saw blade	32	\$14.99	\$479.68
9.00-in. pwr. saw blade	18	\$17.49	\$314.82
Hrd. cloth, 1/4-in., 2x50	15	\$39.95	\$599.25
Hrd. cloth, 1/2-in., 3x50	23	\$43.99	\$1,011.77
B&D jigsaw, 12-in. blade	8	\$109.92	\$879.36
B&D jigsaw, 8-in. blade	6	\$99.87	\$599.22
B&D cordless drill, 1/2-in.	12	\$38.95	\$467.40
Claw hammer	23	\$9.95	\$228.85
Sledge hammer, 12 lb.	8	\$14.40	\$115.20
Rat-tail file, 1/8-in. fine	43	\$4.99	\$214.57
Hicut chain saw, 16 in.	11	\$256.99	\$2,826.89
PVC pipe, 3.5-in., 8-ft	188	\$5.87	\$1,103.56
1.25-in. metal screw, 25	172	\$6.99	\$1,202.28
2.5-in. wd. screw, 50	237	\$8.45	\$2,002.65
Steel matting, 4'x8'x1/6", .5" mesh	18	\$119.95	\$2,159.10
*	0	\$0.00	

# Áp dụng cột tính toán cùng tên riêng và ngày dạng số trong câu truy vấn đơn lẻ

- Giả sử ta cần lập danh sách các sản phẩm hết bảo hành (nằm trong kho hơn 90 ngày). Khi đó P\_INDATE cách ngày hiện tại ít nhất 90 ngày. Câu truy vấn trong Access và Oracle được biểu diễn như sau (kết quả nằm ở slide kế tiếp).

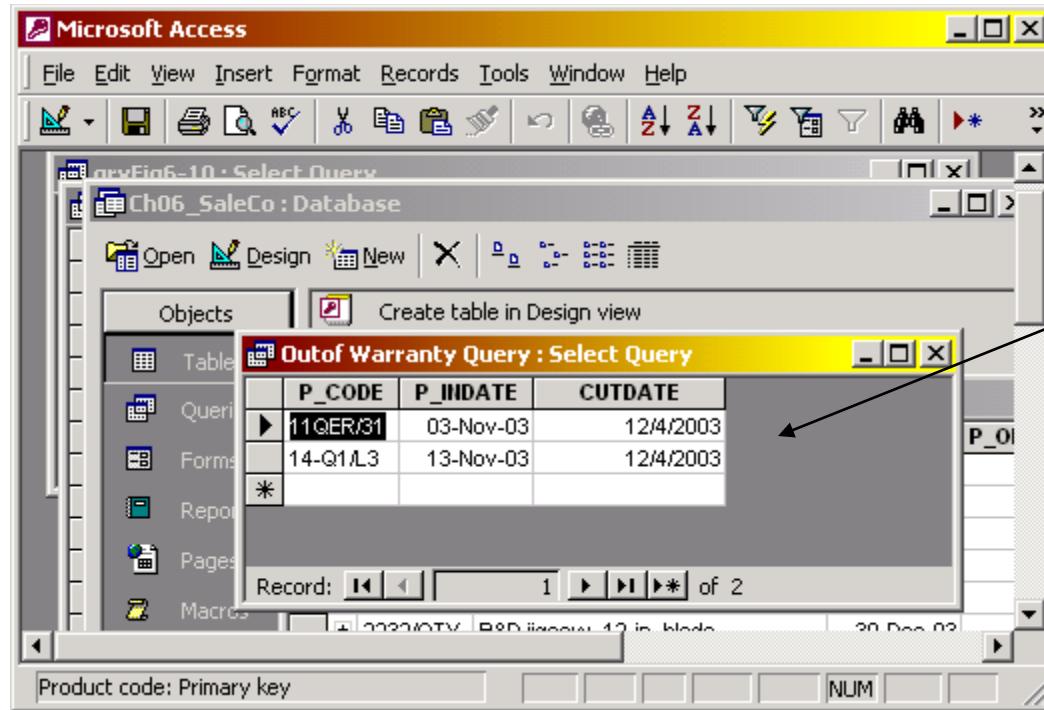
## Access Version

```
SELECT P_CODE, P_INDATE, DATE() – 90 AS CUTDATE
      FROM PRODUCT
     WHERE P_INDATE <= DATE() – 90;
```

## Oracle Version

```
SELECT P_CODE, P_INDATE, SYSDATE – 90 AS CUTDATE
      FROM PRODUCT
     WHERE P_INDATE <= SYSDATE – 90;
```

# Áp dụng cột tính toán cùng tên riêng và ngày dạng số trong câu truy vấn đơn lẻ



Có 2 sản phẩm quá  
hạn bảo hành dựa trên  
ngày nhập hàng vào  
kho trong bảng  
PRODUCTS

# Sử dụng các phép toán logic AND, OR, và NOT

- Trong thực tế, quá trình tìm kiếm dữ liệu thường bao gồm nhiều điều kiện. SQL cho phép gộp nhiều điều kiện vào một câu truy vấn bằng các phép toán logic.
- SQL hỗ trợ các phép toán logic: AND, OR, and NOT.
- Ví dụ, cần liệt kê các phần tử của bảng PRODUCTS thỏa mãn V\_CODE = 21344 hoặc V\_CODE = 24288. Câu lệnh SQL được biểu diễn như sau:

```
SELECT P_DESCRIP,  
       P_INDATE,  
       P_PRICE,  
       V_CODE  
FROM PRODUCT  
WHERE  
       V_CODE = 21344  
   OR  
       V_CODE = 24288;
```

The screenshot shows the Microsoft Access application interface with a query results window titled "qryFig6-13 : Select Query". The results grid displays the following data:

	P_DESCRIP	P_INDATE	P_PRICE	V_CODE
▶	7.25-in. pwr. saw blade	13-Dec-03	\$14.99	21344
	9.00-in. pwr. saw blade	13-Nov-03	\$17.49	21344
	B&D jigsaw, 12-in. blade	30-Dec-03	\$109.92	24288
	B&D jigsaw, 8-in. blade	24-Dec-03	\$99.87	24288
	Rat-tail file, 1/8-in. fine	15-Dec-03	\$4.99	21344
*	Hicut chain saw, 16 in.	07-Feb-04	\$256.99	24288
			\$0.00	0

# Các toán tử đặc biệt trong SQL

- SQL theo chuẩn ANSI cho phép sử dụng các toán tử đặc biệt trong biểu thức nằm trong mệnh đề WHERE. Các toán tử này bao gồm:

**BETWEEN** – được dùng để kiểm tra xem giá trị một thuộc tính có nằm trong một khoảng nào đó không.

**IS NULL** – được dùng để xác định liệu một thuộc tính có nhận giá trị NULL không.

**LIKE** – được dùng để gắn giá trị một thuộc tính với một kiểu chuỗi ký tự. Nhiều ký tự thay thế được sẵn có để sử dụng với toán tử này.

**IN** – được sử dụng để xác định liệu giá trị một thuộc tính có nằm trong một danh sách giá trị không.

**EXISTS** – được dùng để xác định liệu một truy vấn con có trả về một tập rỗng hay không.



# Toán tử đặc biệt BETWEEN

- Giả sử ta cần liệt kê các sản phẩm có giá từ \$50 đến \$100. Toán tử BETWEEN có thể được sử dụng như sau:

```
SELECT all  
      FROM PRODUCT  
     WHERE P_PRICE BETWEEN 50.00 AND 100.00;
```

- Nếu hệ CSDL quan hệ không hỗ trợ hàm BETWEEN câu truy vấn có thể được thực hiện như sau:

```
SELECT all  
      FROM PRODUCT  
     WHERE P_PRICE > 50.00 AND P_PRICE < 100.00;
```



# Toán tử đặc biệt IS NULL

- Giả sử ta cần liệt kê các sản phẩm chưa có thông tin về nơi bán, V\_CODE = null. Thông tin trống có thể được diễn tả như sau:

```
SELECT P_CODE, P_DESCRIP, V_CODE  
      FROM PRODUCT  
 WHERE V_CODE IS NULL;
```

- Lưu ý: SQL sử dụng câu lệnh kiểm tra đặc biệt cho các giá trị trống. Ta **không thể** thực hiện lệnh so sánh V\_CODE = NULL. Lý do cơ bản vì NULL không phải là một “giá trị” mà là một diễn tả đặc biệt thể hiện rằng không có giá trị nào được nhập cho một thuộc tính nào đó.
- Toán tử IS NOT NULL cũng có thể được sử dụng với nghĩa ngược lại.

# Toán tử đặc biệt LIKE

- Toán tử LIKE được sử dụng kết hợp với một hàm đặc biệt để tìm các khối ký tự trong các thuộc tính văn bản.
- SQL chuẩn cho phép sử dụng các ký tự đặc biệt "%" và "\_" để tìm kiếm tương tự cho các xâu ký tự.

“%” được dùng đại diện cho một chuỗi ký tự.

‘M%’ có thể trả về: Mark, Marci, M-234x, v.v.

“\_” được dùng đại diện cho một ký tự.

‘\_07-345-887\_’ trả về: 4**07-345-8871**, **007-345-8875**

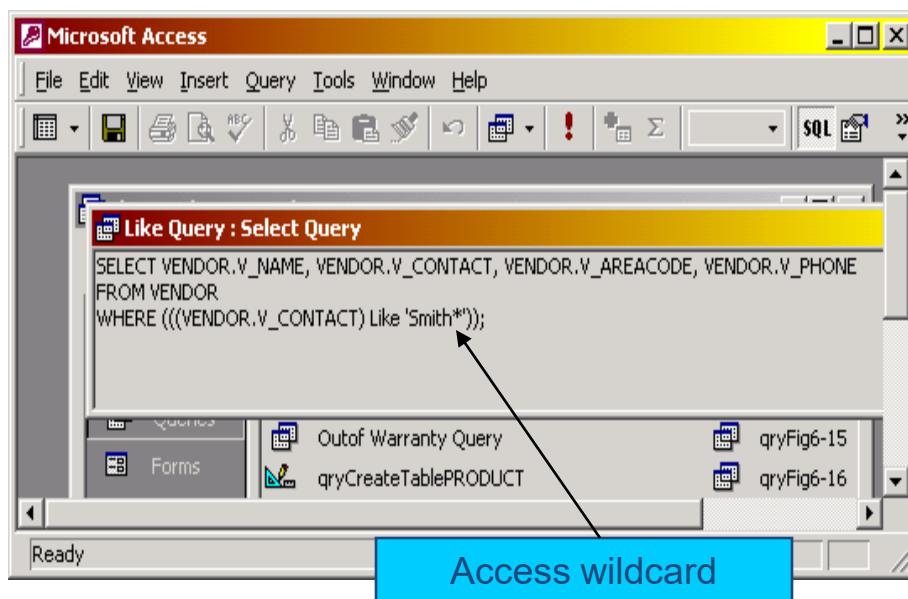
- Lưu ý: trong Access “\*” thay cho “%”, và “?” thay cho “\_”. Oracle phân biệt chữ hoa và chữ thường, Access thì không.



# Toán tử đặc biệt LIKE (cont.)

- Giả sử ta cần tìm các nhà cung cấp có tên giao dịch bắt đầu là Smith.

```
SELECT V_NAME, V_CONTACT, V_AREACODE, V_PHONE  
FROM VENDOR  
WHERE V_CONTACT LIKE 'Smith%';
```



The screenshot shows the Microsoft Access interface with the title bar 'Microsoft Access'. The menu bar includes File, Edit, View, Insert, Format, Records, Tools, Window, Help. The toolbar has various icons for file operations like Open, Save, Print, and a SQL icon. The main window displays the results of the query in Datasheet View. The table has four columns: V\_NAME, V\_CONTACT, V\_AREACODE, and V\_PHONE. The data is as follows:

V_NAME	V_CONTACT	V_AREACODE	V_PHONE
Bryson, Inc.	Smithson	615	223-3234
Dome Supply	Smith	901	678-1419
B&K, Inc.	Smith	904	227-0093
*			

Record: 1 of 3

# Toán tử đặc biệt IN

- Các câu truy vấn có sử dụng toán tử logic OR có thể được thay thế bằng toán tử đặc biệt IN.
- Ví dụ, với câu truy vấn sau:

```
SELECT *  
FROM PRODUCT  
WHERE V_CODE = 21344 OR V_CODE = 24288;
```

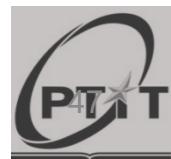
Ta có thể thay thế bằng:

```
SELECT *  
FROM PRODUCT  
WHERE V_CODE IN (21344, 24288);
```

# Toán tử đặc biệt IN (cont.)

- Toán tử IN rất hữu dụng khi được sử dụng kết hợp với câu truy vấn phụ.
- Ví dụ, ta cần liệt kê V\_CODE và V\_NAME của chỉ các nhà cung cấp có sản phẩm. Khi đó, câu truy vấn phụ trong câu truy vấn sử dụng IN sẽ tự động tạo ra danh sách. Toàn bộ câu truy vấn như sau:

```
SELECT V_CODE, V_NAME  
      FROM VENDOR  
 WHERE V_CODE IN ( SELECT V_CODE  
                      FROM PRODUCT);
```



# Toán tử đặc biệt EXISTS

- Toán tử EXISTS được sử dụng khi có một câu lệnh được thực hiện dựa trên kết quả của một lệnh khác. Cụ thể, nếu một câu truy vấn phụ có trả về kết quả thì câu truy vấn chính mới chạy, không thì thôi.
- Cú pháp của toán tử EXISTS là

WHERE EXISTS ( subquery );

- Câu truy vấn phụ subquery là một câu lệnh SELECT nào đó.
- Ta cũng có thể sử dụng toán tử NOT EXISTS với nghĩa ngược lại.
- Lưu ý: Cần hạn chế sử dụng câu lệnh SQL có sử dụng toán tử EXISTS vì câu truy vấn phụ luôn phải chạy lại sau mỗi dòng của câu truy vấn chính.



# Sao chép từng phần của bảng

- Mặc dù ta luôn phải thiết kế cẩn thận CSDL trước khi tiến hành xây dựng, vẫn có nhiều trường hợp cần phải phá bỏ cấu trúc của một bảng thành nhiều phần nhỏ (các bảng nhỏ hơn).
- SQL cho phép sao chép nội dung của các cột nào đó trong bảng để đỡ phải nhập lại một cách thủ công vào bảng mới.
- Ví dụ, ta cần sao chép các cột P\_CODE, P\_DESCRIPT, và P\_PRICE từ bảng PRODUCT sang một bảng mới có tên là PART.
  - Trước hết, ta tạo bảng mới PART như trong slide kế tiếp.

# Sao chép từng phần của bảng (cont.)

```
CREATE TABLE PART (
    PART_CODE          CHAR(8) NOT NULL UNIQUE,
    PART_DESCRPT      CHAR(35),
    PART_PRICE         DECIMAL(8,2),
    PRIMARY KEY (PART_CODE) );
```

- Lưu ý rằng tên và số lượng của các cột trong bảng PART không nhất thiết phải giống với bảng cũ.
  - Trong trường hợp này, cột đầu tiên của bảng PART là PART\_CODE, chứ không phải là P\_CODE như trong bảng PRODUCT. Đồng thời, bảng PART chỉ có 3 cột, chứ không phải là 7 cột như trong bảng PRODUCT.
  - Tuy nhiên, các đặc tính của các cột cần phải giống nhau: ta không thể sao chép dữ liệu kiểu chữ (character) vào cột có dữ liệu kiểu số (numeric) và ngược lại.



# Sao chép từng phần của bảng (cont.)

- Tiếp theo, ta cần phải thêm các bộ dữ liệu cho bảng PART từ bảng PRODUCT. Câu lệnh INSERT có thể được sử dụng.
- Như đã giới thiệu từ trước, cú pháp của câu lệnh này là:

```
INSERT INTO target tablename [(target columnlist)]

  SELECT source columnlist

    FROM source tablename;
```

- Cần phải nhập target-columnlist nếu như các cột từ bảng nguồn không có tên cũng như đặc tính trùng với bảng đích (kể cả thứ tự các cột!). Còn không thì ta không cần nhập target-columnlist.
  - Trong ví dụ đã nêu, ta cần phải nhập target-columnlist vì tên của một hoặc nhiều cột đã bị thay đổi.



# Sao chép từng phần của bảng (cont.)

- Để kích hoạt các thay đổi như mong muốn, ta cần phải có câu lệnh INSERT như sau:

```
INSERT INTO PART (PART_CODE, PART_DESCRIP, PART_PRICE)  
SELECT P_CODE, P_DESCRIP, P_PRICE  
FROM PRODUCT;
```

- Nội dung mới cập nhật của bảng PARTS có thể được hiện lên bằng lệnh:

```
SELECT *  
FROM PART;
```

- Kết quả được trình bày trong slide kế tiếp.

# Sao chép từng phần của bảng (cont.)

The screenshot shows a Microsoft Access window titled "qryFig6-19 : Select Query". The window displays a table with three columns: PART\_CODE, PART\_DESCRIP, and PART\_PRICE. The data consists of 16 rows of product information. The PART\_CODE column contains codes like "11QER/31", "13-Q2/P2", etc. The PART\_DESCRIP column contains descriptions like "Power painter, 15 psi., 3-nozzle", "7.25-in. pwr. saw blade", etc. The PART\_PRICE column contains prices like "\$109.99", "\$14.99", etc. At the bottom of the table, there is a row with an asterisk (\*) and empty cells. Below the table, the status bar shows "Datasheet View".

PART_CODE	PART_DESCRIP	PART_PRICE
11QER/31	Power painter, 15 psi., 3-nozzle	\$109.99
13-Q2/P2	7.25-in. pwr. saw blade	\$14.99
14-Q1/L3	9.00-in. pwr. saw blade	\$17.49
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	\$39.95
1558-QW1	Hrd. cloth, 1/2-in., 3x50	\$43.99
2232/QTY	B&D jigsaw, 12-in. blade	\$109.92
2232/QWE	B&D jigsaw, 8-in. blade	\$99.87
2238/QPD	B&D cordless drill, 1/2-in.	\$38.95
23109-HB	Claw hammer	\$9.95
23114-AA	Sledge hammer, 12 lb.	\$14.40
54778-2T	Rat-tail file, 1/8-in. fine	\$4.99
89-WRE-Q	Hicut chain saw, 16 in.	\$256.99
PVC23DRT	PVC pipe, 3.5-in., 8-ft	\$5.87
SM-18277	1.25-in. metal screw, 25	\$6.99
SW-23116	2.5-in. wd. screw, 50	\$8.45
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	\$119.95
*		

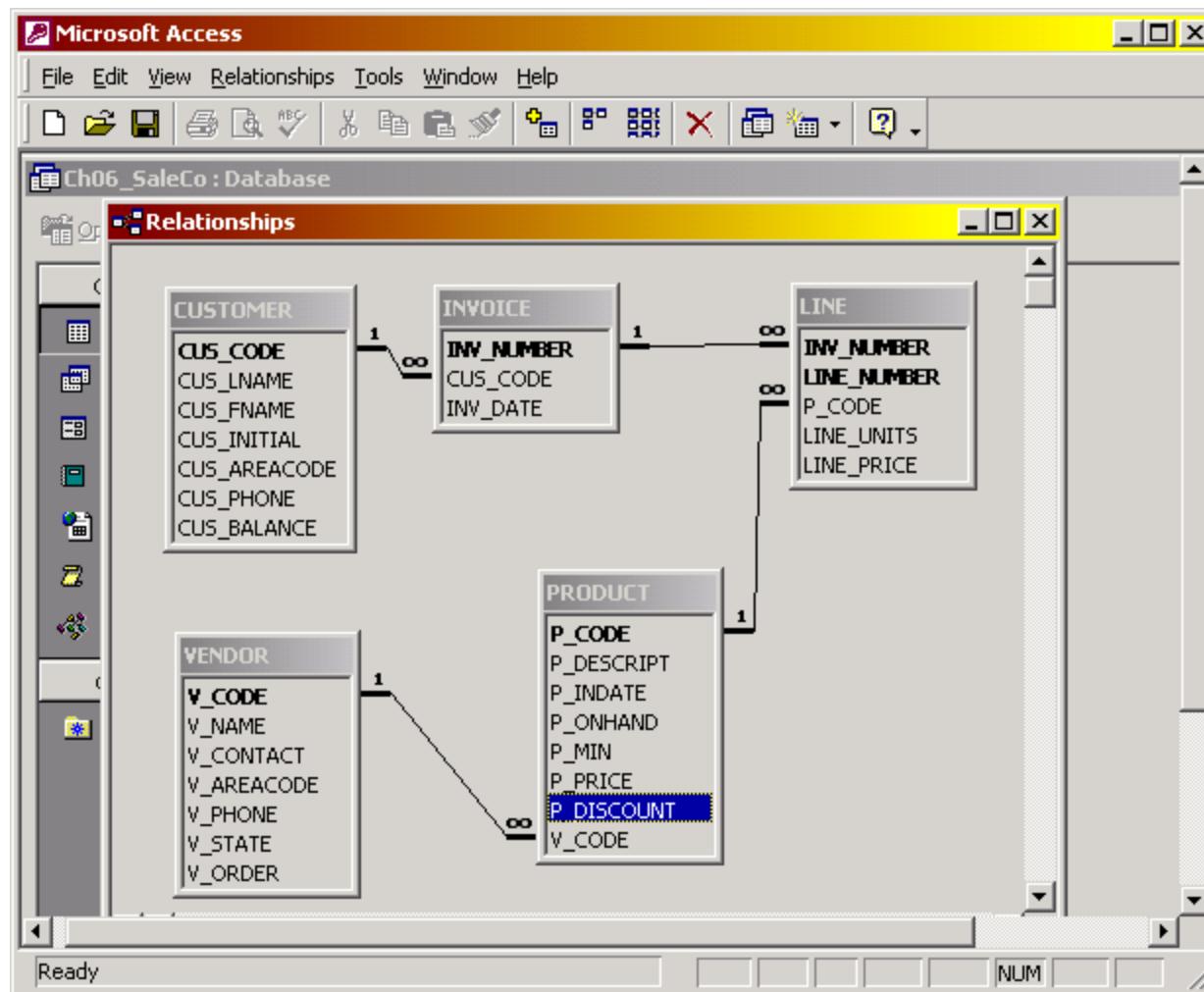
Kết quả của việc sao chép một phần của bảng cùng với thay đổi tên cột.

# Giới thiệu ngôn ngữ SQL – phần 3

Posts and Telecommunications Institute of Technology-PTIT



# Ví dụ về CSDL



# Truy vấn nâng cao SELECT

- Một ưu điểm quan trọng của SQL là khả năng tạo các câu truy vấn phức tạp ở dạng tự do.
- Các phép toán logic đề cập trong bài trước có thể được áp dụng trong các câu truy vấn này.
- Ngoài ra, SQL cung cấp các hàm quan trọng như đếm, tìm lớn nhất, nhỏ nhất, tính trung bình, v.v.
- Hơn nữa, SQL cho phép giới hạn câu truy vấn vào các bản ghi không trùng nhau, hoặc nhóm các bản ghi giống nhau vào một.
- Các đặc tính này sẽ được tìm hiểu sâu hơn trong các phần tiếp theo.



# Sắp xếp danh sách

- Về câu ORDER BY rất hữu dụng trong việc sắp xếp danh sách theo yêu cầu nào đó
- Cú pháp:

```
SELECT columnlist
      FROM tablelist
        [ WHERE conditionlist ]
        [ORDER BY columnlist [ASC | DESC]] ;
```

- Nếu cột được sắp xếp bao gồm giá trị rỗng thì các giá trị này được cho lên đầu hoặc xuống cuối tùy thuộc vào các hệ CSDL quan hệ cụ thể.
- Về câu ORDER BY luôn nằm cuối cùng trong tập lệnh SELECT.
- Ta có thể chọn cụ thể kiểu sắp xếp (tăng dần hoặc giảm dần). Nếu không chọn, mặc định của hệ thống sẽ là tăng dần.



# Sắp xếp danh sách (cont.)

- Câu truy vấn sau liệt kê các bản ghi của bảng PRODUCT được sắp xếp theo thứ tự tăng dần của P\_PRICE:

```
SELECT  
    P_CODE, P_DESCRPT,  
    P_INDATE, P_PRICE  
FROM PRODUCT  
ORDER BY P_PRICE;
```

	P_CODE	P_DESCRPT	P_INDATE	P_PRICE
▶	54778-21	Rat-tail file, 1/8-in. fine	15-Dec-03	\$4.99
	PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	\$5.87
	SM-18277	1.25-in. metal screw, 25	01-Mar-04	\$6.99
	SW-23116	2.5-in. wd. screw, 50	24-Feb-04	\$8.45
	23109-HB	Claw hammer	20-Jan-04	\$9.95
	23114-AA	Sledge hammer, 12 lb.	02-Jan-04	\$14.40
	13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	\$14.99
	14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	\$17.49
	2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	\$38.95
	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	\$39.95
	1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	\$43.99
	2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-03	\$99.87
	2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	\$109.92
	11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	\$109.99
	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	\$119.95
*	89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	\$256.99

# Sắp xếp danh sách (cont.)

- Câu truy vấn sau liệt kê các bản ghi của bảng PRODUCT được sắp xếp theo thứ tự giảm dần của P\_PRICE :

```
SELECT  
    P_CODE, P_DESCRPT,  
    P_INDATE, P_PRICE  
FROM PRODUCT  
ORDER BY P_PRICE DESC;
```

	P_CODE	P_DESCRPT	P_INDATE	P_PRICE
▶	39-WRE-Q	Hicut chain saw, 16 in.	07-Feb-04	\$256.99
	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-04	\$119.95
	11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-03	\$109.99
	2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-03	\$109.92
	2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-03	\$99.87
	1558-Q/W1	Hrd. cloth, 1/2-in., 3x50	15-Jan-04	\$43.99
	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-04	\$39.95
	2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-04	\$38.95
	14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-03	\$17.49
	13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-03	\$14.99
	23114-A.A	Sledge hammer, 12 lb.	02-Jan-04	\$14.40
	23109-HB	Claw hammer	20-Jan-04	\$9.95
	SW-23116	2.5-in. wd. screw, 50	24-Feb-04	\$8.45
	SM-18277	1.25-in. metal screw, 25	01-Mar-04	\$6.99
	PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-04	\$5.87
*	54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-03	\$4.99
				\$0.00

# Sắp xếp theo các cấp

- Sắp xếp danh sách là yêu cầu thường gặp. Ví dụ, ta cần tạo một thư mục số điện thoại của nhân viên. Yêu cầu có thể là sắp xếp danh sách theo từng cấp như sau:
  1. Sắp xếp theo họ.
  2. Trong thứ tự sắp xếp đó, sắp xếp theo tên riêng.
  3. Trong thứ tự sắp xếp ở bước 2, sắp xếp theo tên đệm.
- Chuỗi sắp xếp nhiều cấp có thể được tạo ra bởi danh mục các thuộc tính khác nhau, phân tách bởi dấu phẩy sử dụng vế câu ORDER BY.
- Cách thức cụ thể sẽ được minh họa trong các phần tiếp theo.



# Sắp xếp theo các cấp (cont.)

Microsoft Access

EMPLOYEE : Table

	EMP_NUM	EMP_TITLE	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_DOB	EMP_HIRE_DATE	EMP_YEARS	EMP_AREACODE	EMP_PHONE
▶	100	Mr.	Kolmycz	George	D	15-Jun-42	15-Mar-85	18	615	324-5456
	101	Ms.	Lewis	Rhonda	G	19-Mar-65	25-Apr-86	16	615	324-4472
	102	Mr.	Vandam	Rhett		14-Nov-58	20-Dec-90	12	901	675-8993
	103	Ms.	Jones	Anne	M	16-Oct-74	28-Aug-94	8	615	898-3456
	104	Mr.	Lange	John	P	08-Nov-71	20-Oct-94	8	901	504-4430
	105	Mr.	Williams	Robert	D	14-Mar-75	08-Nov-98	4	615	890-3220
	106	Mrs.	Smith	Jeanine	K	12-Feb-68	05-Jan-89	14	615	324-7883
	107	Mr.	Diante	Jorge	D	21-Aug-74	02-Jul-94	8	615	890-4567
	108	Mr.	Wiesenbach	Paul	R	14-Feb-66	18-Nov-92	10	615	897-4358
	109	Mr.	Smith	George	K	18-Jun-61	14-Apr-89	13	901	504-3339
	110	Mrs.	Genkazi	Leighla	W	19-May-70	01-Dec-90	12	901	569-0093
	111	Mr.	Washington	Rupert	E	03-Jan-66	21-Jun-93	9	615	890-4925
	112	Mr.	Johnson	Edward	E	14-May-61	01-Dec-83	19	615	898-4387
	113	Ms.	Smythe	Melanie	P	15-Sep-70	11-May-99	3	615	324-9006
	114	Ms.	Brandon	Marie	G	02-Nov-56	15-Nov-79	23	901	882-0845
	115	Mrs.	Saranda	Hermine	R	25-Jul-72	23-Apr-93	9	615	324-5505
	116	Mr.	Smith	George	A	08-Nov-65	10-Dec-88	14	615	890-2984
*	0							0		

Record: [◀] [◀] [▶] [▶] [▶] \* of 17

Employee number. (Primary key)

Bảng Employee

# Sắp xếp theo các cấp (cont.)

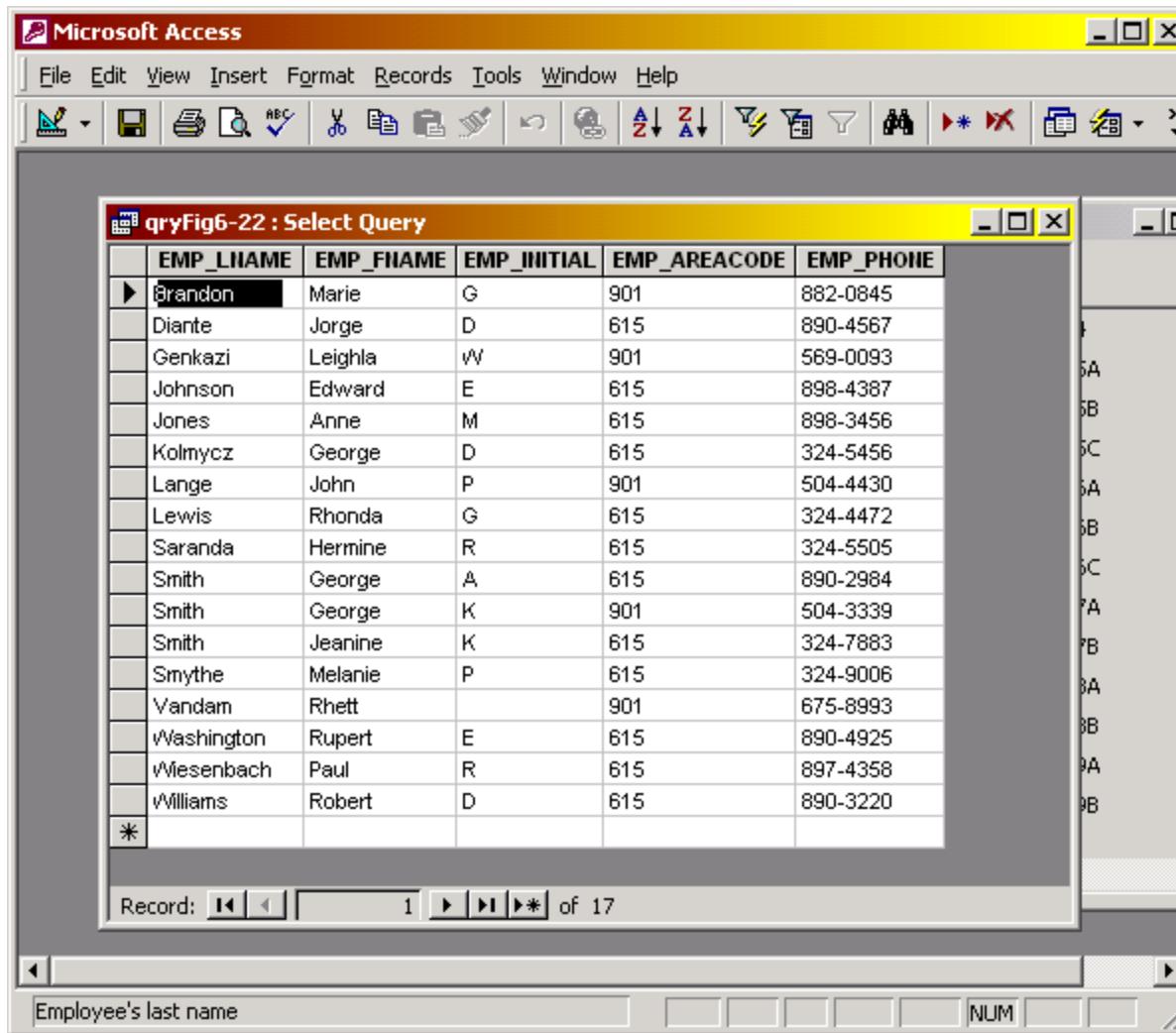
- Để tạo danh mục có sắp xếp từ bảng EMPLOYEE, ta dùng câu truy vấn SQL sau:

```
SELECT EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_AREACODE, EMP_PHONE  
FROM EMPLOYEE  
ORDER BY EMP_LNAME, EMP_FNAME, EMP_INITIAL;
```

- Kết quả câu truy vấn này được minh họa trong slide kế tiếp.



# Sắp xếp theo các cấp (cont.)



The screenshot shows the Microsoft Access application interface with a query results grid. The title bar reads "qryFig6-22 : Select Query". The grid displays 17 records from the Employee table, ordered by LastName, FirstName, MiddleInitial. The columns are labeled: EMP\_LNAME, EMP\_FNAME, EMP\_INITIAL, EMP\_AREACODE, and EMP\_PHONE. The records are as follows:

	EMP_LNAME	EMP_FNAME	EMP_INITIAL	EMP_AREACODE	EMP_PHONE
►	Brandon	Marie	G	901	882-0845
	Dianite	Jorge	D	615	890-4567
	Genkazi	Leighla	W	901	569-0093
	Johnson	Edward	E	615	898-4387
	Jones	Anne	M	615	898-3456
	Kolmycz	George	D	615	324-5456
	Lange	John	P	901	504-4430
	Lewis	Rhonda	G	615	324-4472
	Saranda	Hermine	R	615	324-5505
	Smith	George	A	615	890-2984
	Smith	George	K	901	504-3339
	Smith	Jeanine	K	615	324-7883
	Smythe	Melanie	P	615	324-9006
	Vandam	Rhett		901	675-8993
	Washington	Rupert	E	615	890-4925
	Wiesenbach	Paul	R	615	897-4358
	Williams	Robert	D	615	890-3220
*					

Record: [navigation buttons] 1 [next button] \* of 17

Employee's last name

Bảng Employee – Sắp xếp theo LastName, FirstName, MiddleInitial

# Một số ứng dụng phụ của ORDER BY

- Về câu ORDER BY có thể được sử dụng kết hợp với các lệnh SQL khác.
- Ví dụ, chú ý việc sử dụng các điều kiện ràng buộc trên ngày và giá trong chuỗi truy vấn sau:

```
SELECT P_DESCRIP, V_CODE, P_INDATE, P_PRICE  
      FROM PRODUCT  
     WHERE P_INDATE < '21-Jan-2004' AND P_PRICE <= 50.00  
          ORDER BY V_CODE, P_PRICE DESC;
```

- Kết quả nằm ở slide kế tiếp:



# Một số ứng dụng phụ của ORDER BY (cont.)

The screenshot shows a Microsoft Access application window titled "qryFig6-23 : Select Query". The window displays a table of product data with four columns: P\_DESCRPT, V\_CODE, P\_INDATE, and P\_PRICE. The data is ordered by P\_INDATE in descending order. The records listed are:

P_DESCRPT	V_CODE	P_INDATE	P_PRICE
Sledge hammer, 12 lb.		02-Jan-04	\$14.40
Claw hammer	21225	20-Jan-04	\$9.95
9.00-in. pwr. saw blade	21344	13-Nov-03	\$17.49
7.25-in. pwr. saw blade	21344	13-Dec-03	\$14.99
Rat-tail file, 1/8-in. fine	21344	15-Dec-03	\$4.99
Hrd. cloth, 1/2-in., 3x50	23119	15-Jan-04	\$43.99
Hrd. cloth, 1/4-in., 2x50	23119	15-Jan-04	\$39.95
B&D cordless drill, 1/2-in.	25595	20-Jan-04	\$38.95
*	0		\$0.00

Record: [navigation buttons] 1 [next button] \* of 8

Product description

# Liệt kê các giá trị duy nhất

- Có bao nhiêu nhà cung cấp có tên trong bảng PRODUCT? Câu lệnh liệt kê đơn giản dùng SELECT không hữu hiệu cho câu truy vấn này, đặc biệt nếu trong bảng có hàng nghìn bộ dữ liệu.
- Vẽ câu lệnh DISTINCT của SQL cho phép tạo danh mục các giá trị duy nhất từ một danh sách đã cho.
- Ví dụ, câu lệnh sau:

```
SELECT DISTINCT V_CODE  
FROM PRODUCT;
```

sẽ liệt kê các mã V\_CODE khác nhau có trong bảng PRODUCT.

V_CODE
21225
21231
21344
23119
24288
25595

Trong Oracle, V\_CODE có giá trị rỗng sẽ nằm ở cuối, con trong Access thì sẽ nằm ở đầu. Ta cũng có thể dùng vé ORDER BY để sắp xếp.

# Các hàm trong SQL

Trong quá trình truy vấn dữ liệu có thể sử dụng một số hàm trên các cột dữ liệu với cú pháp sau:

**function\_name(cột, [tham số 1, tham số 2, ...])**

- Các hàm thao tác trên từng bản ghi:

- LOWER(A) – chuyển đổi ký tự thành kiểu chữ thường
- UPPER(a) – chuyển đổi ký tự thành kiểu chữ in hoa

Ví dụ: **SELECT UPPER(*LastName*), LOWER(*FirstName*) FROM student ;**

- ROUND(a) – làm tròn số
- PI() – lấy giá trị pi
- SQRT(a) – căn bậc hai
- POWER(a,b) – hàm mũ



# Các hàm trong SQL

- Các hàm thao tác trên từng bản ghi:

- Hàm chuyển đổi kiểu dữ liệu CONVERT

Ví dụ 1: chuyển đổi số 2011 thành xâu ký tự “2011” như sau

**SELECT CONVERT(2011, CHAR(5));**

Ví dụ 2: Lấy tất cả các thông tin về khóa học bắt đầu vào ngày 25 một tháng bất kỳ. (Ngày bắt đầu của khóa học lưu trong cột *StartDate* của bảng *course*)

**SELECT \***

**FROM course**

**WHERE CONVERT(*StartDate*, CHAR(15)) LIKE "25%";**



# Các hàm trong SQL

- Các hàm thao tác trên nhiều bản ghi:
  - MAX() – tìm giá trị lớn nhất cho các thuộc tính kiểu số
  - MIN() – tìm giá trị nhỏ nhất cho các thuộc tính kiểu số
  - AVG() – tìm giá trị trung bình cho các thuộc tính kiểu số
  - COUNT() – đếm số bản ghi
  - SUM() – tìm tổng các giá trị cho các thuộc tính kiểu số

Ví dụ:

**SELECT AVG(Age) , SUM(Salary)**

**FROM nhanvien;**



# Gom nhóm các kết quả truy vấn

- Tần suất phân bổ các giá trị trả về sẽ được tự động tạo ra bởi về câu lệnh GROUP BY bên trong câu lệnh SELECT.
- Cú pháp là:

```
SELECT columnlist
      FROM tablelist
      [WHERE conditionlist ]
      [GROUP BY columnlist ]
      [HAVING conditionlist ]
      [ORDER BY columnlist [ASC | DESC] ] ;
```

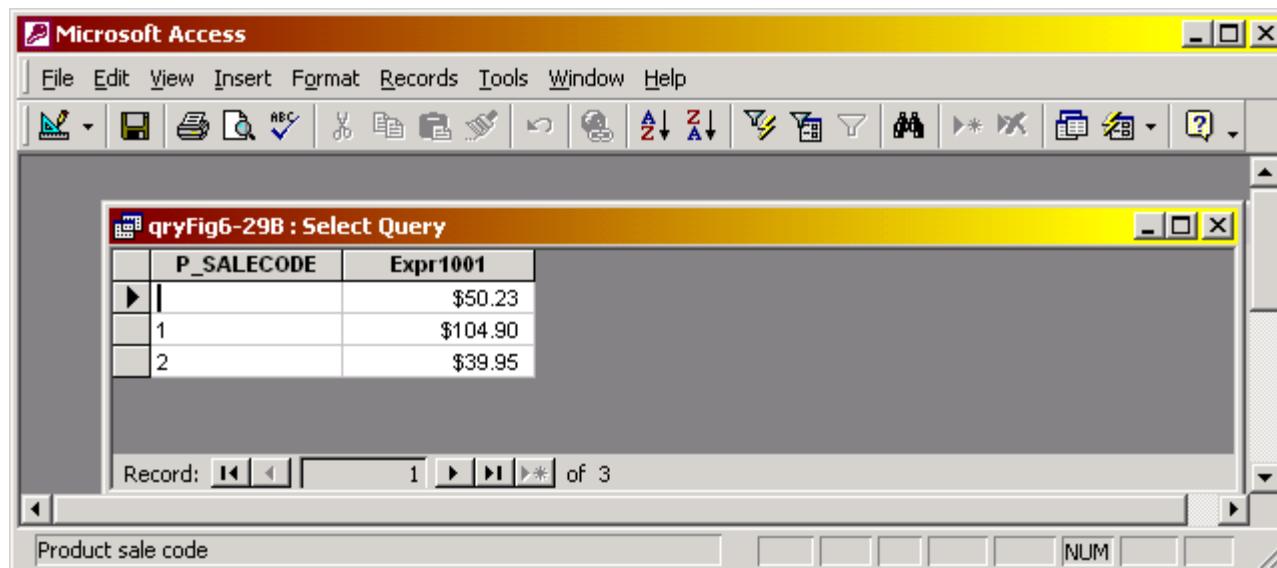
- Về câu lệnh GROUP BY thường được sử dụng trong các cột thuộc tính có các hàm thống kê (MAX, MIN, AVG, SUM, COUNT) trong câu lệnh SELECT.
- Ví dụ, tìm giá bán nhỏ nhất cho mỗi mã hàng. Câu lệnh được thực hiện như ở slide kế tiếp



# Gom nhóm các kết quả truy vấn (cont.)

- Câu truy vấn:

```
SELECT P_SALECODE, MIN(P_PRICE)  
FROM PRODUCT  
GROUP BY P_SALECODE;
```



# Gom nhóm các kết quả truy vấn (cont.)

- Một số quy tắc cần nhớ khi sử dụng về câu GROUP BY trong câu lệnh SELECT:
  - Trong danh mục *columnlist* của SELECT **phải có** sự kết hợp giữa tên cột và các **hàm thống kê**.
  - Trong danh mục *columnlist* của **về GROUP BY** bao gồm các cột trong *columnlist* của phần SELECT mà **không** chứa hàm thống kê. Tùy từng trường hợp, ta có thể gom nhóm theo bất kỳ cột chứa hàm thống kê nào có trong *columnlist* của phần SELECT.
  - Danh mục *columnlist* của **về GROUP BY** có thể bao gồm bất kỳ cột nào trong bảng xác định bởi **về FROM** của câu lệnh SELECT, kể cả khi chúng không xuất hiện trong danh mục *columnlist* của SELECT.



# Về câu HAVING của GROUP BY

- Về cơ bản, HAVING hoạt động giống như vế câu WHERE trong câu lệnh SELECT. Tuy nhiên, vế câu WHERE chỉ đến các giá trị trong từng cột và từng hàng cụ thể của các bảng trong vế câu FROM, trong khi HAVING chỉ đến kết quả đầu ra của câu lệnh GROUP BY.
- Ví dụ, ta cần liệt kê số sản phẩm có trong kho của từng nhà cung cấp, tuy nhiên ta chỉ cần các mặt hàng có giá trung bình dưới \$10.00. Vế thứ nhất được thực hiện bởi lệnh GROUP BY, vế thứ hai được thực hiện bởi lệnh HAVING.



# Về câu HAVING của GROUP BY (cont.)

The screenshot shows two Microsoft Access windows. The top window, titled 'qryFig6-31B : Select Query', displays the following SQL code:

```
SELECT PRODUCT_2.V_CODE, Count(PRODUCT_2.P_CODE) AS CountOfP_CODE, Avg(PRODUCT_2.P_PRICE)  
AS AvgOfP_PRICE  
FROM PRODUCT_2  
GROUP BY PRODUCT_2.V_CODE  
HAVING (((Avg(PRODUCT_2.P_PRICE))<10));
```

A green callout box labeled 'The query' points to this window.

The bottom window, also titled 'qryFig6-31B : Select Query', shows the results of the query in a grid format:

V_CODE	Expr1001	Expr1002
21225	2	\$8.47
21231	1	\$8.45

A pink callout box labeled 'The results' points to this window.

# Các bảng hiển thị: Tạo hiển thị

- Đầu ra của một câu lệnh quan hệ (chẳng hạn, SELECT trong SQL) là các quan hệ (hay là bảng).
- Ví dụ như trong CSDL đã có, giả sử ta cần tạo danh mục các sản phẩm cần nhập thêm vào cuối ngày, các sản phẩm này có số lượng tồn kho nhỏ hơn một ngưỡng cho trước.
- Thay vì phải nhập cùng một câu truy vấn từ ngày này sang ngày khác, ta cần phải lưu vĩnh viễn câu lệnh đó vào CSDL.
- Chức năng hiển thị quan hệ cho phép việc này. Trong SQL, phần hiển thị (**view**) là một bảng tạo ra từ câu truy vấn SELECT. Câu truy vấn này có thể gồm cột, các cột tính toán, cột tên riêng, hoặc các hàm thống kê từ một hoặc nhiều bảng.
- Các bảng cấp dữ liệu cho phần hiển thị gọi là bảng gốc (**base tables**).
- Giao diện hiển thị được tạo ra trong SQL bằng câu lệnh CREATE VIEW.



# Các bảng hiển thị: Tạo hiển thị (cont.)

- Cú pháp của câu lệnh CREATE VIEW là:

```
CREATE VIEW viewname AS SELECT query
```

- Câu lệnh CREATE VIEW thuộc về DDL, nó bao gồm các câu lệnh con bên trong, chẳng hạn như lệnh SELECT để tạo ra các bảng hiển thị của CSDL.

- Ví dụ:

```
CREATE VIEW PRODUCT_3 AS  
    SELECT P_DESCRIP, P_ONHAND, P_PRICE  
    FROM PRODUCT  
    WHERE P_PRICE > 50.00;
```

- Lưu ý: Câu lệnh CREATE VIEW không dùng được trực tiếp trong Access. Để tạo giao diện hiển thị trong Access, ta cần phải tạo câu truy vấn SQL rồi lưu nó lại.



# Các bảng hiển thị: Tạo hiển thị (cont.)

- Một giao diện hiển thị có một vài đặc tính như sau:
  - Ta có thể sử dụng tên của mục hiển thị thay cho tên của bảng trong câu truy vấn SQL.
  - Giao diện hiển thị được cập nhật liên tục. Nói cách khác, giao diện được tự động tạo ra khi câu lệnh được kích hoạt.
  - Giao diện hiển thị cho phép giới hạn người xem theo từng cột và hàng dữ liệu.
  - Giao diện hiển thị có thể được sử dụng làm nội dung báo cáo. Câu lệnh sau tạo ra bảng tổng kết về tổng số giá trị hàng hóa cũng như số lượng tồn kho thống kê theo từng nhà cung cấp:

```
CREATE VIEW SUMPRDXVEN AS
    SELECT V_CODE, SUM(P_ONHAND*P_PRICE) AS TOTCOST,
           MAX(P_ONHAND) AS MAXQTY, MIN(P_OHAND) AS MINQTY,
           AVG(P_ONHAND) AS AVGQTY
      FROM PRODUCT
     GROUP BY V_CODE;
```



```
select fileds  
from table1  
inner join table2 on table1.fildi = table2.fieldi
```



- **inner join** : trả về các bản ghi có giá trị phù hợp giữa hai bảng (*nhớ lại phép giao hai tập hợp*).
- **left join** : mọi bản ghi bảng bên trái được trả về, bản ghi nào phù hợp với bản ghi bên phải thì nó được bổ sung thêm dữ liệu từ bản ghi bảng bên phải (nếu không có thì nhận NULL)
- **right join** : mọi bản ghi bảng bên phải được trả về, sau bổ sung dữ liệu phù hợp từ bảng bên trái.
- **outer join** : (full join) mọi bản ghi ở bảng trái và bảng phải kết hợp lại

