www.mientayvn.com

Khi đọc qua tài liệu này, nếu phát hiện sai sót hoặc nội dung kém chất lượng xin hãy thông báo để chúng tôi sửa chữa hoặc thay thế bằng một tài liệu cùng chủ đề của tác giả khác. Tài liệu này bao gồm nhiều tài liệu nhỏ có cùng chủ đề bên trong nó. Phần nội dung bạn cần có thể nằm ở giữa hoặc ở cuối tài liệu này, hãy sử dụng chức năng Search để tìm chúng.

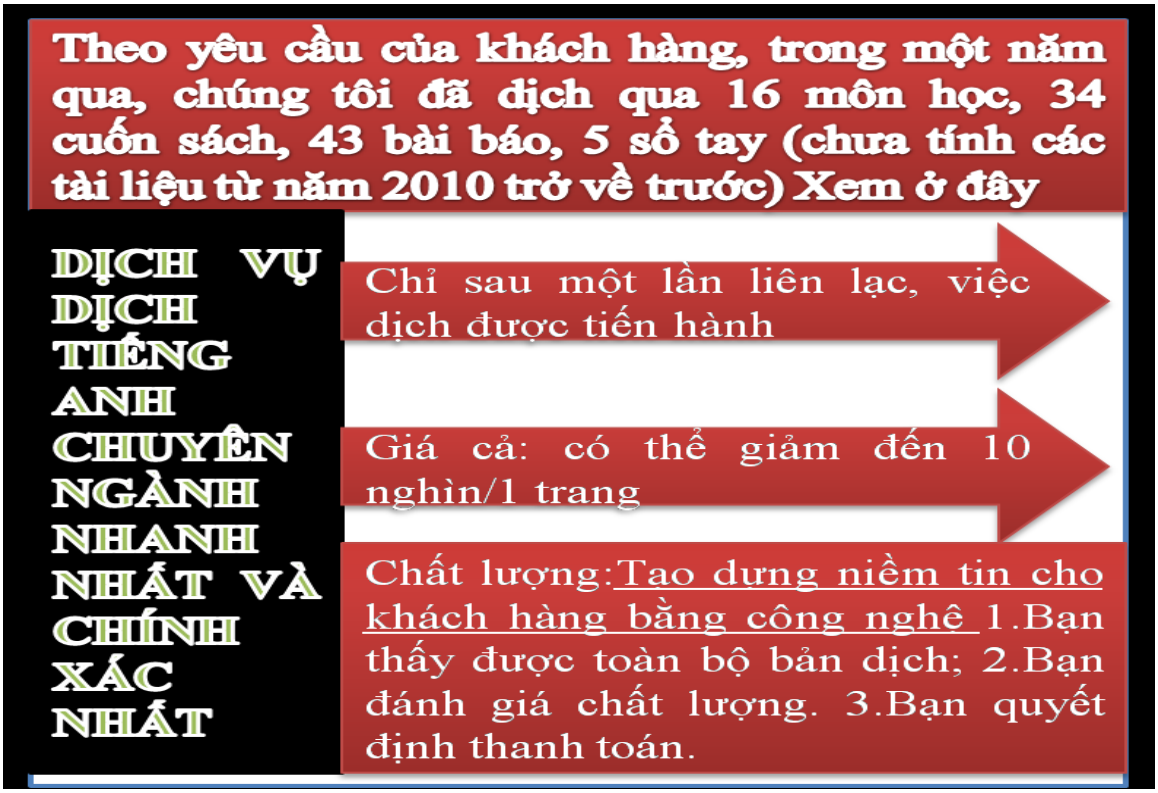
Bạn có thể tham khảo nguồn tài liệu được dịch từ tiếng Anh tại đây:

http://mientayvn.com/Tai_lieu_da_dich.html

Thông tin liên hệ:

Yahoo mail: thanhlam1910_2006@yahoo.com

Gmail: frbwrthes@gmail.com



Đại học Cần thơ

Khoa CNTT-TT

CSDL NÂNG CAO

Phạm Thị Xuân Lộc

10/2008

MỤC LỤC

- A. HQTCSDL phân tán
- B. HQTCSDL hướng đối tượng
- C. Web và HQTCSDL
- D. Kho dữ liệu

A. HQTCSDL phân tán (DDBMS)

A. HQTCSDL phân tán (DDBMS: Distributed DataBase Management System)

- I. Giới thiệu
- II. Chức năng và kiến trúc của DDBMS
- III. Thiết kế CSDL quan hệ phân tán
- IV. Tính trong suốt trong DDBMS
- V. 10 qui tắc của Date cho DDBMS

A.I Giới thiệu

- 1. Nhu cầu có DDBMS
- 2. Khái niệm
- 3. Lợi và bất lợi của DDBMS
- 4. Tính đồng nhất và bất đồng trong DDBMS

A.I.1. Nhu cầu có DDBMS

- Ta đã ng/c các DBMS tập trung:
  - 1 CSDL luận lý
  - tại 1 vị trí (site)
  - dưới sự điều khiển của 1 DBMS
- Trong đó, CSDL như một hòn đảo thông tin vì cách biệt, khó truy xuất, do:
  - cách trở về địa lý,
  - cấu hình các máy tính không tương thích nhau,
  - các nghi thức truyền thông không tương thích nhau, …

A.I.1. Nhu cầu có DDBMS (2)

- Gần đây, phát triển nhanh chóng các công nghệ:
  - Mạng
  - Truyền thông dữ liệu

qua:

  - *Internet*
  - *Tính toán di dộng và không dây (mobile and wireless computing)*
  - *Thiết bị thông minh*
  - *Tính toán lưới (grid computing)*

A.I.1. Nhu cầu có DDBMS (3)

- Công nghệ DDBMS= kết hợp 2 công nghệ trên
- [redacted] thể thay đổi cách thức làm việc từ tập trung sang phân tán
- [redacted] một buớc phát triển lớn trong lĩnh vực DBMS

A.I.1. Nhu cầu có DDBMS (4)

- Cách tiếp cận phân tán phản ánh cấu trúc về tổ chức của các xí nghiệp, vốn đã phân tán về:
  - luận lý: phân hệ, bộ môn, dự án, …
  - vật lý: văn phòng, xưởng, phòng, …

A.I.1. Nhu cầu có DDBMS (5)

- Tính chia sẻ dữ liệu và tính hiệu quả trong truy cập dữ liệu cần được cải thiện bằng một DDBMS.
- Cần làm cho dữ liệu truy xuất được ở mọi đơn vị.
- Cần lưu dữ liệu gần với nơi chúng thường được dùng nhất.

A.I Giới thiệu

- 1. Nhu cầu có DDBMS
- 2. Khái niệm
- 3. Lợi và bất lợi của DDBMS
- 4. Tính đồng nhất và bất đồng trong DDBMS

A.I.2 Khái niệm

- a. CSDL phân tán:
- b. Hệ quản trị CSDL phân tán (DDBMS):
- c. Xử lý phân tán (distributed processing)
- d. Các DBMS song song (parallel DBMSs)

A.I.2 Khái niệm **CSDL phân tán**

Là một tập hợp (collection) các dữ liệu chia sẻ và các mô tả của chúng,

- có liên quan nhau về luận lý
- phân tán về vật lý trên một mạng máy tính

A.I.2 Khái niệm **Hệ quản trị CSDL phân tán (DDBMS)**

- Đinh nghĩa:
- Đoạn (fragment)
- Các loại trình ứng dụng
- Đặc tính của DDBMS
- Tính trong suốt trong DDBMS

A.I.2 Khái niệm **DDBMS** (2)

Định nghĩa:

Là một hệ thống phần mềm, cho phép:

- Quản trị một CSDL phân tán
- Làm cho sự phân tán này trong suốt đối với người sử dụng

A.I.2 Khái niệm **DDBMS** (3)

Đoạn (fragment):

Một DDBMS gồm một CSDL luận lý được chia thành các đoạn.

- Mỗi đoạn được lưu trên một hoặc nhiều máy tính dưới sự kiểm soát của một DBMS riêng thông qua một mạng.
- Mỗi site có khả năng:
  - xử lý độc lập các yêu cầu của người dùng muốn truy xuất dữ liệu cục bộ,
  - xử lý dữ liệu lưu ở các máy khác trên mạng.

A.I.2 Khái niệm **DDBMS** (4)

Các loại trình ứng dụng:

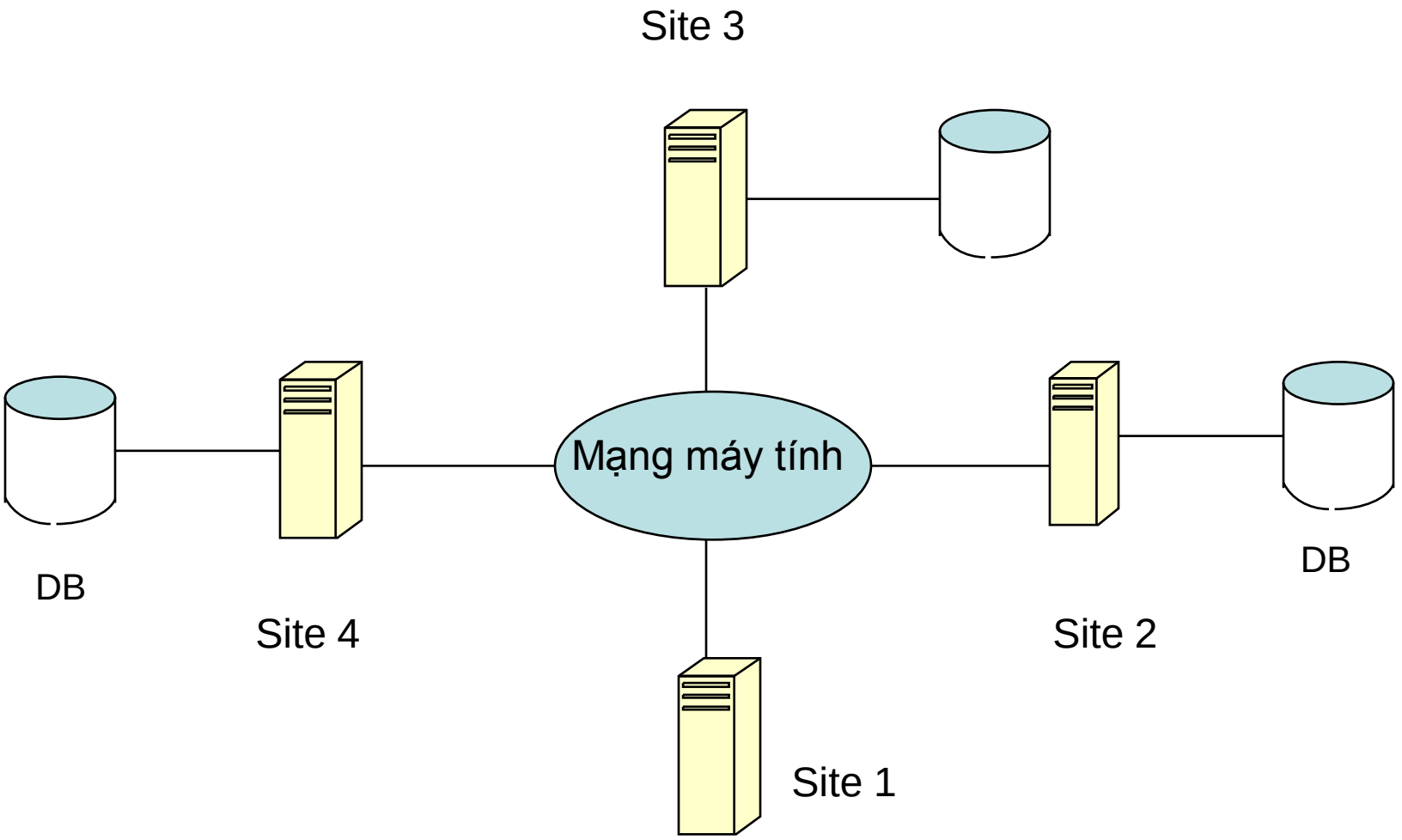
- Ứng dụng cục bộ (local application): không yêu cầu dữ liệu ở nơi khác.
- Ứng dụng toàn cục (global application): ngược lại

A.I.2 Khái niệm **DDBMS** (5)

Đặc tính của DDBMS:

- Là 1 tập hợp các dữ liệu chia sẻ có liên quan về luận lý
- Dữ liệu được chia nhỏ thành các đoạn
- Các đoạn có thể được lặp lại (replicate)
- Các đoạn và các bản sao của chúng được phân phối trên các sites
- Các sites liên kết nhau qua một mạng truyền thông
- Dữ liệu trên mỗi site được điều khiển bởi một hệ quản trị CSDL (DBMS)
- DBMS trên mỗi site có thể điều khiển một cách tự chủ các ứng dụng cục bộ
- Mỗi DBMS tham gia vào ít nhất một ứng dụng toàn cục.

A.I.2 Khái niệm **DDBMS** (6)



A.I.2 Khái niệm **DDBMS** (7)

Tính trong suốt trong DDBMS:

Việc:

- một CSDL phân tán trên nhiều đoạn, trên nhiều máy tính khác nhau
- các đoạn có thể được nhân bản

cần phải được che dấu đối với người sử dụng

làm cho DDBMS có vẻ như một hệ thống tập trung

A.I.2 Khái niệm **Xử lý phân tán**

- Cần phân biệt giữa một DDBMS và một xử lý phân tán
- Xử lý phân tán:
  - Một CSDL tập trung có thể được truy xuất qua một mạng máy tính.

A.I.2 Khái niệm **DBMS song song**

- Cần phân biệt giữa một DDBMS và một DBMS song song
- DBMS song song: là một DBMS
  - chạy trên nhiều bộ xử lý và đĩa,
  - được thiết kế để khai thác song song các hoạt động bất cứ lúc nào có thể,
  - để tăng cường hiệu suất

A.I Giới thiệu

- 1. Nhu cầu có DDBMS
- 2. Khái niệm
- 3. Lợi và bất lợi của DDBMS
- 4. Tính đồng nhất và bất đồng trong DDBMS

A.I.3. Lợi của DDBMS

- Phản ánh cấu trúc về mặt tổ chức
- Cải thiện tính chia sẻ và tính tự chủ cục bộ
- Cải thiện tính sẵn sàng
- Cải thiện tính tin cậy
- Cải thiện hiệu suất
- Tiết kiệm
- Tăng trưởng theo khối (modular growth)
- Tích hợp
- Giữ tính cạnh tranh

A.I.3. Bất lợi của DDBMS

- Phức tạp
- Đắt
- Khó an toàn
- Khó kiểm soát tích hợp hơn
- Thiếu chuẩn
- Thiếu kinh nghiệm
- Thiết kế CSDL phức tạp hơn

A.I Giới thiệu

- 1. Nhu cầu có DDBMS
- 2. Khái niệm
- 3. Lợi và bất lợi của DDBMS
- 4. Tính đồng nhất và bất đồng trong DDBMS

A.I.4. Tính đồng nhất và bất đồng trong DDBMS

DDBMS đồng nhất:

- Mọi site đều dùng chung một DBMS
- Dễ thiết kế và quản trị hơn
- Tăng trưởng dần
- Thêm một site mới dễ dàng
- Tăng hiệu suất bằng cách khai thác khả năng xử lý song song của nhiều site

A.I.4. Tính đồng nhất và bất đồng trong DDBMS (2)

DDBMS bất đồng: ngược lại

- Xảy ra khi mỗi site đã cài đặt CSDL riêng và sự tích hợp mới thực hiện sau đó
- Cần có sự dịch (translation) để giao tiếp giữa các DBMS khác nhau
- Để cung cấp tính trong suốt của DBMS, các người dùng phải có khả năng đặt câu hỏi theo ngôn ngữ của DBMS trên site của họ [redacted] ệ thống phải định vị dữ liệu và dịch.

A.I.4. Tính đồng nhất và bất đồng trong DDBMS (3)

DDBMS bất đồng: Dữ liệu yêu cầu từ site khác có thể có:

- Các phần cứng khác nhau
- Các sản phẩm DBMS khác nhau
- Các phần cứng khác nhau và các sản phẩm DBMS khác nhau

A.I.4. Tính đồng nhất và bất đồng trong DDBMS (4)

Hệ thống đa CSDL (MDBMS: multidatabase system):

- Là một DDBMS, trong đó mỗi site giữ tự chủ hoàn toàn
- Cho phép người dùng truy cập và chia sẻ dữ liệu mà không cần tích hợp toàn bộ các sơ đồ CSDL.
- Tuy nhiên, vẫn cho phép họ quản trị CSDL riêng mà không có kiểm soát tập trung.
- Cho phép người dùng ở site khác truy cập một phần CSDL của họ, bằng cách đặc tả một *export schema*.

A.I.4. Tính đồng nhất và bất đồng trong DDBMS (5)

Hệ thống đa CSDL:

- Có 2 loại MDBS:
  - phi liên hiệp (unfederated): không có người dùng cục bộ
  - liên hiệp (federated): có một hệ thống phân tán cho các người dùng toàn cục, và một hệ thống tập trung cho các người dùng cục bộ

A. HQTCSDL phân tán (DDBMS)

- I. Giới thiệu
- II. Chức năng và kiến trúc của DDBMS
- III. Thiết kế CSDL quan hệ phân tán
- IV. Tính trong suốt trong DDBMS
- V. 10 qui tắc của Date cho DDBMS

A.II Chức năng và kiến trúc của DDBMS

- 1. Chức năng của DDBMS
- 2. Kiến trúc của DDBMS
- 3. Kiến trúc tham chiếu (reference architecture) cho một MDBS liên hiệp
- 4. Kiến trúc thành tố (component architecture) cho một DDBMS

A.II.1 Chức năng của DDBMS

- Giống như HQTCSDL bình thường: mô tả, lưu trữ, cập nhật,…
- Có thêm các dịch vụ truyền thông:
  - để truy xuất từ xa các site và chuyển câu hỏi và dữ liệu giữa các site thông qua mạng.
- Thêm biên mục hệ thống (system catalog):
  - để lưu các chi tiết về phân tán dữ liệu
- Xử lý các câu hỏi phân tán: bao gồm
  - tối ưu hoá câu hỏi
  - truy cập dữ liệu từ xa

A.II.1 Chức năng của DDBMS (2)

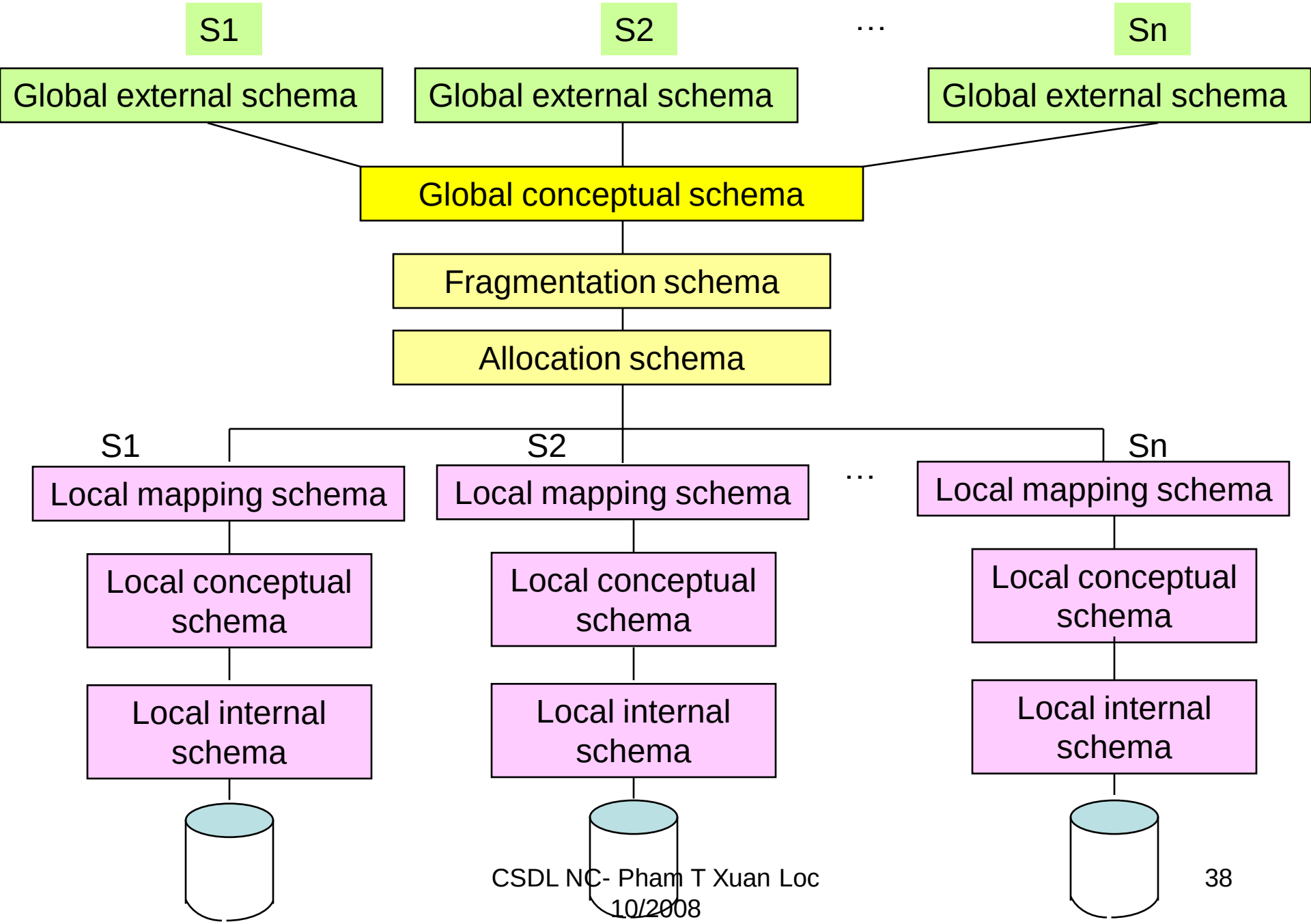
- Tăng cường kiểm soát bảo mật:
  - Để duy trì các cấp phép thích hợp/ đặc quyền truy cập đến các dữ liệu phân tán
- Tăng cường kiểm soát cạnh tranh:
  - Để duy trì tính nhất quán của dữ liệu phân tán (và chúng có thể được nhân bản)
- Tăng cường các dịch vụ phục hồi để khắc phục các hỏng hóc của các site riêng lẻ và của các đường truyền thông

A.II Chức năng và kiến trúc của DDBMS

- 1. Chức năng của DDBMS
- 2. Kiến trúc của DDBMS
- 3. Kiến trúc tham chiếu (reference architecture) cho một MDBS liên hiệp
- 4. Kiến trúc thành tố (component architecture) cho một DDBMS

A.II.2 Kiến trúc tham chiếu của DDBMS

- Kiến trúc phải để ý đến tính phân tán dữ liệu.
- Gồm:
  - 1 tập các sơ đồ ngoài toàn cục
  - 1 sơ đồ quan niệm toàn cục
  - 1 sơ đồ phân đoạn và sơ đồ cấp phát
  - 1 tập các sơ đồ cho mỗi DBMS cục bộ theo kiến trúc 3 lớp của ANSI-SPARC

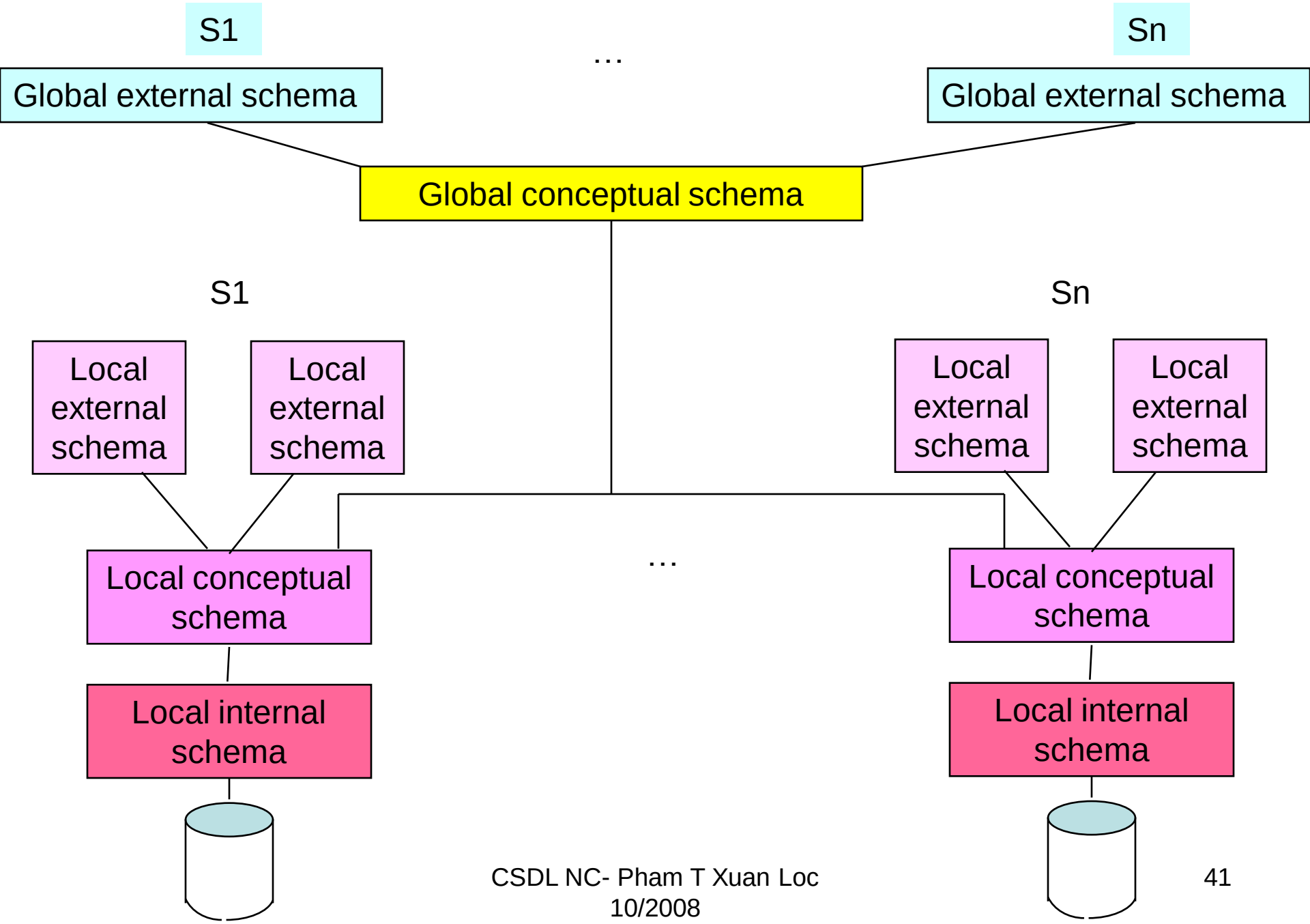

S1
S2
…
Sn
Global external schema
Global external schema
Global external schema
Global conceptual schema
Fragmentation schema
Allocation schema
S1
S2
Sn
Local mapping schema
Local mapping schema
…
Local mapping schema
Local conceptual schema
Local conceptual schema
Local conceptual schema
Local internal schema
Local internal schema
Local internal schema

A.II Chức năng và kiến trúc của DDBMS

- 1. Chức năng của DDBMS
- 2. Kiến trúc của DDBMS
- 3. Kiến trúc tham chiếu (reference architecture) cho một MDBS liên hiệp
- 4. Kiến trúc thành tố (component architecture) cho một DDBMS

A.II.3. Kiến trúc tham chiếu cho một MDBS liên hiệp (FMDBS)

- Có một sơ đồ quan niệm toàn cục (GCS: Global Conceptual Schema) cho các FMDBS *kết hợp chặt* (tighly coupled FMDBS)
- Trong DDBMS, GCS = hội của tất cả các sơ đồ quan niệm cục bộ
- Trong FMDBS, GCS = một tập con của các sơ đồ quan niệm cục bộ, bao gồm các dữ liệu mà mỗi hệ thống cục bộ đồng ý chia sẻ
- GCS của FMDBS *kết hợp chặt* sẽ tích hợp:
  - Hoặc các phần của các các sơ đồ quan niệm cục bộ,
  - Hoặc các sơ đồ ngoài cục bộ

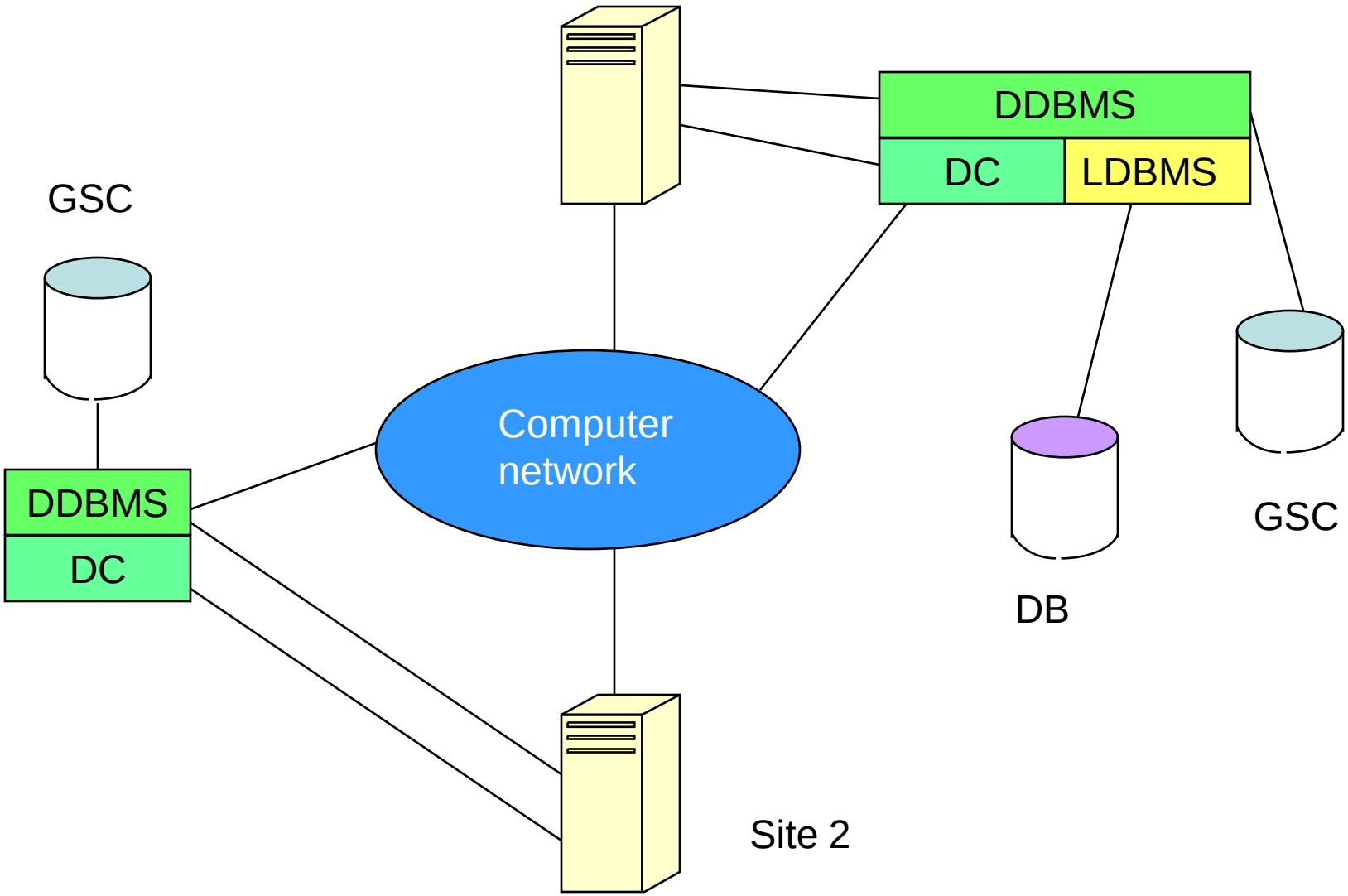

S1
Sn
…
Global external schema
Global external schema
Global conceptual schema
S1
Sn
Local external schema
Local external schema
Local external schema
Local external schema
…
Local conceptual schema
Local conceptual schema
Local internal schema
Local internal schema

A.II Chức năng và kiến trúc của DDBMS

- 1. Chức năng của DDBMS
- 2. Kiến trúc của DDBMS
- 3. Kiến trúc tham chiếu (reference architecture) cho một MDBS liên hiệp
- 4. Kiến trúc thành tố (component architecture) cho một DDBMS

A.II.4 Kiến trúc thành tố cho một DDBMS

- Độc lập với kiến trúc tham chiếu, kiến trúc thành tố cho một DDBMS gồm có :
  - thành tố DBMS cục bộ (LDBMS: local DBMS)
  - thành tố truyền thông dữ liệu (DC: data communication)
  - biên mục hệ thống toàn cục (GSC: global system catalog)
  - thành tố DDBMS


Site 1
DDBMS
DC
LDBMS
GSC
Computer network
DDBMS
DC
DB
GSC
Site 2

A.II.4 Kiến trúc thành tố cho một DDBMS (3)

Thành tố DBMS cục bộ (LDBMS):

- Là một DBMS chuẩn, có nhiệm vụ kiểm soát dữ liệu cục bộ tại chỉ một site có CSDL.
- Có riêng biên mục hệ thống cục bộ để lưu thông tin về lưu dữ liệu trên site đang xét.
- Trong hệ đồng nhất, LDBMS là cùng một sản phẩm được nhân bản ra khắp các site.
- Trong hệ bất đồng, có ít nhất 2 site có LDBMS khác nhau và/ hoặc có platform khác nhau.

A.II.4 Kiến trúc thành tố cho một DDBMS (4)

Thành tố truyền thông dữ liệu (DC):

- Là phần mềm cho phép tất cả các site liên lạc với nhau.
- Chứa thông tin về các site và cá đường liên kết.

A.II.4 Kiến trúc thành tố cho một DDBMS (5)

Biên mục hệ thống toàn cục (GSC):

- Có cùng chức năng như trong một hệ tập trung.
- Giữ thông tin về sự phân tán của hệ thống:

  sự phân đoạn (fragmentation)

  sự nhân bản (replication)

  các sơ đồ cấp phát (allocation schemas)

A.II.4 Kiến trúc thành tố cho một DDBMS (6)

- Bản thân nó cũng được quản lý như một CSDL phân tán: được phân đoạn và phân tán, hoặc được nhân bản, hoặc được tập trung.
  - Được nhân bản hoàn toàn: mọi thay đổi trên GSC sẽ được thông báo cho tất cả các site.
  - Tập trung: như trên, và còn dễ bị tổn hại bởi các hỏng hóc của site trung tâm.

A.II.4 Kiến trúc thành tố cho một DDBMS (7)

Thành tố DDBMS:

- Là đơn vị kiểm soát toàn bộ hệ thống.

A. HQTCSDL phân tán (DDBMS)

- I. Giới thiệu
- II. Chức năng và kiến trúc của DDBMS
- III. Thiết kế CSDL quan hệ phân tán
- IV. Tính trong suốt trong DDBMS
- V. 10 qui tắc của Date cho DDBMS

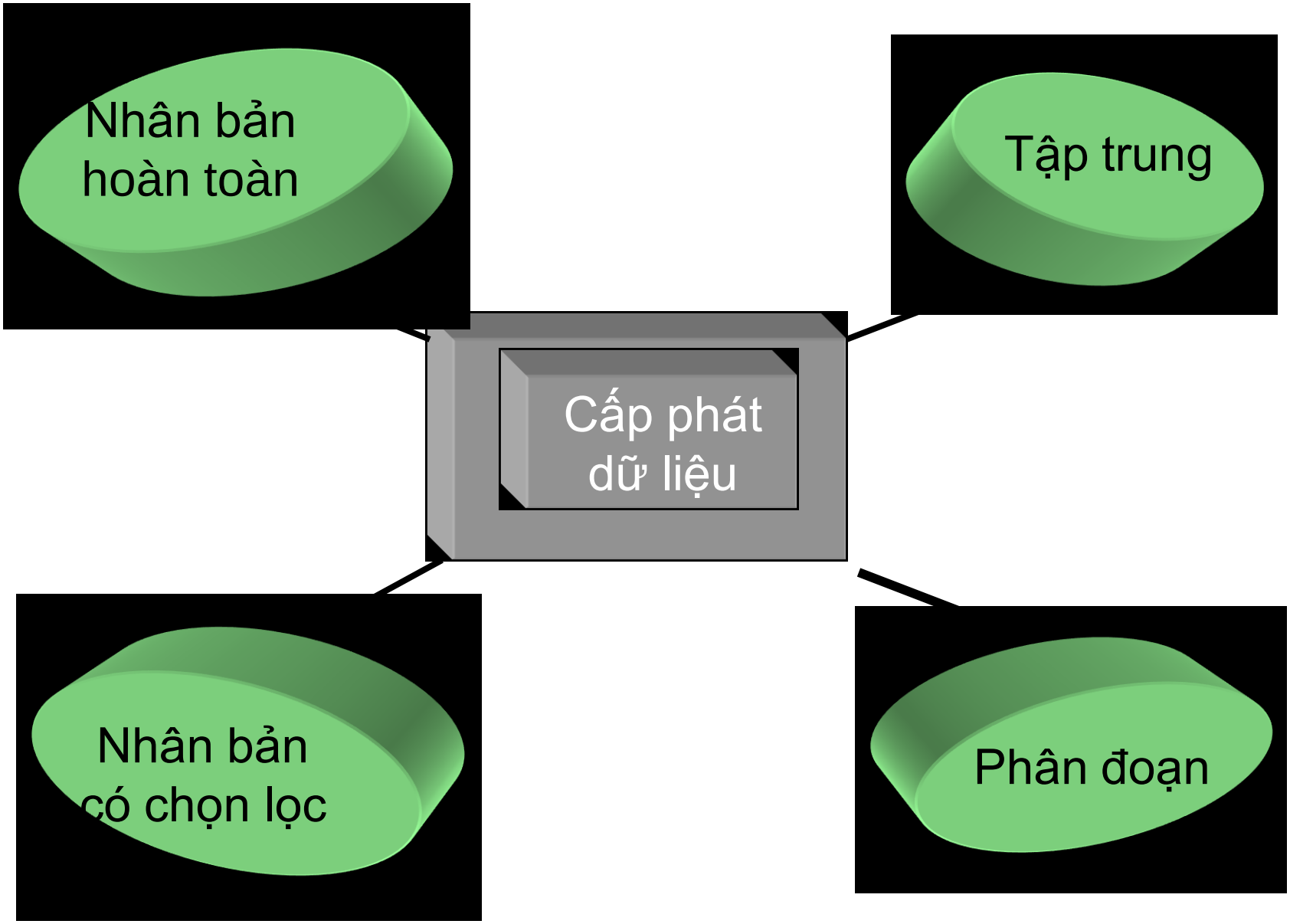
A.III Thiết kế CSDL quan hệ phân tán

- 1. Cấp phát dữ liệu
- 2. Phân đoạn

A.III.1 Cấp phát dữ liệu

Có 4 chiến lược cấp phát tuỳ vào nơi đặt dữ liệu:

- Tập trung (centralized)
- Phân đoạn (fragmented/ partionned)
- Nhân bản hoàn toàn (complete replication)
- Nhân bản có chọn lọc (selective replication)


Nhân bản
hoàn toàn
Tập trung
Cấp phát
dữ liệu
Nhân bản
có chọn lọc
Phân đoạn

A.III.1 Cấp phát dữ liệu (2)

- **Cấp phát tập trung:**
  - Có một CSDL và một DBMS trên một site
  - Người dùng phân tán khắp trên mạng (thực chất đây là xử lý phân tán)
  - Luôn phải dùng mạng cho mọi truy xuất dữ liệu
- *Đánh giá:*
  - Giá thành liên lạc cao
  - Tính tin cậy và tính sẵn sàng kém (khi site trung tâm bị hỏng hóc sẽ làm hỏng luôn toàn bộ hệ CSDL).

A.III.1 Cấp phát dữ liệu (3)

- **<u>Cấp phát có phân đoạn:</u>**
  - Chia CSDL thành các đoạn (fragment) rời nhau, mỗi đoạn trên một site.
- *Đánh giá:*
  - Vì không nhân bản, chi phí lưu trữ thấp
  - Tính tin cậy và tính sẵn sàng cũng kém, mặc dù có cao hơn cấp phát tập trung, vì khi hỏng hóc xảy ra chỉ trên một site nên chỉ làm mất dữ liệu trên site đó.
  - Hiệu suất tương đối tốt
  - Chi phí liên lạc thấp nếu phân bố hợp lý.

A.III.1 Cấp phát dữ liệu (4)

- **<u>Cấp phát có nhân bản hoàn toàn:</u>**
  - Giữ một bản sao đầy đủ của CSDL trên mỗi site.
- *Đánh giá:*
  - Tính cục bộ của tham chiếu cao
  - Tính tin cậy, tính sẵn sàng, hiệu suất đều đạt tối đa
  - Tuy nhiên, chi phí lưu trữ và chi phí liên lạc cho cập nhật thì đắt nhất

  [redacted]ng snapshot= bản sao dữ liệu tại một thời điểm. Các bản sao được cập nhật định kỳ nên có thể không kịp thời.

A.III.1 Cấp phát dữ liệu (5)

- **<u>Cấp phát dùng nhân bản có chọn lọc</u>:**
  - Là sự kết hợp của các chiến lược phân đoạn, nhân bản, và tập trung
  - Một số mục dữ liệu được phân đoạn để nâng cao tính cục bộ
  - Một số mục khác, nếu được dùng trên nhiều site và không được cập nhật thường xuyên, sẽ được nhân bản.
  - Số mục còn lại, sẽ được cấp phát tập trung.

[redacted]ược dùng phổ biến nhất nhờ tính mềm dẽo của nó.

A. HQTCSDL phân tán (DDBMS)

- I. Giới thiệu
- II. Chức năng và kiến trúc của DDBMS
- III. Thiết kế CSDL quan hệ phân tán
- IV. Tính trong suốt trong DDBMS
- V. 10 qui tắc của Date cho DDBMS

A.III Thiết kế CSDL quan hệ phân tán

- 1. Cấp phát dữ liệu
- 2. Phân đoạn

A.III.2 Phân đoạn

- 1. Lợi và bất lợi
- 2. Các qui tắc phân đoạn
- 3. Các cách phân đoạn

A.III.2 Phân đoạn- Lợi và bất lợi

- **Lợi:**
  - *Sự sử dụng:* Thông thường, ứng dụng làm việc với các khung nhìn hơn là với toàn bộ các quan hệ.
  - *Tính hiệu quả:* Dữ liệu được lưu ở nơi nó thường được dùng nhất. Hơn nữ, các dữ liệu không cần cho các ứng dụng cục bộ sẽ không được lưu.
  - *Tính song song:* Khi các đoạn là các đơn vị phân tán dữ liệu, một giao dịch có thể được chia nhỏ thành các câu hỏi con xử lý trên các đoạn [redacted]ng tinh cạnh tranh, tính song song trên các hệ thống cho phép các giao dịch được xử lý song song một cách an toàn.
  - *Tính an toàn:* Các dữ liệu không cần cho các ứng dụng cục bộ sẽ không được lưu, do đó sẽ không sẵn sàng cho các người dùng trái phép.

A.III.2 Phân đoạn- Lợi và bất lợi (2)

- **Bất lợi:**
  - Hiệu suất: Các ứng dụng tổng thể cần dữ liệu trên nhiều đoạn ở các site khác nhau sẽ bị chậm hơn.
  - Tính toàn vẹn: Các ràng buộc toàn vẹn sẽ khó kiểm soát hơn nếu dữ liệu và các phụ thuộc hàm được phân trên nhiều đoạn ở các site khác nhau.

A.III.2 Phân đoạn- Các qui tắc phân đoạn

- (1) Tính đầy đủ (không mất dữ liệu):
  - Nếu một thể hiện quan hệ R chia thành n đoạn thì một mục dữ liệu trong R phải được tìm thấy trên ít nhất một đoạn.
- (2) Tái xây dựng (bảo toàn phụ thuộc hàm):
  - Phải xây dựng lại được R từ các đoạn bằng một phép xử lý trên quan hệ.
- (3) Tính rời rạc (ít dư thừa dữ liệu nhất):
  - Nếu một mục dữ liệu di xuất hiện trong đoạn Ri, nó không được có trên bất kỳ đoạn nào khác.
  - *Ngoại lệ*: Trong cách phân đoạn dọc, khoá chính phải được lặp lại để tái xây dựng quan hệ.

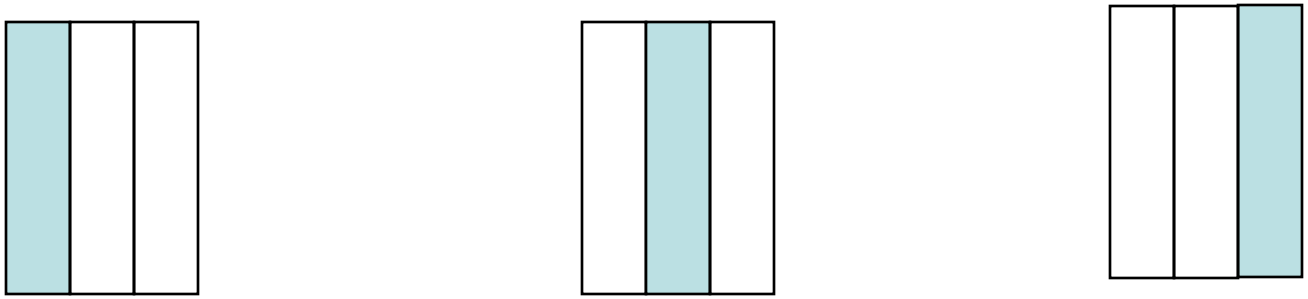
A.III.3 Phân đoạn- Các cách phân đoạn

- Phân đoạn ngang: Đoạn = tập con các bộ
- Phân đoạn dọc: Đoạn = tập con các thuộc tính
- Phân đoạn hỗn hợp: trộn 2 cách trên
- Phân đoạn suy diễn: một cách phân đoạn ngang, dựa trên phân đoạn ngang của quan hệ cha.

A.III.3 Phân đoạn- Các cách phân đoạn

Ngang

Dọc



Hỗn hợp

A.III.3 Phân đoạn- Các cách phân đoạn (2)

- Phân đoạn ngang: Chia DSSV thành 2 đoạn DS1 và DS2

MSSV	HỌ	TÊN	Tổ
1040561	Nguyễn Đình	Luân	5
1040570	Quách Văn	Nhị	5
1040611	Ng T Hoàng	Yến	5
MSSV	**HỌ**	**TÊN**	**Tổ**
1040623	Nguyễn Đức	Duy	6
1040656	Phan Thanh	Nhàn	6

A.III.3 Phân đoạn- Các cách phân đoạn (3)

- Phân đoạn dọc:

MSSV	HỌ	TÊN		MSSV	Tổ
1040561	Nguyễn Đình	Luân		1040561	5
1040570	Quách Văn	Nhị		1040570	5
1040611	Ng T Hoàng	Yến		1040611	5
1040623	Nguyễn Đức	Duy		1040623	6
1040656	Phan Thanh	Nhàn		1040656	6

A.III.3 Phân đoạn- Các cách phân đoạn (4)

- Phân đoạn hỗn hợp:

MSSV	HỌ	TÊN
1040561	Nguyễn Đình	Luân
1040570	Quách Văn	Nhị
1040611	Ng T Hoàng	Yến
1040623	Nguyễn Đức	Duy
1040656	Phan Thanh	Nhàn

MSSV	Tổ
1040561	5
1040570	5
1040611	5

MSSV	Tổ
1040623	6
1040656	6

A.III.3 Phân đoạn- Các cách phân đoạn (5)

- Phân đoạn suy diễn:

MSSV	HỌ	TÊN	Môn
1040561	Nguyễn Đình	Luân	CT106
1040611	Ng T Hoàng	Yến	CT349
1040623	Nguyễn Đức	Duy	CT106
1040656	Phan Thanh	Nhàn	CT349

Môn	Bộ môn
CT106	TH1
CT349	TH1

MSSV	HỌ	TÊN	Môn
1040570	Quách Văn	Nhị	CT101

Môn	Bộ môn
CT101	TH2

A.III.3 Phân đoạn- Các cách phân đoạn (5b)

- Phân đoạn suy diễn:

MS_HT	..	…	MA_CB_QL
1040561	Nguyễn Đình	Luân	CT106
1040611	Ng T Hoàng	Yến	CT349
1040623	Nguyễn Đức	Duy	CT106
1040656	Phan Thanh	Nhàn	CT349

MS_HT	..	…	MA_CB_QL
1040570	Quách Văn	Nhị	CT101

MÃ_CB	MÃ_ĐV
CT106	TH1
CT349	TH1

Môn	Bộ môn
CT101	TH2

A. HQTCSDL phân tán (DDBMS)

- I. Giới thiệu
- II. Chức năng và kiến trúc của DDBMS
- III. Thiết kế CSDL quan hệ phân tán
- IV. Tính trong suốt trong DDBMS
- V. 10 qui tắc của Date cho DDBMS

A.IV Tính trong suốt trong DDBMS

- **Mục đích:**
  - Che đi các chi tiết cài đặt với người sử dụng
  - Làm cho việc sử dụng CSDL phân tán cũng giống như khi dùng CSDL tập trung.

A.IV Tính trong suốt trong DDBMS (2)

- Các kiểu trong suốt trong DDBMS:
  1. Trong suốt về sự phân tán (distribution transparency)
  2. Trong suốt về giao dịch (transaction transparency)
  3. Trong suốt về hiệu suất (performance transparency)
  4. Trong suốt về DBMS (DBMS transparency)

A.IV.1 Trong suốt về sự phân tán

- Gồm có:
  - Trong suốt về sự phân đoạn
  - Trong suốt về vị trí
  - Trong suốt về sự nhân bản
  - Trong suốt về ánh xạ cục bộ
  - Trong suốt về việc đặt tên

A.IV.1 Trong suốt về sự phân tán (2)

Trong suốt về sự phân đoạn:

- Là mức cao nhất của sự trong suốt về sự phân tán.
- Nếu nó được DDBMS cung cấp, người dùng không cần biết dữ liệu có phân đoạn.
- Do đó, các truy xuất CSDL được dựa trên sơ đồ toàn cục, người dùng không cần đặc tả tên đoạn và vị trí dữ liệu.
- Các câu hỏi SQL vẫn như ở hệ tập trung.

A.IV.1 Trong suốt về sự phân tán (3)

Trong suốt về vị trí:

- Là mức giữa của sự trong suốt về sự phân tán.
- Người dùng phải biết cách thức dữ liệu được phân đoạn nhưng vẫn chưa cần phải biết vị trí dữ liệu.
- Cần phải đặc tả tên đoạn trong câu hỏi
- Nếu kết quả cần lấy ở trên các đoạn khác nhau, phải thực hiện kết nối hoặc câu hỏi con.
- Ưu điểm: CSDL có thể tổ chức vật lý lại mà không ảnh hưởng đến trình ứng dụng truy xuất nó.

A.IV.1 Trong suốt về sự phân tán (4)

Trong suốt về sự nhân bản:

- Gần với tính trong suốt về vị trí, người dùng không cần biết sự nhân bản các đoạn.
- Được suy diễn từ tính trong suốt về vị trí.
- Tuy nhiên, chiều ngược lại không đúng.

A.IV.1 Trong suốt về sự phân tán (5)

Trong suốt về ánh xạ cục bộ:

- Ở mức độ trong suốt thấp nhất.
- Người dùng cần đặc tả cả tên đoạn lẫn vị trí dữ liệu, và phải chú ý đến bất kỳ sự nhân bản nào đang có.
- Câu hỏi phải phức tạp và tiêu tốn thời gian hơn nhiều.
- Khó được người dùng chấp nhận nếu chỉ có mình nó trong hệ thống.

A.IV.1 Trong suốt về sự phân tán (6)

Trong suốt về việc đặt tên:

- Như trong một CSDL kiểu tập trung, mỗi mục trong DDBMS phải có một tên duy nhất.
- Do đó, DDBMS phải bảo đảm không có 2 site tạo đối tượng cùng tên.

A.IV.1 Trong suốt về sự phân tán (7)

Trong suốt về việc đặt tên (tt):

- *Giải pháp 1*: Tạo ra một name server trung tâm để bảo đảm tính duy nhất của tất cả các tên trong hệ thống.
- *Bất lợi của giải pháp 1*:
  - Làm mất phần nào tính tự chủ cục bộ
  - Giảm hiệu suất, khi site trung tâm trở thành nút cổ chai.
  - Kém sẵn sàng: nếu site trung tâm có sự cố, các site còn lại không thể tạo được đối tượng CSDL nào mới.

A.IV.1 Trong suốt về sự phân tán (8)

Trong suốt về việc đặt tên (tt):

- *Giải pháp 2*: Đặt định danh của site tạo ra một đối tượng trước tên của nó. Còn phải định rõ tên từng đoạn và từng bản sao của nó.
- *Bất lợi của giải pháp 2*:
  - Làm mất phần nào tính trong suốt về sự phân tán.

A.IV.1 Trong suốt về sự phân tán (9)

Trong suốt về việc đặt tên (tt):

- *Giải pháp 3*: Sử dụng bí danh (alias/ synonym) cho mỗi đối tượng CSDL. DDBMS có nhiệm vụ ánh xạ một bí danh sang một đối tượng CSDL thích hợp.
- *Ưu điểm của giải pháp 3*:
  - Giải quyết được các rắc rối của 2 giải pháp 1 và 2.

A.IV Tính trong suốt trong DDBMS (2)

- Các kiểu trong suốt trong DDBMS:
  1. Trong suốt về sự phân tán (distribution transparency)
  2. Trong suốt về giao dịch (transaction transparency)
  3. Trong suốt về hiệu suất (performance transparency)
  4. Trong suốt về DBMS (DBMS transparency)

A.IV.2 Trong suốt về giao dịch

- Bảo đảm tất cả các giao dịch phân tán duy trì tính toàn vẹn và tính nhất quán của CSDL phân tán.
- Các khái niệm:
  - *Giao dịch phân tán*: truy xuất dữ liệu được lưu ở nhiều hơn một vị trí.
  - Mỗi giao dịch chia thành các *giao dịch con*, một giao dịch con cho mỗi site cần truy xuất.
  - Một giao dịch con được biểu diễn bởi một *agent*.

A.IV.2 Trong suốt về giao dịch (2)

Liên quan 2 khía cạnh:

- Trong suốt về tính cạnh tranh (concurrency transparency)
- Trong suốt về sự cố (failure transparency)

A.IV.2 Trong suốt về giao dịch (3)

Trong suốt về tính cạnh tranh:

- Cung cấp bởi DDBMS, khi các kết quả của mọi giao dịch cạnh tranh (phân tán và không phân tán) được xử lý độc lập và nhất quán về luận lý với các kết quả có được khi các giao dịch được khai thác tuần tự.

A.IV.2 Trong suốt về giao dịch (4)

- *Bất lợi của tính trong suốt về cạnh tranh*:
  - Phức tạp hơn, vì DDBMS phải đảm bảo cả 2 loại giao dịch toàn cục và cục bộ không ảnh hưởng nhau.
  - DDBMS phải đảm bảo tính nhất quán của mọi giao dịch con của một giao dịch toàn cục.
  - Phép nhân bản làm cho sự cạnh tranh rối rắm hơn.

A.IV.2 Trong suốt về giao dịch (5)

Trong suốt về sự cố:

- Nói chung, có các loại sự cố sau:
  - Hỏng hệ thống
  - Hỏng thiết bị
  - Sai trong phần mềm
  - Vô ý
  - Thảm hoạ thiên nhiên
  - Bị phá hoại

A.IV.2 Trong suốt về giao dịch (6)

Trong suốt về sự cố (tt):

- Trong môi trường phân tán, phải để ý đến:
  - Mất thông báo
  - Hỏng đường truyền liên lạc
  - Hỏng một site
  - Phân chia một mạng (networ partitioning)
- *Giải pháp:* Phải đồng bộ hoá giao dịch toàn cục, sao cho mọi giao dịch con phải được hoàn tất thành công trước khi có COMMIT cuối cùng cho giao dịch toàn cục.

A.IV.2 Trong suốt về giao dịch (5)

Trong suốt về sự cố:

- Nói chung, có các loại sự cố sau:
  - Hỏng hệ thống
  - Hỏng thiết bị
  - Sai trong phần mềm
  - Vô ý
  - Thảm hoạ thiên nhiên
  - Bị phá hoại

A.IV.2 Trong suốt về giao dịch (6)

- Phân loại giao dịch: theo mức độ phúc tạp tăng dần trong tương tác giữa các DBMS:
  - Câu hỏi từ xa (remote request)
  - Đơn vị làm việc từ xa (remote unit of work)
  - Đơn vị làm việc phân tán (distributed unit of work)
  - Câu hỏi phân tán (distributed request)
- Có thể hiểu:
  - request= câu hỏi SQL
  - unit of work= giao dịch

A.IV.2 Trong suốt về giao dịch (7)

Câu hỏi từ xa:

- Một trình ứng dụng trên một site có thể gửi một câu hỏi SQL đến một site khác để xử lý.
- Câu hỏi được xử lý hoàn toàn trên site ở xa và chỉ có thể tham chiếu đến dữ liệu trên site ở xa đó.

A.IV.2 Trong suốt về giao dịch (8)

Đơn vị làm việc từ xa:

- Một trình ứng dụng trên một site cục bộ có thể gửi tất cả các câu SQL trong một đơn vị làm việc (giao dịch) đến một site khác để xử lý.
- Tất cả các câu hỏi SQL được xử lý hoàn toàn trên site ở xa và chỉ có thể tham chiếu đến dữ liệu trên site ở xa đó.
- Tuy nhiên, site cục bộ sẽ quyết định xem giao dịch có hoàn tất (commit) được không hay phải thoái lui (roll back).

A.IV.2 Trong suốt về giao dịch (9)

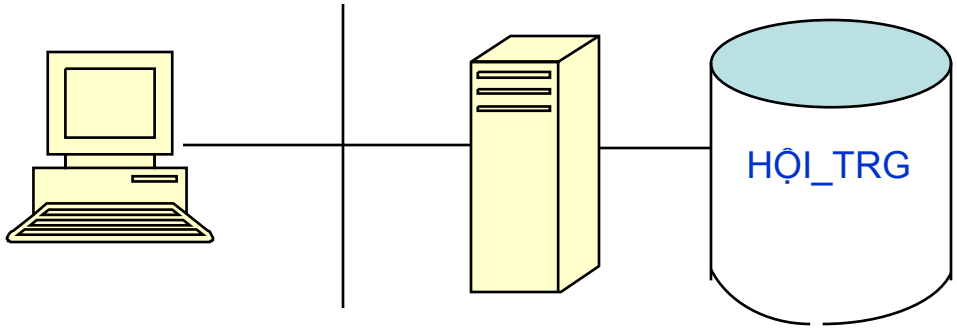
Đơn vị làm việc phân tán:

- Một trình ứng dụng trên một site cục bộ có thể gửi một số hoặc tất cả các câu SQL trong một giao dịch đến một hoặc nhiều site khác để xử lý.
- Mỗi câu hỏi SQL được xử lý hoàn toàn trên site ở xa và chỉ có thể tham chiếu đến dữ liệu trên site ở xa đó.
- Tuy nhiên, các câu hỏi SQL khác nhau có thể được xử lý trên các site khác nhau.
- Như trước, site cục bộ sẽ quyết định xem giao dịch có hoàn tất (commit) được không hay phải thoái lui (roll back).

A.IV.2 Trong suốt về giao dịch (10)

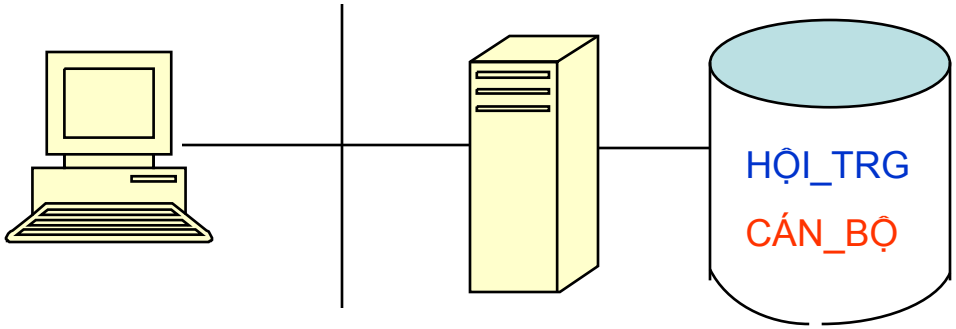
Câu hỏi phân tán:

- Một trình ứng dụng trên một site cục bộ có thể gửi một số hoặc tất cả các câu SQL trong một giao dịch đến một hoặc nhiều site khác để xử lý.
- Tuy nhiên, một câu hỏi SQL có thể yêu cầu dữ liệu trên nhiều hơn một site.
- Có thể cần phép hội hoặc phép kết nối cho các quan hệ hoặc các đoạn trên các site khác nhau.



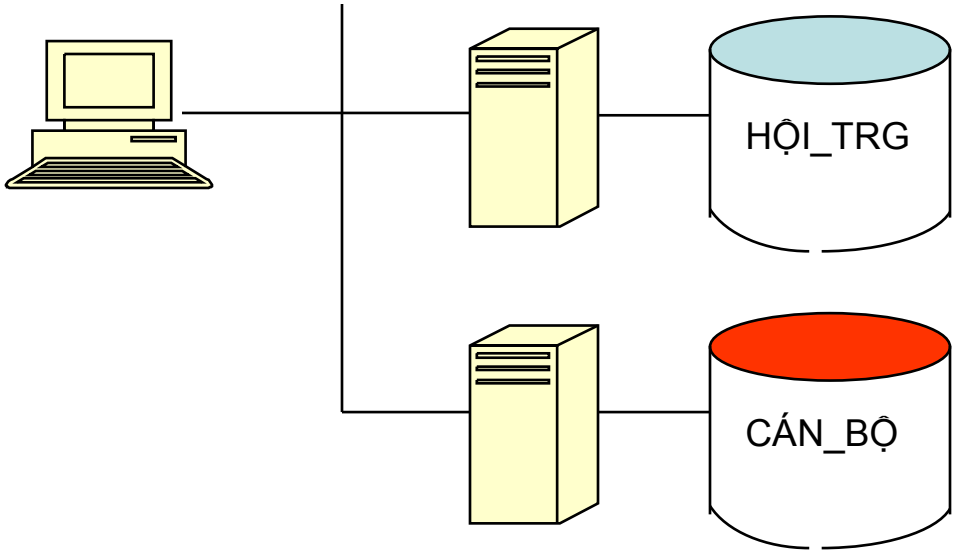

```
SELECT *
FROM HỘI_TRG
WHERE SỐ_CHỖ>200
```

Câu hỏi từ xa



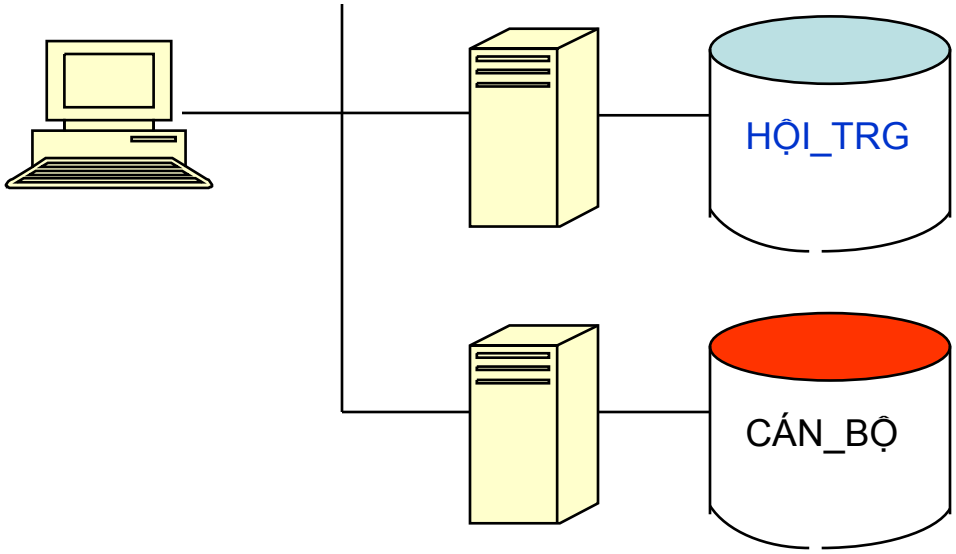

```
UPDATE HỘI_TRG
SET TÊN_HTR="Hội trường Rùa"
WHERE MÃ_HTR="01"
UPDATE CÁN_BỘ
SET ĐT="07103"+ĐT
COMMIT
```

Giao dịch từ xa




```
UPDATE HỘI_TRG
SET TÊN_HTR="Hội trường Rùa"
WHERE MÃ_HTR="01"
UPDATE CÁN_BỘ
SET ĐT="07103"+ĐT
COMMIT
```

Giao dịch phân tán




```
UPDATE HỘI_TRG
SET TÊN_HTR="Hội trường Rùa"
WHERE MÃ_HTR="01"
UPDATE CÁN_BỘ
SET ĐT="07103"+ĐT

SELECT A.*, HỌ, TÊN
FROM HỘI_TRG A, CÁN_BỘ B
WHERE A.MÃ_CB_QL=B.MÃ_CB
COMMIT
```

Câu hỏi phân tán

A.IV Tính trong suốt trong DDBMS (2)

- Các kiểu trong suốt trong DDBMS:
  1. Trong suốt về sự phân tán (distribution transparency)
  2. Trong suốt về giao dịch (transaction transparency)
  3. Trong suốt về hiệu suất (performance transparency)
  4. Trong suốt về DBMS (DBMS transparency)

A.IV.3 Trong suốt về hiệu suất

- Trong môi trường phân tán, hệ thống phải không chịu sụt giảm hiệu suất do kiến trúc phân tán.
- Yêu cầu DBMS xác định chiến lược chi phí- hiệu quả phù hợp khi khai thác một câu hỏi.
- Bộ xử lý câu hỏi (DQP: Distributed Query Processor) ánh xạ câu hỏi yêu cầu dữ liệu thành một chuỗi thứ tự các phép toán trên CSDL cục bộ.

⇨ Phức tạp thêm do phép phân đoạn, phép nhân bản và các sơ đồ cấp phát.

A.IV Tính trong suốt trong DDBMS (2)

- Các kiểu trong suốt trong DDBMS:
  1. Trong suốt về sự phân tán (distribution transparency)
  2. Trong suốt về giao dịch (transaction transparency)
  3. Trong suốt về hiệu suất (performance transparency)
  4. Trong suốt về DBMS (DBMS transparency)

A.IV.4 Trong suốt về DBMS

- Che đi việc các DBMS có thể khác nhau, do đó chỉ có thể áp dụng DDBMS bất đồng.
- Một trong những sự trong suốt khó khăn nhất là cung cấp sự tổng quát hoá.

A. HQTCSDL phân tán (DDBMS)

- I. Giới thiệu
- II. Chức năng và kiến trúc của DDBMS
- III. Thiết kế CSDL quan hệ phân tán
- IV. Tính trong suốt trong DDBMS
- V. 10 qui tắc của Date cho DDBMS

A.V 12 qui tắc của Date cho DDBMS

- 1. Tự chủ cục bộ:
  - Dữ liệu cục bộ được sở hữu và quản trị một cách cục bộ
  - Các phép toán cục bộ vẫn giữ tính chất cục bộ thuần tuý
  - Mọi phép toán trên 1 site sẽ được site đó kiểm soát
- 2. Không phó thác cho site trung tâm:
  - Không nên có site nào mà nếu thiếu nó thì hệ thống không thể hoạt động được.
  - Do đó, không nên có server trung tâm cho các hoạt động như quản trị giao dịch, tìm deadlock, tối ưu hoá câu hỏi, và quản trị biên mục hệ thống toàn cục (GSC).

A.V 12 qui tắc của Date cho DDBMS

- 3. Hoạt động liên tục (continous operation):

  Không nên dự trù có tắt hệ thống khi thực hiện các phép toán:

  – Thêm vào/ tách ra một site không hệ thống
  – Tạo và xoá động các đoạn ở ít nhất một site.

- 4. Độc lập về vị trí:
  – Tương đương với trong suốt về vị trí.
  – Người dùng cần có thể truy cập mọi dữ liệu từ bất cứ site nào, như thể chúng được lưu trên site của họ, không phải bận tâm việc dữ liệu lưu trữ vật lý ở đâu.

A.V 12 qui tắc của Date cho DDBMS (2)

- 5. Độc lập về phân đoạn:
  - Người dùng cần có thể truy cập dữ liệu, không phải bận tâm việc dữ liệu được phân đoạn như thế nào.
- 6. Độc lập về nhân bản:
  - Người dùng không nên biết dữ liệu đã được nhân bản.
  - Như vậy, họ sẽ không thể truy cập trực tiếp một bản sao đặc thù nào đó của một mục dữ liệu, cũng không thể cập nhật mọi bản sao của một mục dữ liệu một cách đặc biệt .

A.V 12 qui tắc của Date cho DDBMS (3)

- 7. Xử lý câu hỏi phân tán:
  - Hệ thống phải có khả năng xử lý các câu hỏi tham chiếu đến dữ liệu chứa ở nhiều hơn một site.
- 8. Xử lý giao dịch phân tán: Hệ thống phải
  - xem giao dịch như một đơn vị phục hồi.
  - Đảm bảo các giao dịch toàn cục và cục bộ đều thoả các tính chất ACID.

A.V 12 qui tắc của Date cho DDBMS (3)

- 9. Độc lập về phần cứng:
  - Phải chạy được DDBMS trên nhiều nền phần cứng khác nhau.
- 10. Độc lập về hệ điều hành:
  - Phải chạy được DDBMS trên nhiều hệ điều hành khác nhau.

A.V 12 qui tắc của Date cho DDBMS (4)

- 11. Độc lập về mạng:
  - Phải chạy được DDBMS trên nhiều mạng truyền thông khác nhau.
- 12. Độc lập về CSDL:
  - Phải có thể có một DDBMS từ nhiều DBMS khác nhau, hỗ trợ nhiều mô hình dữ liệu nền khác nhau.

  ⇨ Hệ thống phải hổ trợ được tính bất đồng.

B. HQTCSDL

hướng đối tượng

Đại học Công Nghệ Thông Tin

Môn: Cơ Sở Dữ Liệu

Thời lượng: 45LT+30TH

Nội dung

- Bài 1: Các khái niệm của một hệ CSDL
- Bài 2: Các mô hình CSDL
- Bài 3: Mô hình dữ liệu quan hệ (của Codd)
- Bài 4: Ngôn ngữ đại số quan hệ
- Bài 5: Ngôn ngữ SQL
- Bài 6: Ngôn ngữ tân từ
- Bài 7: Ràng buộc toàn vẹn trong một CSDL
- Bài 8: Tối ưu hóa câu hỏi bằng đại số quan hệ

Bài 1: Các khái niệm của một hệ CSDL

Bài 1: Các khái niệm của một hệ CSDL

1. Giới thiệu
2. Hệ thống tập tin (File System)
3. Định nghĩa một CSDL
4. Các đối tượng sử dụng CSDL
5. Hệ quản trị CSDL
6. Các mức của một CSDL

1. Giới thiệu

- Tại sao cần phải có một cơ sở dữ liệu?

2. Hệ thống tập tin

- Là tập hợp các tập tin riêng lẻ phục vụ cho một mục đích của đơn vị sử dụng.
- **Ưu điểm**:
  - Triển khai ứng dụng nhanh
  - Khả năng đáp ứng nhanh chóng, kịp thời (vì chỉ phục vụ cho mục đích hạn hẹp)
- **Khuyết điểm**:
  - Trùng lắp dữ liệu [redacted]ng phí, dữ liệu không nhất quán
  - Chi phí cao
  - Chia sẽ dữ liệu kém

3. Cơ sở dữ liệu (1)

- **Định nghĩa**:

  Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc, được lưu trữ trên các thiết bị lưu trữ nhằm thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với những mục đích khác nhau

3. Cơ sở dữ liệu (2)

- **Ưu điểm:**
  - Giảm trùng lắp thông tin xuống mức thấp nhất, đảm bảo tính nhất quán và toàn vẹn dữ liệu.
  - Đảm bảo dữ liệu được truy xuất theo nhiều cách khác nhau.
  - Khả năng chia sẽ thông tin cho nhiều người, nhiều ứng dụng khác nhau.

3. Cơ sở dữ liệu (3)

- Những vấn đề cần giải quyết:
  - Tính chủ quyền dữ liệu.
  - Tính bảo mật và quyền khai thác thông tin của người sử dụng
  - Tranh chấp dữ liệu
  - Đảm bảo dữ liệu khi có sự cố

4. Các đối tượng sử dụng

- Người sử dụng CSDL không chuyên về lĩnh vực tin học và CSDL -> cần công cụ để họ có thể khai thác CSDL khi cần.
- Chuyên viên tin học xây dựng các ứng dụng để phục vụ cho các mục đích quản lý
- Quản trị CSDL: tổ chức CSDL, bảo mật, cấp quyền, sao lưu, phục hồi dữ liệu, giải quyết các tranh chấp dữ liệu …

5. Hệ quản trị cơ sở dữ liệu (1)

- Hệ quản trị cơ sở dữ liệu (DBMS – DataBase Management System) là hệ thống các phần mềm hỗ trợ tích cực cho các nhà phân tích, thiết kế và khai thác CSDL.
- Các DBMS thông dụng: Visual FoxPro, Microsoft Access, SQL Server, DB2, Oracle … hầu hết các DBMS hiện nay đều dựa trên mô hình quan hệ.

5. Hệ quản trị cơ sở dữ liệu (2)

- Một DBMS phải có:
  - Ngôn ngữ giao tiếp giữa người sử dụng và CSDL
  - Từ điển dữ liệu (Data Dictionary)
  - Có biện pháp bảo mật khi có yêu cầu
  - Cơ chế giải quyết tranh chấp dữ liệu
  - Có cơ chế sao lưu (backup), phục hồi (restore)
  - Đảm bảo tính độc lập giữa dữ liệu và chương trình

5. Hệ quản trị cơ sở dữ liệu (3)

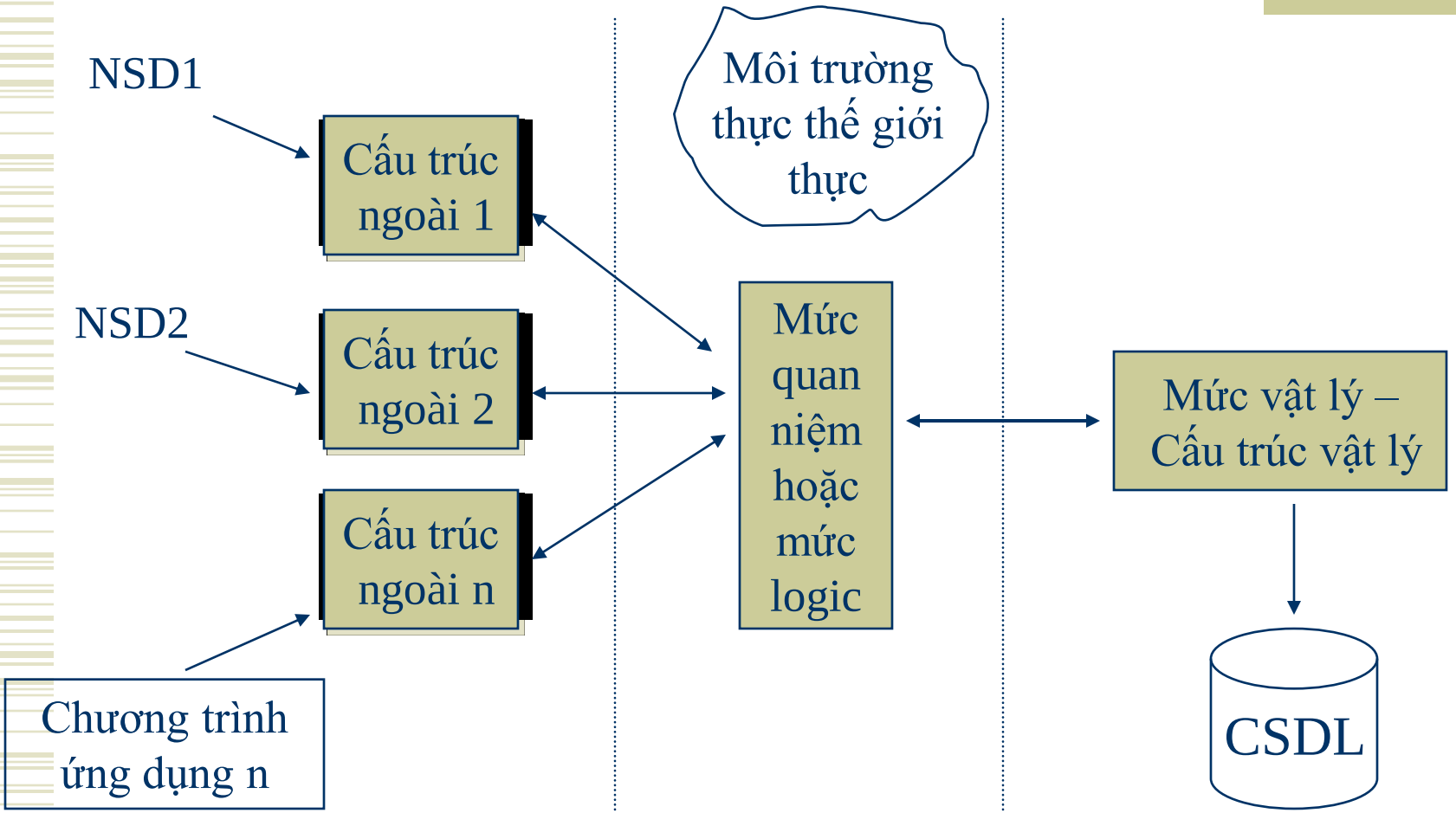
Ngôn ngữ giao tiếp:

- Ngôn ngữ mô tả dữ liệu (*DDL – Data Definition Language*): cho phép khai báo cấu trúc CSDL, các mối liên hệ của dữ liệu, các quy định, ràng buộc dữ liệu.
- Ngôn ngữ thao tác dữ liệu (*DML – Data Manipulation Language*): cho phép thực hiện thao tác thêm, xóa, sửa dữ liệu.
- Ngôn ngữ truy vấn có cấu trúc (SQL – *Structured Query Language*): cho phép người khai thác sử dụng để truy vấn thông tin cần thiết.
- Ngôn ngữ quản lý dữ liệu (DCL – *Data Control Language*) cho phép thay đổi cấu trúc bảng, khai báo bảo mật, cấp quyền cho người sử dụng.

6. Các mức biểu diễn 1 CSDL

- **Mức trong**: (mức vật lý – Physical) là mức lưu trữ CSDL (cần giải quyết vấn đề gì? Dữ liệu gì? Lưu trữ như thế nào? ở đâu? Cần các chỉ mục gì? Truy xuất tuần tự hay ngẫu nhiên. Dành cho người quản trị và người sử dụng chuyên môn.
- **Mức quan niệm**: (Conception hay Logical) cần phải lưu trữ bao nhiêu loại dữ liệu? là dữ liệu gì? mối quan hệ
- **Mức ngoài**: của người sử dụng và các chương trình ứng dụng

6. Các mức biểu diễn 1 CSDL



Bài 2: Các mô hình dữ liệu

Nội dung

1. Giới thiệu
2. Mô hình dữ liệu mạng
3. Mô hình thực thể mối kết hợp

1. Giới thiệu

Mô hình dữ liệu là sự trừu tượng hóa của môi trường thực, biểu diễn dữ liệu ở mức quan niệm. Giới thiệu một số mô hình như:

- Mô hình dữ liệu mạng
- Mô hình dữ liệu phân cấp
- Mô hình dữ liệu thực thể mối kết hợp
- Mô hình dữ liệu quan hệ
- Mô hình dữ liệu hướng đối tượng

2. Mô hình dữ liệu mạng (1)

- Mô hình dữ liệu mạng (*Network Data Model*) còn gọi tắt là mô hình mạng hoặc mô hình lưới là mô hình được biểu diễn bởi một đồ thị có hướng.
  - Mẫu tin (record)
  - Loại mẫu tin
  - Loại liên hệ (set type)
  - Bản số

2. Mô hình dữ liệu mạng (2)

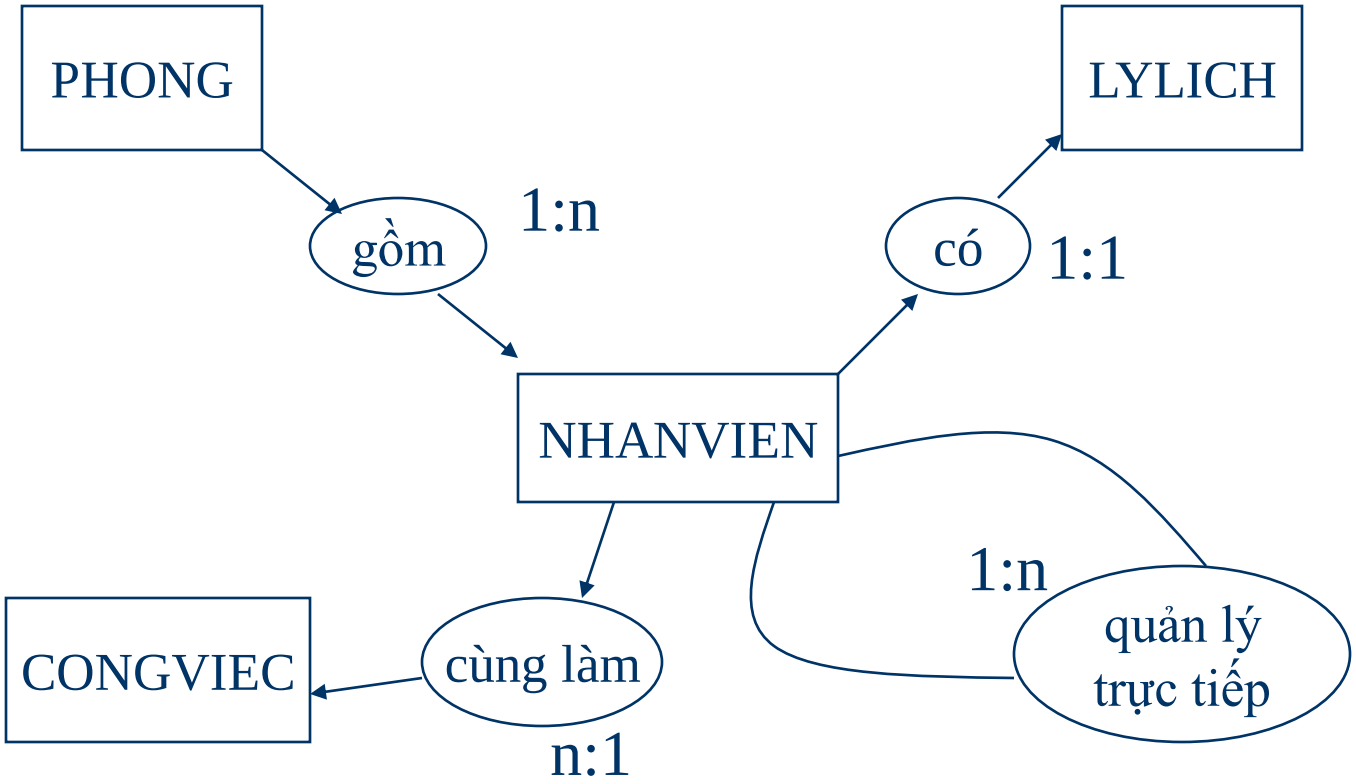
- **Mẫu tin**: mô tả 1 đối tượng trong thế giới thực.
  ('NV001','Nguyen Lam','Nam','10/10/1970','Dong Nai')
- **Loại mẫu tin**: là 1 tập các mẫu tin có cùng tính chất. Ví dụ: NHANVIEN
  - Ký hiệu:

- **Loại liên hệ**: mô tả sự liên kết giữa 1 loại mẫu tin chủ và 1 loại mẫu tin thành viên
  - Ký hiệu:

2. Mô hình dữ liệu mạng (3)

- **Bản số:** chỉ ra số lượng các mẫu tin tham gia trong mối liên hệ
  - **(1:1)** (one-to-one): mỗi mẫu tin của loại mẫu tin chủ kết hợp với đúng 1 mẫu tin của loại mẫu tin thành viên.
  - **(1:n)** (one-to-many): mỗi mẫu tin của loại mẫu tin chủ kết hợp với 1 hay nhiều mẫu tin thành viên.
  - **(n:1)** (many-to-one): nhiều mẫu tin của loại mẫu tin chủ kết hợp với đúng 1 mẫu tin của loại mẫu tin thành viên.
  - (Recursive): một loại mẫu tin chủ cũng có thể đồng thời là loại mẫu tin thành viên với chính nó. Loại liên hệ này là Đệ quy

2. Mô hình dữ liệu mạng (4)



2. Mô hình dữ liệu mạng (5)

- Mô hình dữ liệu mạng:
  - Tương đối đơn giản
  - Dễ sử dụng
  - Không thích hợp biểu diễn CSDL có quy mô lớn
  - Khả năng diễn đạt ngữ nghĩa kém

2. Mô hình dữ liệu mạng (6)

- **Bài tập**:

  Xây dựng mô hình dữ liệu mạng cho cơ sở dữ liệu quản lý bán hàng trong một siêu thị (giáo viên mô tả hiện trạng, xác định các yêu cầu)

3. Mô hình thực thể mối kết hợp

3.1 Giới thiệu
3.2 Loại thực thể, thực thể
3.3 Thuộc tính của loại thực thể
3.4 Khoá của loại thực thể
3.5 Loại mối kết hợp, mối kết hợp
3.6 Thuộc tính của loại mối kết hợp
3.7 Bản số
3.8 Mô hình ER mở rộng

3.1 Giới thiệu

- Mô hình thực thể mối kết hợp (*Entity-Relationship Model* viết tắc ER) được CHEN giới thiệu năm 1976.
- Mô hình ER được sử dụng nhiều trong thiết kế dữ liệu ở mức quan niệm.

3.2 Loại thực thể

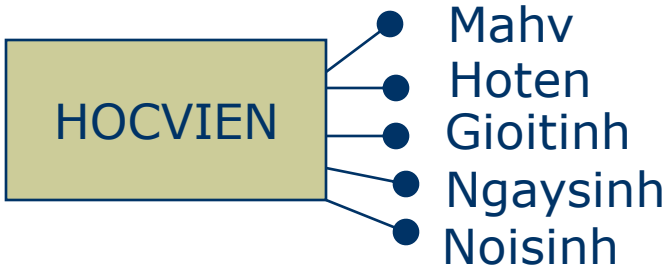
- **Định nghĩa**: loại thực thể (Entity Type) là những loại đối tượng hay sự vật của thế giới thực tồn tại cụ thể cần được quản lý.
- **Ví dụ** : HOCVIEN, LOP, MONHOC, …
- **Ký hiệu**:

3.2 Thực thể (Entity)

- **Định nghĩa**: thực thể là một thể hiện hoặc một đối tượng của một loại thực thể.
- **Ví dụ**: Loại thực thể là HOCVIEN có các thực thể:
  - (‘HV001’, ‘Nguyen Nam’, ‘1/2/1987’,’Nam’)
  - (‘HV002’, ‘Trần Nam’, ‘13/2/1987’, ‘Nam’)

3.3 Thuộc tính của loại thực thể (Entity Attribute)

- **Định nghĩa**: thuộc tính là những tính chất đặc trưng của loại thực thể cần lưu trữ.
- **Ví dụ**: Loại thực thể HOCVIEN có các thuộc tính: Mã học viên, họ tên, giới tính, ngày sinh, nơi sinh
- **Ký hiệu**:



3.3 Các loại thuộc tính (1)

- ***Đơn trị (Simple)***: mỗi thực thể chỉ có một giá trị ứng với các thuộc tính.

  Ví dụ: Mahv,Hoten

- ***Đa hợp (Composite)***: thuộc tính có thể được tạo thành từ nhiều thành phần.

  Ví dụ: DCHI(SONHA,DUONG,PHUONG,QUAN)
  hay thuộc tính HOTEN(HO,TENLOT,TEN).

- ***Đa trị (Multi-valued)***: thuộc tính có thể có nhiều giá trị đối với một thực thể.

  Ví dụ: BANGCAP ký hiệu {BANGCAP}

3.3 Các loại thuộc tính (2)

- Tóm lại, các thuộc tính đa hợp và đa trị có thể lồng nhau tùy ý.
  - **Ví dụ**: thuộc tính BANGCAP của HOCVIEN là một thuộc tính đa hợp được ký hiệu bằng

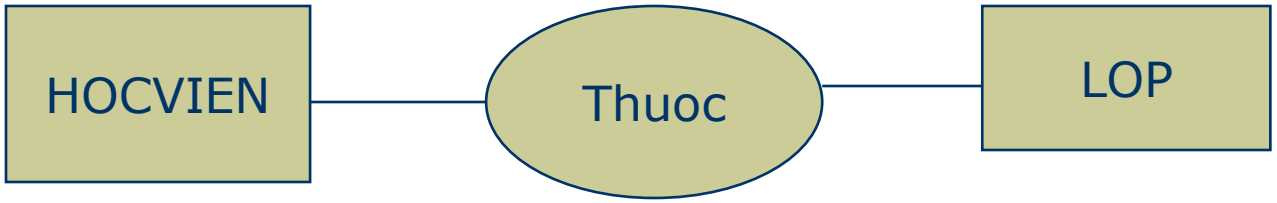
{BANGCAP(TRUONGCAP,NAM,KETQUA, CHUYENNGANH)}

3.4. Khoá của loại thực thể (entity type key)

- Khóa của loại thực thể là thuộc tính nhận diện thực thể.
- Căn cứ vào giá trị của khóa có thể xác định **duy nhất** một thực thể.
- **Ví dụ**:
  - Mỗi học viên có một mã số duy nhất => Khoá của loại thực thể **HOCVIEN** là **Mã học viên**

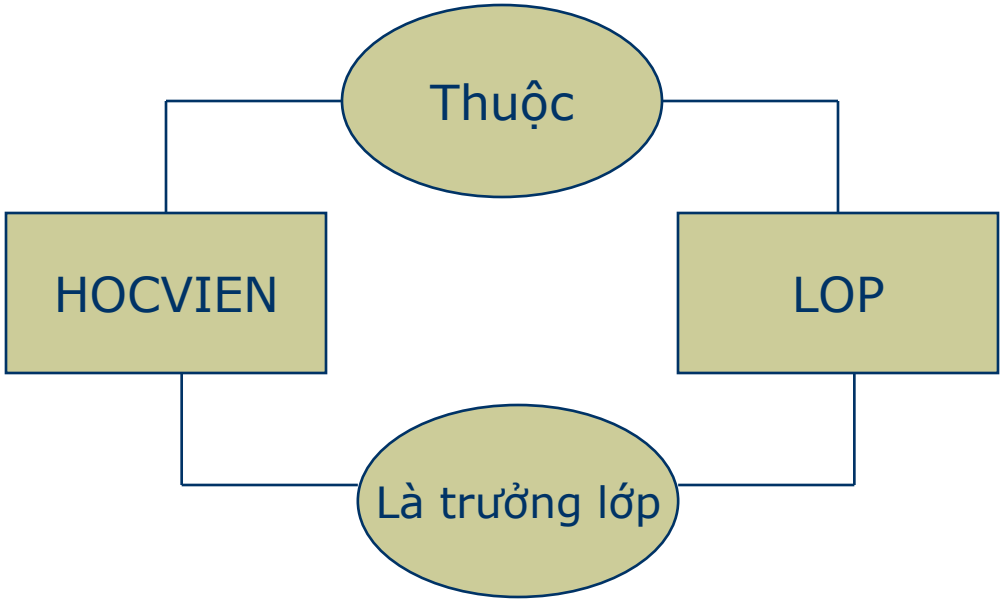
3.5 Loại mối kết hợp (1) (relationship type)

- **Định nghĩa**: loại mối kết hợp là sự kết hợp giữa hai hay nhiều loại thực thể
- **Ví dụ**: giữa hai loại thực thể HOCVIEN và LOP có loại mối kết hợp THUOC
- **Ký hiệu**: bằng một hình oval hoặc hình thoi



3.5 Loại mối kết hợp (2)

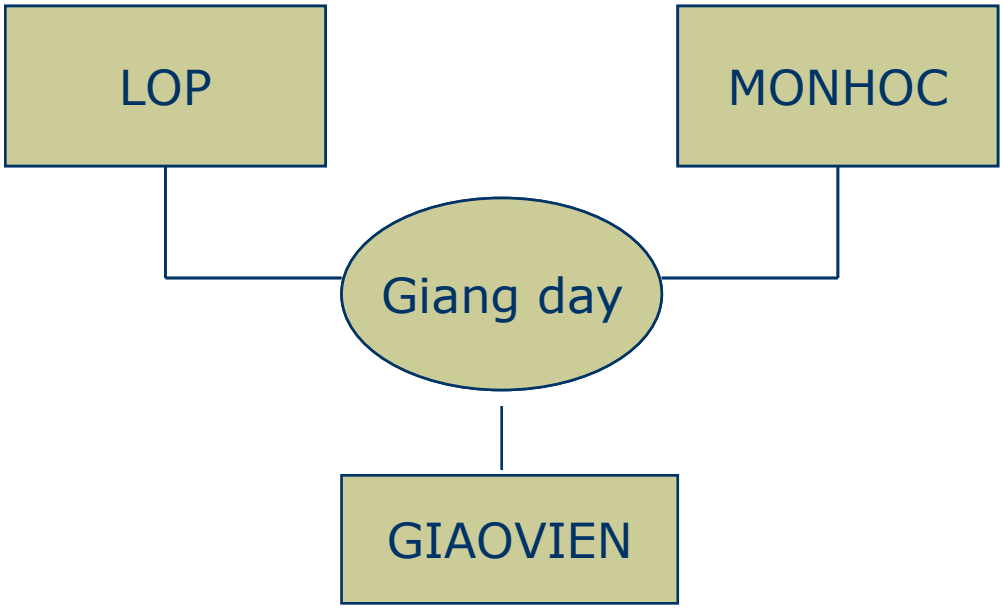
- Giữa hai loại thực thể có thể tồn tại nhiều hơn một loại mối kết hợp.
- **Ví dụ**



3.5 Số ngôi của loại mối kết hợp (relationship degree)

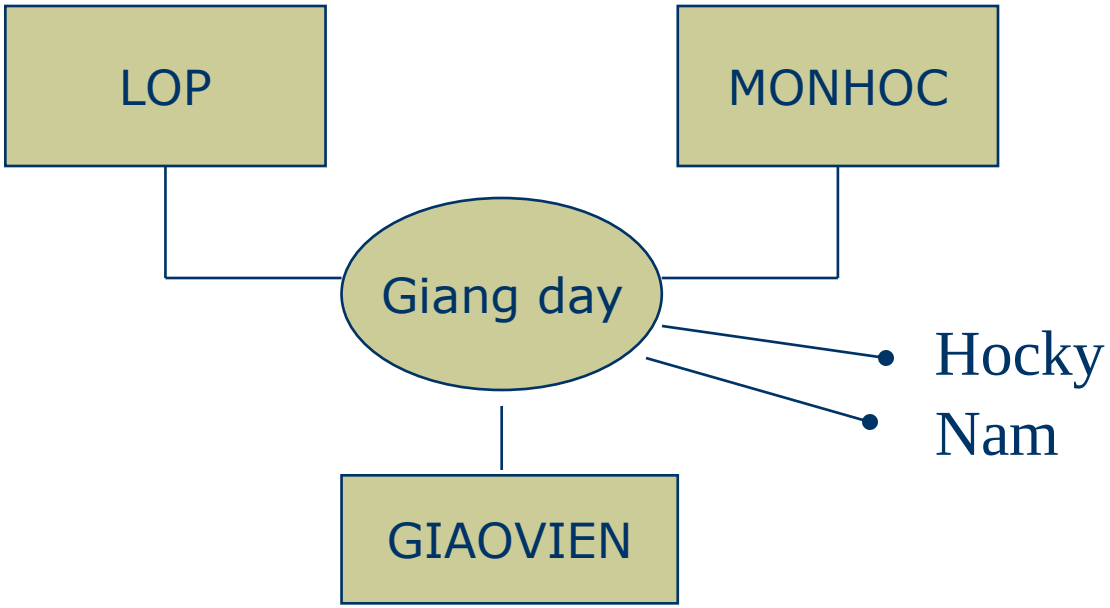
- Số ngôi của loại mối kết hợp là số loại thực thể tham gia vào loại mối kết hợp đó.
- **Ví dụ 1**: Loại mối kết hợp **Thuộc** kết hợp 2 loại thực thể HOCVIEN và LOP nên có số ngôi là 2.
- **Ví dụ 2**: Loại mối kết hợp **Giảng dạy** kết hợp 3 loại thực thể GIAOVIEN, MONHOC, LOP nên có số ngôi là 3.

3.5 Số ngôi của loại mối kết hợp



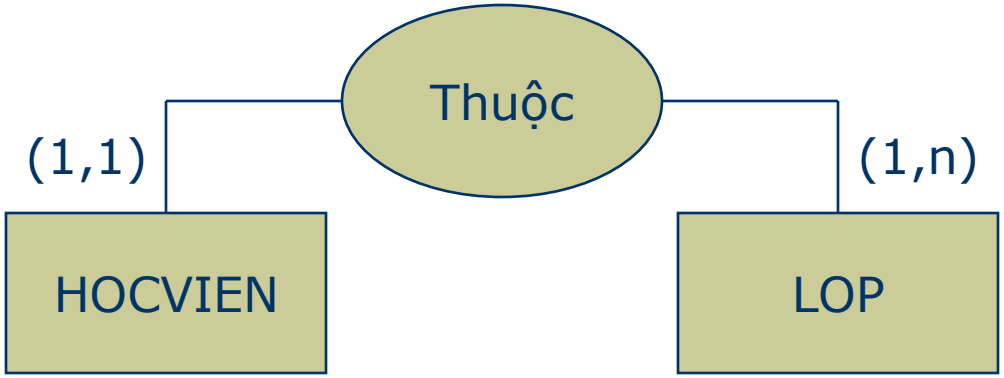
3.6 Thuộc tính của loại mối kết hợp (relationship type attribute)

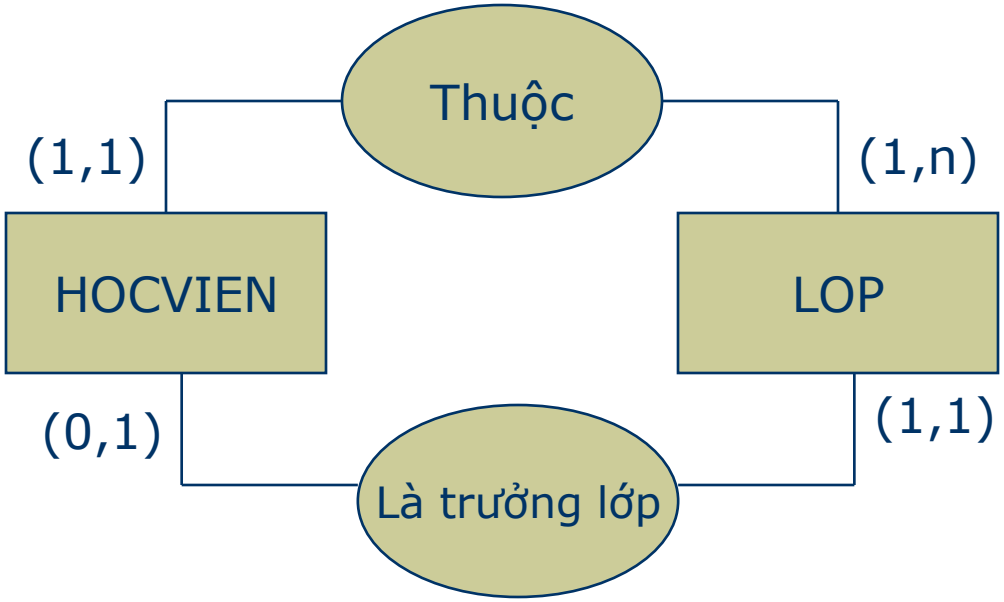
- Thuộc tính của loại mối kết hợp bao gồm các thuộc tính khoá của các loại thực thể tham gia vào loại mối kết hợp đó. Ngoài ra còn có thể có thêm những thuộc tính bổ sung khác.
- **Ví dụ**: Loại mối kết hợp **Giảng dạy** giữa ba loại thực thể GIAOVIEN, MONHOC và LOP có các thuộc tính là Magv,Mamh,Malop, ngoài ra còn có thuộc tính riêng là Hocky, Nam


LOP
MONHOC
Giang day
Hocky
Nam
GIAOVIEN

3.7 Bản số (relationship cardinality)

- Loại mối kết hợp thể hiện liên kết giữa các thực thể, mỗi liên kết được gọi là một nhánh.
- **Định nghĩa**: bản số của nhánh là số lượng tối thiểu và số lượng tối đa các thực thể thuộc nhánh đó tham gia vào loại mối kết hợp.
- **Ký hiệu**: (bản số tối thiểu, bản số tối đa)
- **Ví dụ**: Loại thực thể NhanVien và DeAn có loại mối kết hợp ThamGia.


Thuộc
(1,1)
(1,n)
HOCVIEN
LOP


Thuộc
(1,1)
(1,n)
HOCVIEN
LOP
(0,1)
(1,1)
Là trưởng lớp

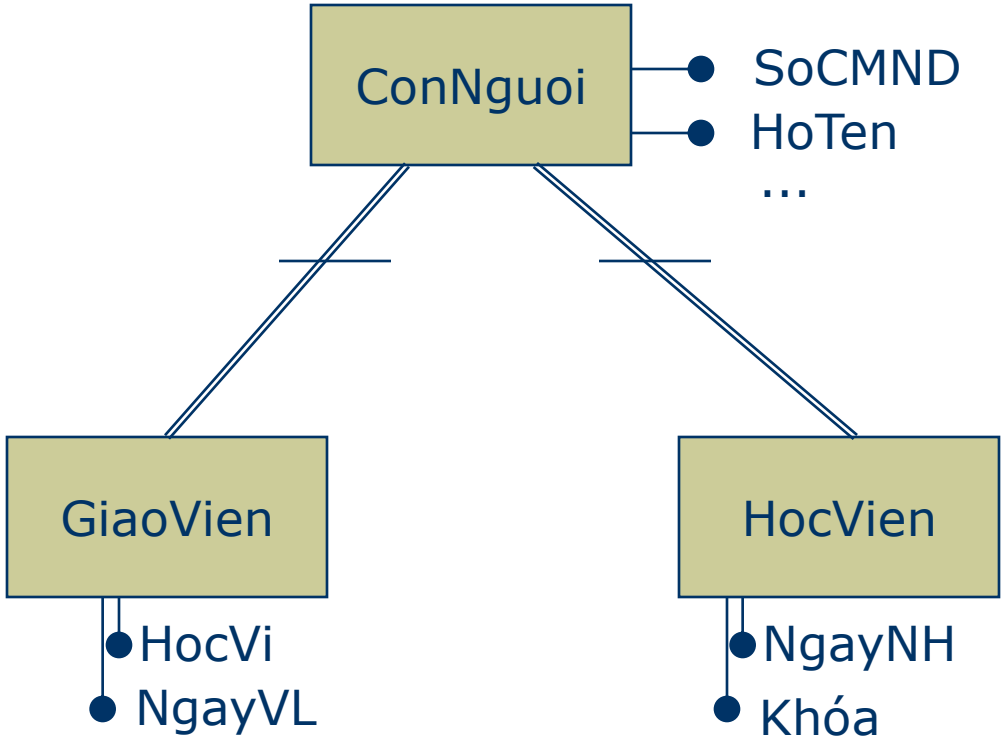
3.7 Mô hình ER mở rộng

3.7.1 Chuyên biệt hoá / Tổng quát hóa

3.7.2 Mối kết hợp đệ quy

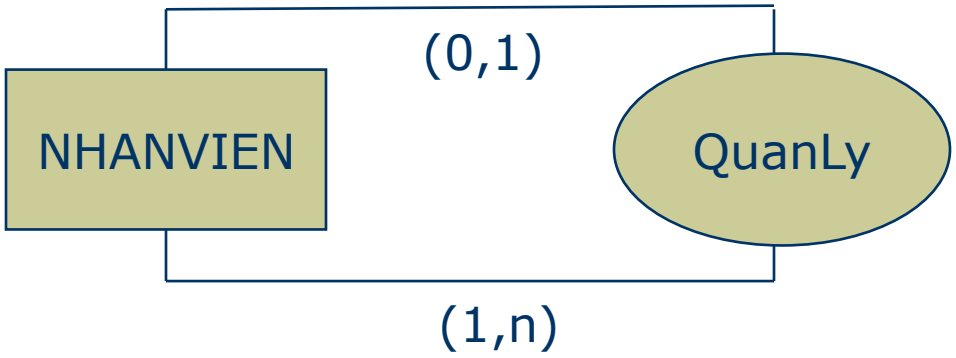
3.7.3 Loại thực thể yếu

3.7.1 Chuyên biệt hóa (tổng quát hóa)



3.7.2 Mối kết hợp đệ quy

- **Định nghĩa**: là loại mối kết hợp được tạo thành từ cùng một loại thực thể (hay một loại thực thể có loại mối kết hợp với chính nó)
- **Ví dụ**: Mỗi nhân viên có một người quản lý trực tiếp và người quản lý đó cũng là một nhân viên

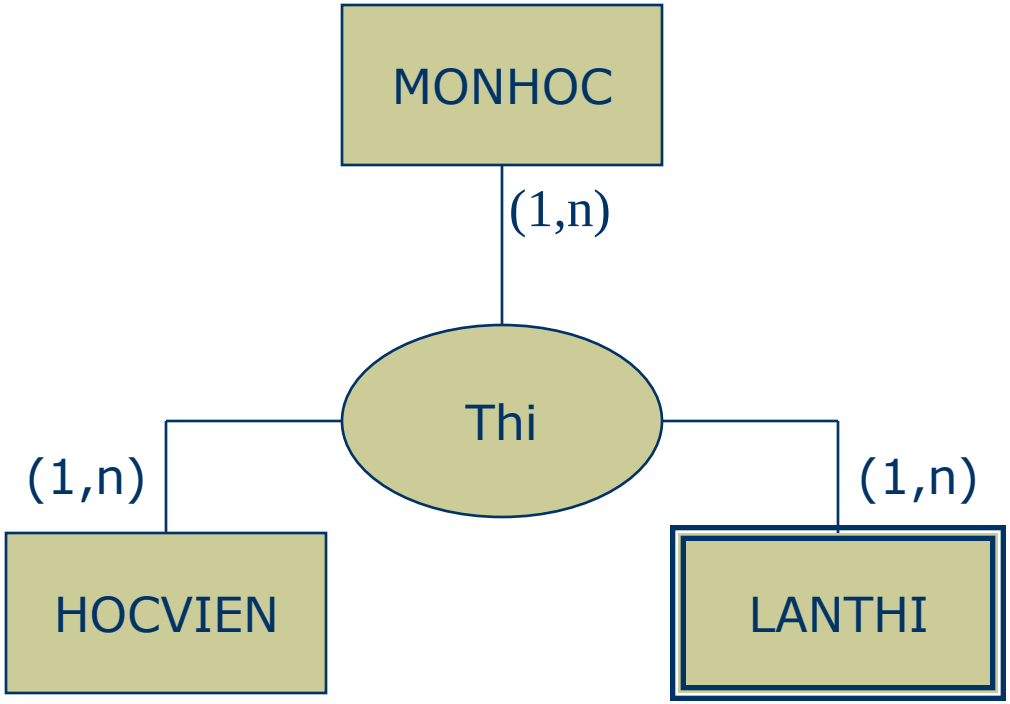


3.7.3 Loại thực thể yếu

- **Định nghĩa:**
  - Là loại thực thể không có thuộc tính khóa
  - Phải tham gia trong một loại mối kết hợp xác định trong đó có một loại thực thể chủ.
- **Ký hiệu**:

- **Ví dụ**: loại thực thể LANTHI có thuộc tính Lần và tham gia trong loại mối kết hợp **Thi** với loại thực thể HOCVIEN và MONHOC là loại thực thể yếu.

3.7.3 Loại thực thể yếu



Bài tập
Xây dựng mô hình ER

- Xây dựng mô hình ER cho CSDL quản lý giáo vụ gồm có các chức năng sau:
  - Lưu trữ thông tin: Học viên , giáo viên, môn học
  - Xếp lớp cho học viên, chọn lớp trưởng cho lớp
  - Phân công giảng dạy: giáo viên dạy lớp nào với môn học gì, ở học kỳ, năm học nào.
  - Lưu trữ kết quả thi: học viên thi môn học nào, lần thi thứ mấy, điểm thi bao nhiêu.

liệu quan hệ (Relational Data Model)

Nội dung

1. Giới thiệu
2. Các khái niệm
   - 2.1 Thuộc tính
   - 2.2 Quan hệ
   - 2.3 Bộ giá trị
   - 2.4 Thể hiện của quan hệ
   - 2.5 Tân từ
   - 2.6 Phép chiếu
   - 2.7 Khóa
   - 2.8 Lược đồ quan hệ và lược đồ CSDL
   - 2.9 Hiện thực mô hình ER bằng mô hình dữ liệu quan hệ.

1. Giới thiệu

- Mô hình Dữ liệu Quan hệ (*Relational Data Model*) dựa trên khái niệm quan hệ.
- *Quan hệ* là khái niệm toán học dựa trên nền tảng hình thức về lý thuyết tập hợp.
- Mô hình này do TS. E. F. Codd đưa ra năm 1970.

2.1 Thuộc tính (attribute)

- Thuộc tính:
  - ***Tên gọi***: dãy ký tự (gợi nhớ)
  - ***Kiểu dữ liệu***: Số, Chuỗi, Thời gian, Luận lý, OLE.
  - ***Miền giá trị***: tập giá trị mà thuộc tính có thể nhận. Ký hiệu miền giá trị của thuộc tính A là ***Dom***(A).
- **Ví dụ**:GIOITINH kiểu dữ liệu là Chuỗi,miền giá trị

  Dom(GIOITINH)=(‘Nam’,’Nu’)
- Tại một thời điểm, một thuộc tính không có giá trị hoặc chưa xác định được giá trị => giá trị Null

2.2 Quan hệ (relation)

- **Định nghĩa**: quan hệ là một tập hữu hạn các thuộc tính.
  - **Ký hiệu**: $Q(A_1, A_2, ..., A_n)$
  - Trong đó Q là tên quan hệ, $Q(A_1, A_2, ..., A_n)$ là tập các thuộc tính của quan hệ Q
  - **Ví dụ**:

HOCVIEN (Mahv, Hoten, Ngsinh, Gioitinh, Noisinh, Malop)

LOP (Malop, Tenlop, Siso, Trglop, Khoa)

2.3 Bộ (tuple)

- **Định nghĩa:** Bộ là các thông tin của một đối tượng thuộc quan hệ, được gọi là mẫu tin (record), dòng.
- Quan hệ là một bảng (table) với các cột là các thuộc tính và mỗi dòng được gọi là bộ.
- Một bộ của quan hệ $Q(A_1, A_2, ..., A_n)$ là $Q(A_1, A_2, ..., A_n)$ với [redacted]$om(A_i)$
- **Ví dụ**: HOCVIEN(Mahv, Hoten, Ngsinh, Noisinh) có q=(1003,Nguyen Van Lam, 1/1/1987,Dong Nai) nghĩa là học viên có mã số là 1003, họ tên là Nguyen Van Lam, sinh ngày 1/1/1987 ở Dong Nai

2.4 Thể hiện của quan hệ (instance)

- **Định nghĩa**: thể hiện của một quan hệ là tập hợp các bộ giá trị của quan hệ tại một thời điểm nhất định.
- **Ký hiệu:** thể hiện của quan hệ Q là T_Q
- **Ví dụ**: T_{HOCVIEN} là thể hiện của quan hệ HOCVIEN tại thời điểm hiện tại gồm có các bộ như sau:

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11

2.5 Tân từ

- **Định nghĩa**: tân từ là một quy tắc dùng để mô tả một quan hệ.
- **Ký hiệu**: ||Q||
- **Ví dụ**: THI (Mahv, Mamh, Lanthi, Diem)
  ||THI||: mỗi học viên được phép thi một môn học nhiều lần, mỗi lần thi lưu trữ học viên nào thi môn gì? lần thi thứ mấy? và điểm là bao nhiêu?

2.6 Phép chiếu (1)

- **Phép chiếu** : Dùng để trích giá trị của một số thuộc tính trong danh sách các thuộc tính của quan hệ.
- **Ký hiệu**: phép chiếu của quan hệ R lên tập thuộc tính A là ***R[A] hoặc R.A***.
- **Ví dụ**:

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11

- hv_1=
- hv_2 =
- hv_3 =

2.6 Phép chiếu (2)

- Phép chiếu của quan hệ HOCVIEN lên thuộc tính NoiSinh của quan hệ HOCVIEN:

HOCVIEN[Noisinh] = {‘Nghe An’,’Kien Giang’,’Tay Ninh’}

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11

2.6 Phép chiếu (3)

- Phép chiếu lên 1 tập thuộc tính K={Hoten,Noisinh} của quan hệ HOCVIEN

HOCVIEN[Hoten, Noisinh] = {{‘Ha Duy Lap’, ‘Nghe An’},{‘Tran Ngoc Han’, ‘Kien Giang’},{‘Tran Ngoc Linh’,’Tay Ninh’}}

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11

2.6 Phép chiếu (4)

- **Chiếu của một bộ lên tập thuộc tính:** dùng để trích chọn các giá trị cụ thể của bộ giá trị đó theo các thuộc tính được chỉ ra trong danh sách thuộc tính của một quan hệ.
- **Ký hiệu**: chiếu của một bộ giá trị t lên tập thuộc tính A của quan hệ R là $t_R[A]$ hoặc $t[A]$. Nếu A có 1 thuộc tính $t_R.A$
- **Ví dụ**: cho quan hệ HOCVIEN với tập thuộc tính $HOCVIEN^+$={Mahv,Hoten,Gioitinh,Noisinh,Malop}, chứa 3 bộ giá trị hv_1,hv_2 và hv_3

2.6 Phép chiếu (5)

- Phép chiếu 1 bộ lên 1 thuộc tính
  - hv_1[Hoten] = {'Ha Duy Lap'}

	HOCVIEN				
	Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
hv_1=	K1103	Ha Duy Lap	Nam	Nghe An	K11
hv_2=	K1102	Tran Ngoc Han	Nu	Kien Giang	K11
hv_3=	K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11

2.6 Phép chiếu (6)

- Phép chiếu 1 bộ lên 1 tập thuộc tính
  - tập thuộc tính K={Hoten, Gioitinh}
  - $hv_2[K]$ = {'Tran Ngoc Han','Nu'}

HOCVIEN

	Mahv	HoTen	Gioitinh	Noisinh	Malop
hv_1 =	K1103	Ha Duy Lap	Nam	Nghe An	K11
hv_2 =	K1102	Tran Ngoc Han	Nu	Kien Giang	K11
hv_3 =	K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11

2.7 Khóa

2.7.1 Siêu khóa (super key)

2.7.2 Khóa (key)

2.7.3 Khóa chính (primary key)

2.7.5 Khóa ngoại (foreign key)

2.7.1 Siêu khóa (super key) (1)

- ***Siêu khóa*** : là một tập con các thuộc tính của Q^+ mà giá trị của chúng có thể phân biệt 2 bộ khác nhau trong cùng một thể hiện $\boldsymbol{T}_Q$ bất kỳ.

  Nghĩa là: [redacted]$_1$, t_2 [redacted]$_Q$, t_1[K][redacted]$_2$[K][redacted]K là siêu khóa của Q.
- Một quan hệ có ít nhất một siêu khóa (Q^+) và có thể có nhiều siêu khóa.

2.7.1 Siêu khóa (super key) (2)

- Ví dụ: các siêu khóa của quan hệ HOCVIEN là: {Mahv};{Mahv,Hoten};{Hoten};{Noisinh,Hoten} …

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11
K1105	Tran Minh Long	Nam	TpHCM	K11
K1106	Le Nhat Minh	Nam	TpHCM	K11

2.7.2 Khóa (key) (1)

Khóa : K là khóa của quan hệ R, thỏa mãn 2 điều kiện:

- K là một siêu khóa.
- K là siêu khóa "*nhỏ nhất*" (chứa ít thuộc tính nhất và khác rỗng) nghĩa là

  ¬∃K_1 ⊂ K, K_1 ⇒ K_1 là siêu khóa.

- Thuộc tính tham gia vào một khóa gọi là *thuộc tính khóa*, ngược lại là *thuộc tính không khóa*.

2.7.2 Khóa (key) (2)

- **Ví dụ**: các siêu khóa của quan hệ HOCVIEN là: {Mahv};{Mahv,Hoten};{Hoten};{Hoten,Gioitinh}; {Noisinh,Hoten};{Mahv,Hoten,Gioitinh,Noisinh}…

  => thì khóa của quan hệ HOCVIEN có thể là {Mahv}; {Hoten}
- **Ví dụ**: khóa của quan hệ GIANGDAY là K={Magv,Mamh,Malop}. Như vậy thuộc tính khóa sẽ là: Magv,Mamh,Malop.

2.7.3 Khóa chính (primary key)

- Khi cài đặt trên một DBMS cụ thể, nếu quan hệ có nhiều hơn một ***khóa***, ta chỉ được chọn một và gọi là ***khóa chính***
- **Ký hiệu**: các *thuộc tính nằm trong khóa chính* khi liệt kê trong quan hệ phải được gạch dưới.
- **Ví dụ**:
  - HOCVIEN (Mahv,Hoten,Gioitinh,Noisinh,Malop)
  - GIANGDAY(Magv,Mamh,Malop,Hocky,Nam)

2.7.4 Khóa ngoại (1)

- Cho R(U), S(V). K_1 [redacted] là khóa chính của R,K_2 [redacted]
- Ta nói **K_2 là khóa ngoại** của S *tham chiếu đến* khóa chính K_1 của R nếu thỏa các điều kiện sau:
  - K_1 và K_2 có cùng số lượng thuộc tính và ngữ nghĩa của các thuộc tính trong K1 và K2 cũng giống nhau.
  - Giữa R và S tồn tại mối quan hệ 1-n trên K_1 và K_2,
  - [redacted], ![redacted] sao cho $r.K_1=s.K_2$

2.7.4 Khóa ngoại (2)

- Ví dụ, cho 2 quan hệ

  LOP (Malop,Tenlop,Siso,Khoahoc)

  HOCVIEN (Mahv,Hoten,Gioitinh,Noisinh,Malop)

- Thuộc tính Malop trong quan hệ LOP là khóa chính của quan hệ LOP. Thuộc tính *Malop trong quan hệ HOCVIEN là khóa ngoại, tham chiếu đến Malop trong quan hệ LOP*

2.7.4 Khóa ngoại (3)

HOCVIEN

Mahv	HoTen	Gioitinh	Noisinh	Malop
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11
K1105	Tran Minh Long	Nam	TpHCM	K11
K1106	Le Nhat Minh	Nam	TpHCM	K11

LOP

Malop	Tenlop	Trglop	Siso	Magvcn
K11	Lop 1 khoa 1	K1106	11	GV07
K12	Lop 2 khoa 1	K1205	12	GV09
K13	Lop 3 khoa 1	K1305	12	GV14

2.8 Lược đồ quan hệ (1)

- ***Lược đồ quan hệ*** nhằm mục đích ***mô tả cấu trúc*** của một quan hệ và ***các mối liên hệ*** giữa các thuộc tính trong quan hệ đó.
- ***Cấu trúc của một quan hệ*** là tập thuộc tính hình thành nên quan hệ đó.
- Một lược đồ quan hệ gồm một tập thuộc tính của quan hệ kèm theo một mô tả để xác định ý nghĩa và mối liên hệ giữa các thuộc tính

2.8 Lược đồ quan hệ (2)

- Lược đồ quan hệ được đặc trưng bởi:
  - Một tên phân biệt
  - Một tập hợp hữu hạn các thuộc tính (A_1, …, A_n)

- Ký hiệu của lược đồ quan hệ ***Q*** gồm ***n*** thuộc tính (A_1, A_2, ... A_n) là :
  - $Q(A_1, A_2, ..., A_n)$

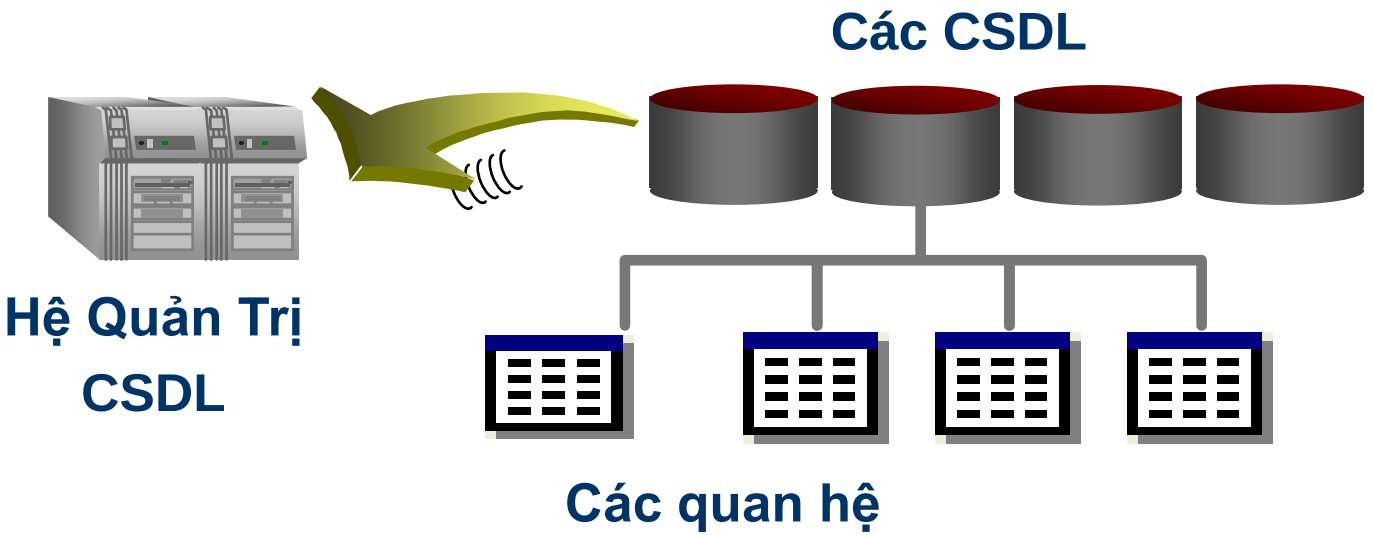
2.8 Lược đồ quan hệ (3)

- HOCVIEN(Mahv,Hoten,Gioitinh,Noisinh,Malop
- Tân từ: mỗi học viên có một mã học viên để phân biệt với các học viên khác. Cần lưu trữ họ tên, giới tính, nơi sinh và thuộc lớp nào.

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11
K1105	Tran Minh Long	Nam	TpHCM	K11
K1106	Le Nhat Minh	Nam	TpHCM	K11

2.8 Lược đồ CSDL (1)

- Là tập hợp gồm các lược đồ quan hệ và các mối liên hệ giữa chúng trong cùng một hệ thống quản lý.



Lược đồ CSDL quản lý sinh viên

HOCVIEN (MAHV, HO, TEN, NGSINH, GIOITINH, NOISINH, MALOP)

Tân từ: mỗi học viên phân biệt với nhau bằng mã học viên, lưu trữ họ tên, ngày sinh, giới tính, nơi sinh, thuộc lớp nào.

LOP (MALOP, TENLOP, TRGLOP, SISO, MAGVCN)

Tân từ: mỗi lớp gồm có mã lớp, tên lớp, học viên làm lớp trưởng của lớp, sỉ số lớp và giáo viên chủ nhiệm.

KHOA (MAKHOA, TENKHOA, NGTLAP, TRGKHOA)

Tân từ: mỗi khoa cần lưu trữ mã khoa, tên khoa, ngày thành lập khoa và trưởng khoa (cũng là một giáo viên thuộc khoa).

MONHOC (MAMH, TENMH, TCLT, TCTH, MAKHOA)

Tân từ: mỗi môn học cần lưu trữ tên môn học, số tín chỉ lý thuyết, số tín chỉ thực hành và khoa nào phụ trách.

DIEUKIEN (MAMH, MAMH_TRUOC)

Tân từ: có những môn học học viên phải có kiến thức từ một số môn học trước.

GIAOVIEN(MAGV,HOTEN,HOCVI,HOCHAM,GIOITINH,NGSINH,NGVL, HESO, MUCLUONG, MAKHOA)

Tân từ: mã giáo viên để phân biệt giữa các giáo viên, cần lưu trữ họ tên, học vị, học hàm, giới tính, ngày sinh, ngày vào làm, hệ số, mức lương và thuộc một khoa.

GIANGDAY(MALOP,MAMH,MAGV,HOCKY, NAM,TUNGAY,DENNGAY)

Tân từ: mỗi học kỳ của năm học sẽ phân công giảng dạy: lớp nào học môn gì do giáo viên nào phụ trách.

KETQUATHI (MAHV, MAMH, LANTHI, NGTHI, DIEM, KQUA)

Tân từ: lưu trữ kết quả thi của học viên: học viên nào thi môn học gì, lần thi thứ mấy, ngày thi là ngày nào, điểm thi bao nhiêu và kết quả là đạt hay không đạt.

Bài 4: Ngôn ngữ đại số quan hệ

Nội dung

1. Giới thiệu
2. Biểu thức đại số quan hệ
3. Các phép toán
4. Biểu thức đại số quan hệ
5. Ví dụ

1. Giới thiệu

- Đại số quan hệ (ĐSQH) có nền tảng toán học (cụ thể là lý thuyết tập hợp) để mô hình hóa CSDL quan hệ. Đối tượng xử lý là các quan hệ trong cơ sở dữ liệu quan hệ.
- **Chức năng**:
  - Cho phép mô tả các phép toán rút trích dữ liệu từ các quan hệ trong cơ sở dữ liệu quan hệ.
  - Cho phép tối ưu quá trình rút trích bằng các phép toán có sẵn của lý thuyết tập hợp.

2. Biểu thức ĐSQH

- Biểu thức ĐSQH là một biểu thức gồm các phép toán ĐSQH.
- Biểu thức ĐSQH được xem như một quan hệ (không có tên).
- Có thể đặt tên cho quan hệ được tạo từ một biểu thức ĐSQH.
- Có thể đổi tên các thuộc tính của quan hệ được tạo từ một biểu thức ĐSQH.

3. Các phép toán

3.1 Giới thiệu
3.2 Phép chọn
3.3 Phép chiếu
3.4 Phép gán
3.5 Các phép toán trên tập hợp
3.6 Phép kết
3.7 Phép chia
3.8 Hàm tính toán và gom nhóm

3.1 Giới thiệu (1)

- Có năm phép toán cơ bản:
  - **Chọn (∨) hoặc (:)**
  - **Chiếu (Π) hoặc ([])**
  - **Tích (×)**
  - **Hiệu (−)**
  - **Hội (∪)**

3.1 Giới thiệu (2)

- Các phép toán khác không cơ bản nhưng hữu ích:
  - Giao ( )
  - Kết ( )
  - Chia ( )
  - Phép bù ( )
  - Đổi tên ( )
  - Phép gán ( )
- Kết quả sau khi thực hiện các phép toán là các quan hệ, do đó có thể kết hợp giữa các phép toán để tạo nên phép toán mới.

3.2 Phép chọn (Selection)

- Trích chọn các bộ (dòng) từ quan hệ R. Các bộ được trích chọn phải thỏa mãn điều kiện chọn p.
- **Ký hiệu**: $\sigma_p(R)$
- **Định nghĩa**: $\sigma_p(R) = \{t / t \in R, p(t)\}$ p(t):thỏa điều kiện p
- Kết quả trả về là một quan hệ, có cùng danh sách thuộc tính với quan hệ R. Không có kết quả trùng.
- Phép chọn có tính giao hoán

$$\sigma_{p1}(\sigma_{p2}(R)) = \sigma_{p2}(\sigma_{p1}(R)) = \sigma_{(p1 \wedge p2)}(R)$$

Lược đồ CSDL quản lý giáo vụ

HOCVIEN (MAHV, HO, TEN, NGSINH, GIOITINH, NOISINH, MALOP)

LOP (MALOP, TENLOP, TRGLOP, SISO, MAGVCN)

KHOA (MAKHOA, TENKHOA, NGTLAP, TRGKHOA)

MONHOC (MAMH, TENMH, TCLT, TCTH, MAKHOA)

DIEUKIEN (MAMH, MAMH_TRUOC)

GIAOVIEN(MAGV,HOTEN,HOCVI,HOCHAM,GIOITINH,NGSINH,NGVL, HESO, MUCLUONG, MAKHOA)

GIANGDAY(MALOP,MAMH,MAGV,HOCKY, NAM,TUNGAY,DENNGAY)

KETQUATHI (MAHV, MAMH, LANTHI, NGTHI, DIEM, KQUA)

3.2 Ví dụ phép chọn

- Tìm những học viên "Nam" có nơi sinh ở 'TpHCM'

$\sigma_{(Gioitinh='Nam') \wedge (Noisinh='TpHCM')}(HOCVIEN)$

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11
K1105	Tran Minh Long	Nam	TpHCM	K11
K1106	Le Nhat Minh	Nam	TpHCM	K11

3.3 Phép chiếu (Project)

- Sử dụng để trích chọn giá trị một vài thuộc tính của quan hệ
- **Ký hiệu**: $\Pi_{A_1,A_2,\ldots,A_k}(R)$

trong đó A_i là tên các thuộc tính được chiếu.

- Kết quả trả về một quan hệ có k thuộc tính theo thứ tự như liệt kê. Các dòng trùng nhau chỉ lấy một.
- Phép chiếu không có tính giao hoán

3.3 Ví dụ

- Tìm mã số, họ tên những học viên "Nam" có nơi sinh ở 'TpHCM'

Mahv,Hoten(Gioitinh='Nam')(Noisinh='TpHCM')(HOCVIEN)

HOCVIEN				
Mahv	**HoTen**	**Gioitinh**	**Noisinh**	**Malop**
K1103	Ha Duy Lap	Nam	Nghe An	K11
K1102	Tran Ngoc Han	Nu	Kien Giang	K11
K1104	Tran Ngoc Linh	Nu	Tay Ninh	K11
K1105	Tran Minh Long	Nam	TpHCM	K11
K1106	Le Nhat Minh	Nam	TpHCM	K11

3.4 Phép gán (Assignment)

- Dùng để diễn tả câu truy vấn phức tạp.
- **Ký hiệu**: A
- **Ví dụ**:
  R(HO,TEN,LUONG) ONV,TENNV,LUONG(NHANVIEN)
- Kết quả bên phải của phép gán được gán cho biến quan hệ nằm bên trái.

3.5 Các phép toán tập hợp

3.5.1 Giới thiệu

3.5.2 Phép hội

3.5.3 Phép trừ

3.5.4 Phép giao

3.5.5 Phép tích

3.5.1 Giới thiệu

- Các phép toán thực hiện trên 2 quan hệ xuất phát từ lý thuyết tập hợp của toán học: phép hội (R[redacted], phép giao (R[redacted], phép trừ (R-S), phép tích (R[redacted]).
- Đối với các phép hội, giao, trừ, các quan hệ R và S phải ***khả hợp:***
  - Số lượng thuộc tính của R và S phải bằng nhau:
    $R(A_1,A_2,\ldots A_n)$ và $S(B_1,B_2,\ldots B_n)$
  - Miền giá trị của thuộc tính phải tương thích
    $dom(A_i)=dom(B_i)$
- Quan hệ kết quả của phép hội, giao, trừ có cùng tên thuộc tính với quan hệ đầu tiên.

3.5.2 Phép hội (Union)

- **Ký hiệu**: R[redacted]
- **Định nghĩa**: R[redacted]$|t$[redacted]$\}$ trong đó R,S là hai quan hệ khả hợp.
- **Ví dụ**: Học viên được khen thưởng đợt 1 hoặc đợt 2

DOT1	
Mahv	**Hoten**
K1103	Le Van Tam
K1114	Tran Ngoc Han
K1203	Le Thanh Hau
K1308	Nguyen Gia

DOT2	
Mahv	**Hoten**
K1101	Le Kieu My
K1114	Tran Ngoc Han

Mahv	Hoten
K1101	Le Kieu My
K1103	Le Van Tam
K1114	Tran Ngoc Han
K1203	Le Thanh Hau
K1308	Nguyen Gia

DOT1 [redacted] **DOT2**

3.5.3 Phép trừ (Set Difference)

- **Ký hiệu**: R-S
- **Định nghĩa**: $R - S = \{t \mid t \in R \wedge t \notin S\}$ trong đó R,S là hai quan hệ khả hợp.
- **Ví dụ**: Học viên được khen thưởng đợt 1 nhưng không được khen thường đợt 2

DOT1	
Mahv	**Hoten**
K1103	Le Van Tam
K1114	Tran Ngoc Han
K1203	Le Thanh Hau
K1308	Nguyen Gia

DOT2	
Mahv	**Hoten**
K1101	Le Kieu My
K1114	Tran Ngoc Han

Mahv	Hoten
K1103	Le Van Tam
K1203	Le Thanh Hau

DOT1- DOT2

3.5.4 Phép giao (Set-Intersection)

- **Ký hiệu**: R
- **Định nghĩa**: R $|t$ trong đó R,S là hai quan hệ khả hợp. Hoặc R $= R - (R - S)$
- **Ví dụ**: Học viên được khen thưởng cả hai đợt 1 và 2

KT_D1	
Mahv	**Hoten**
K1103	Le Van Tam
K1114	Tran Ngoc Han
K1203	Le Thanh Hau
K1308	Nguyen Gia

KT_D2	
Mahv	**Hoten**
K1101	Le Kieu My
K1114	Tran Ngoc Han

Mahv	Hoten
K1114	Tran Ngoc Han

DOT1 DOT2

3.5.5 Phép tích (1)

- **Ký hiệu**: R
- **Định nghĩa**: $R = \{t_r t_s / t_r \in r \wedge t_s \in s\}$
- Nếu R có n bộ và S có m bộ thì kết quả là n*m bộ
  KQ(A_1,A_2,…A_m,B_1,B_2,…B_n) (A_1,A_2,…A_m) S(B_1,B_2,…B_n)
- Phép tích thường dùng kết hợp với các phép chọn để kết hợp các bộ có liên quan từ hai quan hệ.
- **Ví dụ**: từ hai quan hệ HOCVIEN và MONHOC, có tất cả những trường hợp nào "*học viên đăng ký học môn học*", giả sử không có bất kỳ điều kiện nào

3.5.5 Phép tích (2)

HOCVIEN	
Mahv	**Hoten**
K1103	Le Van Tam
K1114	Tran Ngoc Han
K1203	Le Thanh Hau

MONHOC
Mamh
CTRR
THDC
CTDL

Mahv	Hoten	Mamh
K1103	Le Van Tam	CTRR
K1114	Tran Ngoc Han	CTRR
K1203	Le Thanh Hau	CTRR
K1103	Le Van Tam	THDC
K1114	Tran Ngoc Han	THDC
K1203	Le Thanh Hau	THDC
K1103	Le Van Tam	CTDL
K1114	Tran Ngoc Han	CTDL
K1203	Le Thanh Hau	CTDL

HOCVIEN MONHOC

3.6 Phép kết

3.6.1 Phép kết

3.6.2 Phép kết bằng, phép kết tự nhiên

3.6.3 Phép kết ngoài

3.6.1 Phép kết (Theta-Join) (1)

- Theta-join (Tương tự như phép tích kết hợp với phép chọn. Điều kiện chọn gọi là ***điều kiện kết***.
- **Ký hiệu**: $R \overset{p}{\triangleright\triangleleft} S$

  trong đó R,S là các quan hệ, p là điều kiện kết
- Các bộ có giá trị NULL tại thuộc tính kết nối không xuất hiện trong kết quả của phép kết.
- Phép kết với điều kiện tổng quát gọi là kết với à một trong những phép so sánh (

3.6.1 Phép kết (2)

$A_1 \blacksquare B_2$

$R \rhd\lhd S$

R	
A_1	A_2
1	2
1	8
0	0
8	4
0	3

S		
B_1	B_2	B_3
0	2	8
7	8	7
8	0	4
1	0	7
2	1	5

A_1	A_2	B_1	B_2	B_3
1	2	8	0	4
1	2	1	0	7
1	8	8	0	4
1	8	1	0	7
8	4	0	2	8
8	4	8	0	4
8	4	1	0	7
8	4	2	1	5

3.6.2 Phép kết bằng, kết tự nhiên

- Nếu [redacted] là phép so sánh bằng (=), phép kết gọi là phép kết bằng (equi-join).

  Ký hiệu: $HOCVIEN \overset{Mahv[redacted]rglop}{\triangleright\triangleleft} LOP$

- Nếu điều kiện của equi-join là các thuộc tính giống nhau thì gọi là phép kết tự nhiên (natural-join). Khi đó kết quả của phép kết loại bỏ bớt 1 cột (bỏ 1 trong 2 cột giống nhau)

  Ký hiệu: $HOCVIEN \overset{Mahv}{\triangleright\triangleleft} KETQUATHI$ hoặc

  $HOCVIEN \overset{Mahv}{*} KETQUATHI$

3.6.3 Phép kết ngoài (outer join)

- Mở rộng phép kết để tránh mất thông tin
- Thực hiện phép kết và sau đó thêm vào kết quả của phép kết các bộ của quan hệ mà không phù hợp với các bộ trong quan hệ kia.
- Có 3 loại:
  - Left outer join R ⟕ S
  - Right outer join R ⟖ S
  - Full outer join R ⟗ S
- **Ví dụ**: In ra danh sách tất cả các học viên và điểm số của các môn học mà học viên đó thi (nếu có)

3.6.3 Phép kết ngoài (2)

- HOCVIEN $\overset{mahv}{\bowtie}$ KETQUATHI

Mahv	Hoten	Mahv	Mamh	Diem
HV01	Nguyen Van Lan	HV01	CSDL	7.0
HV01	Nguyen Van Lan	HV01	CTRR	8.5
HV02	Tran Hong Son	HV02	CSDL	8.5
HV03	Nguyen Le	HV03	CTRR	9.0
HV04	Le Minh	Null	Null	Null

HOCVIEN	
Mahv	**Hoten**
HV01	Nguyen Van Lan
HV02	Tran Hong Son
HV03	Nguyen Le
HV04	Le Minh

KETQUATHI		
Mahv	**Mamh**	**Diem**
HV01	CSDL	7.0
HV02	CSDL	8.5
HV01	CTRR	8.5
HV03	CTRR	9.0

3.7 Phép chia (Division)

- **Định nghĩa**: Q [redacted] R [redacted] S [redacted] t / [redacted], (t, s) [redacted]}
- R và S là hai quan hệ, R^+ và S^+ lần lượt là tập thuộc tính của R và S. Điều kiện S^+ [redacted] là ***tập con không bằng*** của R^+. Q là kết quả phép chia giữa R và S, $Q^+ = R^+ - S^+$
- Có thể diễn đạt bằng phép toán đại số như sau:

T_1 [redacted] (R)

T_2 [redacted] $((S$ [redacted] $T_1)$ [redacted] $R)$

T [redacted] T_2

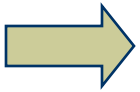
3.7 Phép chia (2)

KETQUATHI		
Mahv	**Mamh**	**Diem**
HV01	CSDL	7.0
HV02	CSDL	8.5
HV01	CTRR	8.5
HV03	CTRR	9.0
HV01	THDC	7.0
HV02	THDC	5.0
HV03	THDC	7.5
HV03	CSDL	6.0

KETQUA

MONHOC	
Mamh	**Tenmh**
CSDL	Co so du lieu
CTRR	Cau truc roi rac
THDC	Tin hoc dai cuong

MONHOC



Mahv
HV01
HV03

KETQUA [redacted] MONHOC

$$KETQUA = [redacted]_{ETQUATHI}[Mahv, Mamh]$$

$$MONHOC = [redacted]_{ONHOC}[Mamh]$$

3.8 Hàm tính toán và gom nhóm (1)

- Hàm tính toán gồm các hàm: avg(giatri), min(giatri), max(giatri), sum(giatri), count(giatri).
- Phép toán gom nhóm:

$$G_1, G_2, \ldots, G_n \; \mathcal{G}_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)} (E)$$

  - E là biểu thức đại số quan hệ
  - G_i là thuộc tính gom nhóm (rỗng, nếu không gom nhóm)
  - F_i là hàm tính toán
  - A_i là tên thuộc tính

3.8 Hàm tính toán và gom nhóm (2)

- Điểm thi cao nhất, thấp nhất, trung bình của môn CSDL ?

$$\square_{\max(Diem),\min(Diem),agv(Diem)}\square_{Mamh='CSDL'}(KETQUATHI)$$

- Điểm thi cao nhất, thấp nhất, trung bình của từng môn ?

$$Mamh\ \square_{\max(Diem),\min(Diem),avg(Diem)}(KETQUATHI)$$

Bài tập

Lược đồ CSDL quản lý bán hàng gồm có các quan hệ sau:

KHACHHANG (MAKH, HOTEN, DCHI, SODT, NGSINH, DOANHSO, NGDK)

NHANVIEN (MANV,HOTEN, NGVL, SODT)

SANPHAM (MASP,TENSP, DVT, NUOCSX, GIA)

HOADON (SOHD, NGHD, MAKH, MANV, TRIGIA)

CTHD (SOHD,MASP,SL)

Mô tả các câu truy vấn sau bằng ĐSQH

1. In ra danh sách các sản phẩm (MASP,TENSP) do "Trung Quốc" sản xuất có giá từ 30.000 đến 40.000
2. In ra danh sách các khách hàng (MAKH, HOTEN) đã mua hàng trong ngày 1/1/2007.
3. In ra danh sách các sản phẩm (MASP,TENSP) do "Trung Quoc" sản xuất hoặc các sản phẩm được bán ra trong ngày 1/1/2007.
4. Tìm các số hóa đơn mua cùng lúc 2 sản phẩm có mã số "BB01" và "BB02".
5. In ra danh sách các sản phẩm (MASP,TENSP) do "Trung Quoc" sản xuất không bán được trong năm 2006.
6. Tìm số hóa đơn đã mua tất cả các sản phẩm do Singapore sản xuất

Câu 1

- In ra danh sách các sản phẩm (MASP, TENSP) do "Trung Quốc" sản xuất có giá từ 30.000 đến 40.000.

$$SANPHAM : ((nuocsx = 'TrungQuoc') \wedge (gia \geq 30.000) \wedge (gia \leq 40.000))[masp, tensp]$$

$$masp, tensp(\sigma_{(nuocsx = 'TrungQuoc') \wedge (gia \geq 30.000) \wedge (gia \leq 40.000)}(SANPHAM))$$

Câu 2

- In ra danh sách các khách hàng (MAKH, HOTEN) đã mua hàng trong ngày 1/1/2007.

$$[KHACHHANG \overset{MAKH}{\triangleright\!\triangleleft} HOADON : (nghd = 1/1/2007\#)[makh, hoten]$$

$$[masp, hoten](nghd = 1/1/2007\#)(HOADON \overset{MAKH}{\triangleright\!\triangleleft} KHACHHANG)$$

Câu 3

- In ra danh sách các sản phẩm do "Trung Quoc" sản xuất hoặc các sản phẩm được bán ra trong ngày 1/1/2007.

$$A \; ANPHAM : (nuocsx \; TrungQuoc')[masp, tensp]$$

$$B \; ANPHAM \overset{MASP}{\triangleright\!\triangleleft} CTHD \overset{SOHD}{\triangleright\!\triangleleft} HOADON : (nghd \; 1/1/2007\#))[masp, tensp]$$

$$C$$

Hoặc

$$A_{masp,tensp \; nuocsx \; TrungQuoc'}(SANPHAM)$$

$$B_{masp,tensp}((_{nghd \; 1/1/2007\#}(HOADON) \overset{SOHD}{\triangleright\!\triangleleft} CTHD) \overset{MASP}{\triangleright\!\triangleleft} SANPHAM)$$

$$C$$

Câu 4

- Tìm các số hóa đơn đã mua cùng lúc các sản phẩm có mã số “BB01” và “BB02”.

A [redacted] $THD : (masp$ [redacted] $BB01')[sohd]$

B [redacted] $THD : (masp$ [redacted] $BB02')[sohd]$

C [redacted]

Hoặc

A [redacted] $(CTHD)$

B [redacted] $(CTHD)$

C [redacted]

Câu 5

- In ra danh sách các sản phẩm do “TrungQuoc” sản xuất không bán được trong năm 2006.

$$A\ [redacted]_{masp,tensp}[redacted]_{nuocsx}[redacted]_{TrungQuoc'}(SANPHAM)$$

$$B\ [redacted]SANPHAM \overset{MASP}{\rhd\lhd} CTHD) \overset{SOHD}{\rhd\lhd} HOADON)$$

$$C\ [redacted]_{masp,tensp}[redacted]_{(nuocsx}[redacted]_{TrungQuoc')}[redacted]_{year(nghd)}[redacted]_{2006)}(B)$$

$$D\ [redacted]A[redacted]C)$$

Câu 6

- Tìm số hóa đơn đã mua tất cả các sản phẩm do Singapore sản xuất

$$A\ [redacted]_{masp}[redacted]_{nuocsx}[redacted]_{Singapore'}(SANPHAM)$$

$$B\ [redacted]_{masp,sohd}[redacted]_{nuocsx}[redacted]_{Singapore'}(SANPHAM \overset{MASP}{\rhd\lhd} CTHD)$$

$$C\ [redacted]A$$

Bài 5: Ngôn ngữ SQL

Nội dung

1. Giới thiệu
2. Các ngôn ngữ giao tiếp
3. Ngôn ngữ định nghĩa dữ liệu
4. Ngôn ngữ thao tác dữ liệu
5. Ngôn ngữ truy vấn dữ liệu có cấu trúc
6. Ngôn ngữ điều khiển dữ liệu

1. Giới thiệu

- Là ngôn ngữ chuẩn để truy vấn và thao tác trên CSDL quan hệ
- Là ngôn ngữ phi thủ tục
- Khởi nguồn của SQL là SEQUEL - *Structured English Query Language*, năm 1974*)*
- *Các chuẩn SQL*
  - SQL89
  - SQL92 (SQL2)
  - SQL99 (SQL3)

2. Các ngôn ngữ giao tiếp

- **Ngôn ngữ định nghĩa dữ liệu** (*Data Definition Language - DDL*): cho phép khai báo cấu trúc bảng, các mối quan hệ và các ràng buộc.
- **Ngôn ngữ thao tác dữ liệu** (*Data Manipulation Language - DML* liệu.
- **Ngôn ngữ truy vấn dữ liệu** (*Structured Query Language – SQL*): cho phép truy vấn dữ liệu.
- **Ngôn ngữ điều khiển dữ liệu** (*Data Control Language – DCL*

u.

3. Ngôn ngữ định nghĩa dữ liệu

3.1 Lệ o bảng (CREATE)

3.1.1 Cú pháp

3.1.2 Một số kiểu dữ liệu

3.2 Lệnh sử u trúc bảng (ALTER)

3.2.1 Thêm thuộc tính

3.2.2 Sửa kiểu dữ liệu của thuộc tính

3.2.3 Xoá thuộc tính

3.2.4 Thêm ràng buộc toàn vẹn

3.2.5 Xoá ràng buộc toàn vẹn

3.3 Lệnh xóa bảng (DROP)

3.1 Lệnh tạo bảng

3.1.1 Cú pháp

CREATE TABLE <tên_bảng>
(
<tên_cột1> <kiểu_dữ_liệu> [not null],
<tên_cột2> <kiểu_dữ_liệu> [not null],
…
<tên_cột1> <kiểu_dữ_liệu> [not null],
khai báo khóa chính, khóa ngoại, ràng buộc
)

3.1 Lệnh tạo bảng (2)

3.1.2 Một số kiểu dữ liệu

Kiểu dữ liệu	SQL Server
Chuỗi ký tự	varchar(n), char(n),nvarchar(n), nchar(n)
Số	tinyint,smallint, int, numeric(m,n), decimal(m,n),float, real, smallmoney, money
Ngày tháng	smalldatetime, datetime
Luận lý	bit

3.1 Lệnh tạo bảng (3)

Lược đồ CSDL quản lý bán hàng gồm có các quan hệ sau:

KHACHHANG (MAKH, HOTEN, DCHI, SODT, NGSINH, DOANHSO, NGDK, CMND)

NHANVIEN (MANV,HOTEN, NGVL, SODT)

SANPHAM (MASP,TENSP, DVT, NUOCSX, GIA)

HOADON (SOHD, NGHD, MAKH, MANV, TRIGIA)

CTHD (SOHD,MASP,SL)

3.1 Lệnh tạo bảng (4)

```
Create table KHACHHANG
(
    MAKH        char(4) primary key,
    HOTEN       varchar(40),
    DCHI        varchar(50),
    SODT        varchar(20),
    NGSINH      smalldatetime,
    DOANHSO     money,
    NGDK        smalldatetime,
    CMND        varchar(10)
)
```

3.1 Lệnh tạo bảng (5)

```
Create table CTHD
(
    SOHD      int foreign key
                    references HOADON(SOHD),
    MASP      char(4) foreign key
                    references SANPHAM(MASP),
    SL        int,
    constraint PK_CTHD  primary key (SOHD,MASP)
)
```

3.2 Sửa cấu trúc bảng(1)

3.2.1 Thêm thuộc tính

ALTER TABLE tênbảng ADD têncột kiểudữliệu

- Ví dụ t Ghi_chu vào bảng khách hàng

ALTER TABLE KHACHHANG ADD GHI_CHU varchar(20)

3.2.2 Sửa kiểu dữ liệu thuộc tính

ALTER TABLE tênbảng ALTER COLUMN têncột kiểudữliệu_mới

- **Lưu ý:**

Không phải sửa bất kỳ kiểu dữ liệu nào cũng được

3.2 Sửa cấu trúc bảng(2)

- **Ví dụ**: Sửa Cột Ghi_chu thành kiểu dữ liệu varchar(50)

ALTER TABLE KHACHHANG ALTER COLUMN GHI_CHU varchar(50)

- Nếu sửa kiểu dữ liệu của cột Ghi_chu thành varchar(5), mà trước đó đã nhập giá trị cho cột Ghi_chu có độ dài hơn 5ký tự thì không được phép.
- Hoặc sửa từ kiểu chuỗi ký tự sang kiểu số, …

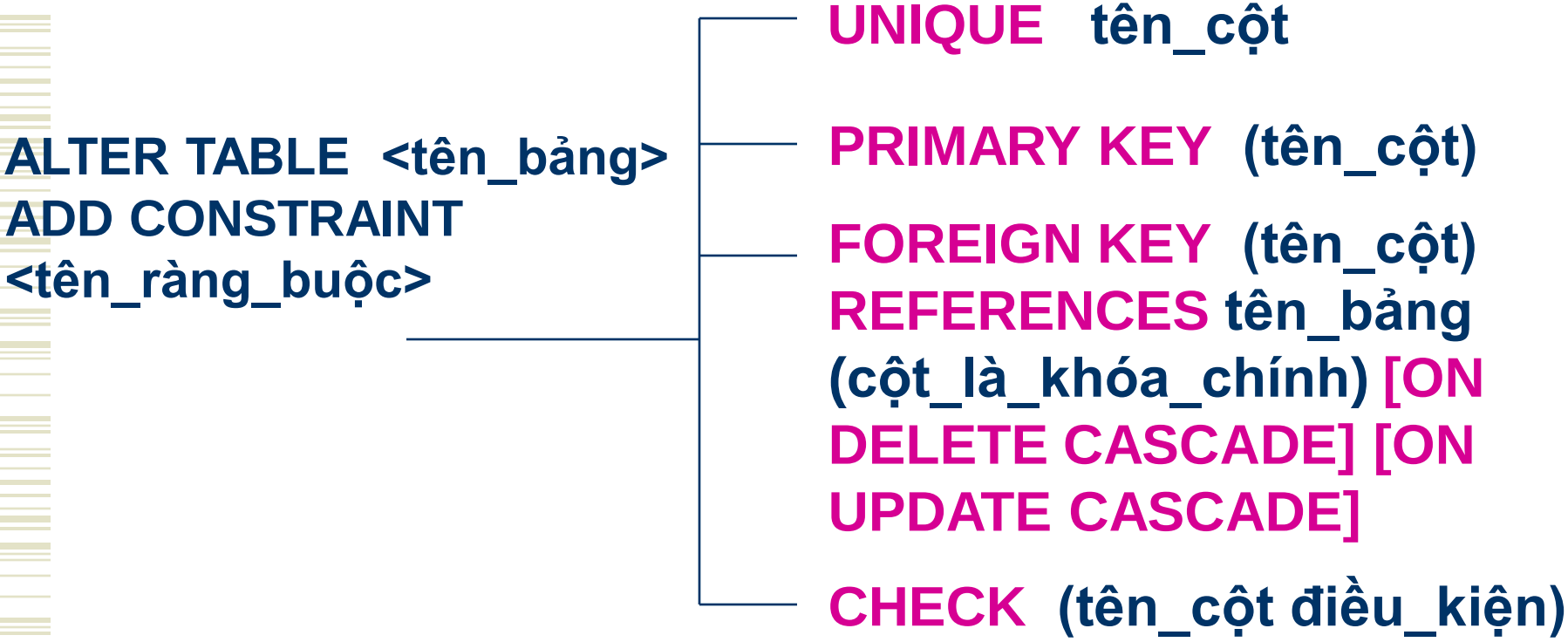
3.2.3 Xóa thuộc tính

ALTER TABLE tên_bảng DROP COLUMN tên_cột

- Ví dụ: xóa cột Ghi_chu trong bảng KHACHHANG
  ALTER TABLE NHANVIEN DROP COLUMN Ghi_chu

3.2 Sửa cấu trúc bảng(3)

3.2.4 Thêm ràng buộc toàn vẹn



3.2 Sửa cấu trúc bảng(4)

- **Ví dụ**
  - ALTER TABLE NHANVIEN ADD CONSTRAINT PK_NV PRIMARY KEY (MANV)
  - ALTER TABLE CTHD ADD CONSTRAINT FK_CT_SP FOREIGN KEY (MASP) REFERENCES SANPHAM(MASP)
  - ALTER TABLE SANPHAM ADD CONSTRAINT CK_GIA CHECK (GIA >=500)
  - ALTER TABLE KHACHHANG ADD CONSTRAINT UQ_KH UNIQUE (CMND)

3.2 Sửa cấu trúc bảng(5)

3.2.5 Xóa ràng buộc toàn vẹn

ALTER TABLE tên_bảng DROP CONSTRAINT tên_ràng_buộc

- **Ví dụ**:
  - Alter table CTHD drop constraint FK_CT_SP
  - Alter table SANPHAM drop constraint ck_gia

- **Lưu ý**: đối với ràng buộc khóa chính, muốn xóa ràng buộc này phải xóa hết các ràng buộc khóa ngoại tham chiếu tới nó

3.3 Lệnh xóa bảng

- **Cú pháp**

  DROP TABLE tên_bảng

- **Ví dụ**: xóa bảng KHACHHANG.

  DROP TABLE KHACHHANG

- **Lưu ý**: khi muốn xóa một bảng phải xóa tất cả những khóa ngoại tham chiếu tới bảng đó trước.

4. Ngôn ngữ thao tác dữ liệu

- G m các lệnh:
  - 4.1 Lệnh thêm dữ u (INSERT)
  - 4.2 Lệnh sửa dữ u (UPDATE)
  - 4.3 Lệnh xóa dữ u (DELETE)

4.1 Thêm dữ liệu

- **Cú pháp**

  INSERT INTO tên_bảng (cột1,…,cộtn) VALUES (giá_trị_1,…., giá_trị_n)

  INSERT INTO tên_bảng VALUES (giá_trị_1, giá_trị_2,…, giá_trị_n)

- **Ví dụ:**
  - insert into SANPHAM values('BC01','But chi', 'cay', 'Singapore', 3000)
  - insert into SANPHAM(masp,tensp,dvt,nuocsx,gia) values ('BC01','But chi','cay','Singapore',3000)

4.2 Sửa dữ liệu

- **Cú pháp**

  UPDATE tên_bảng
  SET cột_1 = giá_trị_1, cột_2 = giá_trị_2 ….
  [WHERE điều_kiện]

- **Lưu ý**: cẩn thận với các lệnh xóa và sửa, nếu không có điều kiện ở WHERE nghĩa là xóa hoặc sửa tất cả.
- **Ví dụ**: Tăng giá 10% đối với những sản phẩm do "Trung Quoc" sản xuất

  UPDATE SANPHAM
  SET Gia = Gia*1.1
  WHERE Nuocsx='Trung Quoc'

4.3 Xóa dữ liệu

- **Cú pháp**

  DELETE FROM tên_bảng [WHERE điều_kiện]

- **Ví dụ:**
  - Xóa toàn bộ nhân viên

    DELETE FROM NHANVIEN
  - Xóa những sản phẩm do Trung Quốc sản xuất có giá thấp hơn 10000

    DELETE FROM SANPHAM
    WHERE (Gia <10000) and (Nuocsx='Trung Quoc')

5. Ngôn ngữ truy vấn dữ liệu có cấu trúc

5.1 Câu truy vấn tổng quát
5.2 Truy vấn đơn giản
5.3 Phép kết
5.4 Đặt bí danh, sử dụng *, distinct
5.5 Các toán tử
5.6 Câu truy vấn con (subquery)
5.7 Phép chia
5.8 Hàm tính toán, gom nhóm

5.1Câu truy vấn tổng quát

SELECT [DISTINCT] *|tên_cột | hàm
FROM bảng
[WHERE điều_kiện]
[GROUP BY tên_cột]
[HAVING điều_kiện]
[ORDER BY tên_cột ASC | DESC]

5.2 Truy vấn đơn giản(1)

- SELECT
  - Tương đương phép chiếu của ĐSQH
  - Liệt kê các thuộc tính cần hiển thị trong kết quả
- WHERE
  - Tương ứng với điều kiện chọn trong ĐSQH
  - Điều kiện liên quan tới thuộc tính, sử dụng các phép nối luận lý AND, OR, NOT, các phép toán so sánh, BETWEEN
- FROM
  - Liệt kê các quan hệ cần thiết, các phép kết

5.2 Truy vấn đơn giản(2)

- Tìm masp, tensp do “Trung Quoc” sản xuất có giá từ 20000 đến 30000

```
Select      masp,tensp
From        SANPHAM
Where       nuocsx=‘Trung Quoc’
      and   gia between 20000 and 30000
```

5.3 Phép kết(1)

- Inner Join, Left Join, Right Join, Full Join
- **Ví dụ**:
  - In ra danh sách các khách hàng (MAKH, HOTEN) đã mua hàng trong ngày 1/1/2007.

```
select   KHACHHANG.makh,hoten
from     KHACHHANG inner join HOADON on
         KHACHHANG.makh=HOADON.makh
where    nghd='1/1/2007'
```

5.3 Phép kết (2)

- **Ví dụ**: In ra danh sách tất cả các hóa đơn và họ tên của khách hàng mua hóa đơn đó (nếu có)
  - Select sohd, hoten
From HOADON left join KHACHHANG on HOADON.makh=KHACHHANG.makh
  - Select sohd, hoten
From HOADON ,KHACHHANG
where HOADON.makh*=KHACHHANG.makh

5.4 Đặt bí danh, sử dụng *, distinct

- Đặt bí danh – Alias: cho thuộc tính và quan hệ: tên_cũ AS tên_mới
  - Select manv,hoten as [ho va ten] From NHANVIEN
- Liệt kê tất cả các thuộc tính của quan hệ:
  - Select * from Nhanvien
  - Select NHANVIEN.* from NHANVIEN
- Distinct: trùng chỉ lấy một lần
  - Select distinct nuocsx from SANPHAM
- Sắp xếp kết quả hiển thị: Order by
  - Select * from SANPHAM order by nuocsx, gia DESC

5.5 Toán tử truy vấn(1)

- Toán tử so sánh: =, >,<,>=,<=,<>
- Toán tử logic: AND, OR, NOT
- Phép toán: +, - ,* , /
- BETWEEN …. AND
- IS NULL, IS NOT NULL
- LIKE (_ %)
- IN, NOT IN
- EXISTS , NOT EXISTS
- SOME, ALL

5.5 Toán tử truy vấn(2)

- **IS NULL, IS NOT NULL**
  - Select sohd from HOADON where makh is Null
  - Select * from HOADON where makh is Not Null
- **Toán tử so sánh, phép toán**
  - Select gia*1.1 as [gia ban] from SANPHAM where nuocsx<>'Viet Nam'
  - Select * from SANPHAM where (gia between 20000 and 30000) OR (nuocsx='Viet Nam')
- **Toán tử IN, NOT IN**
  - Select * from SANPHAM where masp NOT IN ('BB01','BB02','BB03')

5.5 Toán tử so sánh(3)

Toán tử LIKE

- So sánh chuỗi tương đối
- Cú pháp: s LIKE p, p có thể chứa % hoặc _
- % : thay thế một chuỗi ký tự bất kỳ
- _ : thay thế một ký tự bất kỳ
- **Ví dụ**: Select masp,tensp from SANPHAM where masp like 'B%01‘

5.6 Câu truy vấn con (1)

In hoặc Exists

- **Ví dụ**: Tìm các số hóa đơn mua cùng lúc 2 sản phẩm có mã số "BB01" và "BB02".
  - select distinct sohd
    from CTHD where masp='BB01' and sohd **IN**
      (select distinct sohd from CTHD where masp='BB02')
  - select distinct A.sohd
    from CTHD A where A.masp='BB01' and
      EXISTS (select * from CTHD B
        where B.masp='BB02' and A.sohd=B.sohd)

5.6 Câu truy vấn con (2)

Not In hoặc Not Exists

- **Ví dụ**: Tìm các số hóa đơn có mua sản phẩm mã số ‘BB01’ nhưng không mua sản phẩm mã số ‘BB02’.
  - select distinct sohd
    from CTHD where masp='BB01' and sohd **NOT IN**
    (select distinct sohd from CTHD where masp='BB02')
  - select distinct A.sohd
    from CTHD A where A.masp='BB01' and
    NOT EXITST (select * from CTHD B
    where B.masp='BB02‘ and A.sohd=B.sohd)

5.7 Phép chia

Sử dụng NOT EXISTS

- **Ví dụ**: Tìm số hóa đơn đã mua tất cả những sản phẩm do "Trung Quoc" sản xuất.

- Select sohd from HOADON where not exists
  (select * from SANPHAM
  where nuocsx='Trung Quoc' and not exists
  (select * from CTHD where
  HOADON.sohd=CTHD.sohd and
  CTHD.masp=SANPHAM.masp))

5.8 Các hàm tính toán và gom nhóm (1)

5.8.1 Các hàm tính toán cơ bản

- COUNT: Đếm số bộ dữ liệu của thuộc tính
- MIN: Tính giá trị nhỏ nhất
- MAX: Tính giá trị lớn nhất
- AVG: Tính giá trị trung bình
- SUM: Tính tổng giá trị các bộ dữ liệu

NHANVIEN					
MANV	**HOTEN**	**PHAI**	**MANQL**	**PHONG**	**LUONG**
NV001	Nguyễn Ngọc Linh	Nữ	Null	NC	2.800.000
NV002	Đinh Bá Tiến	Nam	NV002	DH	2.000.000
NV003	Nguyễn Văn Mạnh	Nam	NV001	NC	2.300.000
NV004	Trần Thanh Long	Nam	NV002	DH	1.800.000
NV005	Nguyễn Thị Hồng Vân	Nữ	NV001	NC	2.500.000
NV006	Nguyễn Minh	Nam	NV002	DH	2.000.000
NV007	Hà Duy Lập	Nam	NV003	NC	1.800.000
NV008	Trần Kim Duyên	Nữ	NV003	NC	1.800.000
NV009	Nguyễn Kim Anh	Nữ	NV003	NC	2.000.000

Ví dụ

1. Tính lương thấp nhất, cao nhất, trung bình và tổng lương của tất cả các nhân viên.
2. Có tất cả bao nhiêu nhân viên
3. Bao nhiêu nhân viên có người quản lý
4. Bao nhiêu phòng ban có nhân viên trực thuộc
5. Tính lương trung bình của các nhân viên
6. Tính lương trung bình của các nhân viên theo từng phòng ban

1. Tính lương thấp nhất, cao nhất, trung bình và tổng lương của tất cả các nhân viên.

```
SELECT   min(luong) as thapnhat,
         max(luong) as caonhat,
         avg(luong) as trungbinh,
         sum(luong) as tongluong

FROM     NhanVien
```

2. Có tất cả bao nhiêu nhân viên

SELECT **count(*)** FROM NhanVien

3. Bao nhiêu nhân viên có người quản lý

- Select count(*) FROM NhanVien WHERE manql is not null
- SELECT **count(Manql)** FROM NhanVien

4. Bao nhiêu phòng ban có nhân viên trực thuộc

SELECT **count(distinct phong)** FROM NhanVien

5.8 Các hàm tính toán và gom nhóm (2)

5.8.2 Gom nhóm: mệnh đề GROUP BY

- Sử dụng hàm gom nhóm trên các bộ trong quan hệ.
- Mỗi nhóm bộ bao gồm tập hợp các bộ có cùng giá trị trên các thuộc tính gom nhóm
- Hàm gom nhóm áp dụng trên mỗi bộ độc lập nhau.
- SQL có mệnh đề GROUP BY để chỉ ra các thuộc tính gom nhóm, các thuộc tính này phải xuất hiện trong mệnh đề SELECT

5. Tính lương trung bình của các nhân viên

```
SELECT      avg(LUONG) as LUONGTB
FROM        NhanVien
```

6. Tính lương trung bình của các nhân viên theo từng phòng ban.

```
SELECT      phong, avg(LUONG) as LUONGTB
FROM        NhanVien
GROUP BY    phong
```

5.8 Các hàm tính toán và gom nhóm (3)

5.8.3 Điều kiện sau gom nhóm: mệnh đề HAVING

- Lọc kết quả theo điều kiện, sau khi đã gom nhóm
- Điều kiện ở HAVING được thực hiện sau khi gom nhóm, các điều kiện có liên quan đến thuộc tính Group By

- **Ví dụ**: tìm phòng có số lượng nhân viên "Nữ" trên 5 người

```
SELECT      phong
FROM        NhanVien
WHERE       phai = 'Nữ'
GROUP BY    phong
HAVING      count(manv) > 5
```

Bài 6: Ngôn ngữ tân từ

Nội dung

1. Giới thiệu
2. Cú pháp
3. Các định nghĩa
4. Diễn giải của một công thức
5. Quy tắc lượng giá công thức
6. Ngôn ngữ tân từ có biến là n bộ
7. Ngôn ngữ tân từ có biến là miền giá trị

1. Giới thiệu

- Ngôn ngữ tân từ là ngôn ngữ truy vấn hình thức do Codd đề nghị (1972-1973) được Lacroit, Proix và Ullman phát triển, cài đặt trong một số ngôn ngữ như QBE, ALPHA..
- **Đặc điểm:**
  - Ngôn ngữ phi thủ tục
  - Rút trích cái gì chứ không phải rút trích như thế nào
  - Khả năng diễn đạt tương đương với đại số quan hệ
- **Có hai loại:**
  - Có biến là n bộ
  - Có biến là miền giá trị

2. Cú pháp

- ( ) : biểu thức trong ngoặc
- **Biến**: dùng chữ thường ở cuối bộ ký tự: x,y,z,t,s…
- **Hằng**: dùng chữ thường ở đầu bộ ký tự: a,b,c,…
- **Hàm**: là một ánh xạ từ một miền giá trị vào tập hợp gồm 2 giá trị: đúng hoặc sai. Thường dùng chữ thường ở giữa bộ ký tự: h,g,f,…
- **Tân từ**: là một biểu thức được xây dựng dựa trên biểu thức logic. Dùng chữ in hoa ở giữa bộ ký tự P,Q,R…
- **Các phép toán logic**: phủ định (kéo theo (và (hoặc (
- **Các lượng từ**: với mọi (tồn tại (

3. Các định nghĩa (1)

- **Định nghĩa 1: Tân từ 1 ngôi**
  - Tân từ 1 ngôi được định nghĩa trên tập X và biến x có giá trị chạy trên các phần tử của X.
  - Với mỗi giá trị của x, tân từ P(x) là một mệnh đề logic, tức là nó có giá trị đúng (Đ) hoặc sai (S)
  - **Ví dụ**
    - P(x), x là biến chạy trên X, là một tân từ
    - P(gt), gt là một mệnh đề, X = {Nguyen Van A, Tran Thi B}
    - Với tân từ NỮ(x) được xác định: "*x là người nữ*". Khi đó
    - Mệnh đề NỮ(Nguyen Van A): cho kết quả Sai
    - NỮ(Tran Thi B): cho kết quả Đúng

3. Các định nghĩa (2)

- **Định nghĩa 2: Tân từ n ngôi**
  - Tân từ n ngôi được định nghĩa trên các tập $X_1, X_2, \ldots, X_n$ và n biến $x_1, x_2, \ldots, x_n$ lấy giá trị trên các tập X_i tương ứng.
  - Với mỗi giá trị a_i [redacted] $_i$, $x_i=a_i$.Tân từ n ngôi là một mệnh đề.
  - Ký hiệu: $P(x_1, x_2, \ldots, x_n)$
  - **Ví dụ:** $CHA(x_1,x_2)$: "x_1 là CHA của x_2"
  - **Chú ý**:
    - Các X_i không nhất thiết phải là rời nhau
    - Với $x_i=a_i$, $P(x_1, x_2, \ldots, a_i, \ldots, x_n)$ là tân từ n-1 ngôi

3. Các định nghĩa (3)

- **Định nghĩa 3: Từ**
  - Từ là một hằng hay là một biến
  - Nếu $f(t_1, t_2, \ldots, t_n)$ là hàm n ngôi thì f là một từ
- **Định nghĩa 4: Công thức**
  - Công thức nguyên tố: $P(t_1, t_2, \ldots, t_n)$, t_i là các từ
  - Nếu F_1, F_2 là các công thức thì các biểu thức sau cũng là các công thức: F_1 [redacted] F_2, F_1 [redacted] F_2, $F_1 \Rightarrow F_2$, [redacted]
  - Nếu F_1 là một công thức thì [redacted] F_1, [redacted]$:F_1$ cũng là các công thức
  - Nếu F_1 là công thức thì (F_1) cũng là một công thức

3. Các định nghĩa (4)

- **Định nghĩa 4:**
  - Công thức *đóng* là công thức nếu mọi biến đều có kèm với lượng từ. (khẳng định Đ, S)
  - Công thức *mở* là công thức tồn tại 1 biến không kèm lượng từ. (tìm kiếm thông tin)
- **Ví dụ:**
  - C_1: [redacted]P(x,y,a)[redacted](Q(y,z,t)[redacted](x,t)) là công thức đóng vì các biến x,y,z,t đều có kèm lượng từ [redacted]
  - C_2: [redacted](P(x,y,a)[redacted](Q(y,z,t)[redacted](x,t)) là công thức mở vì biến y không có lượng từ [redacted]

4. Diễn giải của một công thức

Gồm 4 phần:

- Miền giá trị của các biến của công thức (ký hiệu là tập M)
- Sử dụng các hằng, các tân từ (ý nghĩa tân từ, xác định được quan hệ n ngôi)
- Ý nghĩa của công thức
- Xác định 1 quan hệ n ngôi trên tập M^n

5. Quy tắc lượng giá công thức

- Lượng giá tân từ: xét tân từ bậc n: $P(x_1,x_2,\ldots x_n)$ và liên kết với quan hệ R, n ngôi.

  $P(a_1,a_2,\ldots,a_n)$: Đ [redacted]$_1,a_2,\ldots,a_n)$ [redacted]

  $P(a_1,a_2,\ldots,a_n)$: S [redacted]$_1,a_2,\ldots,a_n)$ [redacted]
- Các phép toán [redacted]ùng bảng chân trị
- Lượng từ [redacted]gọi x là biến. Công thức [redacted] F(x) là đúng khi có ít nhất một a_i [redacted]I/F(a_i):Đ

  M={$a_1,a_2,\ldots,a_n$} [redacted](a_i), a_i [redacted]
- Lượng từ [redacted]x là biến, [redacted]F(x): Đ với [redacted]$_i$ [redacted]I/F(a_i):Đ

  M={$a_1,a_2,\ldots,a_n$} [redacted](a_i), a_i [redacted]

6. Ngôn ngữ tân từ có biến là n bộ

6.1 Qui tắc

6.2 Định nghĩa

6.3 Công thức an toàn

6.4 Biểu diễn các phép toán

6.1 Quy tắc (1)

1. Biến là 1 bộ của quan hệ
2. Từ: hằng, biến hoặc biểu thức có dạng s[C], s: biến, C: tập các thuộc tính của quan hệ được gọi là từ chiếu.
3. Công thức:
   - Rs (R là quan hệ, s là biến) được gọi là từ. Miền giá trị sẽ định nghĩa miền biến thiên của s.
   - t_1 a , t_1 t_2 ở đây t_1,t_2 là các từ chiếu, còn a là một hằng, à toán tử so sánh dược gọi là công thức nguyên tố

6.1 Quy tắc (2)

4. Một *công thức nguyên tố* là một công thức
5. F_1 và F_2 là công thức: F_1 [redacted] F_2, F_1 [redacted] F_2, F_1 [redacted], [redacted] là công thức
6. F là công thức , s là biến [redacted]F, [redacted]F là công thức
7. F là công thức, (F) là công thức

6.2 Định nghĩa

- Một câu hỏi trong ngôn ngữ tân từ có biến là n bộ được biểu diễn như sau: {s | F} . Trong đó s là biến n bộ, F là một công thức chỉ có một biến tự do là s.
- **Ví dụ**: BIENGIOI(nuoc,tinhtp). Phép toán quan hệ BIENGIOI[nuoc] được chuyển thành câu hỏi trong ngôn ngữ tân từ có biến là bộ: {s[nuoc] BIENGIOI s}

6.3 Công thức an toàn

F là công thức an toàn: nếu nó thoả mãn 3 điều kiện sau:

i) Nếu s là bộ n thỏa: F(s) là đúng thì mọi thành phần của s là phần tử của DOM(F):

$$(F_s{:}Đúng) \blacksquare OM(F)$$

ii) F’ là công thức con của F:

$$\blacksquare F'_s, F'_s{:}Đúng \blacksquare OM(F')$$

iii) $\blacksquare F'_s, F'_s{:}Đúng \blacksquare OM(F')$

6.4 Biểu diễn các phép toán (1)

- **1. Phép hội**
  - Q_1,Q_2 là các quan hệ n chiều
  - F_1, F_2 là các công thức ứng với Q_1, Q_2
  - Công thức của Q= Q_1 Q_2
  - F_s=F_{1s} F_{2s}
- **2. Phép trừ**
  - Q_1,Q_2 là các quan hệ n chiều
  - F_1, F_2 là các công thức ứng với Q_1, Q_2
  - Công thức của Q= Q_1-Q_2
  - F_s=F_1 F_{2s}

6.4 Biểu diễn các phép toán (2)

- **3. Phép tích**
  - $Q_1(x_1,\ldots,x_m)$, $Q_2(y_1,\ldots,y_n)$
  - F_1, F_2 là các công thức ứng với Q_1, Q_2
  - Công thức của $Q = Q_1 \times Q_2$

F_s: $s(x_1,\ldots,x_m, y_1,\ldots,y_n)$

$F_s = (\blacksquare)\ (\blacksquare)\ (F_{1v} \blacksquare F_{2p} \blacksquare$

$s_1 = v_1 \blacksquare \ldots s_m = v_m \blacksquare_{m+1} = p_1 \blacksquare \ldots\ s_{m+n} = p_n)$

6.4 Biểu diễn các phép toán (3)

- **4. Phép chiếu**
  - $Q_1(x_1,\ldots,x_n)$, F_1 là các công thức ứng với Q_1
  - Công thức của $Q = Q_1[x_{i1}, x_{i2},\ldots,x_{ik}]$

    F_s=( ) (F_{1v} $_1$=v_{i1} $_2$=v_{i2} . s_k=v_{ik})
- **5. Phép chọn**
  - Q_1 là quan hệ n chiều, F_1 là công thức ứng với Q_1
  - Công thức Q=Q_1:điều kiện ĐK (ĐK:x_i $_j$ hoặc x_i)

    F_s=F_{1s} $_i$ $_j$ hoặc F_{1s} $_i$ (1 j , i

7. Ngôn ngữ tân từ có biến là miền giá trị

7.1 Quy tắc

7.2 Biểu diễn câu hỏi

7.3 Công thức an toàn

7.4 Biểu diễn các phép toán

7.1 Quy tắc

1. Từ: là hằng hoặc biến
2. Công thức nguyên tố
   - $Q(t_1,t_2,\ldots,t_n)$: t_i là các từ, Q là quan hệ
   - t_i [redacted] t_j , t_i [redacted] với t_i là từ, a là một hằng, [redacted] là phép toán
3. Một *công thức nguyên tố* là một công thức
4. F_1 và F_2 là công thức: F_1 [redacted] F_2, F_1 [redacted] F_2, F_1 [redacted], [redacted] là công thức
5. F là công thức , t:biến tự do, [redacted]F, [redacted]F cũng công thức
6. F là công thức, (F) là công thức

7.2 Biểu diễn câu hỏi

$\{(x_1,x_2,\ldots,x_n) \mid F(x_1,x_2,\ldots,x_n)\}$

- x_i là các biến tự do của F
- $Q= \{(x_1,x_2,\ldots,x_n) \mid F(x_1,x_2,\ldots,x_n)\}$ nên $(x_1,x_2,\ldots,x_n)$ [redacted] $(x_1,x_2,\ldots,x_n)$:Đúng

7.3 Công thức an toàn

F là công thức an toàn: nếu nó thoả mãn 3 điều kiện sau:

i) Nếu s là bộ n thỏa: F(s) là đúng thì mọi thành phần của s là phần tử của DOM(F):

$$(F(x_1,\ldots,x_n):Đúng)\ \text{[redacted]}\ OM(F)\ ,i\ \text{[redacted]}\ldots,n$$

ii) F’ là công thức con của F:

$$\text{[redacted]}\ F':Đúng\ \text{[redacted]}\ OM(F')$$

iii) $\text{[redacted]}\ F':Đúng\ \text{[redacted]}\ OM(F')$

$$(F(x_1,\ldots,x_n):Đúng)\ \text{[redacted]}\ OM(F)\ ,i\ \text{[redacted]}\ldots,n$$

7.4 Biểu diễn các phép toán (1)

- **1. Phép hội**
  - Q_1,Q_2 là các quan hệ n chiều
  - F_1, F_2 là các công thức ứng với Q_1, Q_2
  - Công thức của Q= Q_1 Q_2
  - F=F_1 F_2
- **2. Phép trừ**
  - Q_1,Q_2 là các quan hệ n chiều
  - F_1, F_2 là các công thức ứng với Q_1, Q_2
  - Công thức của Q= Q_1-Q_2
  - F=F_1 F_2

7.4 Biểu diễn các phép toán (2)

- **3. Phép tích**
  - $Q_1(x_1,\ldots,x_m)$, $Q_2(y_1,\ldots,y_n)$
  - F_1, F_2 là các công thức ứng với Q_1, Q_2
  - Công thức của $Q = Q_1 \times Q_2$

    $F(x_1,\ldots,x_m, y_1,\ldots,y_n) = F_1(x_1,\ldots,x_m) \blacksquare F_2(y_1,\ldots,y_n)$

7.4 Biểu diễn các phép toán (3)

- **4. Phép chiếu**
  - $Q_1(x_1,\ldots,x_n)$, $F_1(x_1,\ldots,x_n)$ là các công thức ứng với Q_1
  - Công thức của $Q = Q_1[x_{i1}, x_{i2},\ldots,x_{ik}]$

    $F_s(x_{i1}, x_{i2},\ldots,x_{ik}) = (\square_{i1})(\square_{jz})\ldots(\square_{jn-k})(F_1(x_1,\ldots,x_n))$

    trong đó $(x_{i1}, x_{i2},\ldots,x_{ik})\square_{j1}, x_{j2},\ldots,x_{jn-k}) = (x_1, x_2,\ldots,x_n)$
- **5. Phép chọn**
  - $Q_1(x_1,\ldots,x_n)$, $F_1(x_1,\ldots,x_n)$ là các công thức ứng với Q_1
  - Công thức $Q = Q_1$:điều kiện ĐK (ĐK:$x_i \square x_j$ hoặc $x_i \square$)

    $F_1(x_1,\ldots,x_n) = F_1(x_1,\ldots,x_n) \square x_i \square x_j$ hoặc

    $= F_1(x_1,\ldots,x_n) \square x_i \square$

Bài 7: Ràng buộc toàn vẹn

Nội dung chính

1. Giới thiệu ràng buộc toàn vẹn (RBTV)
2. Các đặc trưng của một RBTV
3. Phân loại RBTV
4. Bảng tầm ảnh hưởng tổng hợp

1. Giới thiệu

- Ràng buộc toàn vẹn là các quy định, điều kiện từ ứng dụng thực tế, các điều kiện này là bất biến.

[redacted] thế phải luôn đảm bảo cơ sở dữ liệu thoả ràng buộc toàn vẹn sau mỗi thao tác làm thay đổi tình trạng của cơ sở dữ liệu.

2. Các đặc trưng của một RBTV

2.1 Nội dung

2.2 Bối cảnh

2.3 Bảng tầm ảnh hưởng

2.1 Nội dung

- Mô tả chặt chẽ ý nghĩa của ràng buộc toàn vẹn.
- Nội dung được phát biểu bằng ngôn ngữ tự nhiên hoặc bằng ngôn ngữ hình thức (ngôn ngữ tân từ, đại số quan hệ, mã giả,…)
  - Ngôn ngữ tự nhiên: dễ hiểu nhưng không chặt chẽ, logic.
  - Ngôn ngữ hình thức: chặt chẽ, cô đọng

2.2 Bối cảnh

- Là tập các quan hệ khi thao tác trên những quan hệ đó có khả năng làm cho ràng buộc bị vi phạm.
- Đó là những quan hệ có thể vi phạm ràng buộc toàn vẹn khi thực hiện các thao tác thêm, xoá, sửa.

2.3 Bảng tầm ảnh hưởng (1)

- Nhằm xác định khi nào tiến hành kiểm tra ràng buộc toàn vẹn. Thao tác nào thực hiện có thể làm vi phạm ràng buộc toàn vẹn.
- Phạm vi ảnh hưởng của một ràng buộc toàn vẹn được biểu diễn bằng một bảng 2 chiều gọi là *bảng tầm ảnh hưởng*.

2.3 Bảng tầm ảnh hưởng (2)

Một số quy định

- Những thuộc tính khoá (những thuộc tính nằm trong khoá chính của quan hệ) không được phép sửa giá trị
- Thao tác thêm và xoá xét trên một bộ của quan hệ. Thao tác sửa xét sửa từng thuộc tính trên bộ của quan hệ
- Trước khi xét thao tác thực hiện có thể làm vi phạm ràng buộc hay không thì CSDL phải thoả ràng buộc toàn vẹn trước.

2.3 Bảng tầm ảnh hưởng (3)

- **Bảng tầm ảnh hưởng của một ràng buộc**

Ràng buộc Ri	Thêm	Xóa	Sửa
Quan hệ 1			
……			
Quan hệ n			

\+ : thực hiện thao tác có thể làm vi phạm RBTV

\- : thực hiện thao tác không thể làm vi phạm RBTV

+(A) : có thể làm vi phạm RBTV khi sửa trên thuộc tính A

$\_^{(*)}$: không vi phạm RBTV do thao tác không thực hiện được

3. Phân loại

3.1 RBTV có bối cảnh trên 1 quan hệ

3.2 RBTV có bối cảnh trên nhiều quan hệ

3.3 Phụ thuộc hàm (functional dependency)

3.1 RBTV có bối cảnh 1 quan hệ

3.1.1 RBTV miền giá trị.

3.1.2 RBTV liên thuộc tính

3.1.3 RBTV liên bộ

Lược đồ CSDL quản lý giáo vụ

HOCVIEN (MAHV, HO, TEN, NGSINH, GIOITINH, NOISINH, MALOP)

LOP (MALOP, TENLOP, TRGLOP, SISO, MAGVCN)

KHOA (MAKHOA, TENKHOA, NGTLAP, TRGKHOA)

MONHOC (MAMH, TENMH, TCLT, TCTH, MAKHOA)

DIEUKIEN (MAMH, MAMH_TRUOC)

GIAOVIEN(MAGV,HOTEN,HOCVI,HOCHAM,GIOITINH,NGSINH,NGVL, HESO, MUCLUONG, MAKHOA)

GIANGDAY(MALOP,MAMH,MAGV,HOCKY, NAM,TUNGAY,DENNGAY)

KETQUATHI (MAHV, MAMH, LANTHI, NGTHI, DIEM, KQUA)

3.1.1 Ràng buộc miền giá trị

- Là tập giá trị mà một thuộc tính có thể nhận.
- **R1:** Giới tính của học viên chỉ là Nam hoặc Nữ
  - Nội dung:

    ∀ hv ∈ HOCVIEN: hv.Gioitinh ∈ {'Nam','Nữ'}
  - Bối cảnh: quan hệ HOCVIEN
  - Bảng tầm ảnh hưởng:

R1	Thêm	Xóa	Sửa
HOCVIEN	+	-	+(Gioitinh)

3.1.2 Ràng buộc liên thuộc tính

- Là ràng buộc giữa các thuộc tính với nhau trên 1 bộ của quan hệ
- **R2:**Ngày bắt đầu (TUNGAY) giảng dạy một môn học cho một lớp luôn nhỏ hơn ngày kết thúc (DENNGAY)
  - Nội dung:

    [redacted]GIANGDAY: gd.TUNGAY < gd.DENNGAY
  - Bối cảnh : GIANGDAY
  - Bảng tầm ảnh hưởng:

R2	Thêm	Xóa	Sửa
GIANGDAY	+	-	+(Tungay, Denngay)

3.1.3 Ràng buộc liên bộ (1)

- Là ràng buộc giữa các bộ trên cùng một quan hệ (có thể liên quan đến nhiều thuộc tính).
- **R3:** Tất cả các học viên phải có mã số phân biệt với nhau
  - Nội dung:
    h_1,h_2 HOCVIEN: Nếu h_1 $\neq$ h_2 thì h_1.Mahv $\neq$ h_2.Mahv
  - Bối cảnh: quan hệ HOCVIEN
  - Bảng tầm ảnh hưởng:

R3	Thêm	Xóa	Sửa
HOCVIEN	+	-	-(*)

3.1.3 Ràng buộc liên bộ (2)

- **R4:** Các giáo viên có cùng học vị, cùng hệ số lương thì mức lương sẽ bằng nhau
  - Nội dung:

    $\forall gv_1, gv_2 \in$ GIAOVIEN:

    Nếu (gv_1.Hocvi=gv_2.Hocvi) $\wedge$ (gv_1.Heso=gv_2.Heso) thì gv.Mucluong=gv.Mucluong
  - Bối cảnh: quan hệ GIAOVIEN
  - Bảng tầm ảnh hưởng:

R4	Thêm	Xóa	Sửa
GIAOVIEN	+	-	+(Hocvi, Heso, Mucluong)

3.2 RBTV có bối cảnh nhiều quan hệ

3.2.1 RBTV tham chiếu (khoá ngoại, phụ thuộc tồn tại)

3.2.2 RBTV liên thuộc tính

3.2.3 RBTV do thuộc tính tổng hợp

3.2.4 RBTV do chu trình trong lược đồ biểu diễn quan hệ

3.2.1 Ràng buộc tham chiếu (1)

- Là ràng buộc quy định giá trị thuộc tính trong một bộ của quan hệ R (tập thuộc tính này gọi là khoá ngoại), phải phụ thuộc vào sự tồn tại của một bộ trong quan hệ S (tập thuộc tính này là khoá chính trong quan hệ S).
- RBTV tham chiếu còn gọi là ràng buộc phụ thuộc tồn tại hay ràng buộc khóa ngoại

3.2.1 Ràng buộc tham chiếu (2)

- **R5:** Học viên thi một môn học nào đó thì môn học đó phải có trong danh sách các môn học
  - Nội dung:
    - [redacted]KETQUATHI, [redacted]MONHOC: k.Mamh = m.Mamh
    - Hoặc: KETQUATHI[Mamh] [redacted]MONHOC[Mamh]
  - Bối cảnh: quan hệ KETQUATHI, MONHOC
  - Bảng tầm ảnh hưởng:

R5	Thêm	Xóa	Sửa
KETQUATHI	+	-	-(*)
MONHOC	-	+	-(*)

3.2.2 Ràng buộc liên thuộc tính (1)

- Là ràng buộc giữa các thuộc tính trên những quan hệ khác nhau
- **R6: N**gày giáo viên giảng dạy một môn học phải lớn hơn hoặc bằng ngày giáo viên đó vào làm.
  - Nội dung: [redacted]d [redacted]GIANGDAY
    Nếu [redacted]v [redacted]GIAOVIEN: gd.Magv = gv.Magv thì
    gv.NGVL [redacted]gd.TUNGAY
  - Bối cảnh: GIANGDAY, GIAOVIEN
  - Bảng tầm ảnh hưởng:

R6	Thêm	Xóa	Sửa
GIANGDAY	+	-	+(Tungay)
GIAOVIEN	-	-	+(Ngvl)

3.2.2 Ràng buộc liên thuộc tính (2)

- **R7:** Ngày thi một môn học phải lớn hơn ngày kết thúc học môn học đó.
  - Nội dung:
    [redacted]q [redacted]KETQUATHI
    Nếu [redacted]d [redacted]IANGDAY, [redacted]v [redacted]OCVIEN:
    (gd.Malop=hv.Malop)[redacted]kq.Mamh=gd.Mamh) thì
    gd.Denngay < kq.Ngthi
  - Bối cảnh: GIANGDAY, HOCVIEN, KETQUATHI

3.2.2 Ràng buộc liên thuộc tính (3)

- Bảng tầm ảnh hưởng:

R7	Thêm	Xóa	Sửa
HOCVIEN	-	-	+(Malop)
GIANGDAY	-	-	+(Denngay)
KETQUATHI	+	-	+(Ngthi)

3.2.3 RBTV do thuộc tính tổng hợp (1)

- Là ràng buộc giữa các thuộc tính, các bộ trên những quan hệ khác nhau.
- Thuộc tính tổng hợp là thuộc tính được tính toán từ giá trị của các thuộc tính khác, các bộ khác.
- **Ví dụ :** SANPHAM(Masp,Tensp, Nuocsx, Gia)
  KHACHHANG(Makh, Hoten, Doanhso)
  HOADON(Sohd, Nghd,Makh,Trigia)
  CTHD(Sohd,Masp,Soluong,Gia)
  - Trị giá của một hoá đơn bằng tổng thành tiền của các chi tiết thuộc hoá đơn đó

3.2.3 RBTV do thuộc tính tổng hợp (2)

- Doanh số của một khách hàng bằng tổng trị giá các hoá đơn mà khách hàng đó đã mua
  - Nội dung:
    [redacted]KHACHHANG,
    kh.Doanhso = [redacted](hd.Trigia)
  - Bối cảnh: KHACHHANG, HOADON
  - Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
KHACHHANG	+	-	+(Doanhso)
HOADON	+	+	+(Makh)

3.2.3 RBTV do thuộc tính tổng hợp (3)

- **R8:** Sỉ số của một lớp là số lượng học viên thuộc lớp đó
  - Nội dung:

    [redacted]LOP,

    l.Siso = Count$_{(\text{hv } [redacted]\text{HOCVIEN: hv.Malop = lp.Malop})}$(*)
  - Bối cảnh: quan hệ LOP, HOCVIEN
  - Bảng tầm ảnh hưởng:

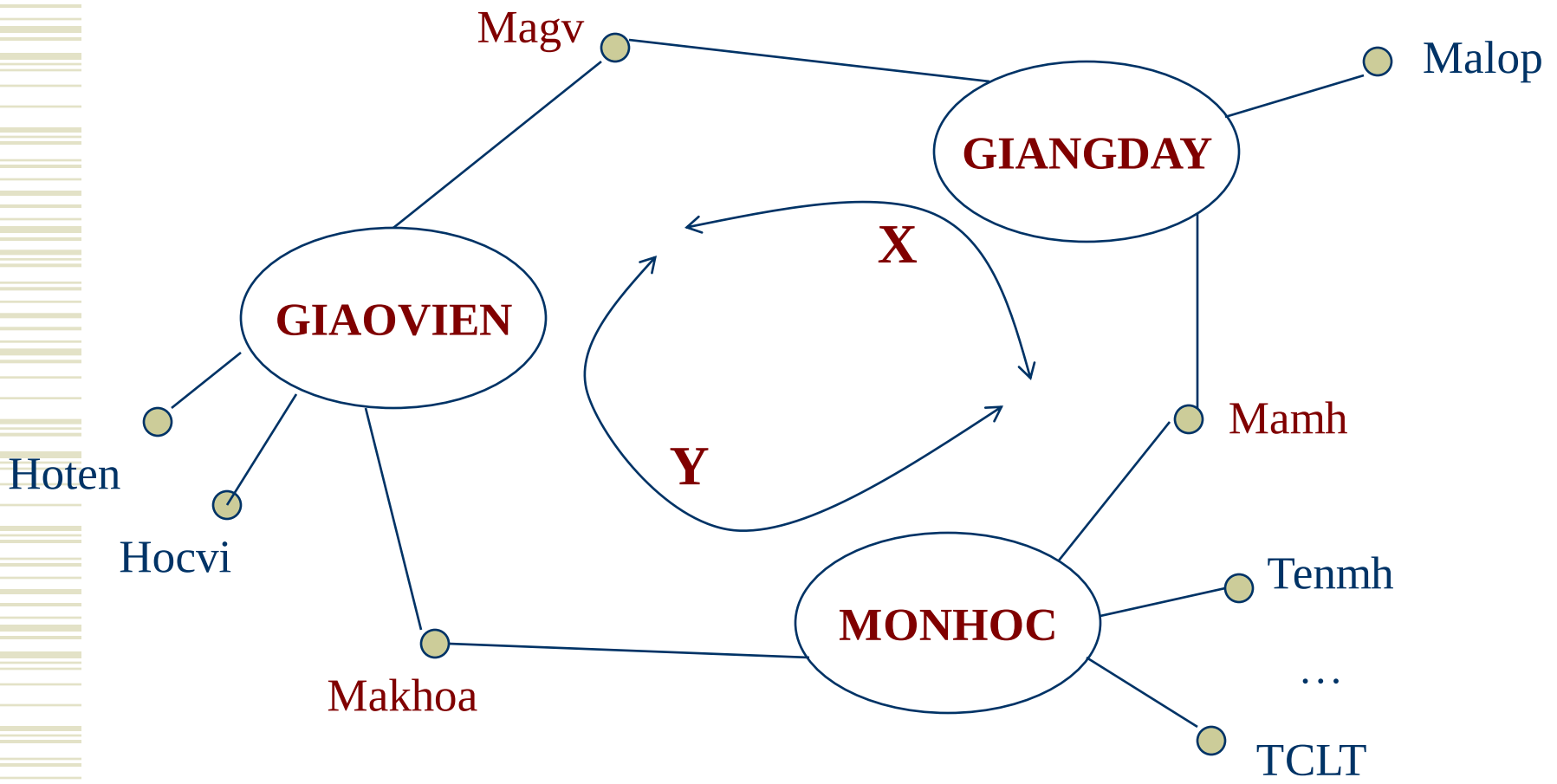
R8	Thêm	Xóa	Sửa
LOP	+	-	+(Siso)
HOCVIEN	+	+	+(Malop)

3.2.4 Do hiện diện của chu trình (1)

Biểu diễn lược đồ quan hệ dưới dạng đồ thị:

- Quan hệ được biểu diễn bằng nút tròn rỗng to
- Thuộc tính được biểu diễn bằng nút tròn đặc nhỏ
- Tất cả các nút đều được chỉ rõ bằng tên của quan hệ hoặc thuộc tính. Thuộc tính thuộc một quan hệ được biểu diễn bởi một cung nối giữa nút tròn to và nút tròn nhỏ
- Nếu đồ thị biểu diễn xuất hiện một đường khép kín => lược đồ CSDL có sự hiện diện của chu trình.

3.2.4 Do hiện diện của chu trình (2)



3.2.4 Do hiện diện của chu trình (3)

- X = GIANGDAY[Magv, Mamh]
- Y = (GIAOVIEN $\overset{\text{Makhoa}}{\bowtie}$ MONHOC) [Magv,Mamh]
- **Ý nghĩa:**
  - X: giáo viên và những môn học đã được phân công cho giáo viên đó giảng dạy
  - Y: giáo viên và những môn học thuộc khoa giáo viên đó phụ trách
- Mối quan hệ giữa X và Y trong các ràng buộc sau:

3.2.4 Do hiện diện của chu trình (4)

- **Ràng buộc 1**: giáo viên chỉ được phân công giảng dạy những môn thuộc khoa giáo viên đó phụ trách **X**
- **Ràng buộc 2**: giáo viên phải được phân công giảng dạy tất cả những môn thuộc khoa giáo viên đó phụ trách **X=Y**
- **Ràng buộc 3**: có thể phân công giáo viên giảng dạy bất kỳ môn học nào **X Y**

3.2.4 Do hiện diện của chu trình (4)

- **R9**: giáo viên chỉ được phân công giảng dạy những môn thuộc khoa giáo viên đó phụ trách **X**

R9	Thêm	Xóa	Sửa
MONHOC	-	-	+(Makhoa)
GIAOVIEN	-	-	+(Makhoa)
GIANGDAY	+	-	+(Magv)

3.3 Phụ thuộc hàm (1)

- Cho quan hệ Q(A, B, C). Phụ thuộc hàm A xác định B. Ký hiệu A [redacted] nếu:

  [redacted],q_2[redacted]: Nếu $q_1.A=q_2.A$ thì $q_1.B=q_2.B$

- A [redacted] được gọi là *phụ thuộc hàm hiển nhiên* nếu B[redacted]

- A [redacted] được gọi là *phụ thuộc hàm nguyên tố* nếu [redacted]', [redacted], A'[redacted] sao cho A'[redacted]

3.3 Phụ thuộc hàm (2)

- Mỗi quan hệ đều có ít nhất một phụ thuộc hàm
- Ràng buộc khoá cũng là một phụ thuộc hàm

  Mamh [redacted]enmh, Tclt, Tcth, Makhoa
- **R4:** Các giáo viên có cùng học vị, cùng hệ số lương thì mức lương sẽ bằng nhau. Ràng buộc này có thể biểu diễn bằng phụ thuộc hàm như sau:

  Hocvi,Heso [redacted]ucluong

4. Bảng tầm ảnh hưởng tổng hợp (1)

- **Bảng tầm ảnh hưởng tổng hợp** của m ràng buộc trên n quan hệ bối cảnh

	QH_1			**QH_2**			**…**	**QH_n**		
	T	X	S	T	X	S	…	T	X	S
R1										
R2										
…										
Rm										

4. Bảng tầm ảnh hưởng tổng hợp (2)

	HOCVIEN			GIAOVIEN			LOP			MONHOC			GIANGDAY			KETQUA THI		
	T	X	S	T	X	S	T	X	S	T	X	S	T	X	S	T	X	S
R1	+	-	+															
R2													+	-	+			
R3	+	-	-*															
R4				+	-	+												
R5										-	+	-*				+	-	-*
R6				-	-	+							+	-	+			
R7	-	-	+										-	-	+	+	-	+
R8	+	+	+				+	-	+									
R9				-	-	+				-	-	+	+	-	+			

Bài 8: Tối ưu hóa câu hỏi

Nội dung

1. Giới thiệu
2. Các nguyên tắc tổng quát để tối ưu hóa câu hỏi
   - 2.1 Biểu thức tương đương
     - 2.1.1 Định nghĩa
     - 2.1.2 Tính chất của phép kết và phép tích
   - 2.2 Nguyên tắc tổng quát
   - 2.3 Các phép biến đổi tương đương
3. Một số kỹ thuật tối ưu hóa câu hỏi bằng ĐSQH
   - 3.1 Kỹ thuật (dãy phép chọn, phép chiếu, hoán vị …)
   - 3.2 Thuật giải tối ưu hoá câu hỏi trong .

1. Giới thiệu (1)

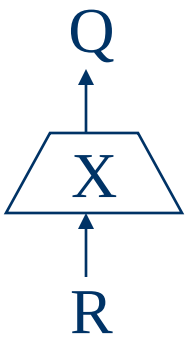
- Mục đích:
  - Giảm thời gian xử lý câu hỏi, giảm khối lượng dữ liệu trung gian.
  - Kết hợp giữa các phép tích, phép kết với phép chọn với phép chiếu.
- Ví dụ:

$$(Q_1 \triangleright\triangleleft Q_2) : A\ q_0)[C]$$

$$(Q_1 : A\ q_0) \triangleright\triangleleft Q_2)[C]$$

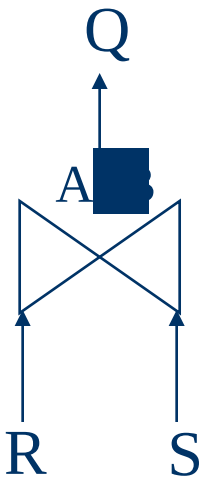
1. Giới thiệu (2)

- Ký hiệu:



Q=R[S]

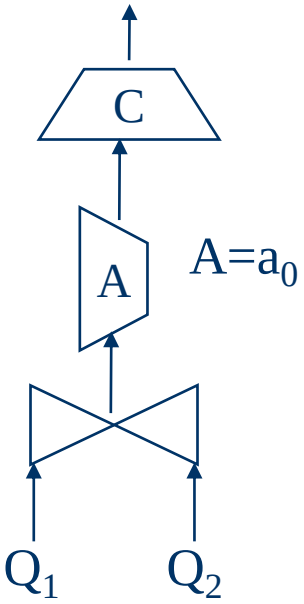
Q=R:D



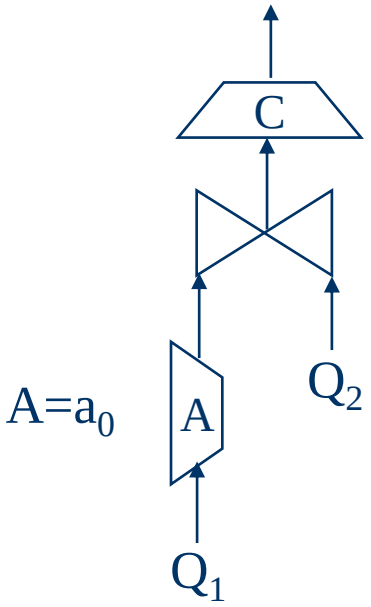
Q=R ⊳⊲ S

1. Giới thiệu (3)

- Ví dụ



$((Q_1 \triangleright\!\triangleleft Q_2) : A = a_0)[C]$



$((Q_1 : A = a_0) \triangleright\!\triangleleft Q_2)[C]$

2.1 Tính tương đương (1)

- **2.1.1 Định nghĩa**: hai biểu thức A, B là tương đương nếu có cùng một tình trạng CSDL thì đều cho một kết quả.
- **2.1.2 Tính chất của phép kết và phép tích**
  - **Phép kết**
    - Giao hoán $Q_1 \square Q_2 \equiv Q_2 \square Q_1$
    - Kết hợp $(Q_1 \square Q_2) \square Q_3 \equiv Q_1 \square (Q_2 \square Q_3)$
  - **Phép tích**
    - Giao hoán: $Q_1 \overset{dk}{\triangleright\!\triangleleft} Q_2 \equiv Q_2 \overset{dk}{\triangleright\!\triangleleft} Q_1$
    - Kết hợp: $(Q_1 \overset{dk1}{\triangleright\!\triangleleft} Q_2) \overset{dk2}{\triangleright\!\triangleleft} Q_3 \equiv Q_1 \overset{dk1}{\triangleright\!\triangleleft} (Q_2 \overset{dk2}{\triangleright\!\triangleleft} Q_3)$

2.1 Tính tương đương (2)

2.1.3 Các phép biến đổi tương đương

1. $Q_1(A,B) \overset{B}{\triangleright\!\triangleleft} Q_2(B,C) \equiv \pi(Q_1 \overset{B}{\triangleright\!\triangleleft} Q_2 : Q_1[B] \cap Q_2[B])$

2. $Q_1(A,B) \overset{B}{\triangleright\!\triangleleft} Q_2(C,D) \equiv \pi(Q_1 \overset{B}{\triangleright\!\triangleleft} Q_2 : B \cap D)$

3. $Q_1 \cap Q_2 \equiv Q_1 - (Q_1 - Q_2)$

4. $\pi(X_1,\ldots,X_n)(Q[X_1] \cap Q[X_2] \cap \ldots \cap Q[X_n]) \equiv \pi(X_1,\ldots,X_n)Q$

5. $Q_1(A,B) \cap Q_2(A,B) \equiv Q_1[B] \cap (Q_1[B] \cap Q_2[A] \cap Q_1(A,B))[B]$

2.2 Nguyên tắc tổng quát

1. Thực hiện phép chiếu, phép chọn càng sớm càng tốt
2. Gom các phép chọn và chiếu cùng quan hệ để thực hiện cùng lúc
3. Biến phép tích thành phép kết tự nhiên hay theta kết
4. Tìm các biểu thức con chung trong một biểu thức
5. Tiền xử lý các quan hệ: lập chỉ mục
6. Đánh giá trước khi thực hiên tính toán

3.1 Các kỹ thuật tối ưu (1)

1. Dãy các phép chọn
2. Dãy các phép chiếu
3. Hoán vị giữa phép chiếu và phép chọn
4. Hoán vị giữa phép chọn và phép tích
5. Hoán vị giữa phép hợp và phép chọn
6. Hoán vị giữa phép chọn và phép trừ
7. Hoán vị giữa phép chiếu và phép hội
8. Hoán vị giữa phép chiếu và phép tích

3.1 Các kỹ thuật tối ưu (2)

1. Dãy các phép chọn

$$(((Q:dk1):dk2)...:dkn) \equiv Q:dk1 \wedge k2 \wedge .dkn$$

2. Dãy phép chiếu

$$(Q[Y])[Z] \equiv Q[Z], \quad Z$$

Ví dụ: *Cho* $Q(A,B,C,D)$

$$(Q[A,C,D])[AD] \equiv Q[AD]$$

3.1 Các kỹ thuật tối ưu (3)

3. Hoán vị giữa phép chiếu và phép chọn

- Nếu X [redacted]

$$(Q : dk(X))[Y] \text{ [redacted] } Q[Y]) : dk(X)$$

- Nếu X [redacted]

$$(Q : dk(X))[Y] \text{ [redacted] } Q[X \text{ [redacted]}]) : dk(X)$$

3.1 Các kỹ thuật tối ưu (4)

4. Hoán vị giữa phép chọn và phép tích:

- Điều kiện dk xác lập trên các thuộc tính của X

$$(Q_1(X)) : dk(X) \quad Q_2(Y) \quad Q_1(X) \quad Q_2(Y)) : dk$$

- Nếu dk $dk1$ $k2$, dk1 xác lập trên các thuộc tính của X, dk2 xác lập trên các thuộc tính của Y.

$$((Q_1(X) \quad Q_2(Y)) : dk1(X) \quad k2(Y) \quad (Q_1(X) : dk1) \quad Q_2(Y) : dk2)$$

- Nếu dk1 xác lập trên các thuộc tính của X và dk2 xác lập trên các thuộc tính của X

$$((Q_1(X) \quad Q_2(Y)) : dk1(X) \quad k2(X \quad) \quad$$

$$((Q_1(X) : dk1) \quad Q_2(Y)) : dk2(X \quad))$$

3.1 Các kỹ thuật tối ưu (5)

5. Hoán vị giữa phép hội và phép chọn

$$(Q_1 \wedge Q_2) : dk \equiv (Q_1 : dk) \wedge (Q_2 : dk)$$

6. Hoán vị giữa phép chọn và phép trừ

$$(Q_1 - Q_2) : dk \equiv (Q_1 : dk) - Q_2 : dk)$$

7. Hoán vị giữa phép chiếu và phép hội

$$(Q_1 \wedge Q_2)[Z] \equiv (Q_1[Z]) \wedge (Q_2[Z])$$

8. Hoán vị giữa phép chiếu và phép tích

$$(Q_1(X) \times Q_2(Y))[Z] \equiv (Q_1[Y \cap Z]) \times (Q_2[Y \cap Z]), \quad Z \cap$$

3.2 Thuật toán

- **Bước 1**: Áp dụng các phép biến đổi tương đương
- **Bước 2**: Áp dụng (1)
- **Bước 3**: Đối với các phép chọn áp dụng (3), (4), (5), (6) nhằm đưa phép chọn càng sâu càng tốt
- **Bước 4**: Đối với các phép chiếu áp dụng (2), (3), (7), (8) nhằm đưa phép chiếu càng sâu càng tốt
- **Bước 5**:
  - Tập trung các phép chọn để áp dụng (1)
  - Kết hợp phép tích và phép chọn để chuyển thành phép kết

Nhập môn Cơ sở Dữ liệu

Phần 1: Giới thiệu về CSDL

Biên soạn : ThS.Phan Võ Minh Thắng

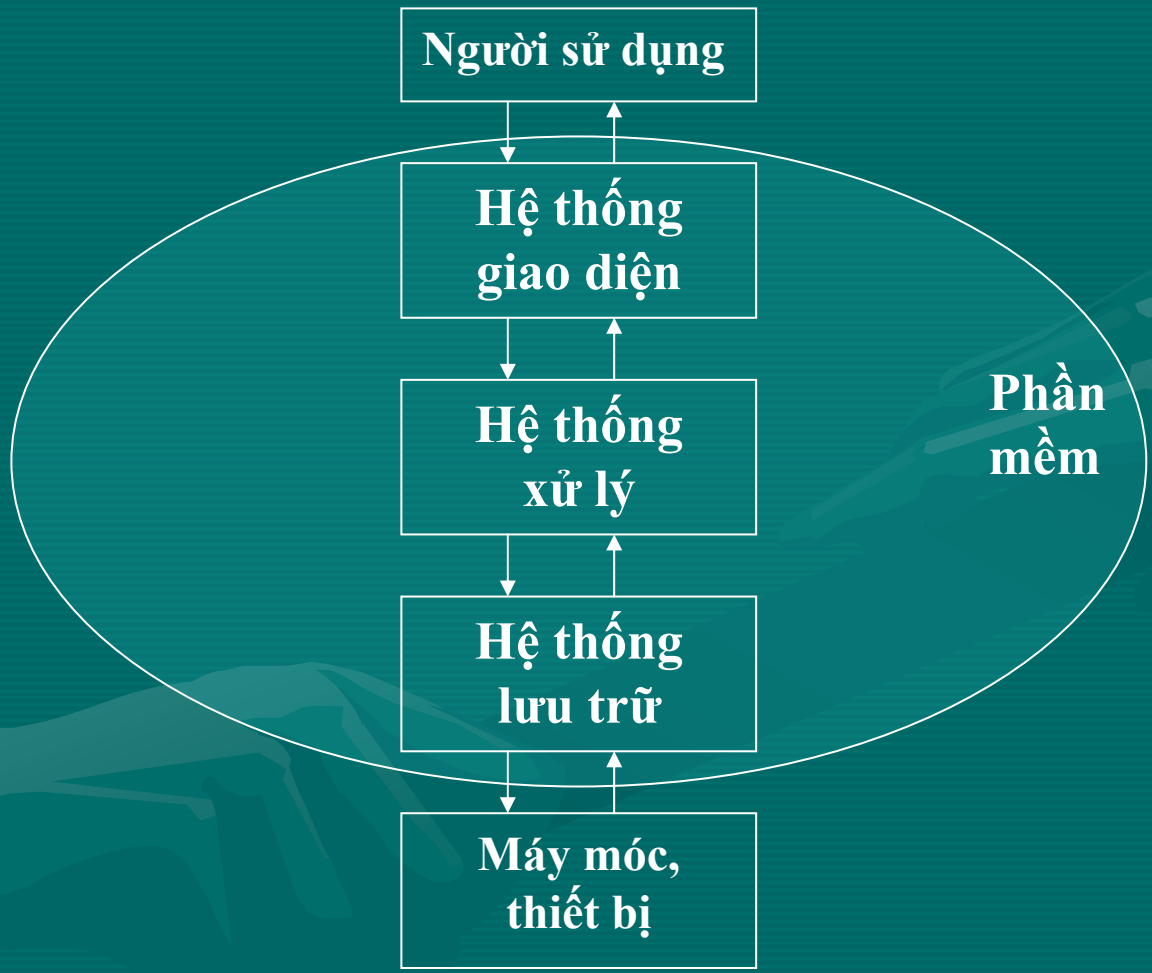
Nội dung

1. Dẫn nhập - Tại sao cần phải có một CSDL
2. Khái niệm về CSDL.
3. Hệ quản trị CSDL.
4. Các loại mô hình CSDL.

Mục tiêu:

Phân biệt rõ ràng giữa một hệ thống các file dữ liệu cổ điển với một cơ sở dữ liệu được tổ chức một cách có hệ thống, có cấu trúc trong môi trường làm việc nhiều người sử dụng đồng thời.

Dẫn nhập

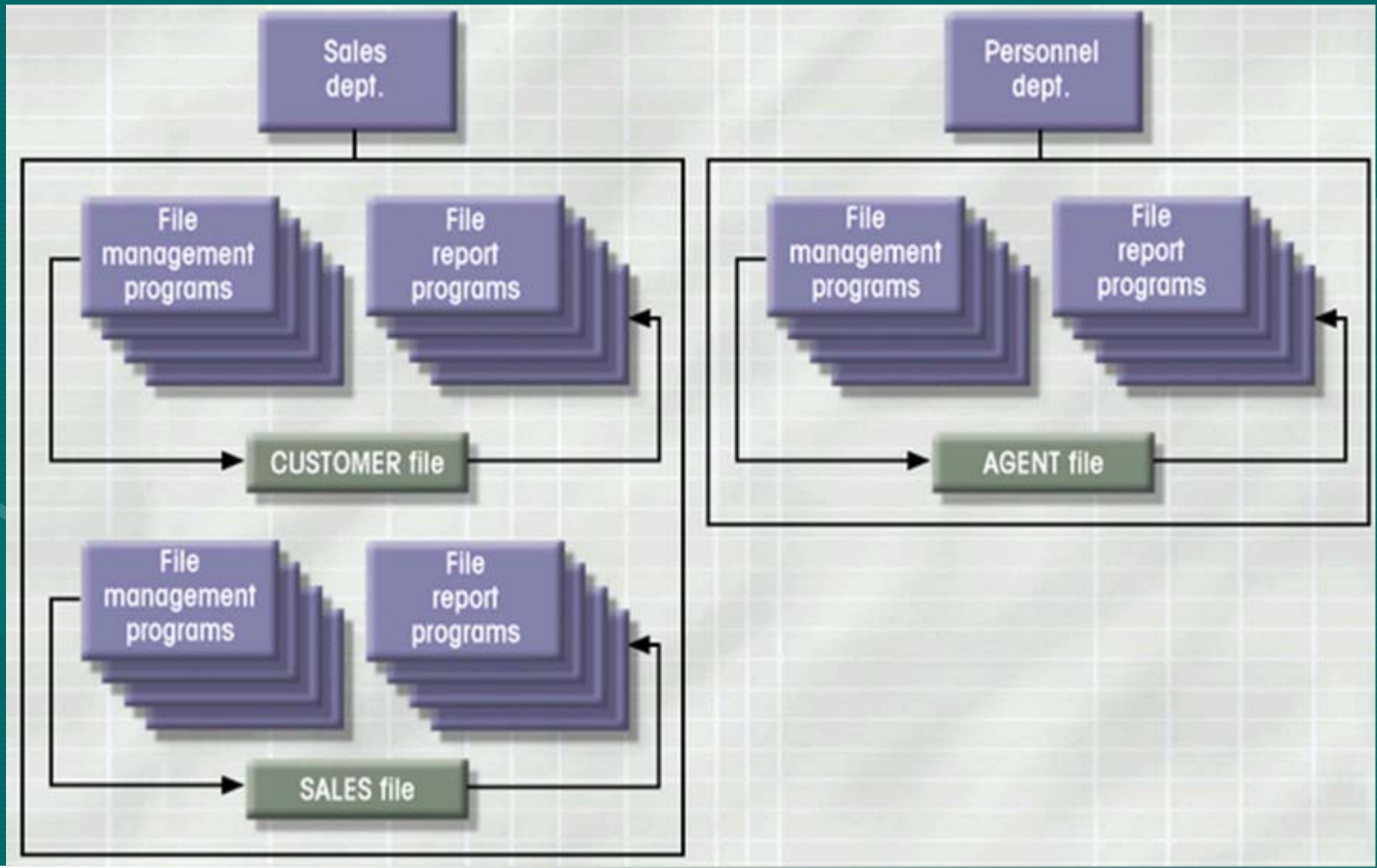


Mô hình phần mềm

Tại sao cần phải có một CSDL

- Hệ thống tập tin cổ điển:
  - Được tổ chức riêng rẽ, phục vụ cho một mục đích của một đơn vị hay một đơn vị con trực thuộc cụ thể .
  - Ưu điểm:
    - triển khai ứng dụng nhanh.
    - có khả năng đáp ứng nhanh chóng, kịp thời.

Ví dụ về hệ thống tập tin cổ điển



Các thuật ngữ trong cách tiếp cận tập tin cổ điển

- Dữ liệu - Data
  - Số liệu thô
- Trường - Field
  - Nhóm các đặc tính có một ý nghĩa nào đó
- Mẫu tin - Record
  - Các trường có liên kết với nhau một cách logic để mô tả một người, một nơi chốn, hay một vật
- Tập tin và thư mục tập tin – File, file folder
  - Tập hợp các mẫu tin có quan hệ

Nhược điểm hệ thống tập tin cổ điển

- Quản trị Dữ liệu của hệ thống tập tin cổ điển
  - Đòi hỏi phải dùng các ngôn ngữ lập trình để lập trình thêm
  - Tốn nhiều thời gian
  - Phụ thuộc vào việc lưu trữ dữ liệu về mặc vật lý
  - Không thể thực hiện những truy vấn thông tin phức tạp
  - Khó hiệu chỉnh các tập tin trong hệ thống tập tin cổ điển (vì mỗi file trực thuộc hệ thống riêng của nó)
  - Dẫn tới các islands of information

Nhược điểm hệ thống tập tin cổ điển

- Phụ thuộc dữ liệu
  - Thay đổi về các đặc tính của tập tin đòi hỏi phải hiệu chỉnh các chương trình truy cập dữ liệu.
  - Chương trình phải biết làm gì và làm thế nào
  - Hệ thống tập tin trở nên cồng kềnh trong việc lập trình và quản lý dữ liệu
- Phụ thuộc cấu trúc
  - Thay đổi về cấu trúc tập tin đòi hỏi phải hiệu chỉnh lại các chương trình có liên quan

Nhược điểm hệ thống tập tin cổ điển

- Định nghĩa trường và cách đặt tên
  - Định nghĩa các mẫu tin thoãi mái nhưng nó sẽ gây khó khăn về các yêu cầu báo cáo.
  - Chọn được các tên trường đúng là quan trọng
  - Chú ý tới độ dài của tên trường
  - Sử dụng các định danh duy nhất cho mẫu tin

Nhược điểm hệ thống tập tin cổ điển

- Dư thừa dữ liệu – data redundancy
  - Các phiên bản khác nhau và mâu thuẫn nhau của cùng một dữ liệu
  - Hậu quả của việc dư thừa dữ liệu không thể kiểm soát được
    - Lãng phí công sức nhập liệu và không gian lưu trữ.
    - Những dị thường về dữ liệu - data anomalies
      - Khi sửa đổi
      - Khi thêm
      - Khi xóa
    - Dữ liệu không nhất quán - Data inconsistency
      - Thiếu các ràng buộc về dữ liệu - data integrity
- Thiếu sự chia sẻ thông tin giữa các nơi.

Tại sao cần phải có một CSDL

- Chúng ta cần gì
  - Một hệ thống thông tin đảm bảo được tính chất nhất quán dữ liệu
  - Không trùng lặp thông tin mà vẫn đáp ứng được nhu cầu khai thác đồng thời của nhiều người dùng khác nhau.
- Cần làm rõ một số khái niệm
  - Dữ liệu
  - Thông tin

Dữ liệu Vs Thông tin

- Dữ liệu - Data
  - Số liệu thô
  - Được lưu trữ và truy cập
  - Chưa được xử lý để thể hiện ý nghĩa của nó cho người sử dụng
  - Ví dụ:
    - Công ty Robcor có hai chi nhánh và hai chi nhánh này lần lược có 1,380,456 và 1,453,907 hóa đơn.
    - Mỗi hóa đơn có mã hóa đơn, ngày hóa đơn, và số lượng
    - Giai đoạn xem xét là từ quý 1 năm 1997 tới quý 1 năm 2002.
    - Tổng cộng có 2,834,363 mẫu tin

Data →

...	...	...
3000124	**12-Jan-2002**	**$121.98**
...	...	

Dữ liệu Vs Thông tin

- Thông tin - Information
  - Dữ liệu đã được xử lý
  - Là mấu chốt để có được các quyết định tốt

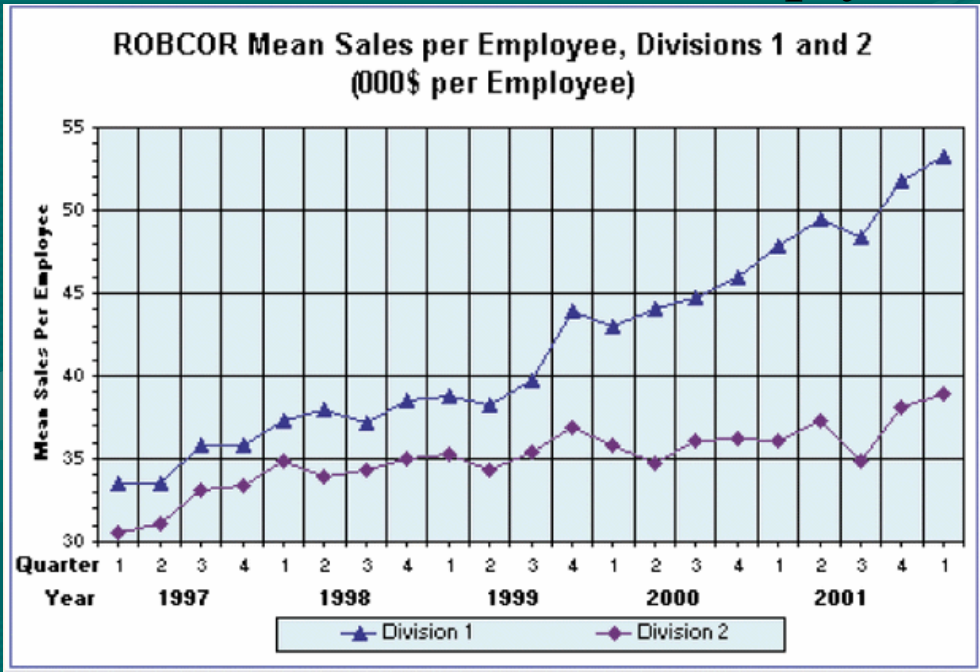


FIGURE 1.1 SALES PER EMPLOYEE FOR ROBCOR'S TWO DIVISIONS

Dữ liệu Vs Thông tin

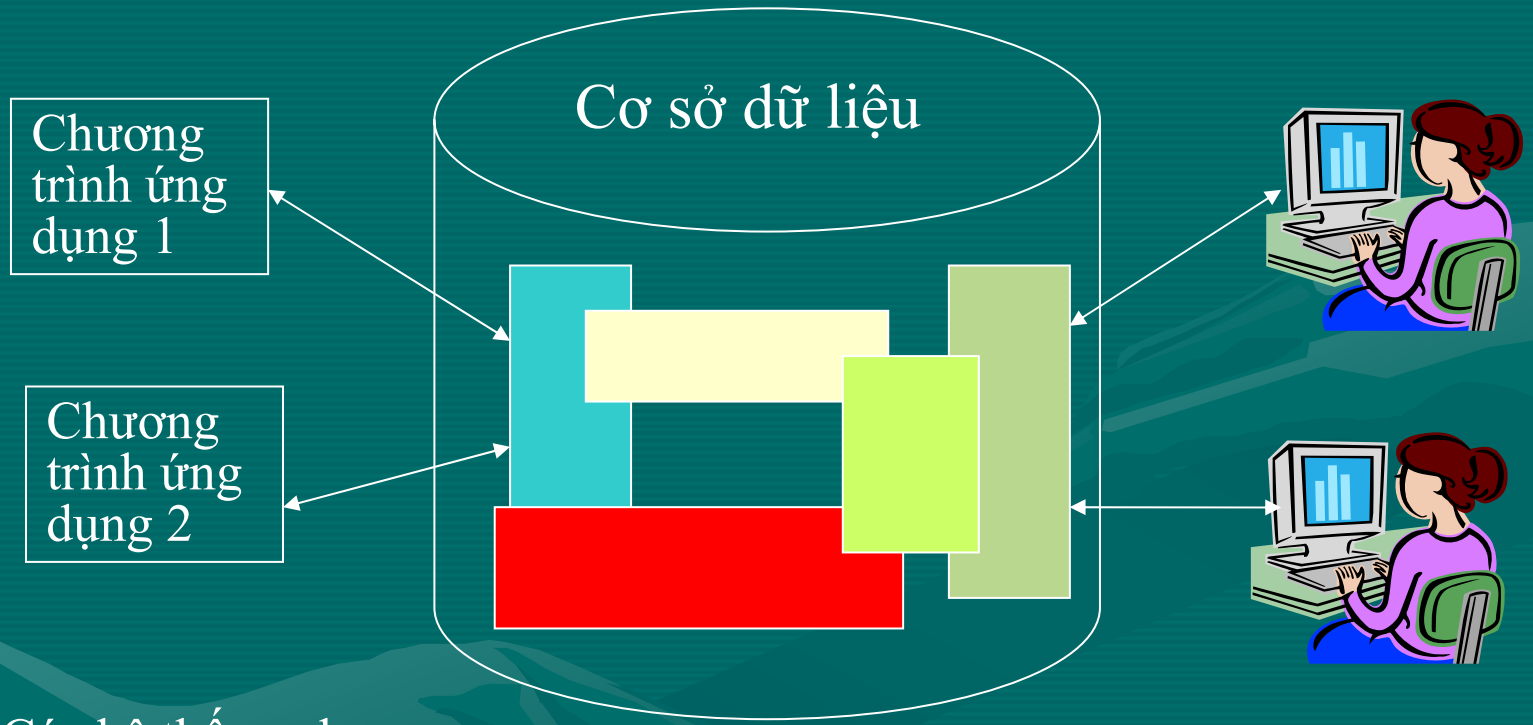
- Dữ liệu hợp thành thông tin
- Thông tin có được từ việc xử lý dữ liệu
- Thông tin lột tả ý nghĩa của dữ liệu
- Thông tin đúng đắng, kịp thời và đầy đủ là mấu chốt để thực hiện những quyết định
- Quyết định đúng đắng là then chốt quyết định sự sống còn của tổ chức

Khái niệm về Cơ sở Dữ liệu

Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc được lưu trữ trên các thiết bị lưu trữ thông tin thứ cấp (như băng từ, đĩa từ ...) để có thể thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với nhiều mục đích khác nhau.

- *CSDL phải là một tập hợp các thông tin mang tính hệ thống chứ không phải là các thông tin rời rạc, không có mối quan hệ với nhau.*
- *Các thông tin này phải có cấu trúc và tập hợp các thông tin này phải có khả năng đáp ứng các nhu cầu khai thác của nhiều người sử dụng một cách đồng thời*

Sơ đồ tổng quát về một Cơ sở dữ liệu



Ưu điểm của CSDL

- Giảm sự trùng lặp thông tin xuống mức thấp nhất và do đó bảo đảm được tính nhất quán và toàn vẹn dữ liệu.
- Đảm bảo dữ liệu có thể được truy xuất theo nhiều cách khác nhau.
- Khả năng chia sẻ thông tin cho nhiều người sử dụng và nhiều ứng dụng khác nhau.

Các cách tiếp cận CSDL

- Mô hình phân cấp
- Mô hình mạng
- Mô hình quan hệ
- Mô hình thực thể kết hợp
- Mô hình đối tượng

Các vấn đề với cách tiếp cận CSDL

- Tính chủ quyền của dữ liệu.
- Tính bảo mật và quyền khai thác thông tin của người sử dụng.
- Tranh chấp dữ liệu.
- Đảm bảo dữ liệu khi có sự cố.

Vấn đề quản trị CSDL

- CSDL được chia sẻ, tích hợp giữa các cấu trúc máy tính giữ các dữ liệu có liên quan:
  - Dữ liệu của người sử dụng (dữ liệu thô)
  - Metadata (dữ liệu về dữ liệu, nó chứa đựng các đặc tính và mối quan hệ giữa các dữ liệu với nhau)

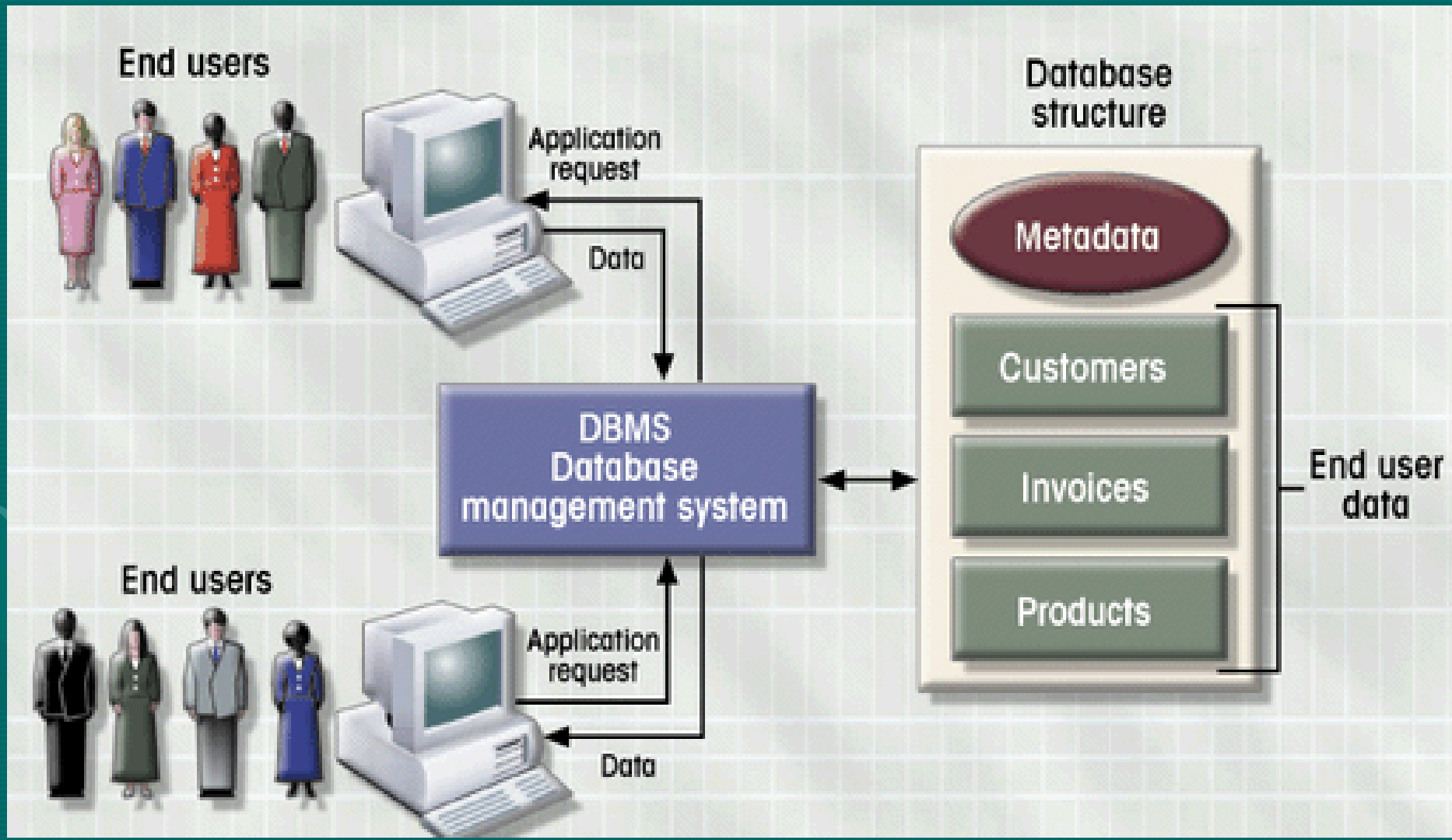
Hệ quản trị Cơ sở Dữ liệu

- Database Management System (DBMS): hệ thống phần mềm giúp giải quyết các vấn đề liên quan đến CSDL:
  - Quản lý nội dung dữ liệu
  - Quản lý cấu trúc CSDL
  - Cung cấp các công cụ để định nghĩa, tạo lập, khai thác và xử lý dữ liệu: môi trường phát triển (đồ họa), ngôn ngữ truy vấn,…
  - Kiểm soát truy cập dữ liệu,
  - …
- Cài đặt dựa trên một mô hình dữ liệu cụ thể.
- Một số hệ quản trị CSDL: Visual FoxPro, MicroSoft Access, SQL-Server, DB2, Sybase, Paradox, Informix, Oracle...

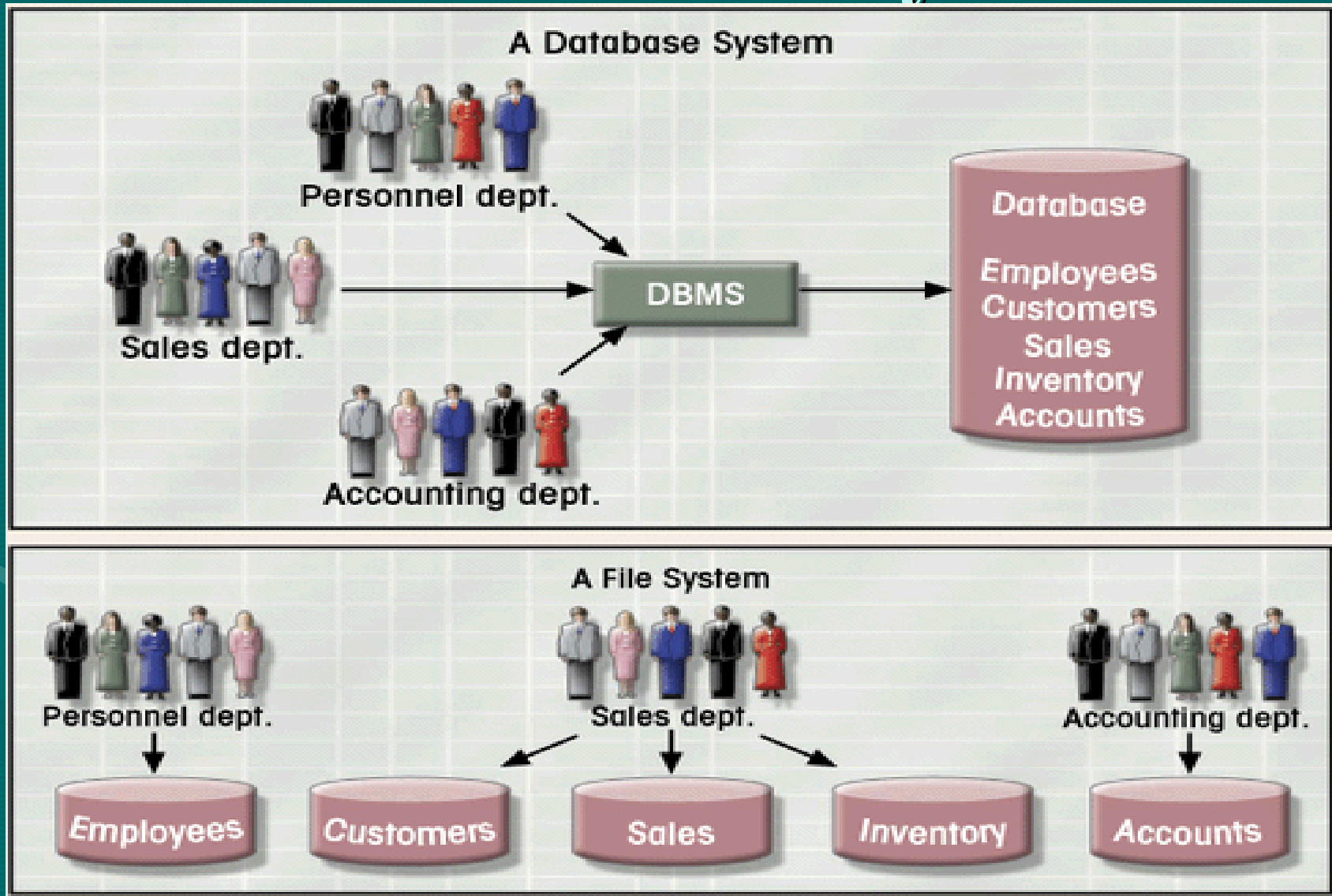
Tầm quan trọng của HQT CSDL

- Việc quản trị dữ liệu được hiệu quả hơn
- Ngôn ngữ truy vấn giúp trả lời các câu hỏi phức tạp một cách nhanh chóng
- Cho phép truy cập tốt hơn tới dữ liệu được quản lý một cách tốt hơn và nhiều hơn
- Tăng cường khả năng xem xét các hoạt động tích hợp của hệ thống
- Giảm khả năng xuất hiện của dữ liệu không nhất quán

DBMS quản lý sự tương tác



Database vs. File Systems



Ba mức độ biểu diễn CSDL

- ***Mức vật lý*** (Physical): Lưu trữ vật lý CSDL: dữ liệu gì và được lưu trữ như thế nào? ở đâu (đĩa từ, băng từ, track, sector ... nào)? Cần các chỉ mục gì?

- ***Mức quan niệm*** (Conception hay Logical)**:** CSDL cần phải lưu giữ bao nhiêu loại dữ liệu? những dữ liệu gì? Mối quan hệ giữa các loại dữ liệu?

CSDL mức quan niệm là một sự biểu diễn trừu tượng CSDL mức vật lý; và CSDL vật lý là sự cài đặt cụ thể của CSDL mức quan niệm.

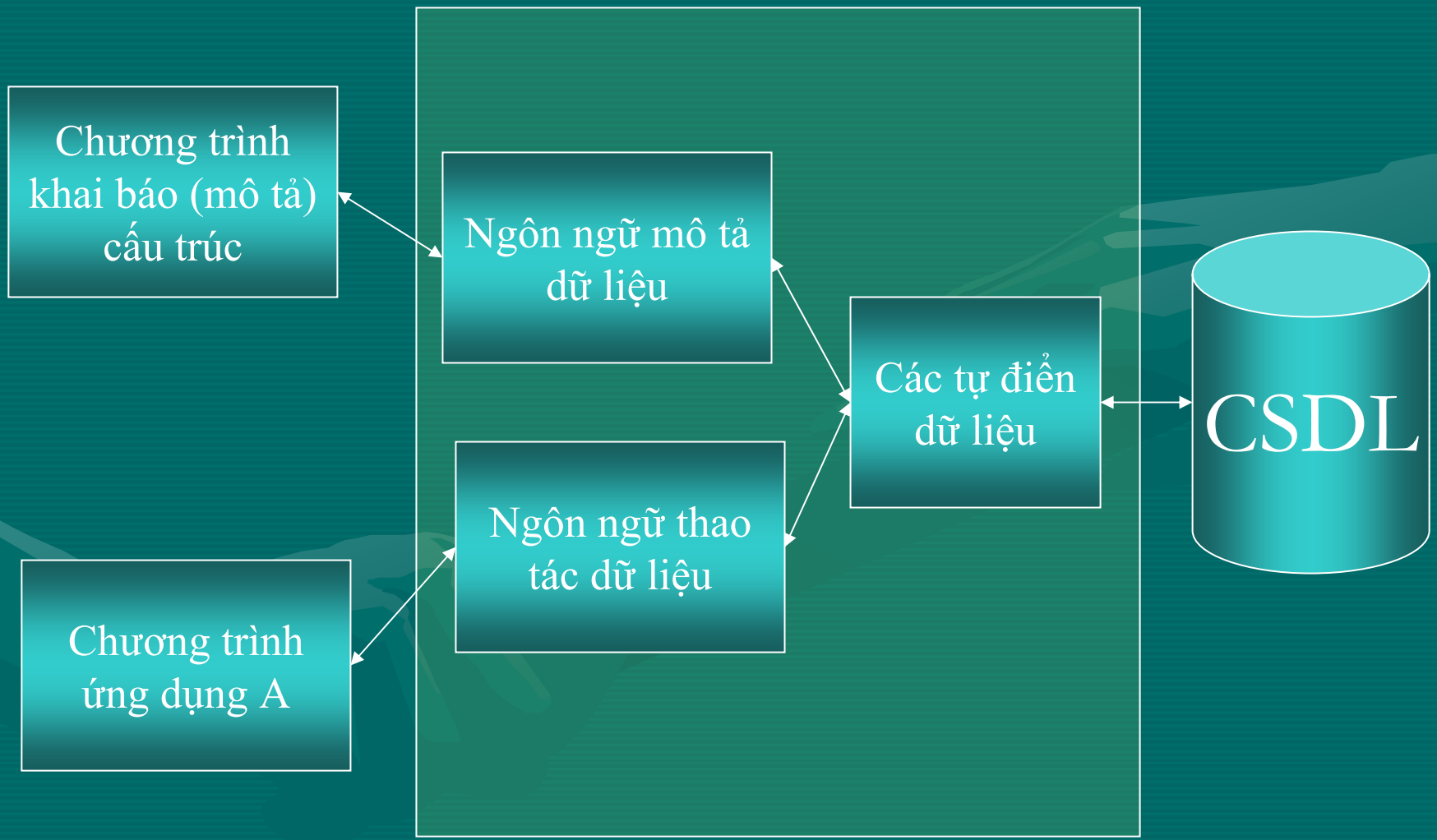
- ***Mức ngoài:*** Người sử dụng hay chương trình ứng dụng "nhìn" CSDL theo một góc độ toàn bộ hay chỉ một phần hoặc chỉ là các thông tin tổng hợp từ CSDL hiện có, do người quản trị hay chương trình ứng dụng quy định. Người sử dụng hay chương trình ứng dụng có thể hoàn toàn không được biết về cấu trúc tổ chức lưu trữ thông tin trong CSDL

Kiến trúc tổng quát ANSI-PARC của 1 cơ sở dữ liệu

Sơ đồ tổng quát của hệ quản trị CSDL

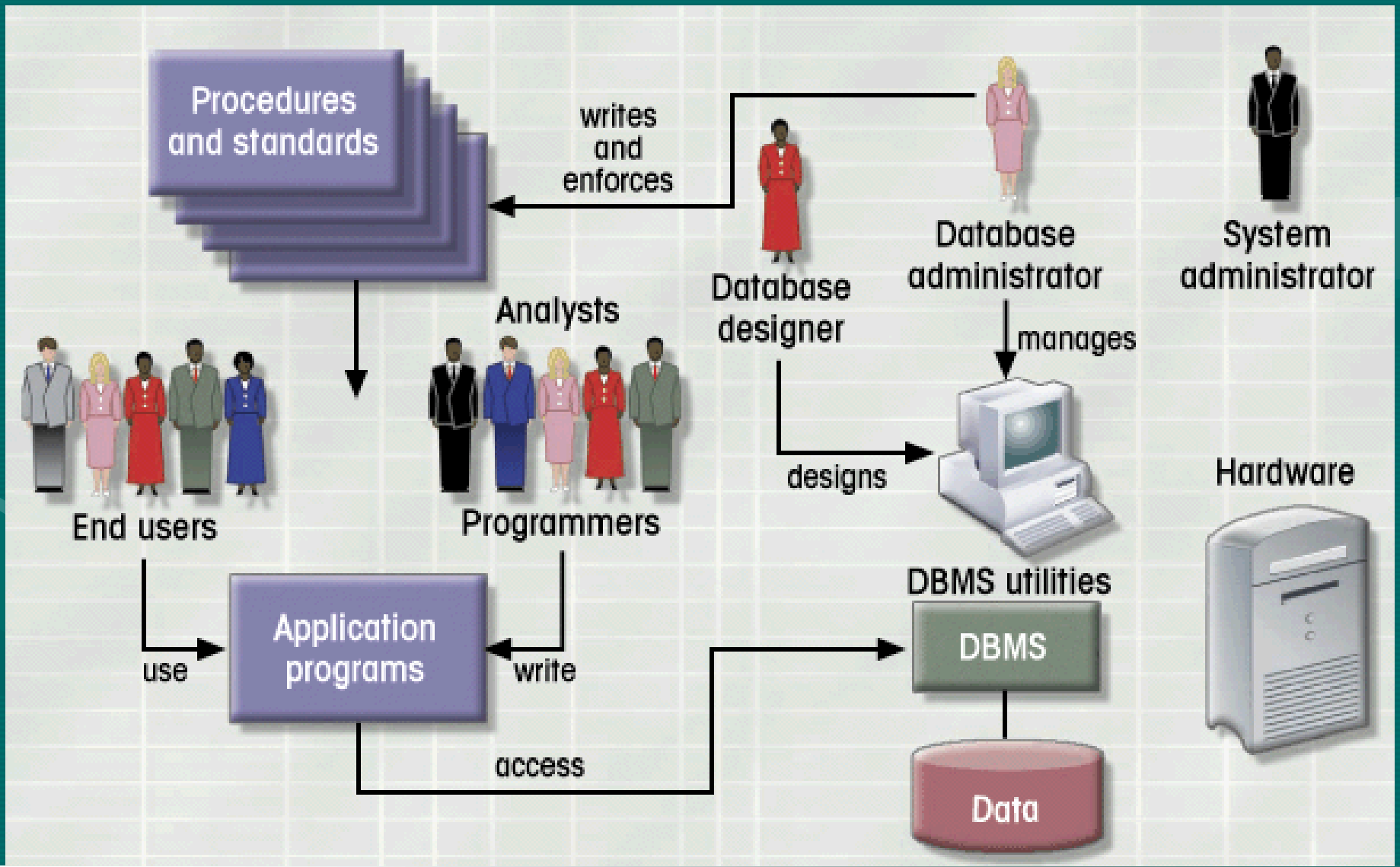
- **Mỗi hệ quản trị CSDL:**
  - Ngôn ngữ khai báo dữ liệu (Data Definition Language - DDL)
  - Ngôn ngữ thao tác CSDL (Data Manipulation Language - DML) với mục đích:
    - Truy xuất dữ liệu
    - Cập nhật dữ liệu (thêm, xóa, sửa dữ liệu)
    - Khai thác dữ liệu
- **Từ điển dữ liệu (*Data Dictionary - DD*):** Là CSDL của hệ quản trị CSDL sử dụng để lưu trữ cấu trúc CSDL, các thông tin bảo mật, an toàn dữ liệu và các cấu trúc ngoài.

Sơ đồ tổng quát của một hệ quản trị CSDL



Quá trình hoạt động của một chương trình ứng dụng thông qua các tầng của CSDL

Môi trường hệ thống CSDL



Các mô hình CSDL

- Định nghĩa: tập hợp logical constructs dùng để thể hiện cấu trúc của dữ liệu và các mối quan giữa chúng trong CSDL
  - Các mô hình quan niệm: thể hiện tính tự nhiên có logic của dữ liệu; nhấn mạnh lên thực thể được thể hiện. Trong thiết kế CSDL, mô hình này được sử dụng như bảng thiết kế.
  - Các mô hình thực thi: nhấn mạnh cách dữ liệu được thể hiện trong CSDL

Các mô hình CSDL

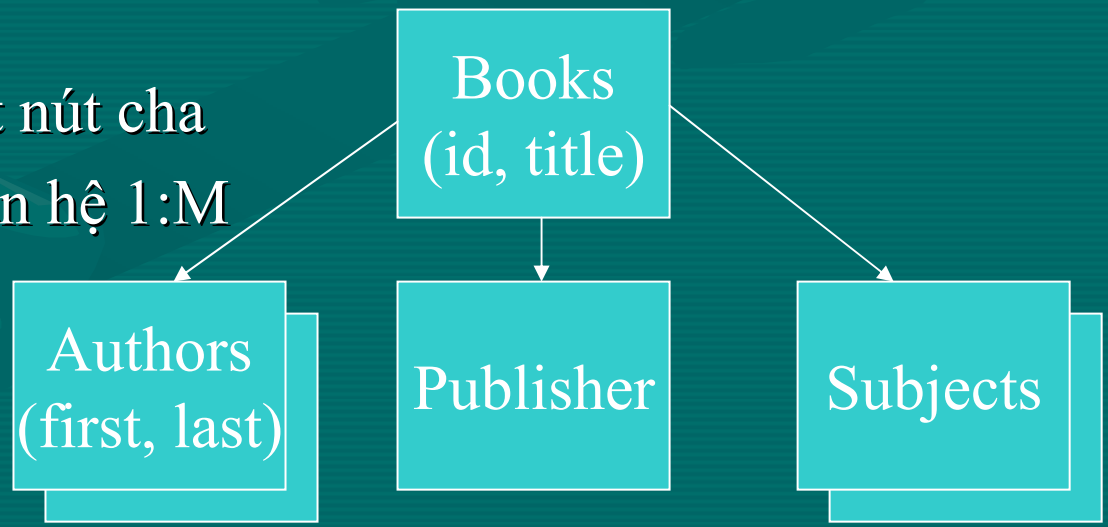
- Các mô hình quan niệm bao gồm
  - Entity-relationship database model (ERDBD)
  - Object-oriented model (OODBM)
- Các mô hình thực thi bao gồm
  - Hierachical database model (HDBM)
  - Network database model (NDBM)
  - Relational database model (RDBM)
  - Object-oriented database model (ODBM)

Các mô hình CSDL

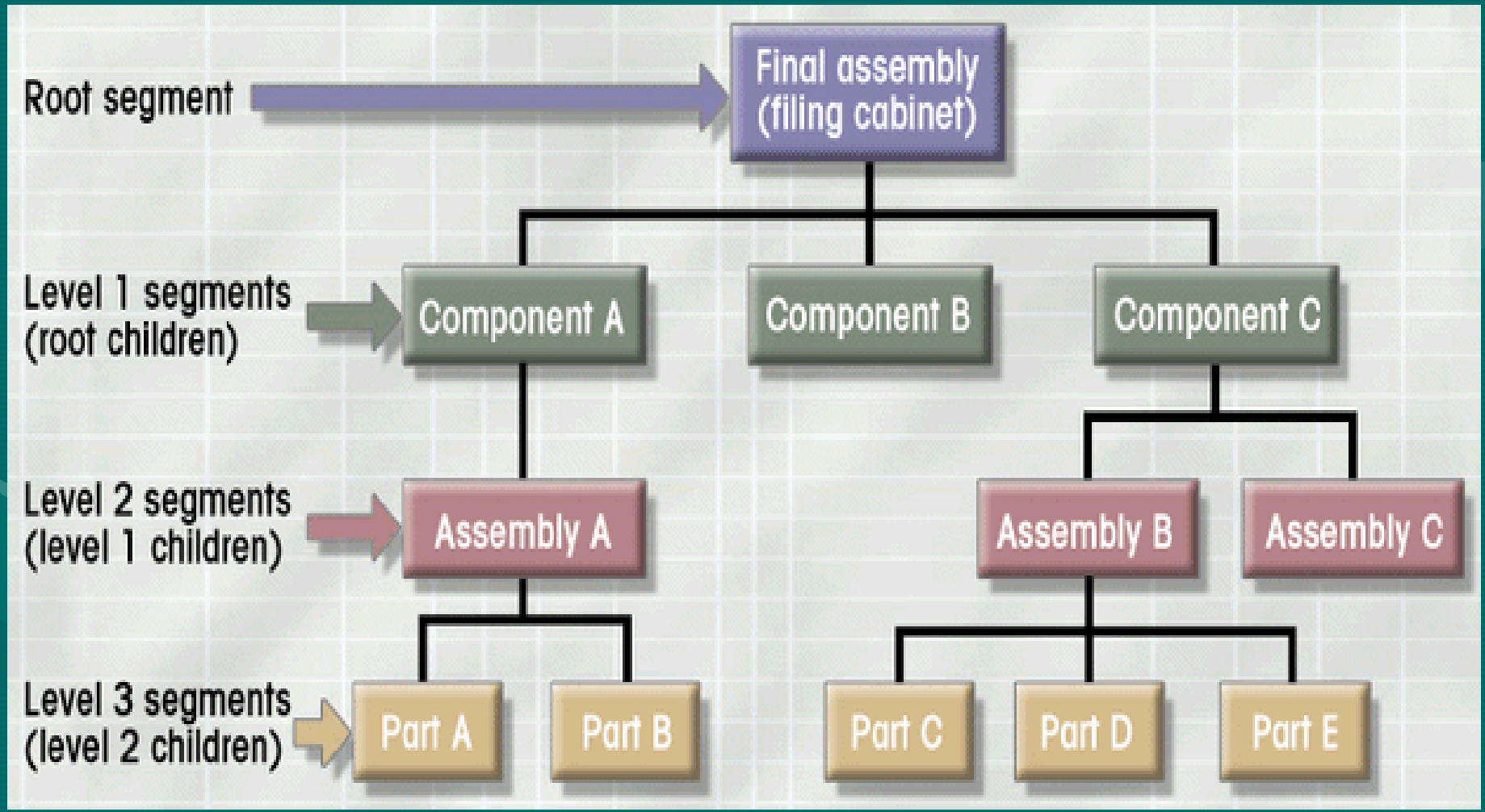
- Các loại quan hệ trong mô hình quan niệm
  - Một-một - One-to-one (1:1)
  - Một-nhiều - One-to-many (1:M)
  - Nhiều-nhiều - Many-to-many (M:N)
- Các mô hình thực thi CSDL
  - Phân cấp - Hierarchical
  - Mạng - Network
  - Quan hệ - Relational
  - Hướng đối tượng - Object-Oriented

Mô hình dữ liệu phân cấp (HDBM)

- Vào các thập kỷ 60, 70
- Tương tự như cấu trúc dữ liệu trong các ngôn ngữ lập trình.
- Được thể hiện một cách logic bằng một cây hướng từ trên xuống
  - Mỗi nút cha có thể có nhiều nút con (được nối với nhau bằng đoạn thẳng)
  - Mỗi nút con chỉ có một nút cha
  - Thể hiện được mối quan hệ 1:M



Mô hình dữ liệu phân cấp



Mô hình dữ liệu phân cấp

- Đường dẫn phân cấp (bắt đầu từ bên trái)
- Left-list hierarchical path, hay preorder traversal, hay hierarchical sequence

Final assembly->Component A->Assembly A-> -> Part A ->Part B -> Component B -> Component C –Assembly B -> Part C ->Part D

- Re-list sequence, nếu phân đoạn được truy cập thường xuyên

Mô hình dữ liệu phân cấp

- Các hệ thống ngân hàng thường sử dụng mô hình dữ liệu phân cấp này
  - Tài khỏan của khách hàng có thể là mục tiêu cho nhiều giao tác - transactions (quan hệ 1:M)
  - Mối quan hệ được cố định (gởi tiền và rút tiền)
  - Một số lượng lớn các giao tác truy cập thường xuyên

Mô hình dữ liệu phân cấp

- Ưu điểm
  - Quan niệm đơn giản: mối quan hệ giữa các lớp (layer) là đơn giản về mặc logic; qui trình thiết kế cũng đơn giản
  - Bảo mật CSDL: buộc đồng nhất trên toàn hệ thống
  - Thống nhất dữ liệu
  - Độc lập dữ liệu
  - Sự hiệu quả trong các mối quan hệ 1:M và khi quá trình sử dụng đòi hỏi cho phép một số lượng lớn các giao tác
  - Chiếm ưu thế vào những năm 1970, khi các hệ thống mainframe có CSDL lớn được sử dụng

Mô hình dữ liệu phân cấp

- Nhược điểm
  - Thực hiện phức tạp: các đặc tính về lưu trữ dữ liệu mức vật lý; thiết kế CSDL phức tạp
  - Khó quản lý và thiếu chuẩn
  - Thiếu tính độc lập cấu trúc
  - Độ phức tạp trong việc lập trình ứng dụng và sử dụng (dựa trên con trỏ)
  - Các hạn chế về thực hiện, như mô hình chỉ xử lý trường hợp kiểu quan hệ là 1:M

Mô hình dữ liệu mạng (NDBM)

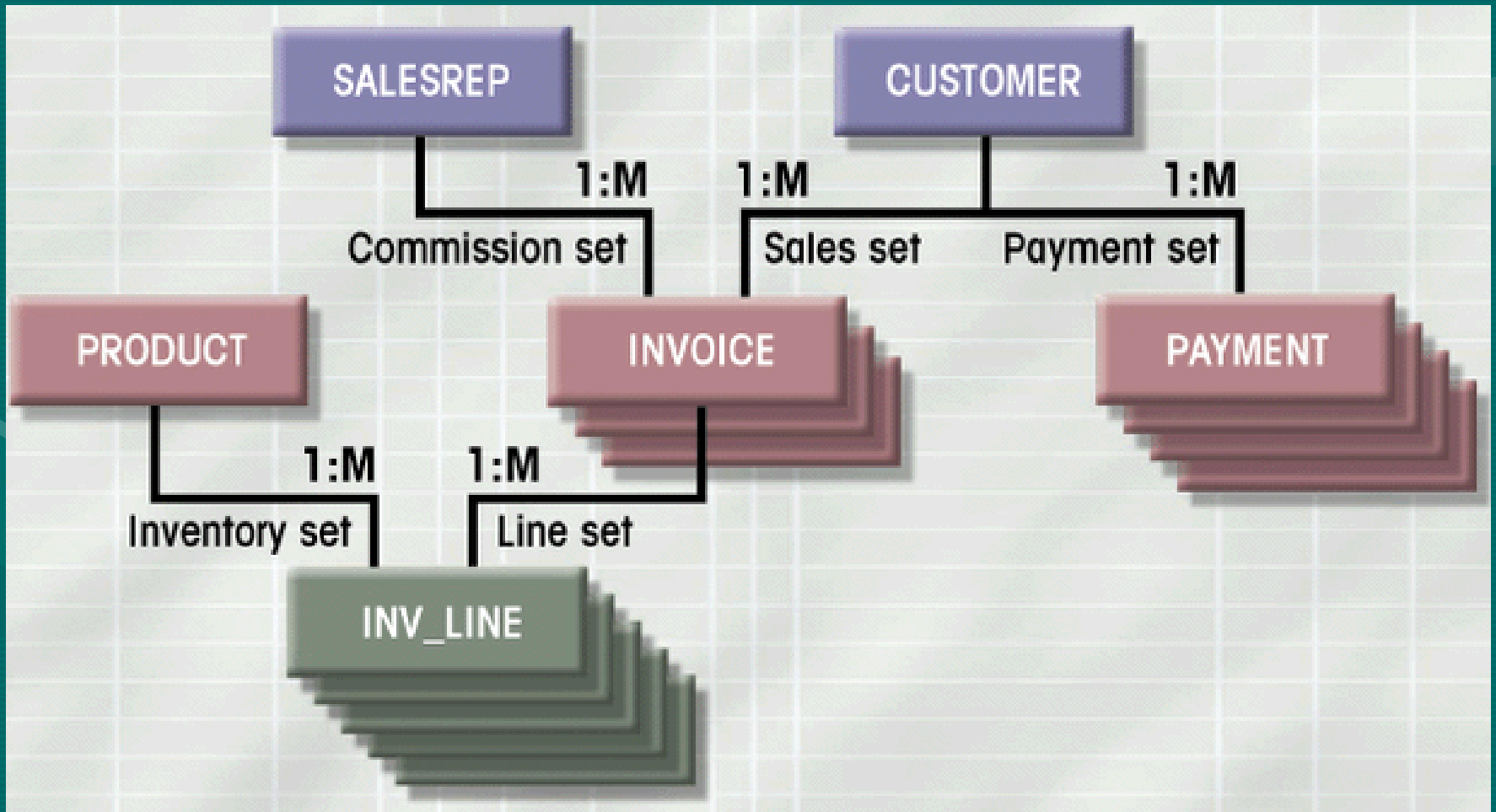
- Mỗi record có thể có nhiều parents
  - Được Database Task Group (DBTG) gọi để định nghĩa chuẩn
  - Ba thành phần quyết định CSDL
    - Network schema: tổ chức mức quan niệm của toàn CSDL
    - Subschema: một phần của CSDL như thông tin cho các chương trình ứng dụng
    - Database management language: định nghĩa các đặc tính và cấu trúc của dữ liệu
      - Schema Data definition language (DDL): định nghĩa các thành phần của schema
      - Subschema Data definition language
      - Data manipulating language: tính toán nội dung dữ liệu

Mô hình dữ liệu mạng

- Mỗi record có thể có nhiều parents
- Giới thiệu khái niệm bộ (set) để mô tả mối quan hệ
- Mỗi set có owner record và member record, tương ứng với parent và child trong HDM
  - Member có thể có nhiều owners
  - One-ownership

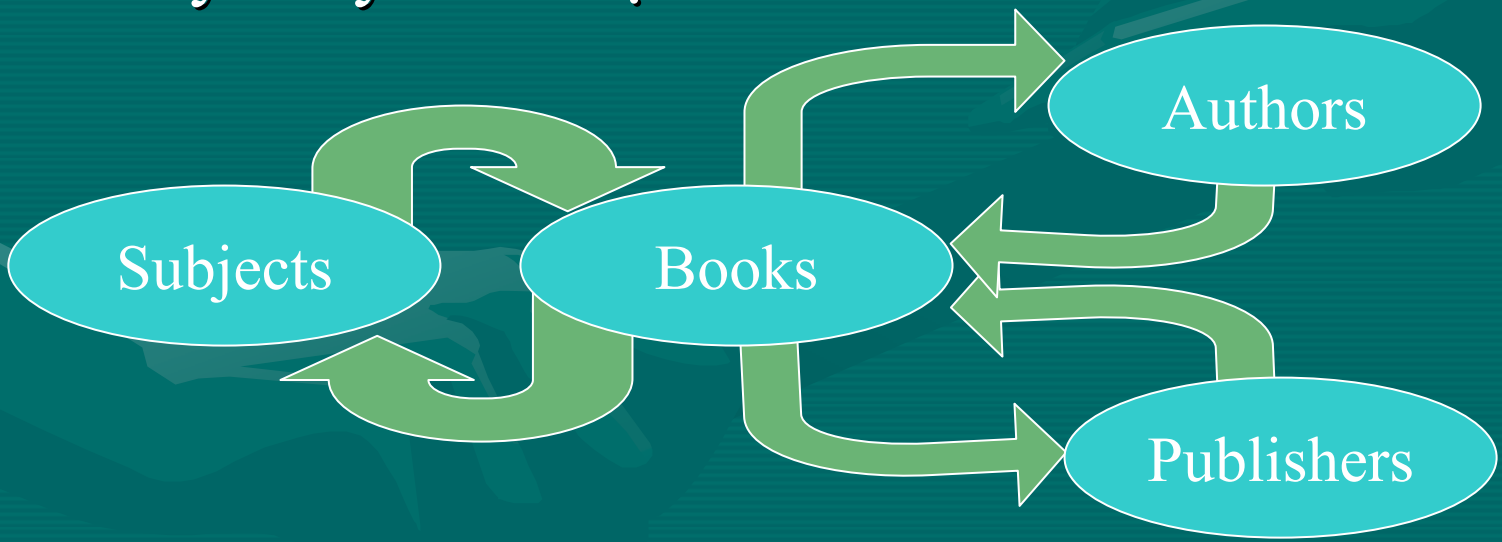
Mô hình dữ liệu mạng

– Member có thể có nhiều owners



Mô hình dữ liệu mạng

- Vào những năm 1970
  - Cho một đầu dữ liệu vào và các liên kết điều hướng navigational "links" để duyệt thông qua các dây chuyền dữ liệu.



Mô hình dữ liệu mạng

- Ưu điểm
  - Quan niệm đơn giản, gần gióng mô hình phân cấp
  - Xử lý được nhiều kiểu quan hệ hơn (nhưng tất cả đều là quan hệ 1:M)
  - Sự linh hoạt trong tuy cập dữ liệu
  - Tăng cường tính thống nhất về CSDL
  - Độc lập dữ liệu
  - Hợp chuẩn
- Nhược điểm
  - Mức độ phức tạp của hệ thống
  - Thiếu sự độc lập về cấu trúc

Mô hình dữ liệu quan hệ (RDBM)

- Vào những năm 1980
- Cho người dùng hay người thiết kế CSDL thực hiện trên một môi trường logic gần gũi với con người hơn
- Người dùng cảm nhận được như là một tập hợp các bảng dùng để lưu trữ dữ liệu, còn việc xử lý chi tiết ở mức vật lý thế nào được giao cho RDBMS.
- Bảng là một dãy các phần giao của hàng và cột
- Các bảng quan hệ với nhau bằng cách chia sẽ các đặc tính chung của thực thể
- Hỗ trợ các loại quan hệ 1:1, 1:M, M:N

Mô hình dữ liệu quan hệ

Table name: AGENT

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT code

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2002	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2002	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2001	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2002	502
10014	Orlando	Myron		615	222-1672	28-Dec-2002	501
10015	O'Brian	Amy	B	713	442-3381	22-Sep-2002	503
10016	Brown	James	G	615	297-1228	25-Mar-2002	502
10017	Williams	George		615	290-2556	17-Jul-2002	503
10018	Farriss	Anne	G	713	382-7185	03-Dec-2002	501
10019	Smith	Olette	K	615	297-3809	14-Mar-2002	503

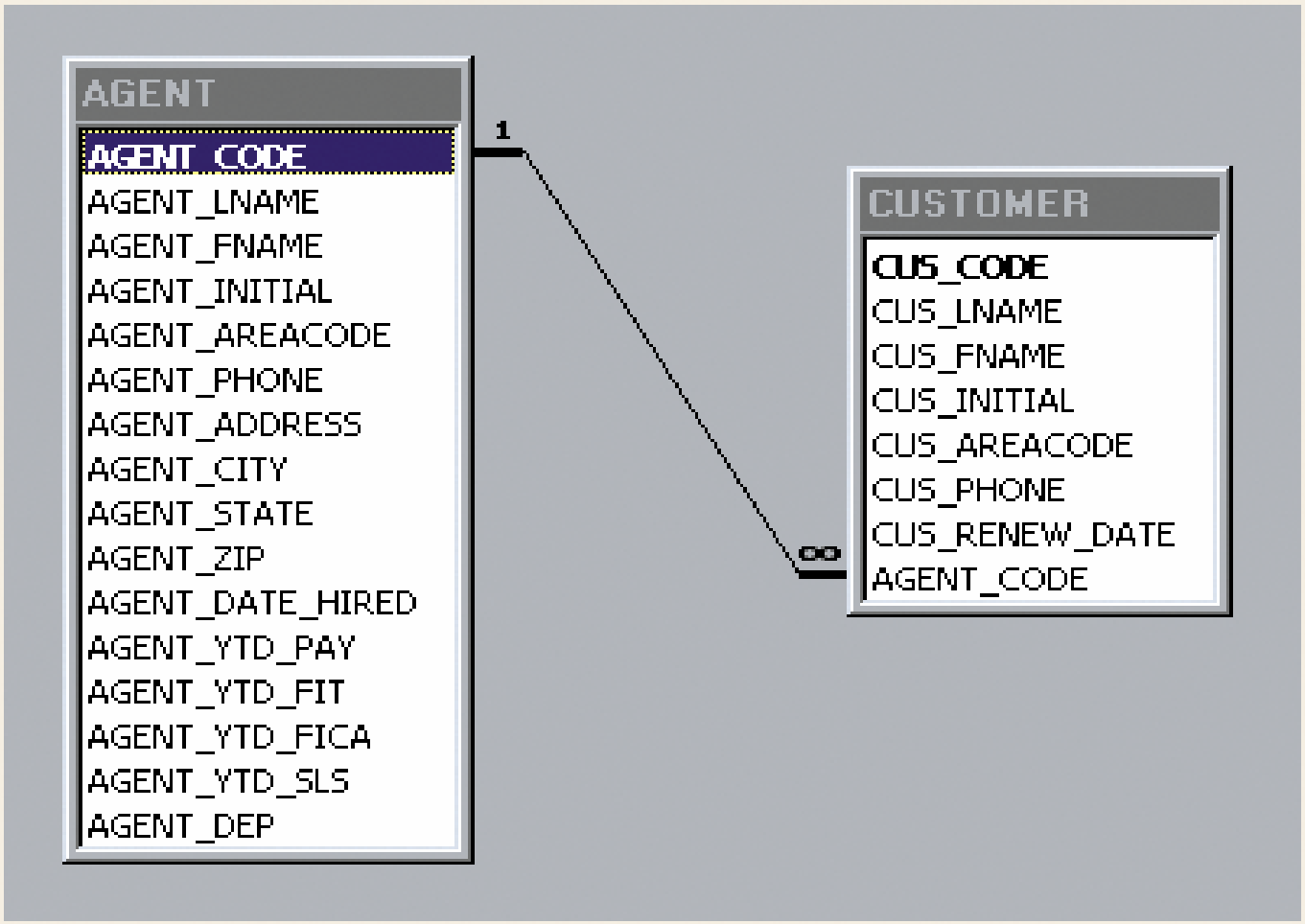


FIGURE 1.12 A RELATIONAL SCHEMA

Mô hình dữ liệu quan hệ

- Ưu điểm
  - Độc lập về cấu trúc: đường dẫn truy cập dữ liệu không liên quan gì đến việc thiết kế CSDL; thay đổi về cấu trúc sẽ không ảnh hưởng đến CSDL
  - Tăng tính đơn giản về quan niệm
  - Việc thiết kế, triển khai, quản trị và sử dụng CSDL trở nên đơn giản hơn
  - Có khả năng truy vấn phức tập với SQL (4GL được thêm vào)
  - Hệ quản trị CSDL manh mẽ

Mô hình dữ liệu quan hệ

- Nhược điểm
  - Cần sự đóng góp nhiều của phần cứng và hệ thống phần mềm
  - Dễ dàng dẫn đến thiết kế và triển khai không tốt
  - Có thể làm tăng vấn đề "islands of information"

Entity Relationship Database Model (ERDBM)

- Bổ sung cho các quan niệm trong mô hình dữ liệu quan hệ
- ERDBM giới thiệu cách thể hiện đồ họa có quan hệ
- ERDBM dựa trên một vài thành phần như Entity, tabled entity (trong RDM)
- Entity và entity set, tập hợp các thực thể giống nhau
- Mỗi entity có các attributes để mô tả thực thể, chúng tương tự như các field trong table
- Relationship và connection

Mô hình thực thể kết hợp (ERDBM)

- Được thể hiện trong entity relationship diagram (ERD): mô hình ERD của Chen và Crow's Foot ERD
- Dựa trên:
  - entities,
  - attributes, và
  - relationships

Mô hình thực thể kết hợp

connection

entity

relationship

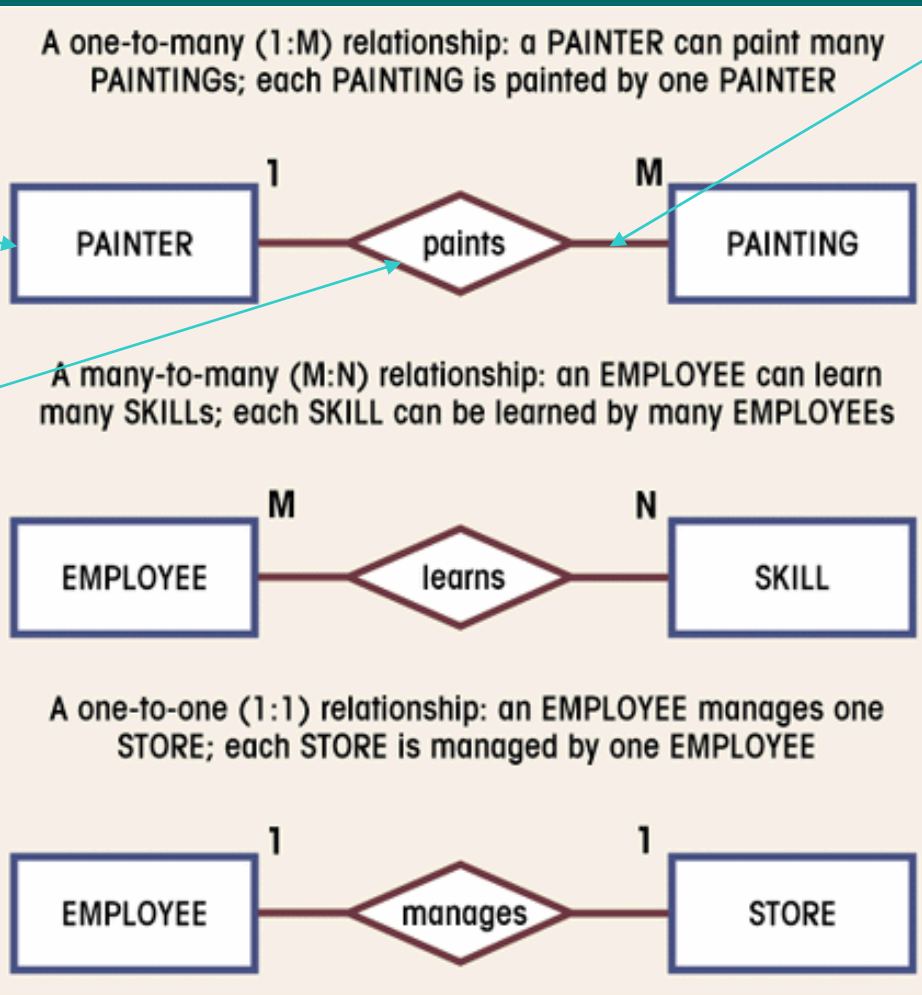
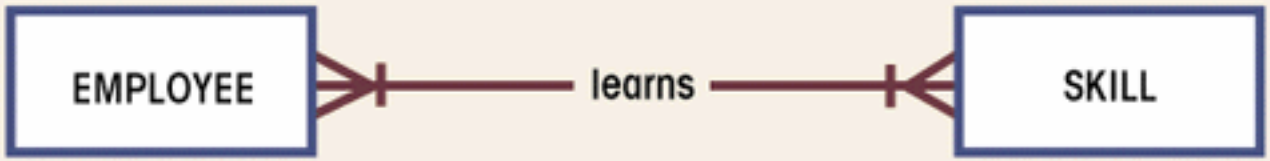


FIGURE 1.13 RELATIONSHIP DEPICTION: THE ERD

A one-to-many (1:M) relationship: A PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER

A many-to-many (M:N) relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs



A one-to-one (1:1) relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE

FIGURE 1.14 RELATIONSHIP DEPICTION: THE CROW'S FOOT ERD

Mô hình thực thể kết hợp

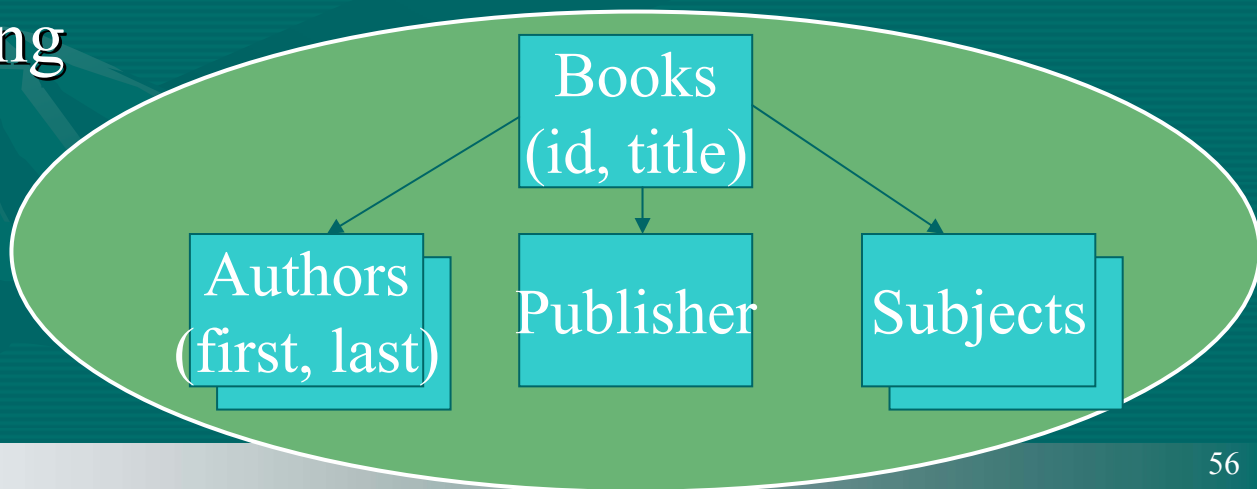
- Ưu điểm
  - Tính đơn giản đặc biệt về quan niệm
  - Thể hiện trực quan
  - Công cụ giao tiếp hiệu quả
  - Được kết hợp với mô hình CSDL quan hệ
- Nhược điểm
  - Hạn chế trong việc thể hiện các ràng buộc
  - Hạn chế trong việc thể hiện các mối quan hệ (không thể miêu tả các quan hệ internal relationship; multiple relationships)
  - Không có ngôn ngữ tính toán dữ liệu (không hoàn chỉnh)
  - Mất nội dung thông tin

Object-Oriented Database Model (OODBM)

- Semantic Data model (SDM)->Object-oriented Data Model (OODM)
- Khái niệm về Object-oriented vào những năm 1990
- Objects hay sự trừu tượng hóa của các thực thể trong thế giới thực được lưu giữ
  - Attributes mô tả đặc tính
  - Tập hợp các objects giống nhau là một class, tương tự như entity set nhưng có chứa procedure methods
    - Methods thể hiện các hành động của các class trong thế giới thực
    - Classes được tổ chức theo một class hierarchy
  - Sự kế thừa là khả năng một object có thể kế thừa inherit các attributes và methods từ các class trên nó.

Mô hình dữ liệu hướng đối tượng

- Gồm các thông tin về sự thực thi và thủ tục vận hành cho các dữ liệu phức tạp như đồ họa, video, và các metadata khác
- Hỗ trợ giao tác và thông tin
- Khả năng tái sử dụng
- Có khả năng mang đi được làm mạnh hơn việc tính tính toán hệ thống



Mô hình dữ liệu hướng đối tượng

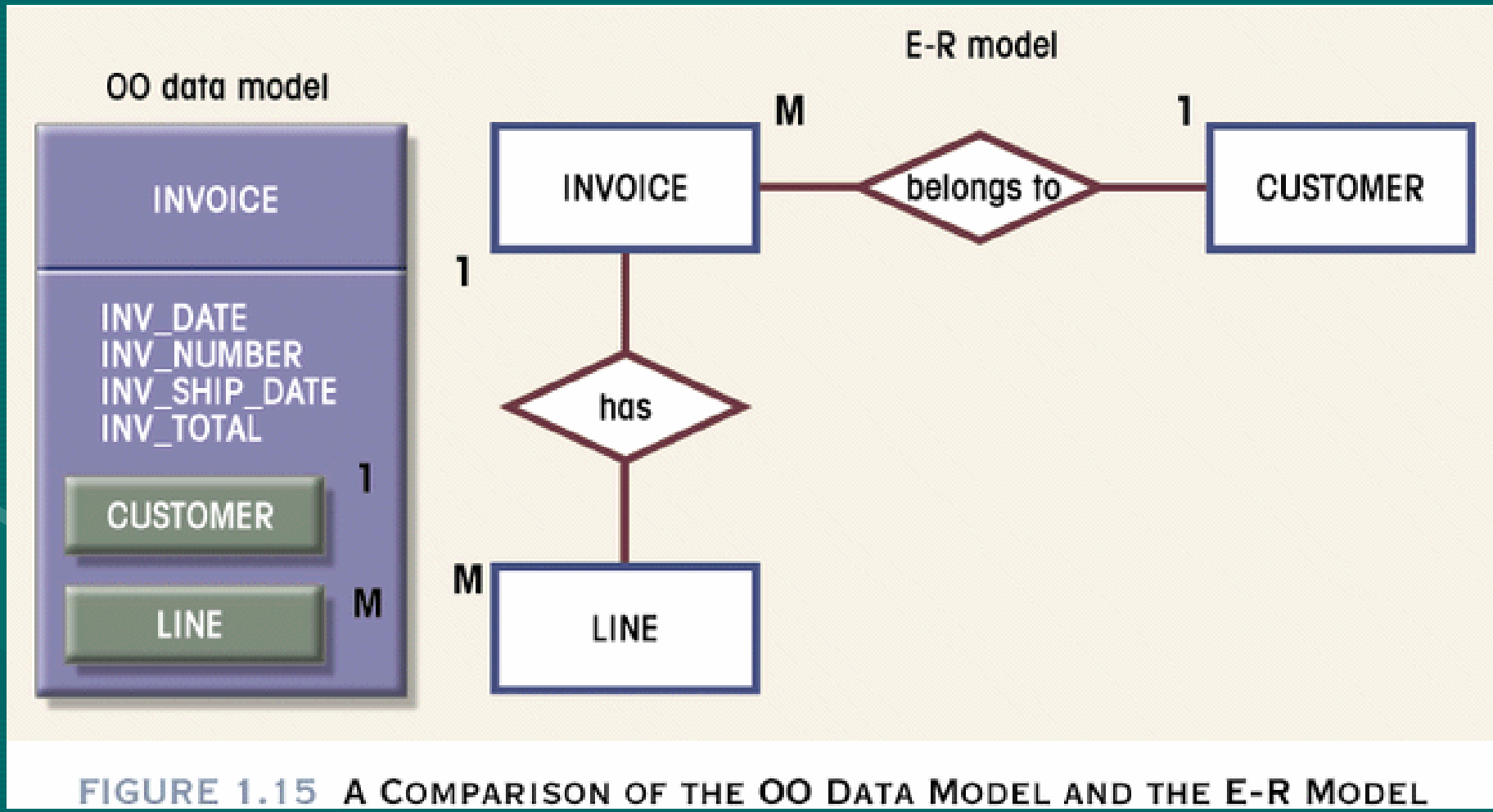


FIGURE 1.15 A COMPARISON OF THE OO DATA MODEL AND THE E-R MODEL

Mô hình dữ liệu hướng đối tượng

- Ưu điểm
  - Thêm vào nội dung ngữ nghĩa (semantic content)
  - Thể hiện trực quan gồm cả semantic content
  - Thống nhất CSDL
  - Độc lập cả cấu trúc và dữ liệu
- Khuyết điểm
  - Thiếu OODM
  - Truy cập dữ liệu điều hướng phức tạp
  - Kiến thức về OO
  - Hệ thống xử lý quá tải làm chậm các giao tác

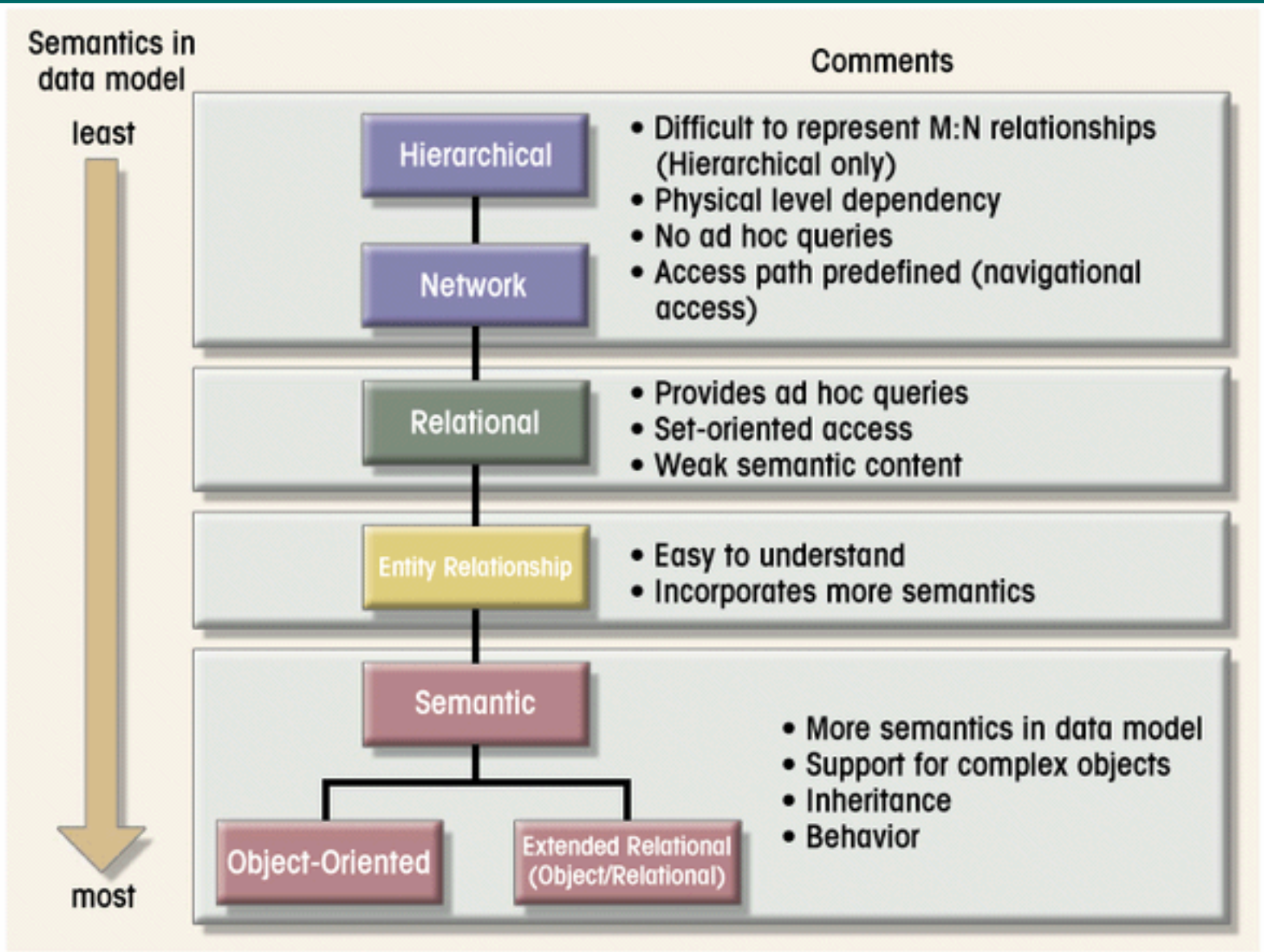


FIGURE 1.16 THE DEVELOPMENT OF DATA MODELS

Các mô hình CSDL và Internet

- Đặc tính của CSDL trong thời đại Internet
  - Truy cập internet mềm dẻo, hiệu quả, và bảo mật
  - Dễ dùng, dễ phát triển, và dễ hỗ trợ
  - Hỗ trợ các kiểu dữ liệu và quan hệ phức tạp
  - Tương tác kép kính giữa nhiều data sources và cấu trúc
  - Tính đơn giản của mô hình CSDL ngữ cảnh
  - Nhiều công cụ thiết kế CSDL, triển khai CSDL và phát triển ứng dụng CSDL
  - Giao diện DBMS mạnh mẽ làm cho công việc DBA dễ dàng hơn

Thông tin

- Nội dung:

  Nhập môn Cơ Sở Dữ Liệu

  Phần 1: Giới thiệu về CSDL

- Tổng số slide: 59
- Ngày cập nhật sau cùng: 19-02-2004
- Biên soạn: ThS. Phan Võ Minh Thắng

  Liên hệ: minhthangpv@hcmuaf.edu.vn

Nhập môn Cơ sở Dữ liệu

Phần 2 – Mô hình dữ liệu quan hệ

Nội dung

- Mô hình CSDL quan hệ là một khía cạnh về mặt logic (logical view) của dữ liệu
- Các thành phần cơ bản của mô hình CSDL quan hệ là thực thể (entities) và các thuộc tính (attributes) của nó và các mối quan hệ (relationships) giữa các thực thể với nhau
- Cách tổ chức các thực thể và các thuộc tính của nó theo dạng bảng
- Các phép toán trên CSDL quan hệ, data dictionary, và system catalog
- Cách xử lý việc dư thừa dữ liệu (data redundancy) trong mô hình CSDL quan hệ
- Tại sao việc indexing lại quan trọng

Logical View of Data

- Với CSDL quan hệ
  - Người thiết kế tập trung vào thể hiện về mặt luận lý (logical) hơn là vật lý (physical)
  - Ưu điểm của việc sử dụng bảng
    - Có cấu trúc và độc lập dữ liệu
    - Các mẫu tin (records) có liên quan được lưu trữ trong các bảng độc lập
    - Tính đơn giản về mặc logic
  - Tạo điều kiện cho các chiến lược thiết kế hiệu quả hơn

Logical View of Data

- Entities và Attributes
  - Thực thể (entity) là một người, vật, nơi chốn, hoặc sự kiện, mà dữ liệu được thu thập về nó
  - Thuộc tính (attributes) là các đặc tính của thực thể
- Tables – Bảng
  - Entity set
  - Giữ các thực thể có liên quan
  - Cũng được gọi là các quan hệ (relations)
  - Gồm các dòng (rows) và các cột (columns)

Các đặc tính của bảng

- Cấu trúc hai chiều gồm các dòng và các cột
- Dòng (bộ - tuples) thể hiện một thực thể đơn lẻ (entity)
- Cột thể hiện các thuộc tính
- Giao giữa dòng và cột thể hiện một giá trị đơn lẻ
- Các bảng phải có một thuộc tính để chỉ định duy nhất từng dòng
- Các giá trị của một cột phải có cùng một định dạng dữ liệu
- Mỗi cộ có một giới hạn các giá trị được gọi là miền giá trị của thuộc tính (attribute domain)
- Thứ tự của các dòng và cột không quan trọng đối với DBMS

Entity set

Column (attribute)

value

Row

entity

Database name: CH2_COLLEGE

Table name: STUDENT

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS
321452	Bowser	William	C	Saturday, February 12, 1972	42	So
324257	Smithson	Anne	K	Tuesday, November 15, 1977	81	Jr
324258	Brewer	Juliette		Tuesday, August 23, 1966	36	So
324269	Oblonski	Walter	H	Sunday, September 16, 1973	66	Jr
324273	Smith	John	D	Friday, December 30, 1955	102	Sr
324274	Katinga	Raphael	P	Thursday, October 21, 1976	114	Sr
324291	Robertson	Gerald	T	Wednesday, April 08, 1970	120	Sr
324299	Smith	John	B	Wednesday, November 30, 1983	15	Fr

STUDENT table, cont.

STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
2.84	No	BIOL	2134	205
3.27	Yes	CIS	2256	222
2.26	Yes	ACCT	2256	228
3.09	No	CIS	2114	222
2.11	Yes	ENGL	2231	199
3.15	No	ACCT	2267	228
3.87	No	EDU	2267	311
2.92	No	ACCT	2315	230

STU_HRS = Credit hours earned
STU_CLASS = Student classification
STU_DOB = Student date of birth

STU_GPA = Grade point average
STU_PHONE = 4-digit campus phone extension
PROF_NUM = Professor number of the professor who is the student's advisor

Kiểu dữ liệu

- Các kiểu dữ liệu:
  - Number
  - Character
  - Date
  - Logical
- Khóa chính (primary key): thuộc tính hoặc tổ hợp các thuộc tính giúp ta chỉ định duy nhất một thực thể (row, entity) cụ thể nào đó trong tập các thực thể (entity set) được lưu trên bảng (table)

Khóa

- Một hay nhiều thuộc tính xác định được các thuộc tính khác
  - Thuộc tính khóa (key attribute)
  - Khóa kết hợp (composite key)
- A xác định B: biết được giá trị của A, có thể truy cứu (xác định) giá trị của B
- A xác định B biểu diễn là A->B
- A xác định B, C, và D biểu diễn là A->B,C,D

Phụ thuộc hàm

- Phụ thuộc hàm toàn phần (full functional dependence)
  - Thuộc tính B phụ thuộc hàm toàn phần vào thuộc tính A, nếu mỗi giá trị trong cột A xác định một và chỉ một giá trị trong cột B.
- Phụ thuộc hàm tổng quát (general)
  - Thuộc tính A xác định thuộc tính B (B là phụ thuộc hàm của A), nếu tất cả các dòng trong bảng được chấp nhận là giá trị cho thuộc tính A thì cũng được chấp nhận là giá trị cho thuộc tính B

Khóa kết hợp

- Composite key – khóa kết hợp
  - Một khóa được tạo thành từ tổ hợp của nhiều hơn một thuộc tính
  - Bất cứ thuộc tính nào trong khóa kết hợp đều được gọi là thuộc tính khóa - key attribute
  - Ví dụ: A,B ->C,B,D
    - A,B là khóa kết hợp
    - A thuộc tính khóa
    - B thuộc tính khóa

Ví dụ: tìm khóa trong bảng sau ?

Table name: STUDENT

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT	STU_DOB	STU_HRS	STU_CLASS
321452	Bowser	William	C	Saturday, February 12, 1972	42	So
324257	Smithson	Anne	K	Tuesday, November 15, 1977	81	Jr
324258	Brewer	Juliette		Tuesday, August 23, 1966	36	So
324269	Oblonski	Walter	H	Sunday, September 16, 1973	66	Jr
324273	Smith	John	D	Friday, December 30, 1955	102	Sr
324274	Katinga	Raphael	P	Thursday, October 21, 1976	114	Sr
324291	Robertson	Gerald	T	Wednesday, April 08, 1970	120	Sr
324299	Smith	John	B	Wednesday, November 30, 1983	15	Fr

STUDENT table, cont.

STU_GPA	STU_TRANSFER	DEPT_CODE	STU_PHONE	PROF_NUM
2.84	No	BIOL	2134	205
3.27	Yes	CIS	2256	222
2.26	Yes	ACCT	2256	228
3.09	No	CIS	2114	222
2.11	Yes	ENGL	2231	199
3.15	No	ACCT	2267	228
3.87	No	EDU	2267	311
2.92	No	ACCT	2315	230

CSDL quan hệ đơn giản

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

Database name: CH2_SALE_CO

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	$12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	$189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	$18.63	6	231
SRE-657UG	Rat-tail file	$2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	$6.79	8	235

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

link

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425

Các loại khóa

- Superkey
  - Chỉ định duy nhất từng thực thể
  - Ví dụ: tất cả những mục sau đều có thể là superkeys
    - STU_NUM
    - STU_NUM, STU_LNAME
    - STU_NUM, STU_LNAME, STU_INIT

Dư thừa

- Candidate key
  - superkey nhỏ nhất (không bị dư thừa)
  - Ví dụ: STU_NUM
- Primary key
  - Candidate key để chỉ định duy nhất tất cả các thuộc tính trong dòng chỉ định

Toàn vẹn thực thể

- Toàn vẹn thực thể - entity integrity
  - Trong một bảng, primary key phải là duy nhất
  - Key không chứa giá trị 'null'

Table name: PRODUCT
Primary key: PROD_CODE
Foreign key: VEND_CODE

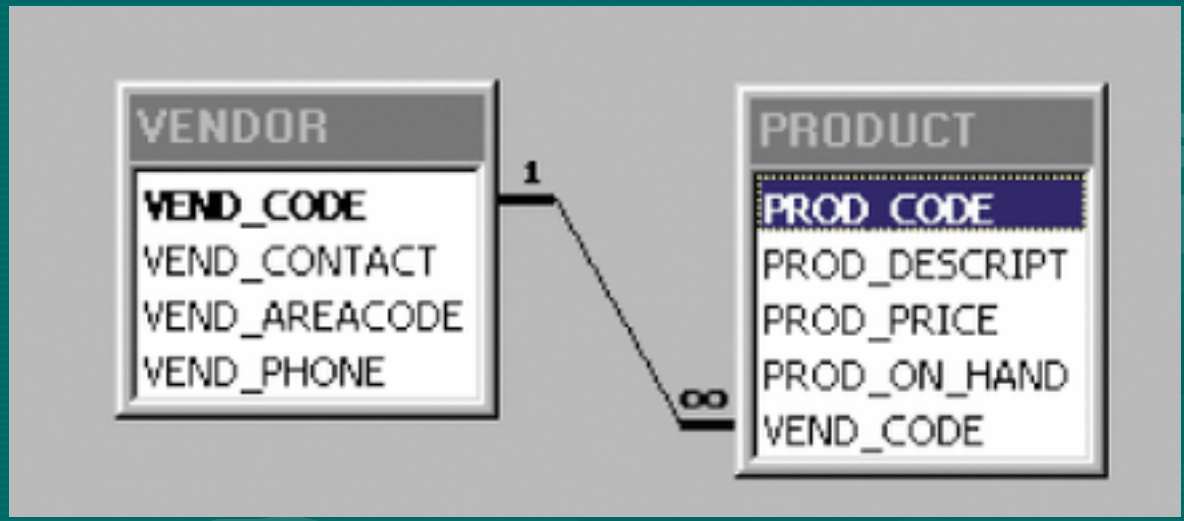
Database name: CH2_SALE_CO

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	PROD_ON_HAND	VEND_CODE
001278-AB	Claw hammer	$12.95	23	232
123-21UUY	Houselite chain saw, 16-in. bar	$189.99	4	235
QER-34256	Sledge hammer, 16-lb. head	$18.63	6	231
SRE-657UG	Rat-tail file	$2.99	15	232
ZZX/3245Q	Steel tape, 12-ft. length	$6.79	8	235

Table name: VENDOR
Primary key: VEND_CODE
Foreign key: none

link

VEND_CODE	VEND_CONTACT	VEND_AREACODE	VEND_PHONE
230	Shelly K. Smithson	608	555-1234
231	James Johnson	615	123-4536
232	Annelise Crystall	608	224-2134
233	Candice Wallace	904	342-6567
234	Arthur Jones	615	123-3324
235	Henry Ortozo	615	899-3425


VENDOR
VEND_CODE
VEND_CONTACT
VEND_AREACODE
VEND_PHONE
1
∞
PRODUCT
PROD_CODE
PROD_DESCRIPT
PROD_PRICE
PROD_ON_HAND
VEND_CODE

Khóa

- Foreign key: thuộc tính mà giá trị của nó phải khớp với các primary keys trong bảng có liên quan.
  - Giá trị phải trùng khớp với primary key trong bảng khác
- Toàn vẹn tham chiếu - reference integrity
  - Nếu foreign key có giá trị, thì giá trị đó phải tham chiếu tới một dòng đã có trong bảng khác. Ví dụ, referential integrity giữa hai bảng PRODUCT và VENDER.
- Secondary key
  - Chỉ được dùng để truy cập dữ liệu

Các loại khóa trong CSDL QH

TABLE 2.3 RELATIONAL DATABASE KEYS

KEY TYPE	DEFINITION
Superkey	An attribute (or combination of attributes) that uniquely identifies each entity in a table.
Candidate key	A minimal superkey. A superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key selected to uniquely identify all other attribute values in any given row. Cannot contain null entries.
Secondary key	An attribute (or combination of attributes) used strictly for data retrieval purposes.
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the primary key in another table or be null.

Các qui luật toàn vẹn

- Toàn vẹn thực thể - entity integrity
  - Yêu cầu (đảm bảo tất cả các thực thể phải duy nhất): tất cả các giá trị của khóa chính phải duy nhất; không phải là giá trị null
  - Mỗi thực thể có một khoá duy nhất
- Toàn vẹn tham chiếu - Referential integrity
  - Foreign key phải khớp với các giá trị primary key
  - Không được xóa dòng mà primary key có tham chiếu thuộc dạng bắt buộc bởi foreign key trong bảng khác

Table name: CUSTOMER
Primary key: CUST_CODE
Foreign key: AGENT_CODE

Database name: CH2_INSURE_CO

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	12-Mar-02	502
10011	Dunne	Leona	K	713	894-1238	23-May-02	501
10012	Smith	Kathy	W	615	894-2285	05-Jan-03	502
10013	Olowski	Paul	F	615	894-2180	20-Sep-02	
10014	Orlando	Myron		615	222-1672	04-Dec-02	501
10015	O'Brian	Amy	B	713	442-3381	29-Aug-02	503
10016	Brown	James	G	615	297-1228	01-Mar-03	502
10017	Williams	George		615	290-2556	23-Jun-02	503
10018	Farriss	Anne	G	713	382-7185	09-Nov-02	501
10019	Smith	Olette	K	615	297-3809	18-Feb-03	503

No null entries

Table name: AGENT
Primary key: AGENT_CODE
Foreign key: none

AGENT_CODE	AGENT_AREACODE	AGENT_PHONE	AGENT_LNAME	AGENT_YTD_SLS
501	713	228-1249	Alby	$1,735,453.75
502	615	882-1244	Hahn	$4,967,003.28
503	615	123-5589	Okon	$3,093,980.41

FIGURE 2.4 AN ILLUSTRATION OF INTEGRITY RULES

Ký pháp các thuật ngữ

- Thuộc tính A_1, A_2, … A_n.MGT của Ai là Dom(A_i)
- Lược đồ quan hệ R bậc n: R(A_1, A_2, A_3, ….A_n)
- Quan hệ - thể hiện của lược đồ quan hệ: $r \in R$ hay r(R)
- Bộ là một tập các giá trị tương ứng với các thuộc tính của một quan hệ: $t \in r(R)$

  $t=(a_1, a_2, a_3, \ldots a_n)$ trong đó $a_i \in Dom(A_i)$.
- Giá trị tham chiếu của bộ t tại thuộc tính A_i là $t[A_i]$, $t.A_i$
- Tập các thuộc tính của R: $R^+=\{A_1, A_2, A_3, \ldots A_n\}$
- Lược đồ cơ sở dữ liệu: $S=\{R_1, R_2, R_3, \ldots R_m\}$

Biểu diễn các phép toán quan hệ

$R(A_1, A_2, A_3, \ldots A_n)$

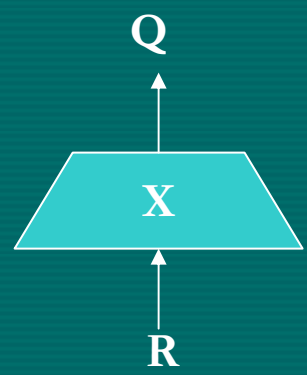
$X=\{A_i, A_k, \ldots A_l\}$

- Phép chiếu

  Phép chiếu quan hệ R trên tập con các thuộc tính X là một quan hệ Q xác định trên tập thuộc tính X, ký hiệu là R[X], được định nghĩa như sau:

$$Q = R[X] = \Pi_X(R) = \Pi_{AiAk\ldots Al}(R)$$

$$Q = \{ q / \exists\ t \in R: q = t.X \}$$



Biểu diễn các phép toán quan hệ

$R(A_1, A_2, A_3, \ldots A_n)$

Đ là điều kiện (hay phát biểu) dựa trên R+

Đánh giá Đ trên bộ giá trị $t \in R$: E(tĐ) hay Đ(t)

- Phép chọn

  Phép chọn các bản ghi của R thỏa mãn điều kiện Đ là một quan hệ Q có cùng ngôi với R, ký hiệu là R:Đ, được định nghĩa như sau:

$$Q = R:Đ = \sigma_{Đ}(R) = \sigma_{\text{mô tả đk Đ}}(R)$$

$$Q = \{ t \in R / Đ(t) = \text{đúng} \}$$

Biểu diễn các phép toán quan hệ

$R(A_1, A_2, A_3, \ldots A_n)$ $\quad t = (a1, a2, ..., an)$

$S(B_1, B_2, B_3, \ldots B_m)$ $\quad u = (b1, b2, ..., bm)$

$v = (t, u) = (a1, a2, ..., an, b1, b2, ..., bm)$

$A \in R+$ và $B \in S+$ là hai thuộc tính có thể so sánh được

θ là một phép toán so sánh

- Phép kết

Phép kết R với S trên các thuộc tính A và B với phép so sánh θ , với giả thiết là giá trị cột R[A] có thể so sánh được (qua phép so sánh θ) với mỗi giá trị của cột R[B], được định nghĩa qua:

$$R \overset{A\theta B}{\bowtie} S = \{v = (t, u) \mid t \in R, u \in S \text{ và } t.A\ \theta\ u.B\}$$

$$R \overset{A\theta B}{\bowtie} S = (R \times S) : (A\ \theta\ B)$$

Các phép toán trong CSDL quan hệ

- Lý thuyết đại số quan hệ định nghĩa cách thức để tính toán nội dung của bảng dùng tám phép toán quan hệ
- Các phép toán chính
  - SELECT – Chọn
  - PROJECT – Chiếu
  - JOIN – Kết
- Các phép toán khác đó là
  - INTERSECT – Giao
  - DIFFERENCE – Hiệu
  - DIVIDE - Chia
  - UNION – Hợp
  - PRODUCT - Nhân

Phép hợp - Union

- Các thuộc tính của các tables phải có cùng đặc tính (như các cột và các miền giá trị của các cột phải giống nhau)
- Các bảng như thế gọi là các bảng khả hợp (UNION compatible)
- Kết hợp tất cả các dòng
- Ví dụ:

Phép hợp - Union

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

UNION

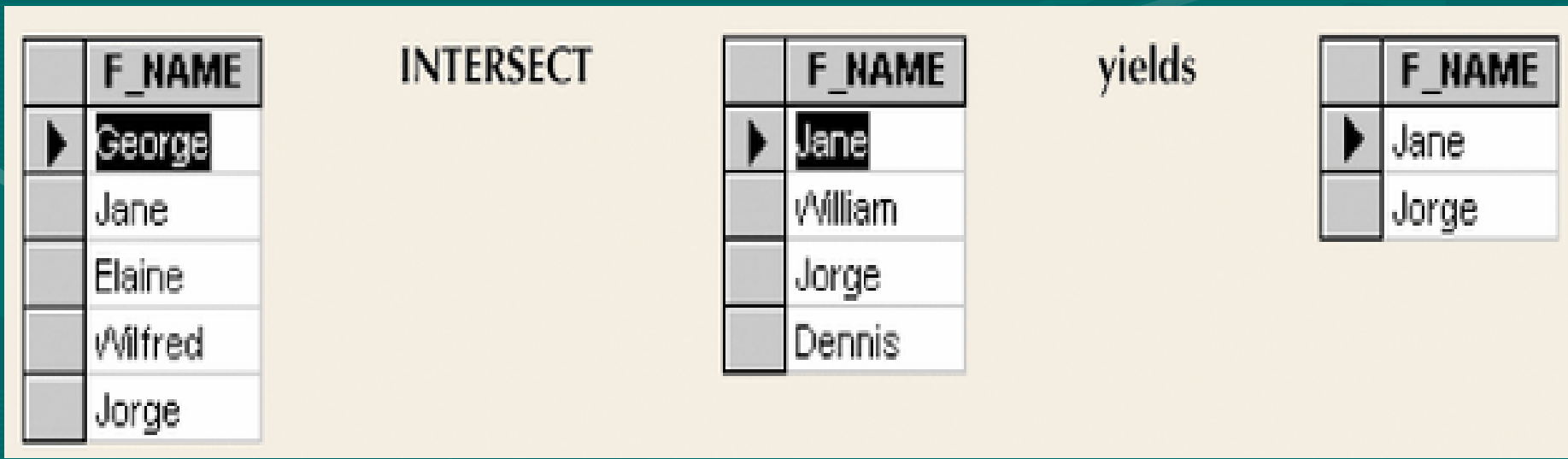
P_CODE	P_DESCRIPT	PRICE
345678	Microwave	160
345679	Dishwasher	500

yields

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99
345678	Microwave	160
345679	Dishwasher	500

Phép giao - Intersect

- Các Tables phải UNION compatible
- Kết quả là các dòng xuất hiện ở cả hai bảng
- Ví dụ:



Phép hiệu - Difference

- Các Tables phải UNION compatible
- Kết quả là các dòng trong bảng này không có trong bảng còn lại.
- Ví dụ:

Phép nhân - Product

- Kết quả là tất cả các cập có thể có (possible pairs) của các dòng trong hai bảng
- Ví dụ:

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PRODUCT

STORE	AISLE	SHELF
23	W	5
24	K	9
25	Z	6

yields

P_CODE	P_DESCRIPT	PRICE	STORE	AISLE	SHELF
123456	Flashlight	5.26	23	W	5
123456	Flashlight	5.26	24	K	9
123456	Flashlight	5.26	25	Z	6
123457	Lamp	25.15	23	W	5
123457	Lamp	25.15	24	K	9
123457	Lamp	25.15	25	Z	6
123458	Box Fan	10.99	23	W	5
123458	Box Fan	10.99	24	K	9
123458	Box Fan	10.99	25	Z	6
213345	9v battery	1.92	23	W	5
213345	9v battery	1.92	24	K	9
213345	9v battery	1.92	25	Z	6
311452	Powerdrill	34.99	23	W	5
311452	Powerdrill	34.99	24	K	9
311452	Powerdrill	34.99	25	Z	6
254467	100W bulb	1.47	23	W	5
254467	100W bulb	1.47	24	K	9
254467	100W bulb	1.47	25	Z	6

Phép chọn - Select

- Kết quả là tập các giá trị của tất cả các dòng tìm thấy trong bảng.
- Phép chọn có thể được dùng để liệt kê tất cả hoặc một phần các dòng có giá trị khớp với tiêu chuẩn chỉ được chỉ định.
- Ví dụ:

Phép chọn - Select

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT ALL will yield →

New table or list

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

SELECT only PRICE less than 2.00 will yield →

P_CODE	P_DESCRIPT	PRICE
213345	9v battery	1.92
254467	100W bulb	1.47

SELECT only P_CODE=311452 will yield →

P_CODE	P_DESCRIPT	PRICE
311452	Powerdrill	34.99

Phép chiếu - Project

- Kết quả là một tập con theo chiều dọc của bảng với các thuộc tính được chọn
- Ví dụ:

Original table

P_CODE	P_DESCRIPT	PRICE
123456	Flashlight	5.26
123457	Lamp	25.15
123458	Box Fan	10.99
213345	9v battery	1.92
254467	100W bulb	1.47
311452	Powerdrill	34.99

PROJECT PRICE yields →

New table or list

PRICE
5.26
25.15
10.99
1.92
1.47
34.99

PROJECT P_DESCRIPT and PRICE yields →

P_DESCRIPT	PRICE
Flashlight	5.26
Lamp	25.15
Box Fan	10.99
9v battery	1.92
100W bulb	1.47
Powerdrill	34.99

PROJECT P_CODE and PRICE yields →

P_CODE	PRICE
123456	5.26
123457	25.15
123458	10.99
213345	1.92
254467	1.47
311452	34.99

Phép kết - Join

- Kết hợp thông tin từ nhiều bảng
- Qui trình kết tự nhiên
  - Phép nhân - Product
  - Phép chọn - Select
  - Phép chiếu - Project
- Ví dụ:

Phép kết - Join

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

Figure 2.11

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

Qui trình kết tự nhiên

- Liên kết các bảng bằng cách chọn các dòng có cùng giá trị của (các) thuộc tính
- Xử lý theo 3 bước
  - Thực hiện phép nhân tạo một bảng
  - Thực hiện phép chọn để lấy các dòng thích hợp
  - Thực hiện phép chiếu trên từng trường đơn để loại bỏ các cột trùng nhau

Thực hiện phép nhân khi kết

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
25	6152439887
167	6153426778
231	6152431124
333	9041234445

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
1132445	Walker	32145	231	125	6152439887
1132445	Walker	32145	231	167	6153426778
1132445	Walker	32145	231	231	6152431124
1132445	Walker	32145	231	333	9041234445
1217782	Adares	32145	125	125	6152439887
1217782	Adares	32145	125	167	6153426778
1217782	Adares	32145	125	231	6152431124
1217782	Adares	32145	125	333	9041234445
1312243	Rakowski	34129	167	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1312243	Rakowski	34129	167	231	6152431124
1312243	Rakowski	34129	167	333	9041234445
1321242	Rodriguez	37134	125	125	6152439887
1321242	Rodriguez	37134	125	167	6153426778
1321242	Rodriguez	37134	125	231	6152431124
1321242	Rodriguez	37134	125	333	9041234445
1542311	Smithson	37134	421	125	6152439887
1542311	Smithson	37134	421	167	6153426778
1542311	Smithson	37134	421	231	6152431124
1542311	Smithson	37134	421	333	9041234445
1657399	Vanloo	32145	231	125	6152439887
1657399	Vanloo	32145	231	167	6153426778
1657399	Vanloo	32145	231	231	6152431124
1657399	Vanloo	32145	231	333	9041234445

Thực hiện phép chọn khi kết

CUS_CODE	CUS_LNAME	CUS_ZIP	CUSTOMER.AGENT_CODE	AGENT.AGENT_CODE	AGENT_PHONE
217782	Adares	32145	125	125	6152439887
1321242	Rodriguez	37134	125	125	6152439887
1312243	Rakowski	34129	167	167	6153426778
1132445	Walker	32145	231	231	6152431124
1657399	Vanloo	32145	231	231	6152431124

Thực hiện phép chiếu khi kết

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124

Các phép kết khác

- Kết bằng - EquiJOIN
  - Liên kết các bảng dựa trên điều kiện bằng (equality condition) trong đó sẽ thực hiện việc so sánh các cột được chỉ định của các bảng
  - Không loại bỏ các cột trùng nhau
  - Điều kiện kết phải được định nghĩa tường minh (explicitly defined)
  - EquiJOIN thường so sánh các cột được chỉ định trong từng bảng sử dụng các phép toán (so sánh) khác hơn là phép toán bằng

Các phép kết khác

- Theta JOIN
  - Bất kỳ phép so sánh nào khác được sử dụng một cách tổng quát được gọi là phép kết theta JOIN
- Outer JOIN
  - Các cập trùng nhau được giữ lại
  - Các giá trị không trùng khớp trong bảng "theo hướng của outer JOIN" được đặt là null
  - Có outer JOIN bên phải và outer JOIN bên trái
  - Ví dụ:

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
25	6152439887
167	6153426778
231	6152431124
333	9041234445

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
1542311	Smithson	37134	421	

FIGURE 2.15 LEFT OUTER JOIN

Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE
1132445	Walker	32145	231
1217782	Adares	32145	125
1312243	Rakowski	34129	167
1321242	Rodriguez	37134	125
1542311	Smithson	37134	421
1657399	Vanloo	32145	231

Table name: AGENT

AGENT_CODE	AGENT_PHONE
125	6152439887
167	6153426778
231	6152431124
333	9041234445

CUS_CODE	CUS_LNAME	CUS_ZIP	AGENT_CODE	AGENT_PHONE
1217782	Adares	32145	125	6152439887
1321242	Rodriguez	37134	125	6152439887
1312243	Rakowski	34129	167	6153426778
1132445	Walker	32145	231	6152431124
1657399	Vanloo	32145	231	6152431124
			333	9041234445

Phép chia - Divide

- Sử dụng trong bảng một cột và bảng hai cột
- Tìm giá trị kết hợp với các giá trị trong bảng chia sao cho tất cả các kết hợp đó đều thuộc bảng bị chia
- Ví dụ:

	CODE	LOC
▶	A	5
	A	9
	A	4
	B	5
	B	3
	C	6
	D	7
	D	8
	E	8

DIVIDE

	CODE
▶	A
	B

yields

	LOC
▶	5

Data Dictionary

- Gồm thông tin chi tiết về tất cả các bảng có trong CSDL
- Thông tin về thông tin - Metadata
- Tên thuộc tính và các đặc tính của thuộc tính
- Tất cả thành viên của nhóm thiết kế và triển khai CSDL dùng chung bảng, các thuộc tính và các đặc tính đó
- DBMS tổ chức lưu trữ data dictionary bên trong nó và additional information bao gồm các loại quan hệ, và các yêu cầu kiểm tra và đảm bảo toàn vẹ trên entity và toàn vẹn về tham chiếu, các hình thức index và các thành phần. Additional information được phát sinh trong suốt giai đoạn thiết kế và triển khai CSDL.
- CSDL của người thiết kế CSDL

TABLE 2.6 A SAMPLE DATA DICTIONARY

TABLE NAME	ATTRIBUTE NAME	CONTENTS	TYPE	FORMAT	RANGE	REQUIRED	PK OR FK	FK REFERENCED TABLE
CUSTOMER								
	CUS_CODE	Customer account code	CHAR(5)	99999	10000–99999	Y	PK	
	CUS_LNAME	Customer last name	VCHAR(20)	Xxxxxxxxxx		Y		
	CUS_FNAME	Customer first name	VCHAR(20)	Xxxxxxxxxxx		Y		
	CUS_INITIAL	Customer middle initial	CHAR(1)	X				
	CUS_RENEW_DATE	Customer insurance renewal date	DATE	DD_MM_YYYY				
	AGENT_CODE	Insurance agent code	CHAR(3)	999	100–999		FK	AGENT
AGENT								
	AGENT_CODE	Insurance agent code	CHAR(3)	999		Y	PK	
	AGENT_AREACODE	Agent's area code	CHAR(3)	999		Y		
	AGENT_PHONE	Agent's telephone number	CHAR(8)	999 – 9999		Y		
	AGENT_LNAME	Agent's last name	VARCHAR(20)			Y		
	AGENT_YTD_SALES	Agent's year-to-date sales	NUMBER(9,2)	9,999,999.99	0.00–9,999,999.99	Y		

FK = Foreign key
PK = Primary key
CHAR = Fixed character length data, 1 to 255 characters.
VARCHAR = Variable character length data, 1 to 2,000 characters. May also be labeled VARCHAR2.
NUMBER = Numeric data. NUMBER(9,2) is used to specify numbers with two decimal places and up to nine digits, including the decimal places. Some RDBMSs permit the use of a MONEY or a CURRENCY data type.

System Catalog

- Data dictionary chi tiết; các phần mềm CSDL quan hệ hiện tại chỉ hỗ trợ một system catalog
- Có thể suy diễn ra data dictionary
- CSDL do hệ thống tạo, các bảng có thể được truy vấn; tài liệu của CSDL cho phép HQT CSDL quan hệ kiểm tra và loại bỏ các trường hợp homonyms (cùng tên cho các thuộc tính khác nhau) và synonyms (khác tên cho cùng thuộc tính)
- Lưu giữ các đặc tính và nội dung của CSDL
- Các table cũng có thể được truy vấn giống như bất cứ table nào khác
- Tự động phát sinh tài liệu cho CSDL

Các loại quan hệ trong CSDL quan hệ

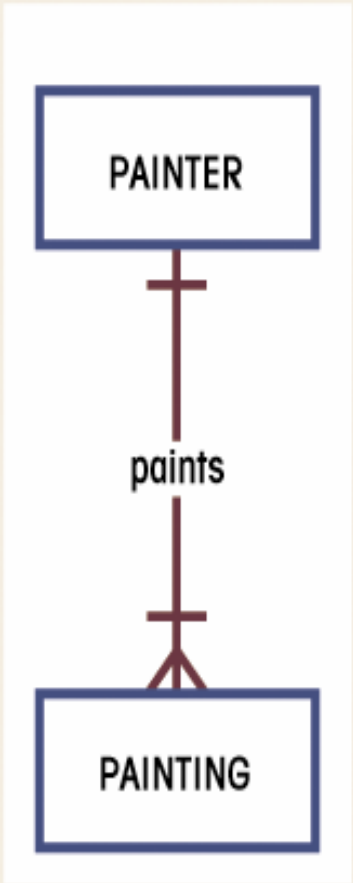
- Phân loại mối quan hệ
  - 1:1
    - một thực thể chỉ có thể quan hệ với một và chỉ một thực thể khác
    - VD: NHÂN VIÊN : LỚP HỌC
  - 1:M (quan trọng)
    - Một thực thể có thể quan hệ với nhiều thực thể khác
    - VD: HÀNG HÓA : NHÀ CUNG CẤP
  - M:N (troublesome)
    - Nhiều thực thể có thể quan hệ với nhiều thực thể khác
    - VD: SINH VIÊN : LỚP HỌC

ERD Symbols

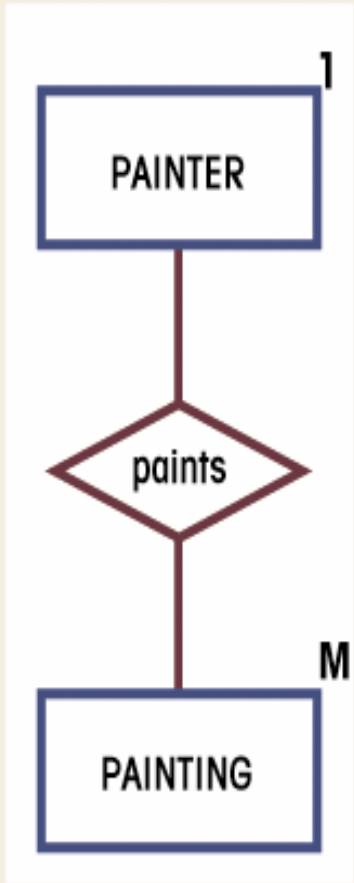
- E-R Model: ERD Maps E-R model: Chen, Crow's Feet
- Hình chữ nhật thể hiện thực thể
- Hình thoi thể hiện mối quan hệ giữa các thực thể
- Bên "1" của quan hệ
  - Số 1 trong Chen Model
  - Lằn kẽ cắt đoạn thẳng trong Crow's Feet Model
- Bên "nhiều" của quan hệ
  - Ký tự "M" và "N" trong Chen Model
  - "Crow's foot" trong Crow's Feet Model

Ví dụ: mối quan hệ 1:M

Crow's Foot model

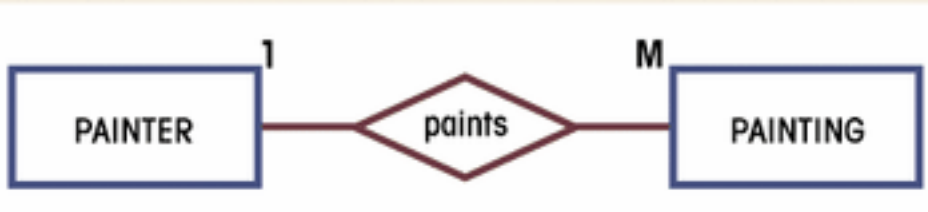


Chen model



Crow's Foot model

Chen model



Ví dụ: mối quan hệ 1:M

Table name: PAINTER
Primary key: PAINTER_NUM
Foreign key: none

Database name: CH2_MUSEUM

PAINTER_NUM	PAINTER_LNAME	PAINTER_FNAME	PAINTER_INITIAL
123	Ross	Georgette	P
126	Itero	Julio	G

link

Table name: PAINTING
Primary key: PAINTER_NUM
Foreign key: PAINTER_NUM

PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM
1338	Dawn Thunder	123
1339	Vanilla Roses To Nowhere	123
1340	Tired Flounders	126
1341	Hasty Exit	123
1342	Plastic Paradise	126

Ví dụ: mối quan hệ 1:M

- Mối quan hệ 1:M dễ dàng được thực hiện trong mô hình dữ liệu quan hệ.
- Khóa chính của bên một được thêm vào thành khóa ngoại của bảng ở bên nhiều

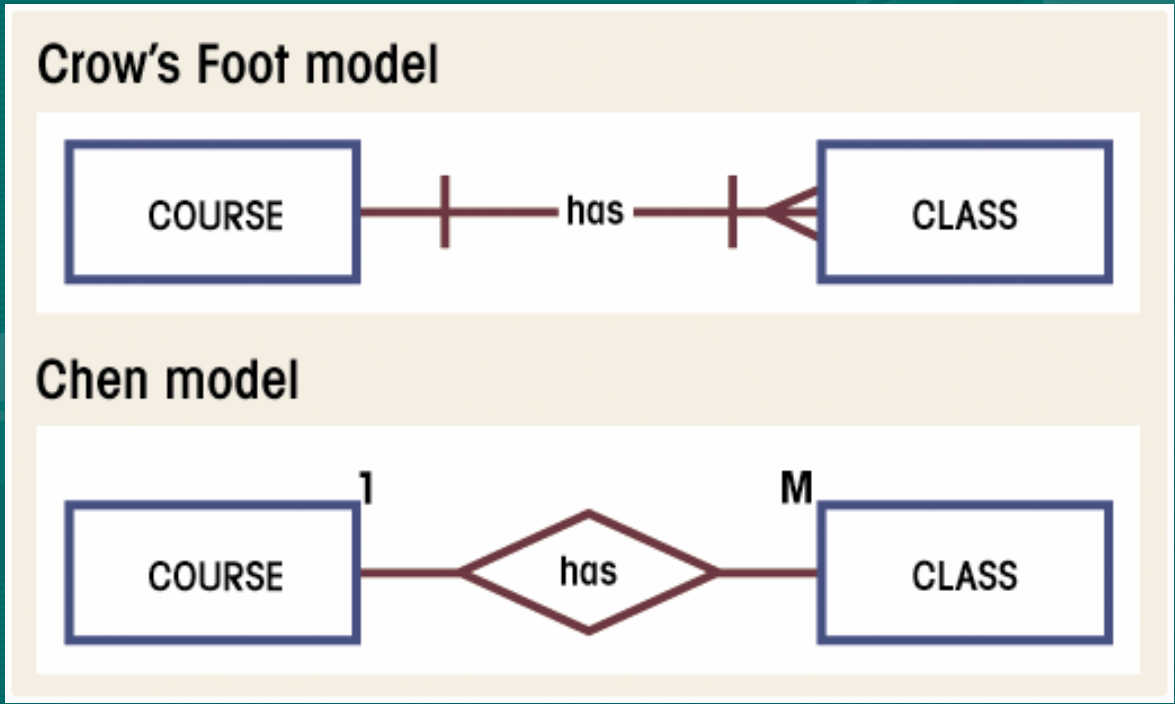


Table name: COURSE
Primary key: CRS_CODE
Foreign key: none

Database name: CH2_COLLEGE

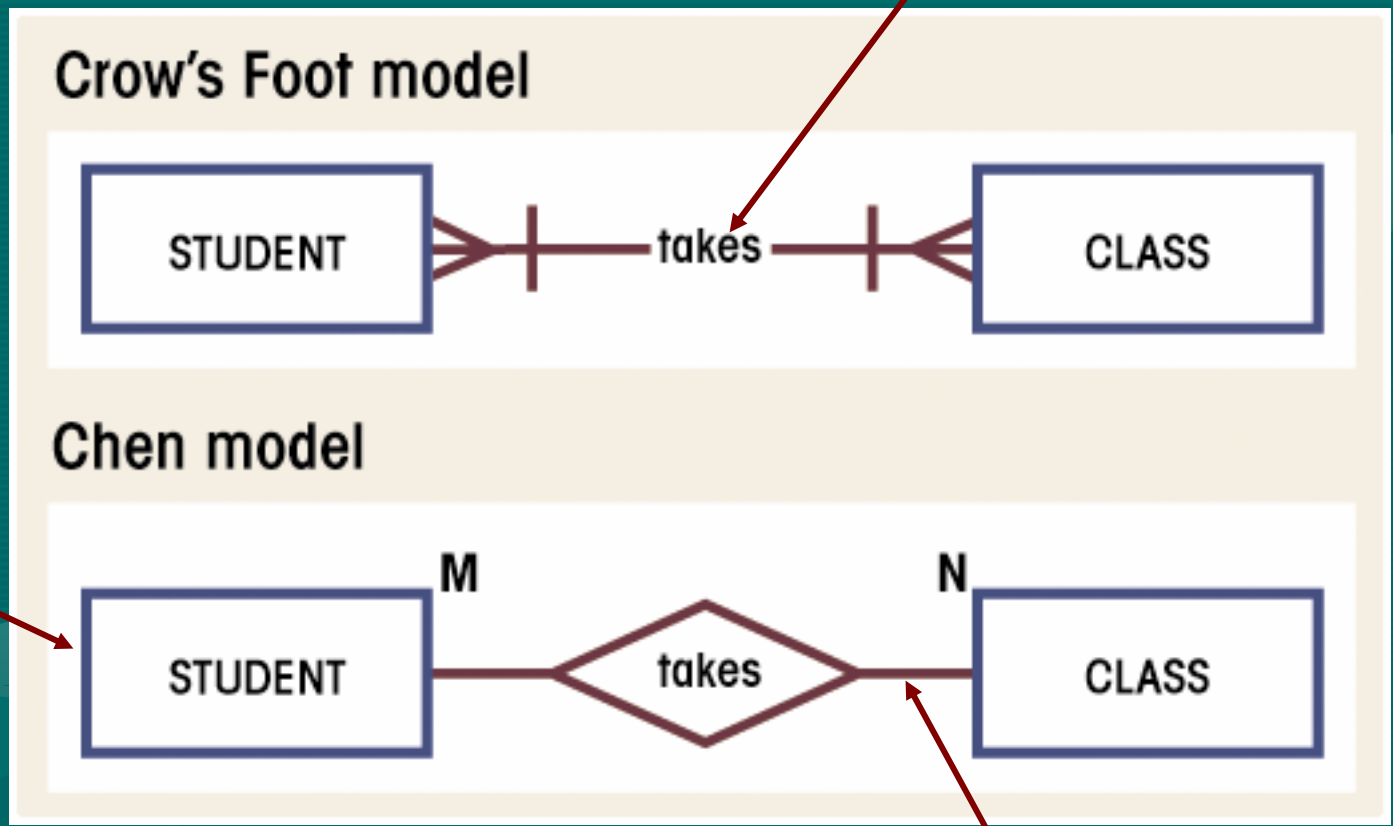
CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
10016	ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
10017	CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10019	CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
10020	CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10022	QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
10023	QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
10024	QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162

Ví dụ: mối quan hệ N:M

Tên mối quan hệ:
Chủ động hoặc bị động



Hình chữ nhật

Thực thể

Viết in hoa

Relationship line
Connector

Ví dụ: mối quan hệ N:M

Database name: CH2_TEXT

Table name: STUDENT_FIG2_24

STU_NUM	STU_LNAME	CLASS_CODE
321452	Bowser	10014
321452	Bowser	10018
321452	Bowser	10021
324257	Smithson	10014
324257	Smithson	10018
324257	Smithson	10021

Table name: CLASS_FIG2_24

CLASS_CODE	STU_NUM	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	321452	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10014	324257	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	321452	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10018	324257	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	321452	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10021	324257	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

Ví dụ: mối quan hệ N:M

- Table có nhiều vị trí dư thừa
- Các phép toán quan hệ trở nên rất phức tạp và có thể dẫn tới system efficiency errors và output errors.

Chuyển quan hệ M:N thành hai quan hệ 1:N

Table name: STUDENT_FIG2_25
Primary key: STU_NUM
Foreign key: none

STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

Table name: ENROLL_FIG2_25
Primary key: CLASS_CODE+STU_NUM
Foreign key: CLASS_CODE, STU_NUM

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS_FIG2_25
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

Chuyển quan hệ M:N thành hai quan hệ 1:N

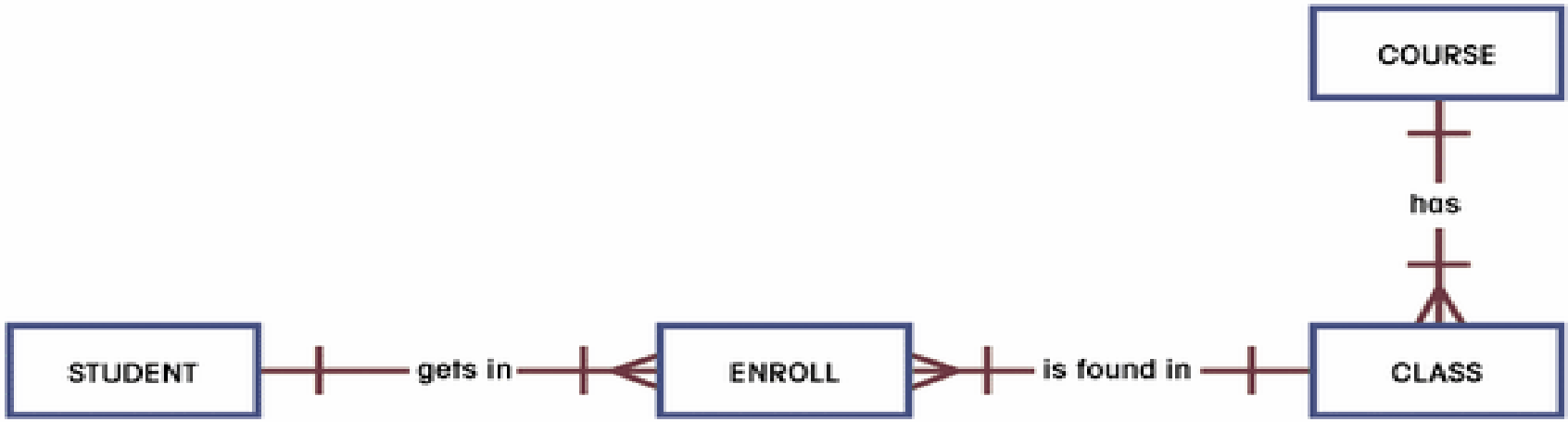
Crow's Foot model

Revised E-R diagram: two sets of 1:M relationships

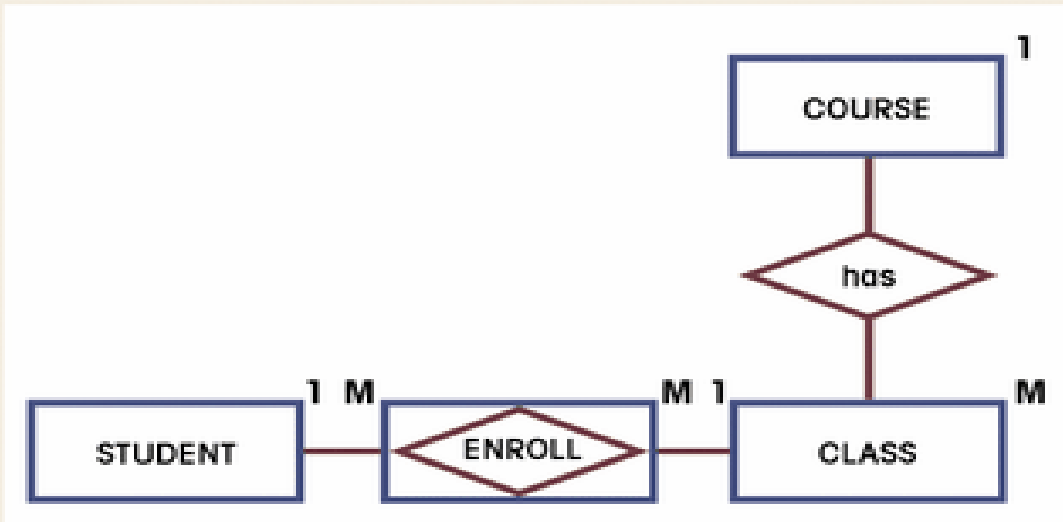
Chen model

Chuyển quan hệ M:N thành hai quan hệ 1:N

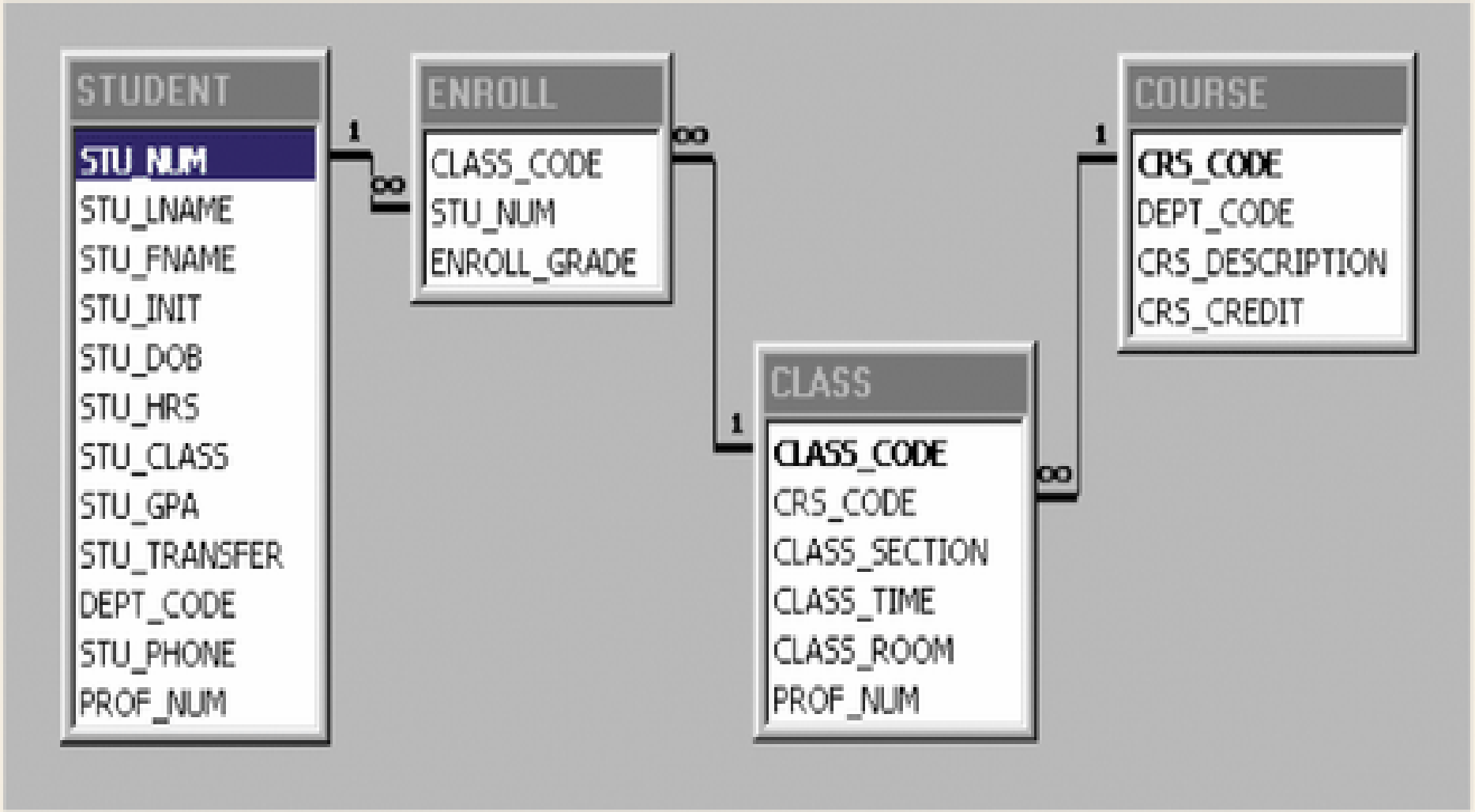
Crow's Foot model



Chen model



Chuyển quan hệ M:N thành hai quan hệ 1:N

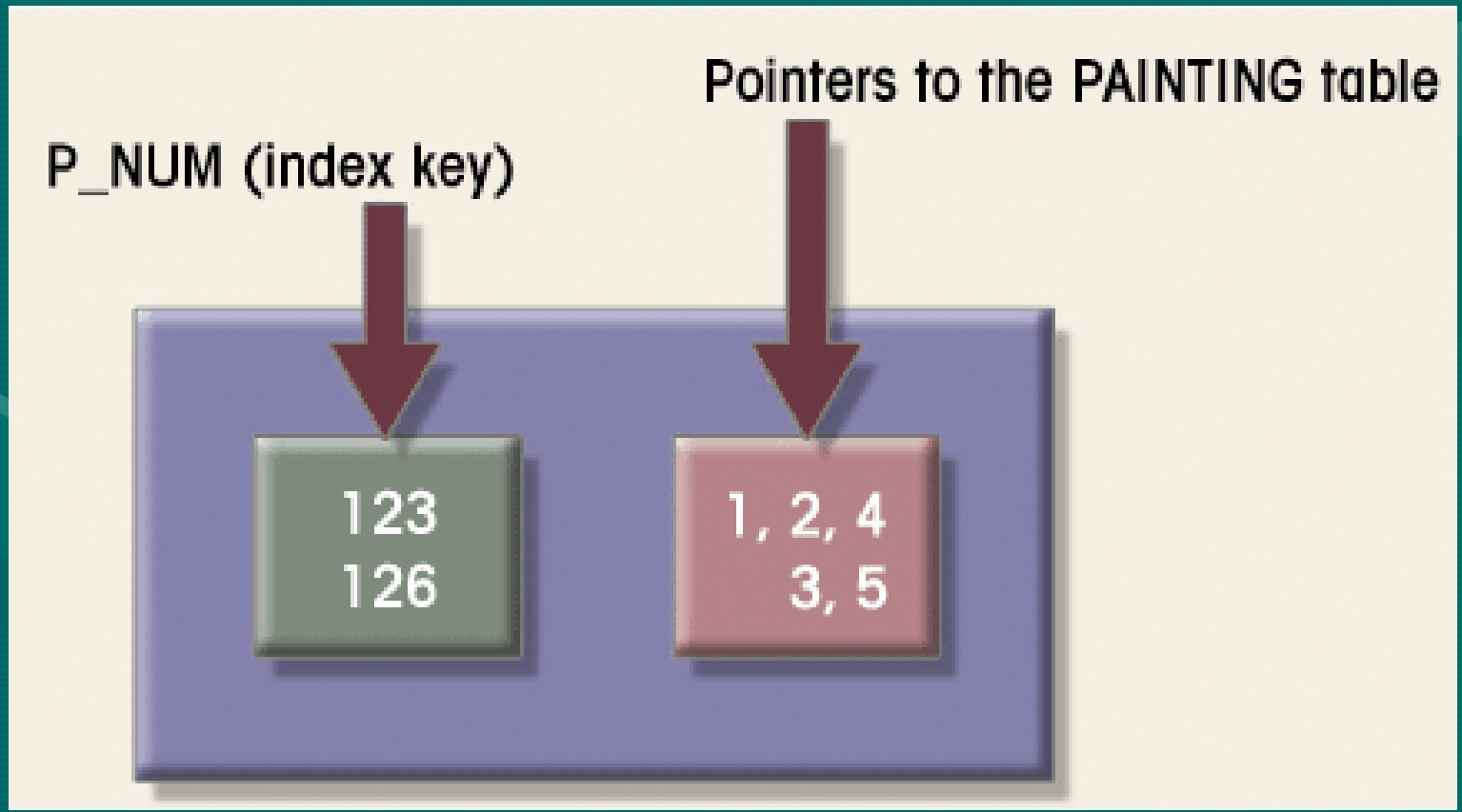


Xem xét sự dư thừa dữ liệu

- Khóa ngoại có thể làm giảm sự dư thừa
- Một vài trường hợp dư thừa là có thể cần thiết
  - Được gọi là các dư thừa được kiểm soát
  - Tốc độ
  - Thông tin yêu cầu

Chỉ mục

- Chỉ tới vị trí
- Làm cho việc truy cập dữ liệu nhanh chóng hơn



Thông tin

- Nội dung:

  Nhập môn Cơ Sở Dữ Liệu

  Phần 2: Mô hình dữ liệu quan hệ

- Tổng số slide: 60
- Ngày cập nhật sau cùng: 19-02-2004
- Biên soạn: ThS. Phan Võ Minh Thắng

  Liên hệ: minhthangpv@hcmuaf.edu.vn

PGS.TS. Vũ Đức Thi

Giáo trình cơ sở dữ liệu

Bài Giảng

Hà Nội

Lời nói đầu

Cơ sở dữ liệu là một lĩnh vực phát triển mạnh của công nghệ thông tin. Cùng với sự phát triển công nghệ thông tin ở nước ta, việc sử dụng các kiến thức về cơ sở dữ liệu vào thực tiễn ngày càng trở lên cần thiết.

Trong bài giảng này chúng tôi cung cấp cho sinh viên những kiến thức cơ bản nhất về cơ sở dữ liệu. Mục tiêu chính là với số kiến thức cơ bản này sinh viên có thể ứng dụng các kiến thức về cơ sở dữ liệu vào thực tiễn và tiếp tục nghiên cứu học tập được các môn tin học khác.

Giáo trình gồm 4 chương chính (Ngoài chương mở đầu và tài liệu tham khảo).

Chương 2 cung cấp cho sinh viên những kiến thức cơ bản về cơ sở dữ liệu, mà cụ thể là về cơ sở dữ liệu quan hệ. Trong chương này, chúng tôi trình bày những khái niệm cơ bản nhất của cơ sở dữ liệu quan hệ, cũng như những thuật toán thiết kế chúng.

Chương 3 trình bày các kiến thức liên quan đến các dạng chuẩn.

Chương 4 giới thiệu các phép toán xử lí các bảng (quan hệ).

Chương 5 và chương 6 là các chương trình bày các ứng dụng của cơ sở dữ liệu vào thực tiễn.

Trong chương 5 chúng tôi nêu một số các ứng dụng của cơ sở dữ liệu trong các hệ quản trị cơ sở dữ liệu hiện có. Trong đó có những vấn đề liên quan đến các thực thể, các khoá, các dạng chuẩn trong các hệ quản trị cơ sở dữ liệu.

Chương 6 trình bày một số các công đoạn xây dựng các dự án thiết kế tổng thể các hệ thống thông tin.

Trong chương 7, chúng tôi trình bày một số các kiến thúc cơ bản về thuật toán và độ phức tạp thuật toán. Những kiến thức này giúp cho bạn đọc tiếp thu các kiến thức của các chương trên.

Giáo trình này phục vụ cho các sinh viên ngành công nghệ thông tin hoặc các cán bộ đang công tác trong lĩnh vực tin học muốn bổ xung kiến thức cho mình.

Tại tất cả các trường đại học có giảng dạy về tin học, cơ sở dữ liệu là môn học chính cho các sinh viên khoa công nghệ thông tin. Vì thế giáo trình này có thể làm tư liệu học tập cho sinh viên hệ cử nhân tin học, cử nhân cao đẳng tin học, kĩ sư tin học, hoặc có thể làm tài liệu tham khảo cho các học viên cao học, nghiên cứu sinh và các giảng viên tin học.

PGS.TS. Vũ Đức Thi

Chương mở đầu

Cơ sở dữ liệu (CSDL) là một trong những lĩnh vực được tập trung nghiên cứu và phát triển của công nghệ thông tin, nhằm giải quyết các bài toán quản lí, tìm kiếm thông tin trong những hệ thống lớn, đa dạng, phức tạp cho nhiều người sử dụng trên máy tính điện tử. Cùng với sự ứng dụng mạnh mẽ công nghệ thông tin vào đời sống xã hội, kinh tế, quốc phòng ...Việc nghiên cứu CSDL đã và đang phát triển ngày càng phong phú và hoàn thiện. Từ những năm 70, mô hình dữ liệu quan hệ do E.F. Codd đưa ra với cấu trúc hoàn chỉnh đã tạo lên cơ sở toán học cho các vấn đề nghiên cứu lí thuyết về CSDL. Với ưu điểm về tính cấu trúc đơn giản và khả năng hình thức hoá phong phú, CSDL quan hệ dễ dàng mô phỏng các hệ thống thông tin đa dạng trong thực tiễn, tạo điều kiện lưu trữ thông tin tiết kiệm, có tính độc lập dữ liệu cao, dễ sửa đổi, bổ sung cũng như khai thác dữ liệu. Mặt khác, việc khai thác và áp dụng các kĩ thuật tổ chức và sử dụng bộ nhớ cho phép việc cài đặt các CSDL quan hệ đưa lại hiệu quả cao và làm cho CSDL quan hệ chiếm ưu thế trên thị trường.

Nhiều hệ quản trị CSDL đã được xây dựng và đưa vào sử dụng rộng rãi như : DBASE, **FOXBASE,**

FOXPRO, PARADOX, ORACLE, MEGA, IBM DB2, SQL for WINDOWS NT...

Mô hình dữ liệu quan hệ đặt trọng điểm hàng đầu không phải là khai thác các tiềm năng của máy mà ở sự mô tả trực quan dữ liệu theo quan điểm của người dùng, cung cấp một mô hình dữ liệu đơn giản, trong sáng, chặt chẽ, dễ hiểu và tạo khả năng tự động hoá thiết kế CSDL quan hệ. Có thể nói lí thuyết thiết kế và cài đặt CSDL, nhất là mô hình dữ liệu quan hệ đã phát triển ở mức độ cao và đạt được những kết quả sâu sắc. Hàng loạt vấn đề đã được nghiên cứu giải quyết như:

- Lí thuyết thiết kế CSDL, các phương pháp tách và tổng hợp các lược đồ quan hệ theo tiêu chuẩn không tổn thất thông tin hay bảo toàn tính nhất thể của các ràng buộc trên dữ liệu .

- Các loại ràng buộc dữ liệu, cấu trúc và các tính chất của chúng, ngữ nghĩa và khả năng áp dụng phụ thuộc dữ liệu ví dụ như phụ thuộc hàm, phụ thuộc đa trị, phụ thuộc kết nối, phụ thuộc lôgic...

- Các vấn đề tối ưu hoá: ở mức vật lí trong việc tổ chức quản lí các tệp; ở mức đường truy nhập với các tệp chỉ số hay các danh sách sắp xếp; ở mức lôgic trên cơ sở rút gọn các biểu thức biểu diễn các câu hỏi, ...vv

.....

Trong Giáo trình này sẽ trình bày một số kiến thức cơ bản nhất về CSDL bao gồm các kiến thức liên quan đến phụ thuộc hàm, khoá và dạng chuẩn, các thuật toán nhận dạng và thiết kế chúng, việc xây dựng các khái niệm này trong các hệ CSDL lớn như MEGA, ORACLE...., việc nghiên cứu và áp dụng chúng để xây dựng các dự án thiết kế tổng thể các hệ thống CSDL hiện nay.

Chương 2

Các kiến thức cơ bản về cơ sở dữ liệu

2.1. Khát quát về mô hình dữ liệu

Thông thường đối với việc thiết kế và xây dựng các hệ thông tin quản lí, chúng ta cần xử lí các file dữ liệu. Những file này bao gồm nhiều bản ghi (record) có cùng một cấu trúc xác định (loại bản ghi). Đồng thời, mỗi bản ghi được phân chia thành các trường dữ liệu (fild). Một cơ sở dữ liệu là một hệ thống các file dữ liệu, mỗi file này có cấu trúc bản ghi khác nhau, nhưng về mặt nội dung có quan hệ với nhau. Một hệ quản trị cơ sở dữ liệu là một hệ thống quản lí và điều hành các file dữ liệu. Nói chung một hệ quản trị cơ sở dữ liệu thường có những đặc tính sau :

- Có tính độc lập với các công cụ lưu trữ,

- Có tính độc lập với các chương trình phần mềm của người sử dụng (có nghĩa là các ngôn ngữ lập trình khác nhau có thể được dùng trong hệ này),

- Có khả năng tại một thời điểm truy nhập vào nhiều nơi trong hệ này ,

- Có khả năng khai thác tốt tiềm năng của máy,

- Người dùng với kiến thức tối thiểu cúng có thể xử dụng được hệ này,

- Bảo đảm an toàn dữ liệu và bảo mật dữ liệu,

- Thuận lợi và mềm dẻo trong việc bổ xung, loại bỏ, thay đổi dữ liệu

- Giảm bớt sự dư thừa dữ liệu trong lưu trữ,

Trong quá trình thiết kế và xây dựng các hệ quản trị cơ sở dữ liệu, người ta tiến hành xây dựng các mô hình dữ liệu. Mô hình dữ liệu phải thể hiện được các mối quan hệ bản chất của các dữ liệu mà các dữ liệu này phản ánh các mối quan hệ và các thực thể trong thế giới hiện thực. Có thể thấy mô hình dữ liệu phản ánh khía cạnh cấu trúc lôgic mà không đi vào khía cạnh vật lí của các cơ sở dữ liệu. Khi xây dựng các mô hình dữ liệu cần phân biệt các thành phần cơ bản sau :

- Thực thể (Entity): Đó là đối tượng có trong thực tế mà chúng ta cần mô tả các đặc trưng của nó.

- Thuộc tính: Đó là các dữ liệu thể hiện các đặc trưng của thực thể.

- Ràng buộc: Đó là các mối quan hệ lôgic của các thực thể.

Tuy vậy, ba thành phần cơ bản trên được thể hiện ở hai mức :

- Mức loại dữ liệu (Type): Đó là sự khái quát hoá các ràng buộc, các thuộc tính, các thực thể cụ thể.

- Mức thể hiện: Đó là một ràng buộc cụ thể, hoặc là các giá trị thuộc tính, hoặc là một thực thể cụ thể

Thông thường chúng ta sẽ nhận được các loại dữ liệu (Type) của các đối tượng cần khảo sát trong quá trình phân tích các thể hiện cụ thể của chúng.

yếu tố quan trọng nhất của cấu trúc cơ sở dữ liệu là dạng cấu trúc dữ liệu mà trong đó các mối quan hệ giữa các dữ liệu lưu trữ được mô tả. Có thể thấy rằng loại dữ liệu nền tảng của việc mô tả các mối quan hệ là loại bản ghi (Record type). Bởi vì các ràng buộc giữa các loại bản ghi tạo ra bản chất cấu trúc của cơ sở dữ liệu. Vì thế, dựa trên việc xác định các ràng buộc giữa các loại dữ liệu được cho như thế nào mà chúng ta phân loại các mô hình dữ liệu. Có nghĩa là từ cách nhìn của người xử dụng việc mô tả các dữ liệu và các ràng buộc giữa các dữ liệu được thực hiện như thế nào. Trên thực tế chúng ta phân biệt hai loại mô hình dữ liệu:

- Mô hình dữ liệu mạng: Trong đó chúng ta thể hiện trực tiếp các ràng buộc tuỳ ý giữa các loại bản ghi,

- Mô hình dữ liệu quan hệ: Trong mô hình này các ràng buộc trên được thể hiện qua các quan hệ (bảng).

Mô hình dữ liệu quan hệ là một công cụ rất tiện lợi để mô tả cấu trúc lôgic của các cơ sở dữ liệu. Như vậy, ở mức lôgic mô hình này bao gồm các file được biểu diễn dưới dạng các bảng. Do đó đơn vị của CSDL quan hệ là một bảng (Một quan hệ được thể hiện trong Định nghĩa 1), trong đó các dòng của bảng là các bản ghi dữ liệu cụ thể (Đó là các thể hiện cụ thể của loại bản ghi), còn tên các cột là các thuộc tính.

Theo cách nhìn của người xử dụng thì một cơ sở dữ liệu quan hệ là một tập hợp các bảng biến đổi theo thời gian.

2.2. Các khái niệm cơ bản và hệ tiên đề Armstrong:

Trong mục này, chúng ta trình bày những khái niệm cơ bản nhất về mô hình dữ liệu quan hệ của E.F. Codd. Những khái niệm cơ bản này gồm các khái niệm về quan hệ, thuộc tính, phụ thuộc hàm, hệ tiên đề Armstrong, khóa, dạng chuẩn....

Những khái niệm này đóng vai trò rất quan trọng trong mô hình dữ liệu quan hệ. Chúng được áp dụng nhiều trong việc thiết kế các hệ quản trị cơ sở dữ liệu hiện nay.

Những khái niệm này có thể tìm thấy trong [1,2,3,4,7,9,10,15,16,17].

Định nghĩa 1. (Quan hệ)

Cho R = $\{a_{1, \dots, a_n}\}$ là một tập hữu hạn và không rỗng các thuộc tính. Mỗi thuộc tính a_i có miền giá trị là D_{ai}. Khi đó r là một tập các bộ $\{h_1, ..., h_m\}$ được gọi là một quan hệ trên R với h_j (j = 1,...m) là một hàm :

$h_j : R \rightarrow \cup D_{ai}$

$a_i \in R$

sao cho: $h_j (a_i) \in D_{ai}$

Chúng ta có thể biểu diễn quan hệ r thành bảng sau:

	a_1	a_2		a_n
h_1	$h_1(a_1)$	$h_1(a_2)$	.....	$h_1(a_n)$
h_2	$h_2(a_1)$	$h_2(a_2)$	.....	$h_2(a_n)$
.	.....			

.

.

h_m $h_m(a_1)$ $h_m(a_2)$ ..... $h_m(a_n)$

Ví dụ: Trong một cơ quan, chúng ta quản lý nhân sự theo biểu gồm các thuộc tính sau:

Nhân sự

Số TT	Họ tên	Giới tính	Năm sinh	Trình độ đào tạo	Lương
001	Nguyễn Văn A	Nam	1970	Đại học	300000
002	Nguyễn Kim Anh	Nữ	1971	Trung cấp	210000
003	Trần Văn ánh	Nam	1969	Đại học	500000
004	Trần Bình	Nam	1965	PTS	450000
....					
120	Trần Thị yến	Nữ	1967	PTS	455000

Chúng ta quy định kích thước cho các thuộc tính (các trường) như sau:

Tên thuộc tính	Kiểu	Kích thước
STT	Kí tự	3
HOTEN	Ký tự	30
GIOITINH	Ký tự	3
NAMSINH	Số	4
TRINHDO	Ký tự	10
LUONG	Số	7

Có nghĩa là qui định cho thuộc tính STT là các dãy gồm 3 kí tự, thuộc tính HOTEN là các dãy gồm 30 kí tự, ..., cho thuộc tính LUONG là các số có nhiều nhất 7 chữ số.

Như vậy chúng ta có tập thuộc tính

NHANSU = {STT, HOTEN, GIOITINH, NAMSINH, TRINHDO, LUONG}

ở đây D_{STT} là tập các dãy gồm 3 kí tự,...., D_{LUONG} là tập các số có nhiều nhất 7 chữ số.

Khi đó chúng ta có quan hệ r = {h_1, h_2,..., h_{120}}, ở đây ví dụ như đối với bản ghi thứ 2 (dòng thứ 2) chúng ta có:

h_2 (STT) = 002, h_2 (HOTEN) = Nguyễn Kim ánh

h_2 (GIOITINH) = Nữ, h_2 (NAMSINH) = 1971

h_2 (TRINHDO) = Trung cấp, h_2 (LUONG) = 240000

Định nghĩa 2. (Phụ thuộc hàm)

1. Cho R = $\{a_1,...,a_n\}$ là tập các thuộc tính, r = $\{h_1,...,h_m\}$ là một quan hệ trên R, và A, B $\subseteq$ R.

2. Khi đó chúng ta nói A xác định hàm cho B hay B phụ thuộc hàm vào A trong r (Kí pháp A $\xrightarrow{f}_{r}$ B) nếu

$(\forall h_i,h_j \in r)((\forall a \in A)(h_i(a)= h_j(a)) \Rightarrow(\forall b \in B)(h_i(b)=h_j(b)))$

Đặt F_r = { (A,B): A,B $\subseteq$ R, A $\xrightarrow{f}_{r}$ B }. Lúc đó F_r được gọi là họ đầy đủ các phụ thuộc hàm của r.

Khái niệm phụ thuộc hàm miêu tả một loại ràng buộc (phụ thuộc dữ liệu) xảy ra tự nhiên nhất giữa các tập thuộc tính. Dù hiện nay đã có nhiều loại phụ thuộc dữ liệu được nghiên cứu, xong về cơ bản các hệ quản trị cơ sở dữ liệu lớn sử dụng phụ thuộc hàm.

Định nghĩa 3.

Phụ thuộc hàm (PTH) trên tập các thuộc tính R là một dãy kí tự có dạng A $\rightarrow$B, ở đây A,B $\subseteq$ R. Chúng

ta nói PTH $A \rightarrow B$ đúng trong quan hệ r if $A \xrightarrow{f}_{r} B$. Chúng ta cũng nói rằng r thoả mãn

$A \rightarrow B$.

Dễ thấy, F_r là tập tất cả các PTH đúng trong r.

Chú ý: Trong giáo trình này chúng ta có thể viết (A,B) hoặc $A \rightarrow B$ thay cho $A \xrightarrow{f}_{r} B$ mà không bị lẫn về mặt kí pháp.

Định nghĩa 4. (Hệ tiên đề của Armstrong)

Giả sử R là tập các thuộc tính và kí pháp P(R) là tập các tập con của R. Cho $Y \subseteq P(R) \times P(R)$. Chúng ta nói Y là một họ f trên R nếu đối với mọi A, B, C, $D \subseteq R$

(1) $(A,A) \in Y$,

(2) $(A,B) \in Y, (B,C) \in Y \Rightarrow (A,C) \in Y$,

(3) $(A,B) \in Y, A \subseteq C, D \subseteq B \rightarrow (C,D) \in Y$,

(4) $(A,B) \in Y, (C,D) \in Y \Rightarrow (A \cup C, B \cup D) \in Y$.

Rõ ràng, F_r là một họ f trên R.

Trong [l] A. A. Armstrong đã chứng minh một kết quả rất quan trọng như sau : Nếu Y là một họ f

bất kì thì tồn tại một quan hệ r trên R sao cho F_r = Y.

Kết quả này cùng với định nghĩa của phụ thuộc hàm chứng tỏ rằng hệ tiên đề Armstrong là đúng đắn và đầy đủ.

Mặt khác, hệ tiên đề này cho ta những đặc trưng của họ các phụ thuộc hàm, mà các đặc trưng này không phụ thuộc vào các quan hệ (bảng) cụ thể. Nhờ có hệ tiên đề này các công cụ của toán học được áp dụng để nghiên cứu làm sáng tỏ cấu trúc lôgic của mô hình dữ liệu quan hệ. Đặc biệt chúng ta xử dụng công cụ thuật toán để thiết kế các công đoạn xây dựng các hệ quản trị cơ sở dữ liệu.

Chúng ta đưa ra ví dụ chỉ ra có nhiều quan hệ khác nhau xong các họ đầy đủ các phụ thuộc hàm của chúng lại như nhau.

Cho r_1 và r_2 là các quan hệ sau:

r_1 =

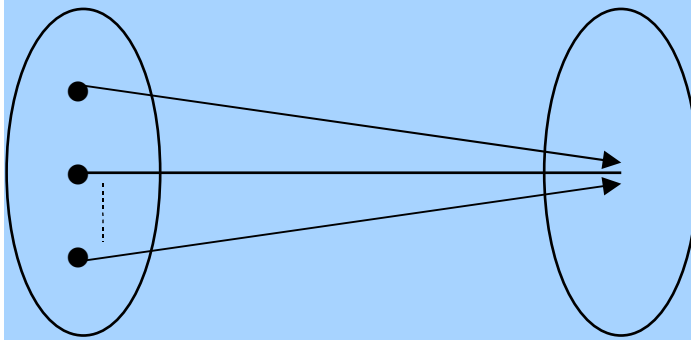
a	b
0	0
1	1
2	1
3	2

r_2 =

a	b
0	0
1	1
2	1
3	1

Có thể thấy r_1 và r_2 khác nhau nhưng $F_{r1} = F_{r2}$.

Như vậy, tương quan giữa lớp các quan hệ với lớp các họ phụ thuộc hàm có thể được thể hiện bằng hình vẽ sau.



Lớp các quan hệ Lớp các phụ thuộc hàm

Định nghĩa 5.

Một hàm L : P(R) $\rightarrow$ P(R) được gọi là một hàm đóng trên R nếu với mọi A, B $\in$ P(R) thì :

- A $\subseteq$ L(A),
- Nếu A $\subseteq$ B thì L(A) $\subseteq$ L(B),
- L(L(A)) = L(A).

Định lí 6.

Nếu F là một họ f và chúng ta đặt

$L_F = \{a : a \in R \text{ và } (A, \{a\}) \in F\}$

thì L_F là một hàm đóng. Ngược lại, nếu L là một hàm đóng thì tồn tại duy nhất một họ f F trên R sao cho $L = L_F$, ở đây

$F = \{ (A,B) : A, B \subseteq R , B \subseteq L(A) \}.$

Như vậy, chúng ta thấy có một tương ứng 1-1 giữa lớp các hàm đóng và lớp các họ f . Chúng ta có hình vẽ sau

Lớp các họ phụ thuộc hàm
Lớp các hàm đóng

Định lí 6 chỉ ra rằng để nghiên cứu phân tích các đặc trưng của họ các phụ thuộc hàm chúng ta có thể dùng công cụ hàm đóng.

Sau này trong mục 2.3 chúng tôi sẽ trình bày nhiều công cụ nữa để nghiên cứu cấu trúc lôgic của họ các phụ thuộc hàm.

Định nghĩa 7. (Sơ đồ quan hệ)

Chúng ta gọi sơ đồ quan hệ (SĐQH) s là một cặp <R,F>, ở đây R là tập các thuộc tính và F là tập các phụ thuộc hàm trên R. Kí pháp F^+ là tập tất cả các PTH được dẫn xuất từ F bằng việc áp dụng các qui tắc trong Định nghĩa 4.

Đặt $A^+ = \{a: A \rightarrow \{a\} \in F^+\}$. A^+ được gọi là bao đóng của A trên s.

Có thể thấy rằng $A \rightarrow B \in F^+$ nếu và chỉ nếu $B \subseteq A^+$.

Tương tự chúng ta đặt $A_r^+ = \{a : A \xrightarrow{f}_r \{a\}\}$. A_r^+ được gọi là bao đóng của A trên r.

Theo [1] chúng ta có thể thấy nếu s = <R,F> là sơ đồ quan hệ thì có quan hệ r trên R sao cho $F_r=F^+$. Quan hệ r như vậy chúng ta gọi là quan hệ Armstrong của s.

Trong trường hợp này hiển nhiên các PTH của s đúng trong r.

Định nghĩa 8. (Khoá)

Giả sử r là một quan hệ , s = <R,F> là một sơ đồ quan hệ, Y là một họ f trên R, và $A \subseteq R$. Khi đó A là một khoá của r (tương ứng là một khoá của s, một khoá của Y) nếu $A \xrightarrow{f}_{r} R$ ($A \to R \in F^+$, $(A,R) \in Y$). Chúng ta gọi A là một khoá tối tiểu của r (tương ứng của s, của Y) nếu

- A là một khoá của r (s, Y),

- Bất kì một tập con thực sự của A không là khoá của r (s, Y).

Chúng ta kí pháp K_r, (K_s, K_y) tương ứng là tập tất cả các khoá tối tiểu của r (s, Y).

Chúng ta gọi K (ở đây K là một tập con của P(R)) là một hệ Sperner trên R nếu với mọi $A,B \in K$ kéo theo $A \subseteq B$).

Có thể thấy K_r,K_s, K_y là các hệ Sperner trên R.

Định nghĩa 9.

Giả sử K là một hệ Sperner trên R. Chúng ta định nghĩa tập các phản khoá của K, kí pháp là K^{-1}, như sau:

$K^{-1} = \{A \subset R : (B \in K) \Rightarrow (B \not\subseteq A)$ and $(A \subset C) \Rightarrow (\exists B \in K)(B \subseteq C)\}$

Dễ thấy K^{-1} cũng là một hệ Sperner trên R.

Tập phản khoá đóng vai trò rất quan trọng trong quá trình nghiên cứu cấu trúc lôgic của các họ phụ thuộc hàm, khóa, dạng chuẩn, quan hệ Armstrong, đặc biệt đối với các bài toán tổ hợp trong mô hình dữ liệu quan hệ.

Trong [5] người ta đã nêu ra rằng nếu s = <R,F> là một sơ đồ quan hệ trên R, thì K_s là hệ Sperner trên R. Ngược lại, nếu K là một hệ Sperner bất kì trên R, thì tồn tại một sơ đồ quan hệ s sao cho $K_s = K$.

Ví dụ: Cho $K = \{ A_1,.....,A_m \}$ là một hệ Sperner. Khi đó s = <R,F>, ở đây $F = \{ A_1 \rightarrow R, ... , A_m \rightarrow R\}$ là sơ đồ quan hệ mà $K_s = K$.

Nhận xét :

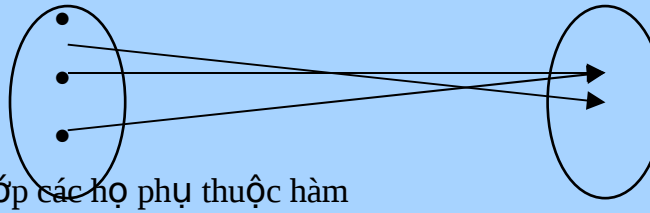
- Có thể cho ví dụ chỉ ra rằng có nhiều sơ đồ quan hệ khác nhau nhưng tập các khoá tối tiểu của chúng giống nhau. Có nghĩa là tồn tại

$s_1 = <R,F_1>, ..., s_t = <R,F_t>$ ($2 \leq t$) mà $F_1^+ \neq .... \neq F_t^+$, nhưng

$K_{s1} = ... = K_{st}$.

Tất nhiên, nếu $F_1 = F_2$ thì $K_{s1} = K_{s2}$.

Mối quan hệ giữa lớp họ phụ thuộc hàm và lớp các hệ Sperner thể hiện qua hình vẽ sau



Lớp các họ phụ thuộc hàm
Lớp các hệ Sperner

- Nếu K đóng vai trò là một tập các khoá tối tiểu của một sơ đồ quan hệ nào đó, thì theo định nghĩa K^{-1} là tập tất cả các tập không phải khoá lớn nhất.

Trong Giáo trình này, chúng ta qui ước rằng nếu hệ Sperner đóng vai trò là tập các khoá tối tiểu (tập các phản khoá), thì hệ này không rỗng (không chứa R).

Định nghĩa 10.

Cho $I \subseteq P(R)$. Khi đó I được gọi là nửa dàn giao nếu

$R \in I$ và $A,B \in I \Rightarrow A \cap B \in I$.

Giả sử $M \subseteq P(R)$. Đặt $M^+ = \{\cap M' : M' \subseteq M\}$. Khi đó chúng ta nói rằng M là một hệ sinh của I nếu $M^+ = I$.

Chú ý rằng $R \in M^+$ nhưng R không là một phần tử của M, bởi vì chúng ta theo thông lệ cho R là giao của một tập rỗng các tập con của M.

Kí pháp $N_I = \{A \in I : A \neq \cap \{A' \in I : A \subset A'\}\}$.

Trong [4] người ta đã chỉ ra rằng N_I là hệ sinh nhỏ nhất và duy nhất của I. Có nghĩa là đối với mọi hệ sinh N' của I chúng ta có $N_I \subseteq N'$.

Định nghĩa 11.

Cho r là một quan hệ trên R. Chúng ta đặt $E_r = \{E_{ij} : 1 \leq i \leq j \leq |r|\}$, ở đây $E_{ij} = \{a \in R: h_i(a) = h_j(a)\}$. E_r được gọi là hệ bằng nhau của r.

Đặt $M_r = \{ A \in P(R) : \exists E_{ij} = A, \nexists E_{pq}: A \subset E_{pq}\}$. Khi đó chúng ta gọi M_r là hệ bằng nhau cực đại của r.

Sau này ta sẽ thấy hệ bằng nhau và hệ bằng nhau cực đại được dùng rất nhiều trong các thuật toán thiết kế.

Mối quan hệ giữa lớp các quan hệ và lớp các phụ thuộc hàm đóng một vai trò quan trọng trong quá trình nghiên cứu cấu trúc lôgic của lớp các phụ thuộc hàm.

Định nghĩa 12.

Cho trước r là một quan hệ r và F là một họ f trên R. Chúng ta nói rằng r là thể hiện họ F nếu $F_r = F$. Chúng ta cũng có thể nói r là một quan hệ Armstrong của F.

Bây giờ, chúng ta đưa ra một điều kiện cần và đủ để một quan hệ là thể hiện một họ f cho trước.

Định lý 13.

Giả sử r = $\{h_1, ..., h_m\}$ là một quan hệ, F là một họ f trên R thì r thể hiện F nếu và chỉ nếu với mọi $A \subseteq R$

$$L_F(A) = \begin{cases} \cap E_{ij} \text{ nếu tồn tại } E_{ij} \in E_r : A \subseteq E_{ij}, \\ R \quad \text{ngược lại.} \end{cases}$$

ở đây $L_F(A) = \{a \in R : (A, \{a\}) \in F\}$ và E_r là hệ bằng nhau của r.

Lời giải: Đầu tiên chúng ta chứng minh rằng trong một quan hệ r bất kì với mọi $A \subseteq R$

$$L_{Fr}(A) = \begin{cases} \cap E_{ij} \text{ nếu tồn tại } E_{ij} \in E_r : A \subseteq E_{ij}, \\ R \quad \text{ngược lại.} \end{cases}$$

Giả sử đầu tiên chúng ta công nhận rằng A là một tập mà không có $E_{ij} \in E_r$ với $A \subseteq E_{ij}$ với mọi $h_i, h_j \in r$, $a \in A : h_i(a) = h_j(a)$. Theo định nghĩa của phụ thuộc hàm điều này kéo theo $A \rightarrow R$ và bởi định nghĩa của L_{Fr} ta thu được $L_F(A) = R$. Rõ ràng là

$$L_{Fr}(\varnothing) = \cap E_{ij}$$

$$E_{ij} \in E_r$$

Nếu $A \neq \varnothing$ và có một $E_{ij} \in E_r$ mà $A \subseteq E_{ij}$ thì chúng ta đặt

$$V = \{ E_{ij} : A \subseteq E_{ij}, E_{ij} \in E_r \}$$

và $E = \cap E_{ij}$

$$E_{ij} \in V$$

Dễ dàng nhận thấy rằng $A \subseteq E$. Nếu $V = E_r$ thì chúng ta nhận thấy rằng $(A,E) \in F_r$ nếu $V \neq E_r$ thì có thể coi như với mọi $E_{ij} \in V$ chúng ta có

(Với mọi $a \in A$) $(h_i(a) = h_j(a)) \rightarrow$ (Với mọi $b \in B$) $(h_i(b) = h_j(b))$ và với mọi $E_{ij} \notin V$ có một $a \in A$ mà $h_i(a) \neq h_j(a)$. Như vậy, $(A, E) \in F_r$.

Từ định nghĩa của L_{Fr} ta có $E \subseteq L_{Fr}(A)$. bởi vì r là một quan hệ trên R, chúng ta có $E \subset R$. Sử dụng $A \subseteq E \subseteq L_{Fr}(A)$ ta thu được $(E, L_{Fr}(A)) \in F_r$.

Bây giờ, ta giả sử rằng c là một thuộc tính mà $c \notin E$. Khi đó có một $E_{ij} \in V$ mà $c \notin E_{ij}$. Điều này kéo theo sự tồn tại của một cặp $h_i, h_j \in r$ mà với mọi $b \in E$: $h_i(b) = h_j(b)$ nhưng $h_i(c) \neq h_j(c)$. Có thể thấy rằng theo định nghĩa phụ thuộc hàm $(E \cup \{c\})$ không phụ thuộc vào E. Như vậy, với mọi thuộc tính $c \notin E$ ta có $(E, E \cup \{c\}) \notin F_r$. Bằng định nghĩa của L_{Fr} ta thu được :

$$L_{Fr}(A) = \bigcap E_{ij}$$

$$E_{ij} \in V$$

Trên cơ sở Định lí 6 chúng ta dễ dàng thấy rằng $F_r = F$ nếu và chỉ nếu $L_{Fr} = L_F$. □

Giả sử L là một hàm đóng. Đặt $Z(L) = \{A \subseteq R : L(A) = A\}$.

Rõ rằng, Z(L) là tập đóng với phép giao.

Có thể thấy là với mọi E_{ii} ($E_{ii} \in E_r$), chúng ta có $E_{ii} \in Z(L_{Fr})$, có nghĩa là $E^+_r \subseteq Z(L_{Fr})$

Nhờ Định lý 13 chúng ta có $Z(L_{Fr}) \subseteq E^+_R$. Như vậy chúng ta có

Hệ quả 14.

Giả sử r quan hệ, F là một họ f trên R. Khi đó r thể hiện F nếu và chỉ nếu $Z(L_F) = E^+_R$.

Trong [5] người ta đã chỉ ra rằng nếu cho một hệ Sperner không rỗng tuỳ ý K thì tồn tại một quan hệ r để $K = K_r$.

Bây giờ, chúng ta đưa ra một định nghĩa dưới đây

Định nghĩa 15.

Cho trước quan hệ r và hệ Sperner K trên R. Chúng ta nói rằng r thể hiện K nếu $K_r = K$.

Định nghĩa 16.

Cho F là một họ f trên R, và (A,B) là một phần tử của F. Chúng ta nói (A,B) là một phụ thuộc có vế phải cực đại của F nếu với mọi B' (B B') và (A,B') ∈ F kéo theo B = B'.

Chúng ta kí pháp M(F) là tập tất cả các phụ thuộc có vế phải cực đại của F. Chúng ta nói rằng B là vế phải cực đại của F nếu có A sao cho (A,B) ∈ M(F). Kí pháp I(F) là tập tất cả các vế phải cực đại của F.

Dưới đây chúng ta cho một điều kiện cần và đủ để một quan hệ thể hiện một hệ Sperner.

Định lý 17.

Giả sử K là một hệ Sperner không rỗng, r là một là một quan hệ trên R. Khi đó r thể hiện K nếu và chỉ nếu $K^{-1} = M_r$, ở đây M_r là hệ bằng nhau cực đại của r.

Lời giải: Có thể xem như là nếu K là một hệ Sperner không rỗng thì K^{-1} tồn tại. Mặt khác, K và K^{-1} là xác định duy nhất. Cho nên, chúng ta có $K_r = K$ nếu và chỉ nếu $K_r^{-1} = K^{-1}$.

Bây giờ chúng ta có thể chỉ cần chứng minh rằng $K_r^{-1} = M_r$. Rõ ràng, F_r là một họ f trên R. Đầu tiên chúng ta có thể giả thiết rằng A là một phản khoá của K_r. Rõ ràng A ≠ R. Nếu có một B sao cho

$A \subset B$ và $A \rightarrow B$, thì bằng định nghĩa của phản khoá, chúng ta có $B \rightarrow R$ và $A \rightarrow R$. Đây là một điều phi lý. Vì vậy $A \in I(F_r)$. Nếu có một B' sao cho $B' \neq R$, $B' \in I(F_r)$ và $A \subset B'$, thì B' là một khoá của r. Đây là một điều mâu thuẫn $B' \neq R$. Do đó $A \in I(F_r) - R$ và không tồn tại B' ($B' \in I(F_r) - R$) để $A \subset B'$.

Mặt khác, theo định nghĩa của một quan hệ, $R \notin M_r$. Rõ ràng, $E_{ij} \in I(F_r)$. Như vậy chúng ta có $M_r \subseteq I(F_r)$. Nếu Dlà một tập sao cho $\forall C \in M_r : D \subseteq C$, thì D là một khoá của r. Bởi vậy, M_r là tập phần tử rời nhau cực đại của $I(F_R)$. Vì vậy chúng ta có $A \in M_r$

Ngược lại, nếu $A \in M_r$, thì theo định nghĩa của quan hệ và M_r Chúng ta có $A \rightarrow R$. Có nghĩa là $\forall K \in K_r : K \subseteq A$. Mặt khác, bởi vì A là một phần tử của tập bằng nhau cực đại, cho nên đối với tất cả D($A \subset D$) chúng ta có $D \rightarrow R$. Đồng thời theo định nghĩa của các phản khoá $A \in K^{-1}_r$. □

Cho trước s = <R,F> là một sơ đồ quan hệ trên R, K_s là tập tất cả các khoá tối tiểu của s. Kí pháp K_s^{-1} là tập các phản khoá của s. Từ Định lí 17 chúng ta có kết quả sau.

Hệ quả 18.

Cho trước s = <R,F> là một sơ đồ quan hệ và r là một quan hệ trên R. Khi đó $K_r = K_s$ nếu và chỉ nếu $K_s^{-1} = M_r$, ở đây M_r là hệ bằng nhau cực đại của r.

Chúng ta đưa ra một kết quả liên quan đến cả K^{-1} và K.

Định lý 19.

Giả sử K là một hệ Sperner trên R. Giả sử s(K) = min{m:K=K_r, |r|=m, r là quan hệ trên R}. Khi đó

$$\sqrt{2^{|K^{-1}|}} \leq s(K) \leq |K^{-1}|+1.$$

Đánh giá này chỉ ra mối quan hệ giữa kích cỡ của quan hệ tối tiểu mà thể hiện một hệ Sperner (ở đây hệ này đóng vai trò là một hệ khoá tối tiểu) cho trước với lực lượng của hệ phản khoá tương ứng của nó.

Cho F là một họ f trên R. Theo Định nghĩa 16 thì dễ thấy I(F) là một nửa dàn giao. Khi đó

$$N_{I(F)} = \{A \in I(F) : A \neq \cap\{A' \in I : A \subset A'\}\}.$$

Trên cơ sở này chúng ta có định lí sau đánh giá quan hệ Armstrong nhỏ nhất (minimal Armstrong relation) của một họ f.

Định lý 20.

Giả sử F là một họ f trên R. Đặt

s(F) = min{m:F=F_r, |r|=m, r là quan hệ trên R}. Khi đó $\sqrt{2^{|N_{I(F)}|}} \leq s(F) \leq |N_{I(F)}|+1$.

Đánh giá này cho chúng ta mối quan hệ giữa kích thước của quan hệ Armstrong nhỏ nhất của họ F với lực lượng của hệ sinh nhỏ nhất của I(F).

Bây giờ, chúng ta đánh giá sâu hơn kích thước của các quan hệ Armstrong nhỏ nhất trên R, cũng như các quan hệ nhỏ nhất mà thể hiện một hệ Sperner K cho trước.

Chúng ta đặt

P(n) = max { s(K) : K là hệ Sperner tùy ý trên R = $\{a_1,..., a_n\}$ }

và Q(n) = max { s(F) : F là họ f tùy ý trên R = $\{a_1,..., a_n\}$ }

Khi đó chúng ta có

Định lý 21.

$- 1/n^2 . C_{[n/2]}^n \leq P(n) \leq C_{[n/2]}^n + 1$

$- 1/n^2 . C_{[n/2]}^n \leq Q(n) \leq (1 + C (1/n^{1/2})).C_{[n/2]}^n$

Đánh giá này cho toàn bộ các hệ Sperner (ở đây hệ này đóng vai trò là một hệ khoá tối tiểu) và các họ f có thể có trên R.

Định nghĩa 22.

Giả sử r là một quan hệ trên R và K_r là tập của tất cả các khoá tối tiểu của r. Chúng ta nói rằng a là

một thuộc tính cơ bản của r nếu tồn tại một khoá tối tiểu K ($K \in K_r$) để a là một phần tử của K.

Nếu a không thoả mãn tính chất trên thì a là thuộc tính thứ cấp.

Trong chương 3 chúng ta có thể thấy các thuộc tính cơ bản và thứ cấp đóng một vai trò quan trọng trong việc chuẩn hoá các sơ đồ quan hệ và các quan hệ.

Trong [24] đã chứng minh kết quả sau

Định lí 23.

Cho trước một sơ đồ quan hệ s = <R,F> và một thuộc tính a. Bài toán xác định a là thuộc tính cơ bản hay không là bài toán NP- đầy đủ.

Có nghĩa rằng cho đến nay không có một thuật toán có độ phức tạp thời gian đa thức để giải quyết bài toán này.

Tuy vậy, chúng ta chỉ ra rằng đối với quan hệ thì bài toán này được giải bằng một thuật toán thời gian đa thức.

Trước tiên chúng ta chứng minh kết quả sau.

Định lý 24.

Giả sử K là một hệ Sperner trên R. thì

$$\cup K = R - \cap K^{-1}.$$

Lời giải:

Nếu $c \in \cup K$, thì tồn tại một khoá tối tiểu K sao cho $c \in K$. Đặt $H = K - c$. Rõ ràng H không chứa một khoá nào. Như vậy, tồn tại một phản khoá B để B chứa H. Có thể thấy c không là phần tử của B, vì ngược lại chúng ta có B chứa K. Điều này là vô lí. Vì thế chúng ta có

$$c \in R - B \subseteq R - \cap K^{-1}.$$

Bây giờ chúng ta giả thiết $c \notin \cup K$ và $B \in K^{-1}$. Có thể thấy $c \in B$. Vì ngược lại $c \notin B$, thì $\{c\} \cup B$ hình thành một khoá chứa khoá tối tiểu K ($K \in K$). Như vậy $K \subseteq B$, và chúng ta có $c \in K$. Điều này là vô lý. □

Trên cơ sở của Định lý 17 và Định lí 24 chúng ta chỉ ra rằng đối với một quan hệ, thì vấn đề về thuộc tính cơ bản có thể là giải quyết bằng một thuật toán thời gian đa thức.

Đầu tiên chúng ta xây dựng một thuật toán xác định tập các thuộc tính cơ bản của quan hệ cho trước.

Thuật toán 25.

Vào: $r = \{h_1, ..., h_m\}$ là một quan hệ trên R

Ra: V là tập tất cả thuộc tính cơ bản của r

Bước 1: Từ r chúng ta xây dựng một tập $E_r = \{E_{ij} : m \geq j > i \geq 1\}$ và $E_{ij} = \{ a \in R : h_j(a) = h_j(a) \}$

Bước 2 : Từ E_r chúng ta xây dựng tập

$M = \{B \in P(R) :$ Tồn tại $E_{ij} \in E_r : E_{ij} = B\}$

Bước 3: Từ M xây dựng tập $M_r = \{ B \in M :$ Với mọi $B' \in M : B \not\subset B'\}$

Có thể thấy rằng M_r tính được bằng một thuật toán thời gian đa thức.

Bước 4: Xây dựng tập $V = R - \cap M_r$.

Rõ ràng $m.(m+1)/2 \geq |E_r| \geq |M| \geq |M_r|$. Bởi vậy thời gian tính của Thuật toán 25 là một đa thức theo số hàng và số cột của r.

Từ các Định lý 17, 24 và Thuật toán 25 chúng ta có hệ quả sau.

Hệ quả 26.

Tồn tại thuật toán đối với một quan hệ r cho trước, xác định một thuộc tính bất kì là cơ bản hay không với thời gian tính đa thức theo số hàng và cột của r.

Một vấn đề thường xuyên hay xảy ra là đối với một sơ đồ quan hệ cho trước s = <R,F>, và một phụ thuộc hàm $A \rightarrow B$, chúng ta muốn biết $A \rightarrow B$ có là phần tử của F^+ hay không. Để trả lời câu hỏi này

chúng ta cần tính bao đóng F^+ của tập các phụ thuộc hàm F. Tuy nhiên tính F^+ trong trường hợp tổng quát là rất khó khăn và tồn kém thời gian vì tập các phụ thuộc hàm thuộc F^+ là rất lớn cho dù F có thể là nhỏ. Chẳng hạn $F = \{A \rightarrow B_1, A \rightarrow B_2, .. A \rightarrow B_n\}$, F^+ khi đó còn bao gồm cả những phụ thuộc hàm $A \rightarrow Y$ với $Y \subseteq \{B_1 \cup B_2 \cup ... \cup B_n\}$. Như vậy sẽ có 2^n tập con Y . Trong khi đó, việc tính bao đóng của tập thuộc tính A lại không khó. Theo kết quả đã trình bày ở trên việc kiểm tra $A \rightarrow B \in F^+$ không khó hơn việc tính A^+.

Ta có thể tính bao đóng A^+ qua thuật toán sau:

Thuật toán 27

Vào: s = <R,F>, ở đây R ={ a_1 ,...., a_n } tập hữu hạn các thuộc tính, F tập các phụ thuộc hàm, $A \subseteq R$

Ra: A^+ bao đóng của A đối với F

Thuật toán thực hiện như sau: Tính các tập thuộc tính A_0, A_1... theo qui tắc:

1) $A_0 = A$

2) $A_i = A_{i-1} \cup \{a\}$ sao cho $\exists (C \rightarrow D) \in F, \{a\} \in Y$ và $C \subseteq A_{i-1}$

Vì $A = A_0 \subseteq ... \subseteq A_i \subseteq R$, và R hữu hạn nên tồn tại một chỉ số i nào đó mà $A_i = A_{i+1}$, khi đó thuật toán dừng và $A^+ = A_i$

Chúng ta có thể thấy độ phức tạp thời gian của thuật toán này là đa thức theo kích thước của s.

Để tiện kí pháp chúng ta thay a $\cup$ b bởi viết ab.

Ví dụ: s = <R,F>, ở đâyR = { a,b,c,d,e,g }, F bao gồm 8 phụ thuộc hàm

$$ab \to c \qquad d \to eg$$

$$c \to a \qquad be \to c$$

$$bc \to d \qquad cg \to bd$$

$$acd \to b \qquad ce \to ag$$

và A = bd.

Dùng Thuật toán 27 chúng ta có thể thấy

A_0 = bd,

A_1 = bdeg,

A_2 = bcdeg,

A_3 = abcdeg,

A^+ = abcdeg.

Để chứng minh tính đúng đắn của Thuật toán 27 chúng ta có thể dùng phương pháp quy nạp để chỉ ra rằng nếu a thuộc A_i thì a cũng thuộc A^+.

Việc chỉ ra điều ngược lại cũng bằng qui nạp nhưng khó khăn hơn là nếu a nằm trong A^+ thì a nằm trong một số A_i nào đó.

Định lý 28.

Thuật toán 27 tính chính xác A^+.

Như vậy, để xác định một phụ thuộc hàm $A \rightarrow B$ có thuộc F^+ hay không chúng ta chỉ cần kiểm tra $B \subseteq A^+$?.

Bây giờ, chúng ta đi tìm thuật toán tìm bao đóng cho một tập các thuộc tính trên một quan hệ bất kì

Đối với một quan hệ bất kì theo Định lí 13 chúng ta đã chúng minh với mọi $A \subseteq R$

$$L_{Fr}(A) = \begin{cases} \cap E_{ij} \text{ nếu tồn tại } E_{ij} \in E_r : A \subseteq E_{ij}, \\ A \subseteq E_{ij} \\ R \quad \text{ngược lại.} \end{cases}$$

Có thể thấy $L_{Fr}(A) = A_r^+$. Do vậy chúng ta có ngay thuật toán sau để tính bao đóng cho một tập bất kì trên quan hệ r.

Thuật toán 29

Vào: $r = \{h_1, ..., h_m\}$ là một quan hệ trên R

Ra: V là tập tất cả thuộc tính cơ bản của r

Bước 1: Từ r chúng ta xây dựng một tập $E_r = \{E_{ij} : m \geq j > i \geq 1\}$ và $E_{ij} = \{ a \in R : h_j(a) = h_j(a) \}$

Bước 2: Từ E_r chúng ta xây dựng một tập $M = \{B \in P(R) : \text{Tồn tại } E_{ij} \in E_r : E_{ij} = B\}$

Bước 3:

$A_r^+ = \begin{cases} \cap B \text{ nếu tồn tại } B \in M : A \subseteq B, \\ R \text{ ngược lại.} \end{cases}$

Dễ thấy, độ phức tạp thời gian của thuật toán này là một đa thức theo kích thước của r.

Định nghĩa 30

Giả sử s = <R,F>, t = <R,G > là hai sơ đồ quan hệ trên R. Khi đó chúng ta nói s tương đương với t nếu $F^+ = G^+$. Nếu s và t tương đương thì đôi khi chúng ta có thể nói rằng s là một phủ của t hoặc t là một phủ của s.

Dễ dàng kiểm tra rằng s và t có tương đương với nhau không.

Mỗi phụ thuộc hàm $Y \rightarrow Z$ ở trong F chúng ta kiểm tra lại $Y \rightarrow Z$ ở trong G^+ bằng vi ệc sử dụng Thuật toán 27 để tính Y^+ và kiểm tra $Z \subseteq Y^+$. Nếu có phụ thuộc hàm $Y \rightarrow Z$ không nằm trong G^+ hoặc ngược lại nếu có phụ thuộc hàm $X \rightarrow W$ ở trong G

nhưng không ở trong F^+ thì điều chắc chắn là F^+ khác G^+. Nếu mỗi phụ thuộc hàm ở trong F thì cũng nằm trong G^+ và mỗi phụ thuộc hàm nằm trong G thì cúng nằm trong F^+, khi đó chúng ta có s và t là tương đương với nhau.

Hiện nay chúng ta cho định nghĩa sau nói về sự tương đương của hai quan hệ.

Định nghĩa 31

Giả sử r và v là hai quan hệ trên R. Khi đó ta nói r và v tương đương với nhau nếu $F_r = F_v$.

Chúng tôi trình bày định lí sau liên quan đến sự tương đương của hai quan hệ.

Định lí 32

Giả sử r và v là hai quan hệ trên R. Khi đó s tương đương với v khi và chỉ khi $N_r = N_v$.

Trên cơ sở Định lí 32 chúng ta đưa ra một thuật toán kiểm tra xem r có tương đương với v hay không.

Thuật toán 33

Vào: r và t là hai quan hệ trên R

Ra: r có tương đương với v hay không

Bước 1: Từ r tính N_r,

Bước 2: Từ v tính N_v

Bước 3 : So sánh M_r với M_v.

Bây giờ, chúng ta quay lại với sơ đồ quan hệ. Chúng ta muốn gọt giũa các phụ thuộc hàm của sơ đồ quan hệ để có tập phụ thuộc hàm tốt hơn.

Định nghĩa 34

Chúng ta nói một sơ đồ quan hệ s = <R,F> là chính tắc nếu

1. Vế phải của mỗi phụ thuộc hàm trong F là thuộc tính đơn.

2. Không có $X \rightarrow a$ nào ở trong F để F- $\{X \rightarrow a\}$ tương đương với F.

3. Không có $X \rightarrow a$ và một tập con Z của X để $F-\{X \rightarrow A\} \cup \{Z \rightarrow a\}$ tương đương với F.

Chúng ta sẽ chỉ ra rằng có thuật toán để tìm một phủ chính tắc cho một sơ đồ quan hệ bất kì.

Trước tiên chúng ta đưa ra mệnh đề sau

Mệnh đề 35:

Mỗi một sơ đồ quan hệ s = <R,F> đều có một phủ tương đương t = <R,G> sao cho vế phải của mỗi phụ thuộc hàm trong G không có hơn một thuộc tính.

Chứng minh: Đặt G là tập phụ thuộc hàm có dạng $X \rightarrow a$, với $X \rightarrow Y$ nằm trong F và a là một

phần tử của Y. Trên cơ sở hệ tiên đề của Armstrong, chúng ta dễ thấy t tương đương với s. □

Chúng ta trình bày thuật toán dưới đây để tìm phủ chính tắc cho một sơ đồ quan hệ cho trước

Thuật toán 36

Vào: s= <R,F>, F = { $A_1 \rightarrow B_1$,..., $A_m \rightarrow B_m$ }

Ra: t = <R,G> là chính tắc và tương đương với s

Do Mệnh đề 35 chúng ta có thể coi s thoả mãn điều kiện 1.

Bước 1: Đặt F_0 = F, với i = 1, ..., m

$F_i = F_{i-1}$ - { $A_i \rightarrow B_i$ } nếu F_{i-1} - { $A_i \rightarrow B_i$ } tương đương với F_i . Trong trương hợp ngược lại thì $F_i = F_{i-1}$.

Bước 2: Nhờ thuật toán tính bao đóng, từ F_m chúng ta lần lượt loại bỏ các thuộc tính thừa trong mỗi vế trái của từng phụ thuộc hàm thuộc F_m.

Kết quả nhận được chính là G.

Dễ thấy rằng thuật toán trên có độ phức tạp thời gian là đa thức theo kích thước của s.

Chúng ta đưa ra một khái niệm sau

Định nghĩa 37

Giả sử s = <R,F> là một sơ đồ quan hệ trên R. Khi đó chúng ta nói rằng s là sơ đồ quan hệ tối tiểu nếu với mọi sơ đồ quan hệ t = < R,G> tương đương với s có | F | < |G|, ở đây |F| là số lượng các phụ thuộc hàm trong F.

David Maier [25] đã chứng minh định lí sau

Định lí 38

Tồn tại một thuật toán có độ phức tạp thời gian đa thức để tìm phủ tối tiểu cho một sơ đồ quan hệ cho trước.

Ví dụ:

Chúng ta xem xét tập các phụ thuộc hàm F trong ví dụ minh hoạ Thuật toán 27. Ban đầu chúng ta tách vế phải ra thành các thuộc tính đơn:

ab $\rightarrow$c,

c $\rightarrow$a,

bc $\rightarrow$d,

acd$\rightarrow$b,

d $\rightarrow$e,

d $\rightarrow$g,

be $\rightarrow$c,

cg$\rightarrow$b,

cg→d,

ce→a,

ce →g.

Rõ ràng ce →a là không cần thiết bởi vì nó suy ra được từ c →a.

cg →b là dư thừa bởi vì có cg →d , c →a và acd →b. Ngoài ra không có một phụ thuộc hàm nào là dư thừa nữa. Dễ thấy acd →b có thể thay thế bởi cd →b. Vậy chúng ta có một phủ chính tắc là :

ab →c,

c →a,

bc→d,

cd→b,

d→e,

d→g,

be→c,

cg→d,

ce→g.

2.3 Các mô tả tương đương của họ các phụ thuộc hàm

Trong mục trên chúng ta đã định nghĩa hàm đóng. Nó là một mô tả tương đương của họ các phụ thuộc hàm. Trong mục này chúng tôi cung cấp cho bạn đọc một số các mô tả tương đương khác của họ này. Chúng chính là các công cụ để chúng ta có thể nghiên cứu phong phú hơn nữa cấu trúc lôgic của họ các phụ thuộc hàm.

Định nghĩa 1

Cho R là tập các thuộc tính và P(R) là tập các tập con của R.

Một hàm C: P(R) $\rightarrow$ P(R) được gọi là một hàm chọn trên R nếu với mọi A $\in$ P(R) thì C (A) $\subseteq$ A.

Giả sử L là một hàm đóng trên R . Chúng ta đặt C(A) = R - L(R-A) (*)

Dễ thấy C là một hàm chọn trên R

Trong [6] người ta đã chứng minh được kết quả sau

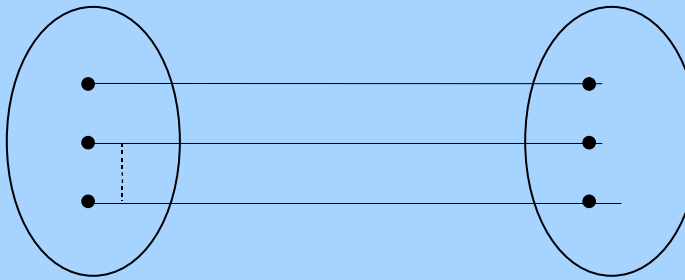
Định lí 2

Tương ứng được xác định như (*) là tương ứng 1-1 giữa tập các hàm đóng và tập các hàm chọn thoả mãn 2 điều kiện sau: với mọi A,B $\subseteq$ R

(1) C(A) $\subseteq$ B $\subseteq$ A kéo theo C(A) = C(B)

(2) A $\subseteq$ B kéo theo C(A) $\subseteq$ C(B).

Chúng ta có hình vẽ sau thể hiện mối quan hệ giữa lớp hàm đóng và lớp hàm chọn đặc biệt trên



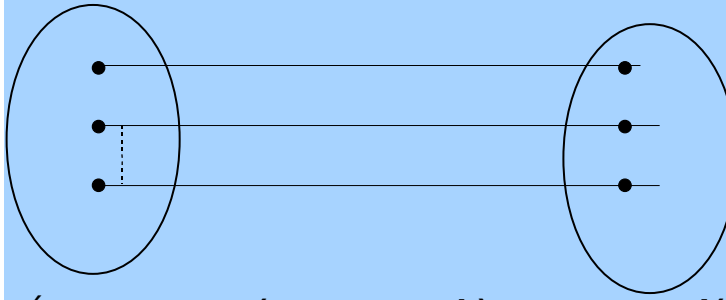
Lớp các hàm đóng Lớp các hàm chọn đặc biệt

Định lí dưới đây chỉ ra một tương ứng 1-1 giữa lớp các hàm đóng và lớp các nửa dàn giao trên R.

Định lí 4

Giả sử L là một hàm đóng trên R. Đặt Z(L) = { A : A $\in$ P(R) và L(A) = A }. Khi đó Z(L) là một nửa dàn giao trên R.

Ngược lại, nếu I là một nửa dàn giao trên R, thì tồn tại duy nhất một hàm đóng L sao cho Z(L) = I, ở đây L(A) = $\cap$ {A' $\in$ I : A $\subseteq$ A'}.



Lớp các hàm đóng Lớp các nửa dàn giao

Như vậy, từ Định lí 6 mục trên và Định lí 4, chúng ta thấy có một tương ứng 1-1 giữa lớp các nửa dàn giao và lớp các họ các phụ thuộc hàm trên R .

Định nghĩa 5

Giả sử $N \subseteq P(R)$. Khi đó N được gọi là tập không giao nếu với mọi $A \in N$ thì $A \neq \cap \{A' \in N : A \subset A'\}$.

Định lí 6

Nếu I là nửa dàn giao, thì N_I là tập không giao. Ngược lại nếu N là tập không giao thì tồn tại duy nhất một nửa dàn giao I sao cho $N_I = N$.

Như vậy, chúng ta thấy có một tương ứng 1-1 giữa lớp các nửa dàn giao và lớp các tập không giao.

Lớp các nửa dàn giao Lớp các tập không giao

Từ Định lí 6 mục trên và các Định lí 4 và 6 chúng ta rút ra kết luận là có một tương ứng 1-1 giữa lớp các họ phụ thuộc hàm với lớp các tập không giao.

Như vậy, để nghiên cứu phân tích các đặc trưng của họ các phụ thuộc hàm chúng ta có thể dùng công cụ nửa dàn giao hoặc tập không giao.

Bây giờ chúng tôi đưa ra khái niệm họ cực đại các thuộc tính. Đồng thời chúng ta chỉ ra rằng họ này là một mô tả tương đương của họ phụ thuộc hàm.

Định nghĩa 7

Giả sử R là tập các thuộc tính. Họ $M=\{(A,\{a\}): A \subset R, a \in R\}$ được gọi là họ cực đại các thuộc tính trên R nếu nó thoả mãn các điều kiện sau

(1) $a \notin A$,

(2) Đối với mọi (B, {b}) ∈ M, a ∉ B và A ⊆ B kéo theo A = B.

(3) ∃(B,{b}) ∈ M : a ∉ B, a ≠ b, và $L_a \cup B$ là một hệ Sperner trên R, ở đây L_a = {A: (A, {a}) ∈ M}.

Nhận xét 8.

- Có thể có (A, {a}), (B, {b}) ∈ M mà a ≠ b, nhưng A = B.

- Do (1) và (2) có thể thấy đối với a ∈ R chúng ta có L_a là một hệ Sperner trên R. Đặc biệt có thể L_a là một hệ Sperner rỗng.

- Trên cơ sở Định nghĩa 7 chúng ta có thể thấy tồn tại một thuật toán thời gian tính đa thức để xác định một tập Y ⊆ P(R) x P(R) có là một họ cực đại các thuộc tính trên R hay không.

Giả sử H là một hàm đóng trên R. Đặt Z(H) = {A : H(A) = A} và M(H) = {(A, {A}) : A ∉ A, A ∈ Z(H) và B ∈ Z(H), A ⊆ B, A ∉ B kéo theo A = B}.

Z(H) là họ các tập đóng của H. Dễ thấy, với mỗi (A, {a}) ∈ M(H), A là tập đóng cực đại mà không chứa a.

Có thể tồn tại (A, {a}), (B, {b}) ∈ M(H) mà a ≠ b, nhưng A = B.

Nhận xét 9.

Giả sử r là một quan hệ trên R và F_r là họ các phụ thuộc hàm của r. Đặt $A_r^+ = \{a: A \rightarrow \{a\} \in F_r\}$ và $Z_r = \{A: A = A_r^+\}$ và N_r là hệ sinh cực tiểu của nó. Có thể thấy $N_r \subseteq E_r$ với

$N_r = \{A \in E_r : A \neq \cap\{B: B \in E_r, A \subset B\}\}$, ở đây E_r là hệ bằng nhau r.

Chúng ta cho định lí dưới đây chỉ ra rằng giữa các hàm đóng và họ cực đạI các thuộc tính có tương ứng 1-1.

Định lí 10.

Giả sử H là một hàm đóng trên R. Khi đó M(H) là họ cực đại các thuộc tính. Ngược lại, nếu M là họ cực đạI các thuộc tính trên R thì tồn tại đúng một hàm đóng H trên R để M(H) = M, ở đây với mọi $B \in P(R)$.

$$H(B) = \begin{cases} \cap A \text{ if } \exists A \in L(M) : B \subseteq A, \\ B \subseteq A \\ R \quad \text{ngược lại,} \end{cases}$$

và $L(M) = \{A: (A, \{a\} \in M\}$.

Lời giải:

Giả sử H là hàm đóng trên R. Cơ sở trên định nghĩa của M(H) ta có (1) và (2). Ta đặt $L'_a = \{ A : (A,$

{a} ∈ M(H)}. Giả thiết có S (B, {b}) ∈ M(H) : a ≠ b, a ∉ B, $L'_a \cup B$ là hệ Sperner trên R (*). Khi đó ta chọn (B,{b}) ∈ M(H) sao cho B là lớn nhất cho (*). Do (2) trong Định nghĩa 7 ta có L_a' là hệ Sperner trên R. Do đó, không có một A nào mà A ∈ L_a' và B ⊆ A. Phù hợp với định nghĩa của M(H) ta có (B,{a}) ∈ M(H). Như vậy, B ∈ L_a'. Điều này là vô lí. Từ đó ta có (3) trong Định nghĩa 7. Có nghĩa là M(H) là họ cực đại các thuộc tính trên R.

Ngược lại, giả sử M là họ cực đại các thuộc tính trên R. Đặt L(M) = {A : (A, {a}) ∈ M}. Đầu tiên ta chứng tỏ rằng L(M) là tập không giao trên R. Đối với bất kì (A, {a}) ∈ M do Nhận xét 2.2 ta có A ≠ A' ∩ A" và A ≠ A' ∩ B, ở đây A', A" ∈ L_a và B ∈ L(M) : A ≠ B.

Nếu tồn tại (B, {b}), (C, {c}) ∈ M sao cho b ≠ a, c ≠ a, A ⊂ B, A ⊂ C thì bởi (2) trong Định nghĩa 2.1 ta có a ∈ B, a ∈ C. Từ đó, A ⊂ B ∩ C. Như vậy, đối với A,B,C, ∈ L(M) nếu A = B ∩ C thì A = B hoặc A = C. Do đó, L(M) là hệ không giao trên R. Chúng ta biết rằng các họ không giao và các hàm đóng xác định duy nhất lẫn nhau. Mặt khác phù hợp với Nhận xét 8 và Định lí 13 mục trên ta có H là một hàm đóng trên R và L(M) là hệ sinh tối tiểu của Z(H).

Hiện ta sẽ chứng minh M(H) = M. Nếu (A, {a}) $\in$ M thì A $\in$ L(M). Giả thiết rằng đối với mỗi b $\notin$ A có B $\in$ Z(H) : A $\subset$ B, b $\notin$ B. Có thể thấy rằng A là giao của những B như thế. Điều này mâu thuẫn với A $\in$ L(M). Nếu (A, {a}) $\in$ M thì có b $\notin$ A để (A, {b}) $\in$ M(H) (**).

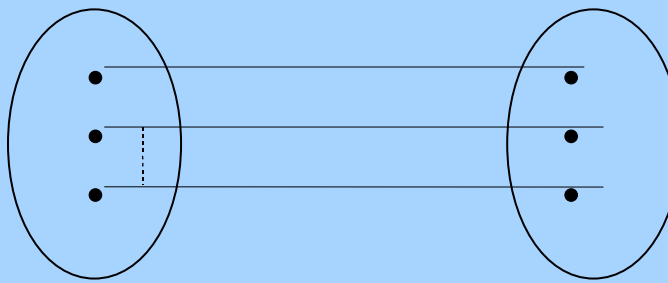
Nếu (A, {a}) $\in$ M(H) thì phù hợp với định nghĩa của M(H) B $\in$ Z(H), và A $\subset$ B kéo theo a $\in$ B. Vì a $\notin$ A, ta thấy A không là giao của các B như thế. Theo cấu trúc của H ta có A $\in$ L(M). Như vậy, nếu (A, {a}) $\in$ M(H) thì A $\in$ L(M) (***).

Bây giờ ta giả thiết là (A,{a}) $\in$ M, nhưng (A, {a}) $\notin$ M(H). Vì A là tập đóng của H,a $\notin$ A và bởi định nghĩa của M(H) thì tồn tại (B,{a}) $\in$ M(H) sao chot A $\subset$ B. Do (***) ta có B $\in$ L(M). Điều này mâu thuẫn với điều kiện (2) của Định nghĩa 7. Từ đó, ta có (A,{a}) $\in$ M(H).

Giả thiết (A,{a}) $\in$ M(H), nhưng (A,{a}) $\notin$ M. Nếu có A' $\in$ L_a để A $\subset$ A' thì bởi (**) ta có A' là một tạp đóng của H. Phù hợp với định nghĩa của M(H) ta có (A,{a}) $\notin$ M. Điều này là vô lí. Nếu A' $\subset$ A thì do (***) ta có (A',{a}) $\notin$ M, nó cũng vô lí. Nếu A $\cup$ L_a là hệ Sperner trên R thì bởi (***) ta có thể thấy nó trái với điều kiện (3) của Định nghĩa 7. Do đó tồn tại

một A' mà $A' \in L_a$ và $A = A'$. Từ đó ta có M(H) = M.

Nếu ta giả thiết rằng có H' mà M(H') = M. Đặt L(H') = {A : (A,{a}) ∈ M(H'). Theo (**) và (***) của chứng minh trên ta có L(H') là một hệ sinh nhỏ nhất của Z(H'). Do M(H') = M(H) = M ta có L(H') = L(M) = L(H). Vì các hàm đóng và các tập không giao là xã định duy nhất lẫn nhau nên H = H'. □



Lớp các hàm đóng cực đại các thuộc tính

Lớp các họ

Vì các hàm đóng và các họ các phụ thuộc hàm tương ứng xác định duy nhất lẫn mhau, từ Định lí 2.4 ta có

Hệ quả 11.

Tồn tại tương ứng 1-1 giữa lớp các họ cực đại các thuộc tính và họ các phụ thuộc hàm.

2.4. Các thuật toán liên quan đến các khoá

Khi giải quyết các bài toán thông tin quản lí, chúng ta thường sử dụng các hệ quản trị cơ sở dữ liệu mà trong đó chứa cơ sở dữ liệu quan hệ. Các phép xử lí đối với lớp bài toán này thường là tìm kiếm bản ghi sau đó thay đổi nội dung bản ghi, thêm bản ghi mới hoặc xoá bản ghi cũ. Trong các thao tác trên việc tìm kiếm bản ghi là rất quan trọng. muốn tìm được bản ghi trong một file dữ liệu chúng ta phải xây dựng khoá cho file dữ liệu đó.

Việc xây dựng khoá ở đây chính là xây dựng khoá tối tiểu. Vì thế trong mục này chúng tôi cung cấp cho bạn đọc hai thuật toán tìm khoá tối tiểu.

2.4.1 Thuật toán tìm khoá tối tiểu của một sơ đồ quan hệ

Vào : sơ đồ quan hệ s = < F, R> trong đó :

F là tập các phụ thuộc hàm và

R={ a_1 ,...., a_n } là tập các thuộc tính

Ra : K là một khoá tối tiểu

Thuật toán thực hiện như sau:

Tính liên tiếp các tập thuộc tính K_0, K_1....., K_n như sau:

$K_o = R = \{ a_1 , ..., a_n \}$

$K_I = \begin{cases} K_{i-1} \text{ nếu } K_{i-1} - \{a_i\} \rightarrow R \not\in F^+, \\ K_{i-1} - \{a_i\} \quad \text{ngược lại} \end{cases}$

...

K = K_n là khoá tối tiểu

2.4.2. Thuật toán tìm một khoá tối tiểu của một quan hệ

Cho trước r = {h_1,.... h_m} là một quan hệ trên tập thuộc tính R = { a_1 ,...., a_n }

Hệ bằng nhau của quan hệ r được định nghĩa ở phần trên như sau:

E_r ={E_{ij} : 1≤ i ≤ j ≤ m}, ở đây $E_{i\,j}$ = {a ∈ R : h_i (a) = h_j (a)}.

Dễ dàng thấy rằng E_r được tính bằng thời gian đa thức từ r

Thuật toán tìm một khoá tối tiểu của một quan hệ r:

Vào: r = {h_1,.... h_m} là một quan hệ trên tập thuộc tính R= { a_1 ,...., a_n }

Ra: K là một khoá tối tiểu của r

Thuật toán thực hiện như sau

Bước 1: Tính E_r ={ A_1 ....A_t}

Bước 2 : Tính M_r = { A : có A_j ∈ E_r : A = A_j và không có A_i : A_i∈E_r: A ⊂ A_i }

Bước 3 : Lần lượt tính các tập thuộc K_1, K_2, ...K_n tính theo qui tắc :

$K_0 = R = \{ a_1, ..., a_n \}$ hoặc K_0 = một khoá đã biết

$K_i = K_{i-1} - \{ a_i \}$ nếu không tồn tại $A \in M_r$ sao cho $K_{i-1} - \{a_i\} \subseteq A$ hoặc $K_i = K_{i-1}$ trong trường hợp ngược lại

Bước 4: Đặt $K = K_n$, khi đó K là khoá tối tiểu

2.5. Mối quan hệ giữa quan hệ Armstrong và sơ đồ quan hệ

Việc xây dựng quan hệ Armstrong của một sơ đồ quan hệ cho trước và ngược lại từ quan hệ cho trước ta xây dựng một SĐQH sao cho quan hệ cho trước này là quan hệ Armstrong của nó có vai trò rất quan trọng trong việc phân tích cấu trúc lôgic của mô hình dữ liệu quan hệ cả trong thiết kế lẫn trong ứng dụng. Đã có nhiều tác giả nghiên cứu vấn đề này. Trong mục này chúng tôi trình bày hai thuật toán giải quyết bài toán trên và đưa ra việc đánh giá các thuật toán này cũng như đánh giá độ phức tạp của bài toán trên.

Trước tiên, chúng ta cho một thuật toán tìm tập tất cả các phản khoá của hệ Spernner cho trước.

Thuật toán 1 (Tìm tập phản khoá)

Vào: K = $\{B_1,...,B_n\}$ là hệ Sperner trên R.

Ra: K^{-1}.

Bước 1: Ta đặt $K_1 = \{R - \{a\}: a \in B_1\}$. Hiển nhiên $K_1 = \{B_1\}^{-1}$.

Bước q + 1: (q < m). Ta giả thiết rằng $K_q = F_q \cup \{X_1... X_{tq}\}$, ở đây $X_{1,}....,X_{tq}$ chứa B_{q+1} và $F_q = \{ A \in K_q : B_{q+1} \subseteq A \}$. Đối với mỗi I (I = 1,..., t_q) ta tìm các phản khoá của { B_{q+1} } trên X_i tương tự như K_1. Kí pháp chúng là A^i_1,..., $A_{ri}{}^i$. Đặt

$K_{q+1} = F_q \cup \{A^i_p : A \in F_q$ kéo theo $A^i_p \not\subset A, 1 \leq i \leq t_q. 1 \leq p \leq r_i\}$

Cuối cùng ta đặt $K^{-1} = K_m$

Định lí 2.

Với mọi q ($1 \leq q \leq m$), $K_q = \{ B_{1,}.... B_q\}^{-1}$, có nghĩa là $K_m = K^{-1,,}$

Rõ ràng, K và K^{-1} là xác định duy nhất lẫn nhau và từ định nghĩa của K^{-1} có thể thấy thuật toán của chúng ta không phụ thuộc vào thứ tự của dãy $B_1,...,$ B_m. Đặt $K_q = F_q \cup \{X_1,..., X_{tq}\}$ và l_q ($1 \leq q \leq$ m-1) là số các phần tử của K_q. Khi đó ta có

Mệnh đề 3

Độ phức tạp thời gian tồi nhất của Thuật toán **2.1** là

m-1

$$O(|R|^2 \sum_{q=1} t_q.u_q).$$

ở đây $u_q = \begin{cases} l_q - t_q, & \text{nếu } l_q > t_q, \\ 1 & \text{nếu } l_q = t_q \end{cases}$

Rõ ràng trong mỗi bước thuật toán ta có K_q là hệ Sperner trên R. Ta biết rằng[5] kích thước của hệ Sperner bất kì trên R không vượt quá $C_n^{[n/2]}$, ở đây $n = |R|$. Có thể thấy $C_n^{[n/2]}$ xấp xỉ bằng $2^{n+1/2}/(\Pi . n^{1/2})$. Từ đó độ phức tạp thời gian tồi nhất của thuật toán trên không nhiều hơn hàm số mũ theo n. Trong trường hợp mà $l_q \leq l_m$ (q = 1,..., m-1), dễ thấy rằng độ phức tạp thuật toán không lớn hơn $O(|R|^2 |K| |K^{-1}|^2)$. Như vậy, trong các trường hợp này độ phức tạp của Thuật toán 1 tìm K^{-1} là đa thức theo $|R|, |K|$, and $|K^{-1}|$. Có thể thấy nếu số lượng các phần tử của K là nhỏ thì Thuật toán 1 là rất hiệu quả. Nó chỉ đòi hỏi thời gian đa thức theo |R|

Định nghĩa 4

Cho s = (R,F) là SĐQH trên R và $a \in R$. Đặt

$K_a = \{ A \subseteq R: A \rightarrow \{a\}, \nexists B: (B \rightarrow \{a\})(B \subset A)\}$. K_a được gọi là họ các tập tối tiểu của thuộc tính a.

Rõ ràng, $R \notin K_a$, $\{a\} \in K_a$ và K_a là hệ Sperner trên R.

Thuật toán 5. (Tìm tập tối tiểu của thuộc tính a)

Vào: Cho s = (R = {a_1,..., a_n}, F) là SĐQH, a = a_1.

Ra: A ∈ K_a.

Bước 1: Ta đặt L(0) = R.

Bước i + 1: Đặt

$$L(i+1) = \begin{cases} L(i) - a_{i+1} & \text{nếu } L(i) - a_{i+1} \to \{a\}, \\ L(i), & \text{ngược lại.} \end{cases}$$

Khi đó A = L(n).

Bổ đề 6.

L(n) ∈ K_a

Lời giải: Bằng phương pháp chứng minh qui nạp có thể thấy L(n) → {a}, và L(n) ⊆ ... ⊆ L(0) (1). Nếu L(n) = a, thì bởi định nghĩa của tập tối tiểu của thuộc tính a ta thu được L(n) ∈ K_a. Bây giờ ta giả thiết là tồn tại một B sao cho B ⊂ L(n) và B ≠ 0. Như vậy sẽ có a_j sao cho a_j ⊄ B, a_j ∈ L(n). Theo các xây dựng thuật toán này ta có L(j-1) - a_j → {a}. Rõ ràng bởi (1) ta thu được L(n) - a_j ⊆ L(j-1) - a_j (2). Dễ thấy B ⊆ L(n) - a_j.

Từ (1), (2) ta có B → {a}. □

Dễ thấy, vì thuật toán xác định một phụ thuộc hàm bất kì có phải là phụ thuộc hàm của một SĐQH hay không là có độ phức tạp thời gian đa thức, nên độ phức tạp của Thuật toán 5 là $O(|R|^2|F|)$.

Bổ đề 7.

Cho s = (R,F) là SĐQH trên R và $a \in R$, K_a là họ các tập tối tiểu của a, $L \subseteq K_a$, $\{a\} \in L$. Khi đó $L \subset K_a$ nếu và chỉ nếu tồn tại C, $A \rightarrow B$ sao cho $C \in L$ và $A \rightarrow B \in F$ và $\forall E \in L \Rightarrow E \subseteq A \cup (C - B)$.

Lời giải. =>: Ta giả thiết rằng $L \subset K_a$. Do đó, tồn tại $D \in K_a - L$. Bởi $\{a\} \in L$ và K_a là hệ Sperner trên R, chúng ta có thể xây dựng một tập cực đại Q sao cho $D \subseteq Q \subset U$ và $L \cup Q$ là hệ Spernner. Từ định nghĩa của K_a, chúng ta thu được $Q \rightarrow \{a\}$ (1) và $a \notin Q$ (2). Nếu $A \rightarrow B \in F$ kéo theo (A tb Q, $B \subseteq Q$) hoặc $A \subseteq Q$ thì $Q^+ = Q$. Bởi (2) ta có $Q \rightarrow \{a\}$. Điều này mâu thuẫn với (1). Do đó, tồn tại một phụ thuộc hàm $A \rightarrow B$ sao cho $A \subseteq Q$, $B \subseteq Q$. Từ cách xây dựng của Q có C sao cho $C \in L$, $A \subseteq Q$, $C - B \subseteq Q$. Hiển nhiên rằng $A \cup (C-B) \subseteq Q$.

Rõ ràng, $E \subseteq A \cup (C-B)$ đối với mọi $E \in L$.

<=: Ta giả thiết rằng có C, và $A \rightarrow B$ sao cho $C \in L$, $A \rightarrow B \in F$ và $E \subseteq A \cup (C-B)$ đối với mọi $E \in L$ (3). Bởi định nghĩa của L chúng ta thu được $A \cup$

U(C-B) $\rightarrow$ {a}. Bởi {a} thuộc L nên có D sao cho D $\in$ K_a, a $\notin$ D, D thuộc A $\cup$ (C-B). Bởi (3) ta có D $\in$ K_a - L. □

Cơ sở trên bổ đề này và thuật toán 5, chúng ta xây dựng một thuật toán sau đây bằng qui nạp.

Thuật toán 8. Tìm họ các tập cực tiểu của thuộc tính a.

Vào: Cho s = (R, F) là một sơ đồ quan hệ và a thuộc R.

Ra: K_a

Bước 1: Đặt L(1) = E_1 = {a}

Bước i + 1: Nếu có C và A $\rightarrow$ B mà C $\in$ L(i), A $\rightarrow$ B $\in$ F, $\forall$ E $\in$ L(i) $\rightarrow$ E $\notin$ A $\cup$ (C - B), thì bởi thuật toán 5 chúng ta xây dựng E_{i+1}, ở đây $E_{i+1} \subseteq$ A $\cup$ (C - B), $E_{i+1} \in K_a$. Chúng ta đặt K(i+1) = K(i) $\cup$ E_{i+1} . Trong trường hợp ngược lại ta đặt K_a = L(i).

Bởi bổ đề 7 hiển nhiên rằng tồn tại một số tự nhiên t để K_a = L(t)

Có thể thấy rằng độ phức tạp thời gian tồi nhất của thuật toán là O (| U| |F| |K_a| (| U| + ||K_a |)). Như vậy, độ phức tạp thời gian của thuật toán này là đa thức theo| U | , | F | , và| K_a | .

Rõ ràng, nếu số lượng các phần tử của K_a đối với sơ đồ quan hệ s = <R, F> là đa thức theo kích thước của s, thì thuật toán này là rất hiệu quả. Đặc biệt khi $|K_a|$ là nhỏ.

Hiển nhiên rằng nếu đối với mỗi $A \to B \in F$ kéo theo $a \in A$ hoặc $a \notin B$, thì $K_a = \{a\}$

Nhận xét 9

Biết rằng [27] nếu s = <R, F> là một sơ đồ quan hệ, $Z(F) = \{A : A^+ = A\}$ và N(F) là hệ sinh nhỏ nhất của Z(F), thì

$$N(F) = MAX(F^+) = \bigcup_{a \in R} MAX(F^+, a)$$

ở đây $MAX(F^+,a) = \{A \subseteq U : A \to \{a\} \notin F^+, A \subset B \to B \to \{a\} \in F^+$. Rõ ràng rằng, K_a là một hệ Sperner trên R. Có thể thấy MAX (F^+, a) là tập các phản khoá của K_a đối với mọi $a \in R$. Như vậy, $MAX(F^+, a) = K_a^{-1}$.

Định lý 10

Cho $r = \{h_1,...h_m\}$ là một quan hệ, và F là một họ f trên R. Khi đó $F_R = F$ nếu và chỉ nếu với mọi $A \in P(R)$

$$\bigcap E_{ij} \quad \text{nếu } \exists E_{ij} \in E_r ; A \subseteq E_{ij}$$

$L_F(A) = \begin{cases} A \subseteq E_{ij} \\ R \end{cases}$ ngược lại,

ở đây $L_F(A) = \{a \in R : (A, \{a\}) \in F\}$ và E_r là hệ bằng nhau của r.

Trên cơ sở nhận xét 9, định lý 10, các thuật toán 1, 8, chúng ta xây dựng một thuật toán tìm quan hệ Armstrong từ một sơ đồ quan hệ cho trước như sau:

Thuật toán 11 (Tìm quan hệ Armstrong)

Vào: Cho s = <R, F> là một sơ đồ quan hệ

Ra: r là quan hệ sao cho $F_r = F^+$

Bước 1: Đối với mỗi $a \in R$ bởi thuật toán 2.8 chúng ta tính K_a, và từ thuật toán 2.1 xây dựng tập các phản khoá K_a^{-1} .

Bước 2: $N = \bigcup_{a \in R} K_a^{-1}$

Bước 3: Giả sử các phần tử của N là $A_1,...,A_t$, chúng ta xây dựng quan hệ $r = \{ h_0, h_1, ..., h_0 \}$ như sau: Với mỗi $a \in R$, $h_0(a) = 0$, $\forall i = 1,...,t$

$h_i(a) = 0$ nếu $a \in A_i$, hoặc $h_i(a) = 1$ trong trường hợp ngược lại.

Do nhận xét 9 rõ ràng rằng, nếu chúng ta có N(F), thì chúng ta có thể trực tiếp xây dựng r. Độ phức tạp của việc xây dựng này phụ thuộc vào

N(F) | . Dễ thấy độ phức tạp của thuật toán 11 là độ phức tạp của bước 1. Bởi bổ đề 3 và đánh giá của thuật toán 8, dễ thấy rằng độ phức tạp tồi nhất của thuật toán 11 là

$$O\left(n \sum_{i=1}^{n} \left(\sum_{q-1}^{m_i - 1} t_{iq}\, u_{iq} + |F|\, m_i (m_i + n) \right)\right).$$

ở đây $R = \{a_1,...,a_n\}$, $m_i = |K_a,|$ and

$$u_{iq} = l_{iq} \text{ nếu } l_{iq} > t_{iq} \text{ hoặc } u_{iq} = 1 \text{ nếu } l_{iq} = t_{iq}$$

Trong trường hợp $l_{iq} \leq (\forall i, \forall q : 1 \leq q \leq m_i)$, độ phức tạp thuật toán của chúng ta là

$$O\left(n \sum_{i=1}^{n} |K_{ai}| \, (n|F| + |K_{ai}|F| + n|K_a^{-1}|^2)\right).$$

Như vậy, độ phức tạp thuật toán 2.11 là đa thức theo $|R|$, $|F|$, $|K_{ai}|$, $|K_a^{-1}|$. Rõ ràng, trong các trường hợp này nếu $|K_{ai}|$ và $|K_a^{-1}|$ là đa thức (đặc biệt nếu chúng là nhỏ) theo $|R|$ và $|F|$, thì thuật toán của chúng ta là hiệu quả.

Bây giờ chúng ta sử dụng thuật toán 11 để xây dựng quan hệ Armstrong cho sơ đồ quan hệ trong ví dụ dưới đây.

Ví dụ 13

Cho s = <R, F> là một sơ đồ quan hệ, ở đây R = {a,b,c,d} và F = {{a,d} $\rightarrow$R, {a} $\rightarrow${a,b,c},{b,d} $\rightarrow$ {b,c,d}}.

Bởi thuật toán 8, chúng ta thu được K_a = {a}.K_b = {{a},{b}},K_c ={{a},{b,d},{c}}, K_d = {d}.

Trên cơ sở thuật toán 1, ta có K_a^{-1} = {b,c,d }, K_b^{-1} = {c,d}, K_c^{-1} = {{b},{d}}, K^{-1}_d = {a,b,c}.

Do đó, N(F) = {{a,b,c }, {b,c,d}, {c,d}, {b}, {d}}. Khi đó ta xây dựng quan hệ r như sau:

a	b	c	d
0	0	0	0
0	0	0	1
2	0	0	0
3	3	0	0
4	0	4	4
5	5	5	0

Bây giờ chúng ta xây dựng một thuật toán tìm một sơ đồ quan hệ s từ một quan hệ cho trước sao cho quan hệ này là quan hệ Armstrong của s.

Thuật toán 14 (Tìm một khoá tối thiểu từ tập các phản khoá).

Vào: Cho K là một hệ Sperner, H là một hệ Sperner, và C = $\{b_1,...,b_m\} \subseteq R$ sao cho $H^{-1} = K$ và $\exists B \in K : B \subseteq C$.

Ra : $D \in H$

Bước 1: Đặt $T(0) = C$

Bước i+1: Đặt $T = T(i) - b_{i+1}$

$$T(i+1) = \begin{cases} T & \text{nếu } \forall B \in K : T \not\in B \\ T(i) & \text{ngược lại} \end{cases}$$

Cuối cùng đặt D = T(m)

Bổ đề 15. Nếu K là tập các phản khoá thì $T(m) \in H$.

Bổ đề 16. Cho H là một hệ Sperner trên R, và $H^{-1} = \{ B_1,...,B_m\}$ là tập các phản khoá của H, $T \subseteq H$ .Khi đó $T \subset H$, $T \neq \emptyset$ nếu và chỉ nếu tồn tại $B \subseteq U$ sao cho $B \in T^{-1}$, $B \not\in B_i$ ($\forall i : 1 \leq i \leq m$).

Cơ sở trên bổ đề 16 và thuật toán 14 chúng ta xây dựng thuật toán sau.

Thuật toán 17. Tìm tập các khoá tối thiểu từ tập các phản khoá.

Vào: Cho $K = \{B_1,...,B_k \}$ là một hệ Sperner trên R

Ra: H mà $H^{-1} = K$

Bước 1: Nhờ thuật toán 2.14 chúng ta tính A_1, đặt $K(1) = A_1$

Bước i+1: Nếu có $B \in K_i^{-1}$ sao cho $B \notin B_j$ ($\forall$ j: $1 \leq j \leq k$), thì bởi Thuật toán 2.14 chúng ta tính A_{i+1}, ở đây $A_{i+1} \in H$, $A_{i+1} \subseteq B$. Đặt $K(i+1) = K(i) \cup A_{i+1}$. Trong trường hợp ngược lại ta đặt H=K(i).

Mệnh đề 18. Độ phức tạp của Thuật toán 17 là

$$O(n (\sum_{q=1}^{m-1} (k l_q + n t_q u_q) + k^2 + n))$$

ở đây $|R| = n$, $|K| = k$, $|H| = m$, ý nghĩa của l_q, t_q, u_q, xem trong mệnh đề 3.

Rõ ràng , trong các trường hợp mà $l_q \leq k$ ($\forall$ q : $1 \leq q \leq m-1$) độ phức tạp thời gian của thuật toán là $O (|R|^2 |K|^2 |H|)$. Dễ thấy trong các trường hợp này thuật toán 2.17 tìm tập các khoá tối tiểu có độ phức tạp thời gian là đa thức trong kích thước của R, K, H.

Nếu $|H|$ là đa thức theo $|R|$ và $|K|$, thì thuật toán là hiệu quả. Có thể thấy rằng nếu số lượng các phần tử của H là nhỏ thì thuật toán 17 là rất hiệu quả.

Bổ đề 19.

Cho F là một họ f trên R, $a \in R$. Đặt $L_F(A) = \{a \in R: (A , \{a\}) \in F\}$, $Z_F = \{A : L_F (A) = A\}$. Rõ ràng,

$R \in Z_F$, $A,B \in Z_F \rightarrow A \cap B \in Z_F$. Kí pháp N_F là hệ sinh tối tiểu Z_F. Đặt $M_a = \{ A \in N_F : a \notin A, \exists B \in N_F: a \notin B, A \subset B\}$. Khi đó $M_a = MAX(F,a)$, ở đây $MAX(F,a) = \{A \subseteq U : A$ là một tập cực đại không rỗng mà $(A,\{a\}) \notin F\}$.

Lời giải:

Biết rằng [27] $MAX(F,a) \subseteq N_F$ (1). Giả thiết rằng $A \in M_a$. Bởi $A \in N_F$, có nghĩa là $L_F(A) = A$, và $a \notin A$, ta thu được $(A, \{a\}) \notin F$. Từ (1) và phù hợp với định nghĩa của M_a ta có $A \in MAX(F,a)$.

Ngược lại, Nếu $A \in MAX(F,a)$ thì do (1) ta có $A \in N_F$ (2). Do $(A,\{a\}]) \notin F$ và từ (2) ta thu được $a \notin A$. Phù hợp với định nghĩa của $MAX(F,a)$ ta có $A \in M_a$.□

Trên cơ sở Thuật toán 17 và Bổ đề 19, ta xây dựng thuật toán dưới đây để tìm SĐQH $s = <R,F>$ cho một quan hệ r cho trước sao cho $F^+ = F_r$.

Thuật toán 20. (Tìm SĐQH)

Vào: r là quan hệ trên R

Ra: $s = <R,F>$ mà $F^+ = F_R$

Bước1: Từ r ta tính hệ bằng nhau E_r

Bước 2: Đặt $N_r = \{ A \in E_r : A \neq \cap \{ B \in E_r : A \subset B\}\}$

Bước 3: Với mỗi $a \in R$ ta xây $N_a = \{A \in N_r : a \notin A \exists B \in N_r : a \notin B, A \in B\}$. Sau đó, bởi Thuật toán 17 ta xây họ H_a($H_a^{-1} = N_a$)

Bước 4: Xây s = <R,F>, ở đây $F = \{A \rightarrow \{a\} : \forall a \in R, A \in H_a, A \neq \{a\}\}$

Mệnh đề 21.

$F_R = F^+$

Lời giải: Vì F_R là một họ f trên R, có thể thấy $N_{F\,r} \subset E_r$, ở đây N_{Fr} là hệ sinh nhỏ nhất của Z_{Fr}. Do định nghĩa của hệ sinh nhỏ nhất ta có $N_r = N_{Fr}$. Do đó ta có $N_a = M_a$. Từ định nghĩa của tập phản khoá và định nghĩa của tập K_a ta có $H_a = K_a$. Tù đó ta thu được $F^+ \subseteq F_R$.

Ngược lại, nếu $A \rightarrow B = \{b_1,...,b_t\} \in F_R$ thì bởi việc xây dựng của F ta thu được $A \rightarrow \{b_i\} \in F^+$ với mỗi i=l,...,t. Vì không có phụ thuộc hàm tầm thường $\{a\} \rightarrow \{a\}$ trong F, dễ thấy với mọi i=1,...,t, nếu không có phụ thuộc hàm $B \rightarrow \{b_i\} \in F$, ở đây $B \subseteq U - b_i$, thì $b_i \in A$. Từ đó ta có $A \rightarrow B \in F^+$. □

Có thể thấy E_r, N_r, N_a với $a \in R$ được xây trong thời gian đa thức theo kích thước của r. Rõ ràng, việc xây dựng F phụ thuộc vào kích thước của H_a($\forall a \in R$). Do đó, độ phức tạp thời gian tồi nhất của Thuật toán 20 là

$$O\left(n \sum_{i=1}^{n} \left(\sum_{q=1}^{m_i - 1} (k_i l_{iq} + n t_{iq} u_{iq}) + k_i^2 + n \right)\right)$$

ở đây $R = \{ a_1,..., a_n \}$, $|N_{ai}| = k_i$, $|H_{ai}| = m_i$, ý nghĩa của các l_{iq}, t_{iq}, u_{iq} xem các Mệnh đề 3 và 18.

Dễ thấy, nếu $l_{iq} \leq k_i$ ($\forall$ i, $\forall$ q : $1 \leq q \leq m_i - 1$), thì độ phức tạp thời gian của thuật toán của chúng ta là

$$\mathbf{O}\left(n^2 \sum_{i=1}^{n} k_i^2 m_i\right)$$

Bởi vì k_i là đa thức theo kích thước của r, trong các trường hợp nếu is m_i là đa thức theo kích thước của r, thì thuật toán của chúng ta là hiệu quả. Lúc đó độ phức tạp của nó là đa thức theo kích thước của r. Nếu $|H_a|$ là nhỏ thì thuật toán của ta rất hiệu quả.

Bây giờ, nhờ Thuật toán 20 chúng ta xây dựng SĐQH s = <R,F> cho quan hệ sau đây.

Ví dụ 22 r là quan hệ sau đây trên R = {a, b, c, d}:

a	b	c	d
6	6	6	0
0	2	0	2
0	0	0	0
0	0	0	3
4	4	0	0
5	0	5	5
1	0	0	0

Dễ thấy là

E_R = {{a,b,c}, {b,c,d}. { a,c}, { b,c}, {c,d}.{b}, {c},{d}, ∅},

N_R = {[a,b,c}, {b,c,d},{a,c},{c,d},{b},{d}},

N_a = {b,c,d}, N_b= {{a,c},{c,d}},

N_c = {{b},{d}}, N_d ={a,b,c}

Ta có H_a = {a}, H_b = {{b}, {a,d}}, H_c = {{a}, {b,d},{c}}, H_d = {d}.

Ta xây dựng s = (R,F) như sau:

R = {a,b,c,d}, F = {{a,d} $\rightarrow${b},{a} $\rightarrow${c},{b,d} $\rightarrow${c}}.

Chúng ta trình bày hai kết quả cơ bản về độ phức tạp thuật toán cho việc xây dựng quan hệ Armstrong cho một SĐQH cho trước và ngược lại.

Định lí 23

Độ phức tạp thời gian cho việc tìm kiếm một quan hệ Armstrong của một SĐQH cho trước là hàm số mũ theo số lượng của các thuộc tính.

Định lí 24

Độ phức tạp thời gian cho việc tìm kiếm một SĐQH s = <R,F> từ một quan hệ r cho trước sao cho $F_r = F^+$ là hàm số mũ theo số lượng các thuộc tính.

Chương 3

Các dạng chuẩn và các thuật toán liên quan

Việc chuẩn hoá các quan hệ cũng như các sơ đồ quan hệ đóng một vai trò cực kì quan trọng trong việc thiết kế các hệ quản trị cơ sở dữ liệu trên mô hình dữ liệu của Codd. Nhờ có chuẩn hoá các quan hệ và các sơ đồ quan hệ chúng ta tránh được việc dư thừa dữ liệu và tăng tốc độ của các phép toán xử lí quan hệ.

3.1 Các khái niệm cơ bản

Chúng ta định nghĩa các dạng chuẩn như sau.

Cho r = $\{h_1,...,h_m\}$ là quan hệ trên R = $\{a_1 ...., a_n\}$

Định nghĩa 1. (Dạng chuẩn 1 - 1NF):

r là dạng chuẩn 1 nếu các phần tử của nó là sơ cấp.

Khái niệm sơ cấp hiểu ở đây là giá trị $h_i(a_j)$ (i=1,...,m; j=1,...,n) không phân chia được nữa.

Định nghĩa 2 (Dạng chuẩn 2 - 2NF)

r là dạng chuẩn 2 nếu:

- r là dạng chuẩn 1

- $A \rightarrow \{a\} \notin F_r$ đối với mọi khoá tối thiểu K, $A \subset K$ và a là thuộc tính thứ cấp.

Định nghĩa 3. (Dạng chuẩn 3 - 3NF):

r là dạng chuẩn 3 nếu:

$A \rightarrow \{a\} \notin F_r$ đối với A mà $A^+ \neq R$, $a \notin A$, $a \notin \cup K$

Có nghĩa rằng :

- K là một khoá tối tiểu

- a là thuộc tính thứ cấp

- A không là khoá

- $A \rightarrow \{a\}$ không đúng trong r

Định nghĩa 4. (Dạng chuẩn Boye-Codd - BCNF)

r là dạng chuẩn của Boye-Codd nếu:

$A \rightarrow \{a\} \notin F_r$ đối với A mà $A^+ \neq R$, $a \notin A$

Nhận xét 5

Qua định nghĩa, ta có thể thấy dạng chuẩn

BCNF là 3NF và 3NF là 2NF. Tuy vậy, chúng ta có thể đưa ra các ví dụ chứng tỏ có quan hệ là 2NF nhưng không là 3NF và có quan hệ là 3NF nhưng không là BCNF.

Nói cách khác là lớp các quan hệ BCNF là lớp con thực sự của lớp các quan hệ 3NF và lớp các quan hệ 3NF này lại là lớp con thực sự của lớp các quan hệ 2NF.

Đối với s = <F,R> thì các dạng chuẩn 2NF, 3NF, BCNF trong đó ta thay F_r bằng F^+.

Chú ý là đối với sơ đồ quan hệ ta không có dạng 1NF.

Nhận xét 5 cũng đúng cho các dạng chuẩn của sơ đồ quan hệ. Chúng ta xem ví dụ sau

Ví dụ 6.

Cho s = <R, F>, s' = <R, F' > là hai SĐQH trên R = {a, b, c, d} và

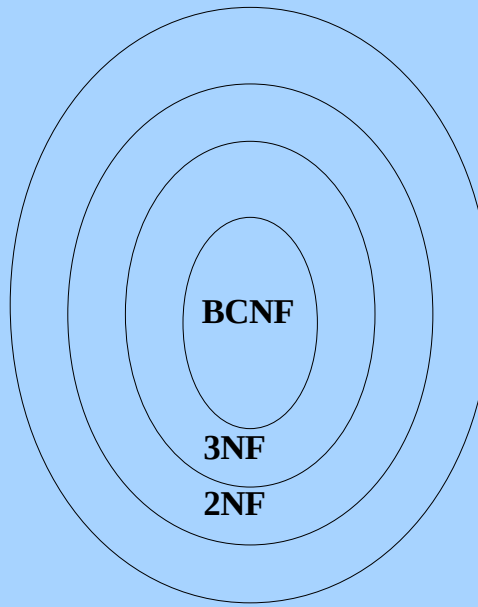
F = {{a} $\rightarrow${c}, {b} $\rightarrow${d}, {c} $\rightarrow${a, b, d}}.

F' = {{a, b} $\rightarrow${c}, {d} $\rightarrow${b}, {c} $\rightarrow${a, b, d}}.

Dễ thấy {a}, {c} là các khoá tối tiểu của s, {b} là thuộc tính thứ cấp. Do đó, s là 2NF, nhưng không là 3NF.

Rõ ràng, {a, b}, {c} là các khoá tối tiểu của s'. Hiển nhiên s là 3NF. Vì ta có {d} $\rightarrow${b}, nên s không là BCNF.

Như vậy việc phân lớp các dạng chuẩn có thể được thể hiện quan hình vẽ sau



2NF

3.2 Dạng chuẩn 2NF

Bây giờ chúng ta nêu ra loại phụ thuộc hàm đặc biệt, mà phụ thuộc dữ liệu này đóng vai trò quan trọng trong dạng chuẩn 2.

Định nghĩa 1.

Một phụ thuộc hàm $A \rightarrow B$ được gọi là sơ cấp nếu không tồn tại một tập hợp $A' \subset A$ sao cho $A' \rightarrow B$. Trong trường hợp này ta cũng nói B phụ thuộc hoàn toàn vào A. Như vậy nếu A là một thuộc tính sơ cấp thì phụ thuộc hàm $A \rightarrow B$ cũng là sơ cấp. Trong trường hợp ngược lại, ta nói B phụ thuộc bộ phận vào A

Định lí 2.

Cho r là một quan hệ trên R. Khi đó r là 2NF khi và chỉ khi

- r là 1NF
- Mỗi thuộc tính thứ cấp của r đều phụ thuộc hoàn toàn vào mọi khoá tối tiểu.

Vì SĐQH không có dạng chuẩn 1, từ Định lí 2 ta có mệnh đề sau

Mệnh đề 3.

Cho s là một sơ đồ quan hệ trên R. Khi đó s là 2NF khi và chỉ khi mọi thuộc tính thứ cấp của s đều phụ thuộc hoàn toàn vào khoá tối tiểu bất kì.

Có thể thấy, bản chất dạng chuẩn 2NF là loại bỏ các phụ thuộc bộ phận giữa các thuộc tính thứ cấp với các khoá tối tiểu.

Định lí 4.

Giả sử s = <R, F> là sơ đồ quan hệ. Đặt M_s = {A - a; a ∈ A, A ∈ K_s}, và F_n là tập tất cả các thuộc tính thứ cấp của s. Đặt l_s = {B : B = C^+ , C ∈ M_s}. Khi đó ta có các tương đương sau:

(1) s là 2NF.

(2) Với mỗi C ∈ M_s: $C^+ \cap F_n = \emptyset$;

(3) Với mỗi B ∈ l_s và a ∈ F_n : $(B - a)^+ = B - a$.

Lời giải: Giả sử s là 2NF. Nếu $F_n = \emptyset$ thì (2) là rõ ràng. Giả thiết rằng $F_n \neq \emptyset$. Do định nghĩa của F_n và của M_s, (3) là hiển nhiên.

Giả sử rằng chúng ta có (2) và $F_n \neq \emptyset$. Nếu có B ∈ l_s, và a ∈ F_n : B - a ⊂ $(B - a^+)$. Từ định nghĩa của l_s có C ∈ M_s : $C^+ = B$. Rõ ràng rằng a ∈ $(B - a)^+$. Phù hợp với định nghĩa của bao đóng ta có $(B - a)^+ = C^+ = B$. Từ đó ta thu được a ∈ C^+. Do vậy, $C^+ \cap F_n \neq \emptyset$. Điều này là vô lý. Do đó ta thu được (3).

Bây giờ, giả sử chúng ta có (3) và $F_n \neq \emptyset$. Giả thiết rằng tồn tại D ⊂ A, A ∈ K_s(*) và a ∈ F_n, a ∉ D, nhưng D → {a} ∈ F^+. Do (*) và phù hợp với việc xây

dựng của M_s và l_s thì có $C \in M_s : D \subseteq C$. Hiển nhiên rằng $a \notin C$. Rõ ràng, $D^+ \subseteq C^+$ và $a \in C^+$. Đặt $B = C^+$. Có thể thấy $C \subseteq B - a$. Do đó, $B - a \subset C^+ = (B - a)^+$. Điều này mâu thuẫn với $(B - a)^+ = B - a$. Như vậy chúng ta có (1). □

Từ định lý 4 trực tiếp suy ra kết quả sau:

Hệ quả 5.

Giả sử s = (R, F) là một sơ đồ quan hệ. Ký pháp F_n là tập tất cả những thuộc tính thứ cấp của s, và $G_s = \{B - F_n : B \in K_s^{-1}\}$. Khi đó nếu đối với mọi $C \in G_s : C^+ = C$ thì s là 2NF.

Cho s = <R, F> là một sơ đồ quan hệ trên R. Đặt $Z(s) = \{X^+ : X \subseteq R\}$. Chúng ta nói rằng s là đơn nếu F chứa chỉ các phụ thuộc hàm dạng $\{a\} \rightarrow \{b\}$. Biết rằng [16] s là đơn nếu và chỉ nếu đối với mọi $A, B \in Z(s) : A \cup B \in Z(s)$. Rõ ràng, từ điều đó ta có $(A \cup B)^+ = A^+ \cup B^+$.

Mệnh đề 6.

Cho s = <R, F> là một sơ đồ quan hệ đơn. Đặt F_n là tập tất cả những thuộc tính thứ cấp của s, và $G_s = \{B - F_n : B \in K_s^{-1}\}$. Khi đó s là 2NF nếu và chỉ nếu với mọi $C \in G_s : C^+ = C$.

Lời giải: Giả sử rằng s là 2NF. Nếu $F_n = \phi$ thì bởi định nghĩa của phản khoá ta có $C^+ = C$. Nếu $F_n \neq$

ϕ thì giả thiết rằngcó $C \in G_s : C^+ \neq C$. Bởi định nghĩa của G_s thì tồn tại $B \in K_s^{-1} : C \cup F_n = B$. Biết rằng [9] F_n là giao của tất cả các phản khoá chúng ta có $C \subset B$. Do đó, $C^+ \subseteq B$, $C^+ \cap F_n \neq \phi$. Ta ký pháp các phần tử của C là $c_1,..., c_l$. Bởi vì s là đơn chúng ta thu được $\{c_1\}^+ \cup ... \cup \{c_l\}^+ = C^+$. Do đó tồn tại $c_1 \in C$ sao cho $\{c_1\}^+ \cap F_n \neq \varnothing$. Hiển nhiên c_1 là thuộc tính cơ bản. Điều này trái với việc s có 2NF. Do đó, $C^+ = C$.

Ngược lại bởi hệ quả 5 nếu ta có $C^+ = C$ đối với mọi $C \in G_s$, thì s là 2NF. □

Cho r là một quan hệ trên R. Đặt $A^+_r = \{a : a \in R, A \rightarrow^f_r \{a\}\}$, r là quan hệ đơn nếu đối với mọi $A, B \subseteq R : (A \cup B)^+_r = A^+_r \cup B^+_r$.

Biết rằng đối với một quan hệ r cho trước thì tập hợp tất cả các phản khoá của r được xây dựng trong thời gian đa thức. Trong [9] chúng ta đã chỉ ra rằng giao của tất cả các phản khoá đúng bằng tập tất cả các thuộc tính thứ cấp. Mặt khác biết rằng [6] nếu sơ đồ quan hệ s = <R, F> là đơn thì độ phức tạp thời gian tìm quan hệ r sao cho $F^+ = F_r$ là đa thức. Từ điều này và mệnh đề 6 ta có

Mệnh đề 7.

Cho s là một sơ đồ quan hệ đơn và r là một quan hệ đơn trên R. Khi đó tồn tại một thuật toán đa thức xác định rằng s (r, tương ứng) có là 2NF.

3.3 Dạng chuẩn 3NF

Trong mục này, chúng ta đưa ra một khái niệm quan trọng thường dùng để mô tả dạng chuẩn 3NF

Định nghĩa 1.

Một phụ thuộc hàm $A \rightarrow C$ được gọi là trực tiếp nếu không có B ($B \neq A$ và $B \neq C$) sao cho $A \rightarrow B$ và $B \rightarrow C$ nhưng B không phụ thuộc hàm vào A hoặc C không phụ thuộc hàm vào B. Trong trường hợp nếu có B như vậy thì B được gọi là tập thuộc tính bắc cầu và $A \rightarrow C$ là phụ thuộc bắc cầu.

Ta có một đặc trưng sau cho dạng chuẩn 3.

Định lí 2.

Cho r là một quan hệ trên R. Khi đó r là 3NF nếu và chỉ nếu

- r là 2NF

- Không có một thuộc tính thứ cấp nào của r phụ thuộc bắc cầu vào một khoá tối tiểu.

Từ Định lí 2 đối với SĐQH ta cũng có kết quả sau

Mệnh đề 3.

Giả sử s là một sơ đồ quan hệ trên R. Khi đó s là 3NF nếu và chỉ nếu

- s là 2NF

- Mọi thuộc tính thứ cấp của s phụ thuộc trực tiếp vào mỗi khoá tối tiểu.

Trong dạng chuẩn 3NF chúng ta loại bỏ các phụ thuộc bộ phận, phụ thuộc bắc cầu giữa các thuộc tính thứ cấp với các khoá tối tiểu.

Chúng ta trình bày thêm một số đặc trưng của các SĐQH dạng 3NF.

Cho s = <R, F> là một sơ đồ quan hệ trên R. Từ s chúng ta xây dựng Z(s) = $\{X^+ : X \subseteq R\}$, và tính hệ sinh N_s của Z(s). Chúng ta đặt

$T_s = \{A \in N_s : \exists B \in N_s : A \subset B\}$

Biết rằng [1] đối với một sơ đồ quan hệ s cho trước thì tồn tại một quan hệ r là quan hệ Amstrong của s. Mặt khác bởi Định lý 17 và Hệ quả 18 mục 2.2, mệnh đề dưới đây là rõ ràng

Mệnh đề 4.

S là một SĐQH. Khi đó $K_s^{-1} = T_s$.

Định lí 5.

Cho s = <R, F> là một sơ đồ quan hệ. Đặt F_n là tập tất cả các thuộc tính thứ cấp của s. Khi đó s là

3NF nếu và chỉ nếu $\forall B \in K_s^{-1}, a \in F_n : (B - a)^+ = B - a$.

Lời giải: Giả sử $F_n \neq \emptyset$. Có thể thấy rằng s là 3NF thì đối với mỗi $B \in K_s^{-1}, a \in F_n : B - a = (B - a)^+$.

Ngược lại, nếu s không là 3NF thì tồn tại một tập A và $a \in F_n : a \notin A$ sao cho $A \rightarrow \{a\} \in F^+$ và $A^+ \neq R$. Phù hợp với Mệnh đề 4 có $B \in K^{-1}_r$ sao cho $A^+ \subseteq B$. Từ $a \in A^+$ chúng ta có $a \in B$. Do $a \notin A$ ta có $A \subseteq B - a$. Do đó ta thu được $B - a \subset (B - a)^+$. □

Định lí 6.

Giả sử r là một quan hệ trên R. Khi đó r là 3NF nếu và chỉ nếu với mọi $A \in E_r$, $a \in A$ và a là thuộc tính thứ cấp thì $\{A- a\}_r^+ = A- a$, ở đây E_r là hệ bằng nhau của r.

Từ Định lí 5 ta có hệ quả sau

Hệ quả 7.

Giả sử s là một sơ đồ quan hệ trên R. Khi đó s là 3NF nếu và chỉ nếu với mọi A : $A^+ = A$, $a \in A$ và a là thuộc tính thứ cấp thì $\{A - a\}^+ = A- a$.

3.4. Dạng chuẩn BCNF

Trong mục này, chúng ta đưa ra một số các đặc trưng của dạng chuẩn BCNF cho sơ đồ quan hệ và quan hệ.

Định nghĩa 1.

Giả sử r là một quan hệ trên R, $A,B \subseteq R$ và $A \rightarrow B$.

Khi đó ta nói A là tập sinh của B nếu

- $|A| < |B|$,

- Không tồn tại tập con thực sự của A mà xác định hàm cho B.

Tập C là tập sinh của quan hệ r nếu có một tập D nào đó để C là tập sinh của D.

Định lí 2

Giả sử r là quan hệ trên R. Khi đó r là BCNF khi và chỉ khi mọi tập sinh của r đều là khoá.

Mệnh đề 3

Cho s = <R, F> là một sơ đồ quan hệ. Đặt F_n là tập tất cả các thuộc tính thứ cấp của s. Khi đó

s là BCNF nếu và chỉ nếu $\forall B \in K_s^1, a \in B : (B - a)^+ = B - a$.

Lời giải: Dễ thấy nếu s là BCNF thì $(B - a)^+ = B - a$ đối với $B \in K_s^{-1}$ và $a \in B$.

Ngược lại giả thiết s không là BCNF. Khi đó tồn tại $A \rightarrow \{a\} \in F^+$ ở đây $A^+ \neq R$ và $a \notin A$. Bởi mệnh đề 4 mục trên, ta có $B \in K^{-1}$ sao cho $A^+ \subseteq B$. Rõ ràng, $a \in B$ và $A \subseteq B - a$. Từ đó, ta có $(B - a)^+ = B$. □

Định lí 4.

Giả sử r là một quan hệ trên R. Khi đó r là BCNF nếu và chỉ nếu với mọi $A \in M_r$, $a \in A$ thì $\{A- a\}_r^+ = A- a$, ở đây M_r là hệ bằng nhau cực đại của r.

Giả sử $A \rightarrow B$ là một phụ thuộc hàm. Chúng ta gọi phụ thuộc hàm này là tầm thường nếu $B \subseteq A$. Ngược lại trong trường hợp này, chúng ta gọi nó là phụ thuộc hàm không tầm thường.

Định lí 5

Giả sử s = < R,F> là một sơ đồ quan hệ trên R. Khi đó s là BCNF nếu và chỉ nếu với mọi $A \rightarrow B \in F$ và $A \rightarrow B$ không tầm thường thì $A^+ = R$.

3.5. Các thuật toán liên quan

Trên cơ sở các định lí đã trình bày ở các mục trên, chúng ta xây dựng các thuật toán để xác định dạng chuẩn cho các quan hệ hoặc sơ đồ quan hệ cho trước.

Đầu tiên chúng ta xây dựng thuật toán xác định một quan hệ cho trước có là 3NF hay không.

Thuật toán 1.

Đầu vào: r = {h_1, ..., h_m }là một quan hệ trên R

Đầu ra : r là 3NF ?

Bước 1: Từ r chúng ta xây dựng một tập E_r = {E_{ij} : m ≥ j > i ≥ 1}, ở đây E_{ij} = { a ∈ R : $h_j(a) = h_j(a)$}.

Bước 2: Từ E_r chúng ta xây dựng một tập M = {B ∈ P(R) : Tồn tại $E_{ij} \in E_r$: E_{ij} = B}.

Bước 3: Từ M xây dựng tập M_r = { B ∈ M : Với mọi B' ∈ M : B ⊄ B'}.

Có thể thấy rằng M_r tính được bằng một thuật toán thời gian đa thức.

Bước 4: Xây dựng tập V = ∩M_r.

Bước 5: r là 3NF nếu với mọi B ∈ M_r , a ∈ V : $\{B - a\}_r^+$ = B - a. Ngược lại r không là 3NF.

Ví dụ : Cho quan hệ r sau

A	B	C	D	E
0	0	1	0	1
1	1	0	0	1
2	2	0	1	3
1	2	3	1	0

1 1 1 0 2

Khi đó E_{12}= DE, E_{13}= ∅, E_{14}= ∅, E_{15}= D, E_{23}= C, E_{24}= A, E_{25}=AB,

E_{34}=BD, E_{35}=∅, E_{45}=A.

Như vậy ta có M_r= {DE,AB,BD,C}. Dễ thấy DE ∩ AB ∩ BD ∩ C = ∅.

Cho nên r là 3NF.

Trên cơ sở Định lí 4 mục trên chúng ta xây dựng thuật toán dưới đây

Thuật toán 2.

Đầu vào: r = {h_1, ..., h_m }là một quan hệ trên R

Đầu ra: r là BCNF ?

Bước 1: Từ r chúng ta xây dựng một tập E_r = {E_{ij} : m ≥ j > i ≥ 1} và E_{ij} = {a ∈ R : h_j(a) = h_j(a)}

Bước 2: Từ E_r chúng ta xây dựng một tập M = {B ∈ P(R) : Tồn tại E_{ij} ∈ E_r : E_{ij} = B}

Bước 3: Từ M xây dựng tập M_r = {B ∈ M : Với mọi B' ∈ M : B ⊄ B'}. Có thể thấy rằng M_r tính được bằng một thuật toán thời gian đa thức.

Bước 4: r là BCNF nếu với mọi $B \in M_r$, $a \in B : \{B - a\}_r^+ = B - a$. Ngược lại r không là BCNF.

Ví dụ : Cho quan hệ r :

A	B	C	D	E
2	0	1	1	1
1	1	0	0	1
2	2	0	3	3
1	2	3	3	0
1	1	1	0	2

Khi đó E_{12}= {E}, E_{13}= {A}, E_{14}=∅ , E_{15}= {C}, E_{23}= {C}, E_{24}= {A}, E_{25}={A,B}, E_{34}={B,D}, E_{35}=∅, E_{45}={A}.

Như vậy ta có M_r= {{A,B},{B,D},{C},{E}}. Có thể kiểm tra rằng {A,D} - A = D và $\{D\}^+_r$ = {B,D}. Vì thế r không là BCNF.

Nhờ Định lí thuật toán dưới đây được xây dựng

Thuật toán 3.

Đầu vào: s = <R,F> là một sơ đồ quan hệ trên R, với

$F = \{ A_1 \rightarrow B_1, ..., A_m \rightarrow B_m \}$

Đầu ra: s là BCNF ?

Bước 1: Nếu $A_1 \rightarrow B_1$ là phụ thuộc hàm không tầm thường và A_1^+ # R thì dừng và kết luận s không là BCNF. Ngược lại thì chuyển sang bước tiếp theo.

.....

Bước m: Giống như bước 1 nhưng đối với $A_m \rightarrow B_m$.

Bước m+1: s là BCNF.

Ví dụ : Cho sơ đồ quan hệ s =<R,F>

R = {a,b,c,d,e}

F={{a,b}→{d},{b,c}→{e}, {d}→{c}}

Ta có $\{a,b\}^+ = R$, nhưng $\{b,c\}^+ \neq R$. Vậy s không là BCNF.

Vì thời gian tính bao đóng của một tập thuộc tính bất kì của một sơ đồ quan hệ hoặc một quan hệ là đa thức. Cho nên chúng ta có các kết luận sau.

Định lí 4.

Cho trước một quan hệ r và một sơ đồ quan hệ s. Khi đó đều tồn tại một thuật toán có độ phức tạp thời gian đa thức theo kích thước của r (s) để kiểm tra r (s) có là BCNF hay không.

Định lí 5

Cho trước r là một quan hệ trên R. Khi đó tồn tại một thuật toán có độ phức tạp thời gian đa thức để kiểm tra r có là 3NF hay không.

Tuy vậy, đối với đầu vào là s thì đây lại là bài toán NP đầy đủ.

Định lí 6

Cho trước s là một sơ đồ quan hệ trên R. Khi đó bài toán xác định s có là 3NF hay không là NP - đầy đủ.

Có nghĩa là cho đến nay, độ phức tạp thời gian của bài toán này không là đa thức.

- Với trường hợp 2NF, các câu hỏi tương tự cho cả r lẫn s còn là bài toán mở (Chúng tôi phỏng đoán có độ phức tạp thời gian là hàm mũ trở lên)

3.6 Dạng chuẩn của các hệ khoá

Bây giờ chúng ta khảo sát các sơ đồ quan hệ mà tập các khoá tối thiểu của nó là một hệ Sperner cho trước.

Định nghĩa 1.

Cho K là hệ Sperner trên R. Ta nó rằng K là 2NF (3NF, BCNF, tương ứng) nếu với mỗi sơ đồ quan hệ s = <R, F> mà K_s = K thì s là 2NF (3NF, BCNF tương ứng).

Bây giờ chúng ta cho một điều kiện cần và đủ để một hệ Sperner bất kỳ là 2NF.

Cho K là một hệ Sperner trên R. Đặt K_p= {a ∈ R : ∃A ∈ K : a ∈ A}, và K_n = R - K_p. K_p (K_n) được gọi là tập các thuộc tính cơ bản (thứ cấp) của K.

Cho trước sơ đồ quan hệ s = <R, F>, chúng ta nói rằng phụ thuộc hàm A →B ∈ F là thừa nếu hoặc A = B hoặc có C →D ∈ F sao cho C ⊆ A và B ⊆ D.

Định lí 2.

Cho K là hệ Sperner trên R. Khi đó K là 2NF nếu và chỉ nếu K_n = ∅.

Lời giải: Theo định nghĩa của quan hệ 2NF, hệ Sperner 2NF và K_n ta có thể thấy nếu K_n = ∅ thì K là 2NF.

Bây giờ ta giả thiết là K là 2NF. Kí pháp K^{-1} là tập các phản khoá của K. Từ K, K^{-1} ta xây một SĐQH như sau

Đối với mỗi A ⊂ R có B ∈ K^{-1} sao cho A ⊆ B . Đặt C = ∩ {B ∈ K^{-1} :A ⊆ B}. Ta lập A →C . Kí pháp T là tập tất cả các phụ thuộc hàm như thế. Đặt F = { E →R : E ∈ K} ∪ (T - Q) , ở đây Q = {X →Y ∈ T : X →Y là phụ thuộc hàm thừa}. Từ Định lí 13 phần 2.2 và định nghĩa của hệ Sperner ta thu được K_s = K .

Rõ ràng, với mỗi SĐQH bất kì s_1 = (R, F_1) sao cho K_{s1} = K và A ⊆ R ta có $A^+_{s1} ⊆ A^+_s$, ở đây A^+_{s1} = {a: A →{a} ∈ F_1^+}. Chúng ta đã chứng minh rằng [9]

K_n là giao của các phản khoá của s. Trên cơ sở việc xây dựng s = <R,F> và phù hợp với định nghĩa của hệ Sperner 2NF ta có $K_n = \emptyset$. □

Dễ thấy SĐQH 3NF là 2NF và nếu tập các thuộc tính thứ cấp là rỗng thì SĐQH này 3NF. Do đó từ Định lí 2 ta suy ra ngay hệ quả sau

Hệ quả 3.

Cho K là hệ Sperner trên R. Khi đó K là 3NF nếu và chỉ nếu $K_n=\emptyset$.

Định nghĩa 4.

Cho K là hệ Sperner trên R. Ta nói K là đơn nhất nếu K xác định duy nhất SĐQH s = <R,F>, theo nghĩa đối với mọi SĐQH s' =<R,F '> mà $K_{s'}$ = K thì ta có $F^+ = F'^+$.

Từ định nghĩa hệ Sperner BCNF và Định nghĩa 4 ta có

Mệnh đề 5.

K là BCNF nếu và chỉ nếu K là đơn nhất.

Như đã biết [5] đối với hệ Sperner cho trước K tồn tại SĐQH s (tương ứng quan hệ r) sao cho K_s =K (tương ứng K_r =K). Ta nói s (tương ứng r) là đơn nhất nếu K_s (tương ứng K_r) xác định duy nhất s

(tương ứng r). Có nghĩa là K_s (tương ứng K_r) là đơn nhất.

Bây giờ ta cho một điều kiện cần và đủ để một SĐQH là đơn nhất.

Định lí 6.

Cho s = (R,F) là SĐQH trên R. Khi đó s là đơn nhất nếu và chỉ nếu đối với mọi a ∈ A, A ∈ $K^{-1}{}_s$: A-a = ∩ {B ∈ $K^{-1}{}_s$: (A- a) ⊂ B}

Lời giải:

Chúng ta biết rằng hệ Sperner K là đơn nhất khi và chỉ khi đối với mọi B ⊆ A,A ∈ K^{-1}, B là giao của các phản khoá. Đặt P_s = {A-a: ∈ $K^{-1}{}_s$, a ∈ A}

Có thể thấy nếu s=<R,F> là đơn nhất thì B ∈ P_s kéo theo B là giao của các phản khoá. Có nghĩa là B = ∩ {A ∈ K_s^{-1}:B ⊂ A}.

Ngược lại giả sử với mỗi B ∈ P_s ta có B = ∩ {A ∈ K_s^{-1}:B ⊆ A}(*).. Do mệnh đề 3 mục 3.4 và Mệnh đề 4 mục 3.3 ta có N_s ⊆ (P_s ∪ K_s^{-1}). Có thể thấy s là BCNF. Trên cơ sở định nghĩa của N_s và Mệnh đề 4 mục 3.3 ta có K_s^{-1} ⊆ N_s. Phù hợp với (*) ta thu được K_s^{-1} = N_s . Vì s là BCNF nên đối với mọi B ⊆ A, A ∈ K_s^{-1} : B^+ = B. Như vậy B là giao của các phản khoá của s. □

Theo định nghĩa của hệ Sperner BCNF, Định lí 6 và Mệnh đề 5 ta cho một điều kiện cần và đủ để một hệ Sperner là BCNF.

Địnhlí 7.

Giả sử K là hệ Sperner trên R. Khi đó K là BCNF nếu và chỉ nếu đối với mọi $a \in A$, $A \in K^{-1} : A - a = \cap \{B \in K^{-1} : (A - a) \subset B\}$.

Phù hợp với Định lí 6 và thuật toán đa thức tìm tập các phản khoá của một quan hệ cho trước, ta có mệnh đề sau

Mệnh đề 8.

Tồn tại thuật toán xác định một quan hệ cho trước có là đơn nhất hay không. Độ phức tạp thời gian của thuật toán này là đa thức theo kích thước của R và r.

Từ Định lí 7 và Mệnh đề 8 trực tiếp kéo theo mệnh đề sau

Mệnh đề 9.

Tồn tại thuật toán đa thức xác định tập các khoá tối tiểu của một quan hệ cho trước là BCNF hay không.

Từ Định lí 6 suy ra ngay hệ quả sau

Hệ quả 10.

Cho K là hệ Sperner trên R . Khi đó có thuật toán đa thức xác định hệ Sperner H có là đơn nhất hay không, ở đây $H^{-1} = K$.

3.7. Ví dụ

Dưới đây chúng ta cho ví dụ minh hoạ việc phân tách một bảng (quan hệ) thành các bảng ở dạng chuẩn 3NF.

Trong một nhà máy, hàng ngày người ta xuất vật tư theo phiếu xuất kho như sau:

Phiếu xuất kho

Số phiếu	Ngày xuất	Người nhận	Địa chỉ người nhận	Mã vật tư
1010001	26/10/96	Phạm An	2 Phố Huế, Hà Nội	10100 20018 10703
1020004	12/01/97	Trần Hà	14 Lê Lợi, TP. HCM	30101
1170003	17/03/97	Trần Hà	14 Lê Lợi, TP. HCM	10100 20904

Trong ví dụ này có hai thuộc tính không sơ cấp. Đó là :

- 'Địa chỉ người nhận' là một thuộc tính tổng hợp những thuộc tính sơ cấp sau: 'Số và phố', 'Tên TP'.

- 'Mã vật tư' là danh sách các vật tư của một hoá đơn, có chiều dài không nhất định, cần được tách riêng ra từng vật tư.

Ta có thể biến đổi quan hệ phiếu xuất kho sang dạng chuẩn 1 như sau

Phiếu xuất kho

Số phiếu	Ngày xuất	Người nhận	Số và phố	Thành phố	Mã vật tư
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội	10100
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội	20018
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội	10703
1020004	12/01/97	Trần Hà	14 Lê Lợi	TPHCM	30101
1170003	17/03/97	Trần Hà	14 Lê Lợi	TPHCM	10100
1170003	17/03/97	Trần Hà	14 Lê Lợi	TPHCM	20904

Trong quan hệ Phiếu xuất kho ta nhận thấy là tập {Số phiếu, Mã vật tư} là khoá tối tiểu. Dễ thấy các thuộc tính Ngày xuất, Người nhận, Số và phố, Thành phố phụ thuộc hàm vào thuộc tính số phiếu. Như vậy, quan hệ Phiếu xuất kho không là 2NF (Nếu lưu trữ và xử lí trên quan hệ này sẽ dẫn đến trùng lặp dữ liệu). Do đó, ta tách thành 2 quan hệ riêng biệt:

Phiếu kho

Số phiếu	Ngày xuất	Người nhận	Số và phố	Thành phố
1010001	26/10/96	Phạm An	2 Phố Huế	Hà Nội
1020004	12/01/97	Trần Hà	14 Lê Lợi	TPHCM
1170003	17/03/97	Trần Hà	14 Lê Lợi	TPHCM

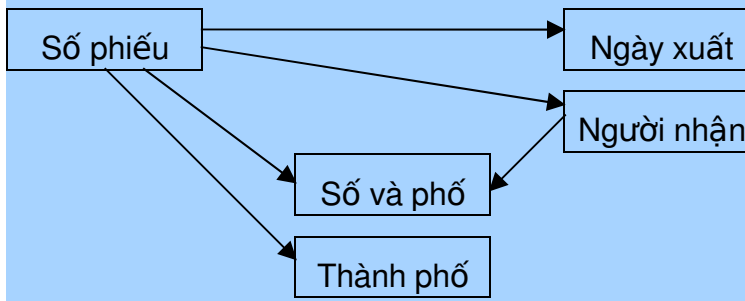
Vật tư

Số phiếu	Mã vật tư
1010001	10100
1010001	20018
1010001	10703

1020004	30101
1170003	10100
1170003	20904

Ta có thể thấy quan hệ vật tư là 3NF.

Trong quan hệ Phiếu kho là 2NF ở trên, ta thấy trên đồ thị của các phụ thuộc hàm có hai con đường để đi từ Số phiếu đến {Số và phố, Thành phố} hoặc đi qua thuộc tính Người nhận. Như vậy tồn tại phụ thuộc bắc cầu trong quan hệ này.



Điều này chứng tỏ quan hệ này chưa là 3NF, Nếu ta lưu quan hệ này thì khi xử lí sẽ dẫn đến trùng

lặp địa chỉ của người nhận. Do vậy ta tách nó thành hai thực thể riêng biệt:

Phiếu

Số phiếu	Ngày xuất	Người nhận
1010001	26/10/96	Phạm An
1020004	12/01/97	Trần Hà
1170003	17/03/97	Trần Hà

Người nhận

Người nhận	Số và phố	Thành phố
Phạm An	2 Phố Huế	Hà Nội
Trần Hà	14 Lê Lợi	TP. HCM

Như vậy, ta đã tách quan hệ phiếu xuất kho thành 3 quan hệ dạng chuẩn 3 là phiếu, người nhận, vật tư.

Chương 4
Một số phép toán xử lí bảng

Ngôn ngữ xử lí dữ liệu là một phần quan trọng trong các hệ quản trị cơ sở dữ liệu. Ngay từ năm 1970 E.F.Codd đã đưa ra hai ngôn ngữ xử lí dữ liệu chính. Đó là ngôn ngữ đại số quan hệ và ngôn ngữ tính toán quan hệ (Chủ yếu dựa vào phép toán tân từ). Hầu hết ngôn ngữ xử lí của các hệ quản trị cơ sở dữ liệu lớn hiện nay đều chứa ngôn ngữ đại số quan hệ. Trong chương này chúng tôi trình bày các phép toán cơ bản của ngôn ngữ đại số quan hệ này.

Trong Giáo trình này chúng ta coi bảng, file dữ liệu, quan hệ (theo dạng Codd) là tương đương nhau.

4.1 Các phép toán cơ bản

Để minh hoạ bằng các ví dụ làm sáng tỏ tính chất của các phép toán xử lí bảng chúng ta cho 2 quan hệ sau:

A	B	C
a	a	b
a	c	b
b	c	d
a	a	d

Quan hệ r

D	A	E
a	a	b
b	c	d
e	f	g

Quan hệ t

Hình 1

1. Phép hợp (r ∪ t)

Giả sử r và t là các quan hệ n cột

Khi đó quan hệ hợp r ∪ t là quan hệ n cột bao gồm các bản ghi (dòng) của cả r lẫn t.

Chú ý những bản ghi giống nhau chỉ giữ lại một.

Nếu r và t là các quan hệ có tên các cột khác nhau thì quan hệ hợp không có tên các cột

Với r, t trong hình 1, ta có r ∪ t =

a	a	b
a	c	b
b	c	d
a	a	d
e	f	g

2. Phép trừ (r - t)

Giả sử r và t là hai quan hệ n cột.

Quan hệ hiệu (kí hiệu là r-t) là quan hệ n cột mà bao gồm các dòng của r nhưng không có mặt trong t

Nếu r và t có các tên cột khác nhau, thì quan hệ hiệu không có tên các cột.

Ví dụ: r - t =

a c b

a a d

3. Phép giao (r ∩ t)

Giả sử t và t là hai quan hệ n cột.

Khi đó quan hệ giao của r và t là quan hệ n cột bao gồm các bản ghi có mặt cả ở trong r lẫn t.

Trong trường hợp nếu r và t có các tên cột khác nhau (tên các thuộc tính) thì các cột của quan hệ giao không có tên.

Ví dụ: r ∩ t =

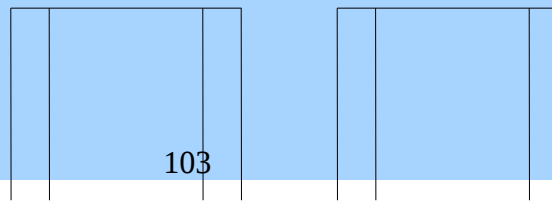
a a b

b c d

4. Tích Đề các

Giả sử có quan hệ r có n cột (R_1= {a_1,...,a_n}) và t có m cột (R_2= {b_1,...,b_m})

a_1 .....a_n b_1 .....b_m



r = ... t = ...

Tích Đề các:

r x t nếu $(r_1.....r_n) \in r$ và

$(t_1.....t_m) \in t$

Thì $(r_1,....,r_n, t_1,....., t_m) \in rxt$

Ví dụ:

A	B	C
a	a	b
a	c	b
b	c	d
a	a	d

r =

D	A	E
a	a	b
b	c	d
e	f	g

t =

Tích Đề các:

r x t =

r.A	B	C	D	t.A	E
a	a	b	a	a	b
a	a	b	b	c	d
a	a	b	e	f	g
a	c	b	a	a	b

a	c	b	b	c	d
a	c	b	e	f	g
b	c	d	a	a	b
b	c	d	b	c	d
b	c	d	e	f	g
a	a	d	a	a	b
a	a	d	b	c	d
a	a	d	e	f	g

Tích Đề các là một quan hệ. Trong trường hợp có cột giống nhau (tên giống nhau) cần đánh dấu để phân biệt. Trong ví dụ trên đó là r.A và t.A

Số lượng bản ghi (dòng) là n.m.

5. Phép chiếu

Giả sử có r là một quan hệ gồm m cột ($R_1 = \{a_1...a_m\}$). Khi ấy phép chiếu (Ký hiệu là: Π) lên tập r:

$\Pi_{i1,i2...ip}$ (r)

ij là số thứ tự lấy trong tập từ 1 đến m

j = 1,...,p, ở đây chỉ số $p \leq m$.

hoặc $\Pi_{a,b,...,\ t}$ (r), ở đây a, b, ... t là tên các thuộc tính.

Khi đó ta thực hiện phép chiếu như sau:

Giữ lại p cột có số hiệu là i_1, i_2, i_p, loại bỏ các dòng bị trùng nhau.

Ví dụ: (lấy ví dụ trước).

$\Pi_{1,2}$ (r) =

A	B
a	a
a	c
b	c

6. Phép chọn

Phép toán này rút ra các bản ghi thoả mãn điều kiện nào đấy.

Gọi F là một điều kiện nếu nó bao gồm:

- Các toán hạng (hằng, tên thuộc tính)

- Các quan hệ số học (<, =, >, ≤ , ≥ , #)

- Các phép toán logic: và ($\wedge$), hoặc ($\vee$) và phủ định. Riêng với hằng ta bao bằng dấu ' '

Ví dụ: F là điều kiện A = 'a' v C = 'b'

Ký hiệu phép chọn: δ_F (r) (F: điều kiện).

Khi đó δ_F là phép chọn được thực hiện bằng việc rút ra từ r (và loại bỏ các dòng trùng) các bản ghi thoả mãn F.

Ví dụ:

$\delta_{A = 'a' \vee C = 'b'}(r)$ =

A	B	C
a	a	b
a	c	b
a	a	d

4.2 Các phép toán khác

Các phép toán trình bày tiếp theo đây được biểu diễn qua các phép toán ở mục trên.

1. Phép chia (r ÷ s):

Giả sử r là quan hệ n cột, t là quan hệ m cột

ở đây n > m và t ≠ ∅. Quan hệ thương của r và t là quan hệ được tạo ra như sau:

- Tích $A_1 = \prod_{1,2,..., n-m}(r)$
- Tích $A_2 = A_1 \times t$
- Tích $A_3 = A_2 - r$
- Tích $A_4 = \prod_{1, 2,...., n-m}(A_3)$

Cuối cùng $r \div s = \prod_{1, 2,...., n-m}(r) - A_4$

Như vậy chúng ta có:

$r \div s = \prod_{1, 2,...., n-m}(r) - \prod_{1, 2,...., n-m}((\prod_{1, 2,...., n-m}(r) \times t) - r)$

Ví dụ:

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	c
b	b	b	b

Quan hệ r

c	d
e	f

Quan hệ t

a	b
e	d

Quan hệ r ÷ t

2. Phép nối θ

Giả sử r là quan hệ n cột, t là quan hệ m cột, θ là một trong các quan hệ số học: < , ≤ , = , # , > , ≥

i và j là tên hoặc là số cột tương ứng của r và t

Khi đó kết quả của phép nối θ hai quan hệ r và t là quan hệ (n + m) cột bao gồm các bản ghi của tích đề các r x t, mà các bản ghi này thoả mãn quan hệ số học θ giữa các giá trị thuộc tính i và j.

Ký pháp nó là r >< t

θ

Có thể thấy r >< t = $\delta_{i\,\theta\,(n+j)}$ (r x t)

iθj

Với ví dụ như ở mục trên ta có:

r.A	B	C	D	t.A	E

a	a	b	a	a	b
a	c	b	a	a	b
a	a	d	a	a	b
b	c	d	b	c	d

r >< t

r.A=D

3. Phép nối

Giả sử có r là một quan hệ n cột (R_1 = {a_1...a_n}), r là quan hệ m cột (R_2 = {b_1,...,b_m}). Khi đó chúng ta ký hiệu r ⋈ t là phép nối.

- Nó được thực hiện bởi biểu thức trên cơ sở các phép toán sau:

r ⋈ t = ∏ i_1, i_2...i_{n+m-q} (δr.A_1 = t.A_1 ∧... ∧r.A_q = t.A_q (r x t))

A_1,...,A_q là q cột có tên trùng nhau ở hai quan hệ r và t.

- Phép toán được thực hiện:

+ Tích Đề các r và t.

+ Chọn trong tích Đề các những bản ghi thoả mãn những cột giống tên nhau phải trùng giá trị trên p cột.

+ Phép chiếu loại bỏ p cột giống nhau (Chỉ giữ lại một).

Ví dụ như hai quan hệ trong hình 1 của phần này, chúng ta có thể thấy r ⋈ t là bảng sau.

A	B	C	D	E
a	a	b	a	b
a	c	b	a	b
a	a	d	a	b

Phép nối là một phép xử lý bảng rất quan trọng. Thông thường chúng ta phân tách file dữ liệu lớn ban đầu thành các file dữ liệu nhỏ ở dạng chuẩn 3NF. Nhờ phép nối các file dữ liệu nhỏ với nhau chúng ta có thể phục hồi file dữ liệu lớn ban đầu và dung tích bộ nhớ để lưu trữ các file dữ liệu nhỏ thường nhỏ hơn dung tích dùng để lưu trữ file dữ liệu lớn ban

đầu. Như vậy, nhờ phép nối chúng ta tiết kiệm được việc lưu trữ các bảng.

4.3. Các ví dụ

Bây giờ chúng ta đưa ra ví dụ minh hoạ việc sử dụng các phép toán xử lý bảng

Một cửa hàng tổng hợp mỗi ngày có bản tổng kết bán hàng như sau:

1. Bản tổng kết bán hàng một ngày từ những hoá đơn bán ra

Đơn vị tính 1000đ

Ngày tháng	Mã hàng	Tên hàng	Đơn vị tính	Số lượng
210397	M1	Radio	1 000	1
	M3	TV	4 000	2
	M6	Xe đạp	1 000	1
			Tổng giá tiền: 10 000	
			Đã thanh toán: 6 000	

ở đây 210397 là ngày bán: Ngày 21 tháng 03 năm 1997. Trên cơ sở của bản tổng kết này chúng ta xây

dựng một quan hệ (bảng) bán hàng như sau gồm 7 cột và cho các số liệu cụ thể.

Bán hàng

Ngày tháng	Mã hàng	Tên hàng	Đơn giá	Số lượng	Tổng	Thanh toán
210397	M1	Radio	1 000	1	10 000	6 000
210397	M3	TV	4 000	2	10 000	6 000
210397	M6	Xe đạp	1 000	1	10 000	6 000
220397	M2	Máy giặt	3 000	2	6 000	2 000
230397	M1	Radio	1 000	3	15 000	11 000
230397	M4	Video	5 000	2	15 000	11 000

230397	M9	Máy ảnh	2 000	1	15 000	11 000

Có thể thấy quan hệ bán hàng có khoá tối tiểu là Ngày tháng, Mã hàng.

2. Chúng ta sẽ tách từ bảng bán hàng thành 4 bảng sau:

khối lượng

Ngày tháng	Mã hàng	Số lượng
210397	M1	1
210397	M3	2
210397	M6	1
220397	M2	2
230397	M1	3
230397	M4	2
230397	M9	1

Khoá tối tiểu của quan hệ Khối lượng là Ngày tháng, Mã hàng

Doanh số

Ngày tháng	Tổng	Thanh toán

210397	10000	6000
220397	6000	2000
230397	15000	11000

Khoá tối tiểu là Ngày tháng

Hàng

Mã hàng	Tên hàng
M1	Radio
M2	Máy giặt
M3	TV
M4	Video
M6	Xe đạp
M9	Máy ảnh

Khoá tối tiểu là Mã hàng

Mặt hàng

Tên hàng	Đơn giá
Radio	1000
Máy giặt	3000

TV	4000
Video	5000
Xe đạp	1000
Máy ảnh	2000

Khoá tối tiểu là Tên hàng

3. Có thể thấy (Nếu không kể đến thứ tự của cột và hàng):

bán hàng = khối lượng >< doanh số >< Hàng ⋈ Mặt hàng

Không khó khăn lắm chúng ta thấy 4 quan hệ khối lượng, doanh số, Hàng, mặt hàng là 3NF, còn quan hệ bán hàng chưa được chuẩn hoá. Để thực hiện việc xử lí thông tin, chúng ta lưu trữ 4 quan hệ đã được chuẩn hóa, chứ không lưu trữ quan hệ bán hàng.

Như vậy nhờ phép nối chúng ta có thể hồi phục được quan hệ Bán Hàng

Bây giờ chúng ta xử dụng các phép toán xử lí bảng để tìm kiếm và in ra các thông tin sau

- Chúng ta muốn biết doanh số bán ra sau ngày 21 tháng 03 năm 1997 đó là

$\Pi_{\text{Ngày tháng,Tổng}}(\delta_{\text{Ngày tháng' > '210397'}}(\text{doanh số}))$ và in ra bảng sau:

Ngày tháng	Tổng
220397	6 000
230397	15 000

- Chúng ta muốn biết các tên hàng và số lượng đã bán trong ngày 21 tháng 03 năm 1997.

$\Pi_{\text{Tên hàng, số lượng}} (\delta_{\text{Ngày tháng='210397'}} (\text{Khối lượng}) \bowtie \text{Hàng})$

Tên hàng	Số lượng
Radio	1
TV	2
Xe đạp	1

- Tìm các ngày mà trong các ngày đó doanh số bán ra ít nhất là 10.000.000đ

$\Pi_{\text{Ngày tháng}} (\delta_{\text{Tổng}\geq\text{'10 000'}} (\text{doanh số}))$

Ngày tháng
210197
230397

- In ra các mã và tên hàng mà đơn giá của nó nhỏ hơn 3.000.000đ

$$\Pi_{\text{Tên hàng, Mã hàng}} (\delta_{\text{Đơn giá'<'3 000'}} (\text{Mặt hàng})).$$

Tên hàng	Mã hàng
Radio	M1
Xe đạp	M6
Máy ảnh	M9

- Cho các mã hàng và đơn giá của chúng trong ngày 23 tháng 03 năm 1997.

$\Pi_{\text{Mã hàng}} (\delta_{\text{Ngày tháng = '230397'}} (\text{khối lượng})) >< \Pi_{\text{Mã hàng, Đơn giá}} (\text{Hàng} \bowtie \text{Mặt Hàng})$.

Mã hàng	Đơn giá
M1	1000
M4	5000
M9	2000

- Tìm tên, đơn giá của mã hàng M1 và số lượng bán ra của mặt hàng này trong ngày 23 tháng 03 năm 1997

$\Pi_{\text{Tên hàng, Đơn giá, Số lượng}}(\Pi_{\text{Mã hàng, Số lượng}}(\delta_{\text{Ngày tháng = '230397' } \wedge \text{Mã hàng = M1}}(\text{khối lượng}) >< \text{hàng} \bowtie \text{Mặt hàng}))$.

Tên hàng	Đơn giá	Số lượng
Radio	1000	3

Ví dụ: Cho 2 quan hệ

r_1 =

Chuyến bay	Máy bay
83	727
83	747
84	727
84	747
109	707

r_2 =

Phi công	Máy bay

Tuấn	707
Tuấn	727
Thành	747
Thắng	727
Thắng	747

Chúng ta cần in ra một bảng chỉ ra các phi công có thể lái cho mỗi chuyến bay. Khi đó chúng ta chỉ cần thực hiện phép nối tự nhiên giữa r_1 và r_2.

$r_3 = r_1 \bowtie r_2$

Chuyến bay	Máy bay	Phi công
83	727	Tuấn
83	727	Thắng
83	747	Thành
83	747	Thắng

84	727	Tuấn
84	727	Thắng
84	747	Thành
84	747	Thắng
109	707	Tuấn

Đơn giản hơn ta thực hiện

$\Pi_{\text{Chuyến bay, Phi công}}(r_3)$

Chuyến bay	Phi công
83	Tuấn
83	Thắng
83	Thành
84	Tuấn
84	Thắng
84	Thành

109	Tuấn

Chương 5
Một số áp dụng mô hình dữ liệu trong các hệ quản trị CSDL hiện có

5.1. Mô tả chung

Chương này mô tả việc áp dụng các khái niệm của chương 2, 3 trong các hệ QTCSDL hiện có trên thị trường. Các khái niệm này bao gồm thực thể,

thuộc tính, khoá , quan hệ, phụ thuộc hàm, các dạng chuẩn.

5.2. Những khái niệm cơ bản

Trong phần này chúng tôi nêu lại một vài khái niệm đã được trình bày sơ bộ ở chương 2.

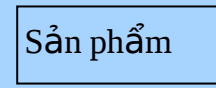
5.2.1. Thực thể

Thực thể là một hình ảnh tượng trưng cho một đối tượng cụ thể hay một khái niệm trừu tượng nhưng có mặt trong thế giới thực.

Ví dụ:

Dự án, con người, sản phẩm, ...

Thông thường khi xây dựng mô hình dữ liệu các thực thể được biểu diễn bằng những hình chữ nhật ví dụ như



5.2.2. Thuộc tính

Trong một hệ thông tin, ta cần lựa chọn một số tính chất đặc trưng để diễn tả một thực thể, các tính chất này được gọi là thuộc tính của thực thể được mô tả và đây cũng chính là các loại thông tin dữ liệu cần quản lí.

Ví dụ:

Họ tên, địa chỉ, ngày sinh của thực thể 'sinh viên'

Nhãn hiệu, giá của thực thể 'sản phẩm'

Giá trị các thuộc tính của một thực thể cho phép diễn tả một trường hợp cụ thể của thực thể, gọi là một thể hiện của thực thể đó .

Ví dụ:

('Trần Văn Sơn', '204 Triệu Việt Vương - Hà Nội', 12-5-1975) là một thể hiện của 'sinh viên'

('Máy vi tính ACER', 1349) là một thể hiện của 'sản phẩm'

Một thuộc tính là sơ cấp khi ta không cần phân tích nó thành nhiều thuộc tính khác, tuỳ theo nhu cầu xử lí trong hệ thông tin đối với một thực thể.

Thông thường một thực thể tương ứng với một bảng (hay một quan hệ của Codd).

Mỗi thực thể phải có ít nhất một thuộc tính mà mỗi giá trị của nó vừa đủ cho phép nhận diện một cách duy nhất một thể hiện của thực thể, gọi là thuộc tính nhận dạng hay là khoá. Có nhiều trường hợp chúng ta phải dùng một tập các thuộc tính để nhận diện thực thể. Khi một thực thể có nhiều khoá, người ta chọn một trong số đó làm khoá chính (khóa

tối tiểu). Giá trị của một khoá luôn luôn được xác định.

Ví dụ:

Số hoá đơn là thuộc tính nhận dạng của thực thể "Hoá đơn".

Không thể có hai hay nhiều hoá đơn có cùng số hoá đơn trong cùng một hệ thông tin.

Ví dụ:

Hoá Đơn
Số Hoá Đơn
Khách Hàng
Giá Tiền

5.2.3. Quan hệ

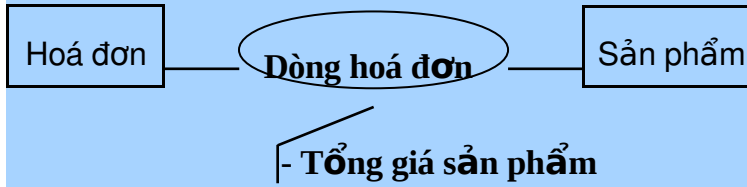
Khái niệm quan hệ ở mục này (khác với khái niệm quan hệ của Codd) được dùng để nhóm họp 2 hay nhiều thực thể với nhau nhằm biểu hiện một mối liên quan tồn tại trong thế giới thực giữa các thực thể này. Kích thước của một quan hệ là số thực thể đã cấu thành nên quan hệ, và có thể là một số

nguyên bất kỳ. Tuy vậy, trong thực tiễn, người ta luôn tìm cách tránh dùng đến những quan hệ có kích thước lớn hơn 3.

Trong một mô hình dữ liệu các quan hệ được biểu diễn bằng những hình tròn hoặc elipse. Trong một số trường hợp, mối quan hệ cũng có thể có những thuộc tính riêng.

Ví dụ:

Hoá đơn dùng để thanh toán một số sản phẩm bán ra. Mỗi dòng hoá đơn cho biết tổng giá của mỗi sản phẩm.Đây là một quan hệ có kích thước là 2, còn gọi là quan hệ nhị nguyên.



Người ta đưa ra khái niệm những vai trò khác nhau của cùng một thực thể để có thể biểu diễn mối quan hệ giữa thực thể này với chính nó. Vì loại quan hệ này ít dùng nên trong Giáo trình này chúng tôi không trình bày loại quan hệ đó.

5.2.4. Phân loại các quan hệ

Xét R là một tập quan hệ và E là một thực thể cấu thành của R, mỗi cặp (E,R) được biểu thị trên sơ đồ khái niệm dữ liệu bằng một đoạn thẳng. Với thực thể E, ta có thể xác định được:

- X là số tối thiểu các thể hiện tương ứng với E mà R có thể có trong thực tế.

Giá trị của X như vậy chỉ có thể bằng 0 hay 1.

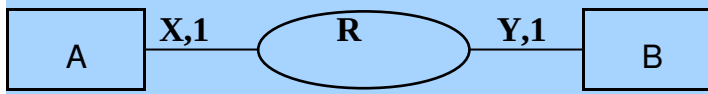
- Y là số tối đa các thể hiện tương ứng với E mà R có thể có trong thực tế.

Giá trị của Y có thể bằng 1 hay một số nguyên N > 1 .

Cặp số (X, Y) được định nghĩa là bản số của đoạn thẳng (E,R) và có thể lấy các giá trị sau: (0,1), (1,1), (0,N) hay (1,N), với N > 1.

Đối với loại quan hệ nhị nguyên R liên kết giữa hai thực thể A và B, ta phân thành ba loại quan hệ cơ bản sau:

- Quan hệ 1-1 (một - một): mỗi thể hiện của thực thể A được kết hợp với 0 hay 1 thể hiện của B và ngược lại.

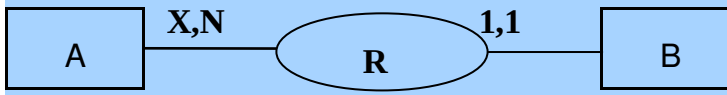


X và Y có thể lấy các giá trị 0 và 1

Ví dụ:

Mỗi độc giả ở một thời điểm chỉ được đọc một quyển sách.

- Quan hệ 1 - N (một - nhiều): Mỗi thể hiện của thực thể A được kết hợp với 0,1 hay nhiều thể hiện của B và mỗi thể hiện của B được kết hợp với một thể hiện duy nhất của A. Đây là một loại quan hệ thông dụng và đơn giản nhất.

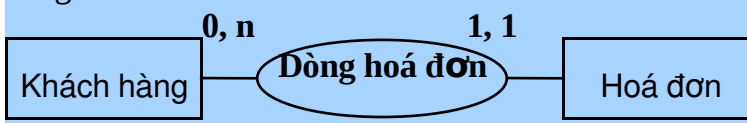


X có thể lấy các giá trị 0 và 1

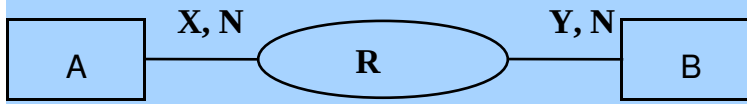
Ví dụ:

Một khách hàng có thể có nhiều hoá đơn

Một hoá đơn chỉ có thể mang tên một khách hàng.



- Quan hệ N - P (nhiều - nhiều): Mỗi thể hiện của một thực thể A được kết hợp với 0, 1 hay nhiều thể hiện của B và ngược lại, mỗi thể hiện của B được kết hợp 0, 1 hay nhiều thể hiện của A



X và Y có thể lấy các giá trị 0 hoặc 1

Ví dụ:

Một hoá đơn dùng để thanh toán một hay nhiều sản phẩm

Một sản phẩm có thể xuất hiện trong 0, 1 hay nhiều hoá đơn.

Thông thường quan hệ N - P chứa các thuộc tính. Chúng ta biến đổi loại quan hệ này thành thực thể và thực thể này cần được nhận dạng bởi một khoá chính.

5.3. Các dạng chuẩn trong các hệ QTCSDL hiện có

Trong mục này chúng tôi trình bày một số ý nghĩa của phụ thuộc hàm và mối liên hệ của nó với việc chuẩn hoá trong thực tiễn

5.3.1. Một số phụ thuộc hàm đặc biệt

Trên cơ sở của định nghĩa phụ thuộc hàm đã trình bày ở chương 2 chúng ta có thể thấy:

- Nếu B phụ thuộc hàm vào A (A→B), thì với mỗi giá trị của A tương ứng với một giá trị duy nhất của B. Hay nói cách khác, tồn tại một hàm (ánh xạ) từ tập hợp những giá trị của A đến tập hợp những giá trị của B.

Ví dụ: Trong một hoá đơn bao gồm các thuộc tính 'số hoá đơn', 'tên khách hàng', 'mã sản phẩm', 'tổng giá trị sản phẩm'....

Ta thấy 'Số hoá đơn' →'Tên khách hàng'

'Số hoá đơn', 'Mã sản phẩm' →'Tổng giá trị sản phẩm'

Chúng ta có thể mở rộng khái niệm phụ thuộc hàm khi cho phép A hoặc B là một thực thể hoặc là một quan hệ.

Ví dụ: Ta có hai thực thể 'Hoá đơn' và 'Khách hàng'

Khi đó: Thực thể 'Hoá đơn' →'Thuộc tính 'Tên khách hàng'

Thực thể 'Hoá đơn' →thực thể 'Khách hàng'

Thuộc tính 'Số hoá đơn' →thực thể 'Khách hàng'

Trong chương 3 ta trình bày phụ thuộc hàm hoàn toàn và phụ thuộc hàm trực tiếp. Hai loại phụ thuộc hàm này đóng vai trò quan trong các dạng chuẩn.

Các dạng chuẩn được đề ra với mục đích để đảm bảo tính nhất quán và tránh việc trùng lặp các thông tin. Trong mục này chúng ta sẽ quay trở lại với các dạng chuẩn. Các dạng chuẩn này có những biến đổi điều kiện ràng buộc đơn giản hơn so với các dạng chuẩn đã trình bày trong chương 3.

5.3.2. Dạng chuẩn 1

Chúng ta nói rằng một thực thể hay quan hệ ở dạng chuẩn 1 nếu tất cả giá trị các thuộc tính của nó là sơ cấp. Điều kiện ràng buộc giống như 1NF của chương 3. Định nghĩa của dạng chuẩn 1 mang tính mô tả. Thông thường giá trị các thuộc tính là các dãy kí tự hoặc là các số như trong FOXPRO, khi đó chúng ta cho các giá trị này là sơ cấp

Để minh họa ta đưa ra thực thể sau đã trình bày trong mục 4.3.

Bán hàng

Ngày tháng	Mã hàng	Tên hàng	Đơn giá	Số lượng	Tổng	Thanh toán
210397	M1	Radio	1000	1	10000	6 000

210397	M3	TV	4000	2	10000	6 000
210397	M6	Xe đạp	1000	1	10000	6 000
220397	M2	Máy giặt	3000	2	6 000	2 000
230397	M1	Radio	1000	3	15000	11 000
230397	M4	Video	5000	2	15000	11 000
230397	M9	Máy ảnh	2000	1	15000	11 000

Chúng ta chọn khoá chính (nó là một khoá tối tiểu) cho thực thể bán hàng là tập {Ngày tháng, Mã hàng}

5.3.3. Dạng chuẩn 2

Một thực thể hay quan hệ là 1NF được xem là dạng chuẩn 2 nếu tất cả các phụ thuộc hàm giữa khoá chính và các thuộc tính khác của nó đều là hoàn toàn .

Chú ý rằng định nghĩa dạng chuẩn 2 trong chương 2 chặt hơn vì điều kiện phụ thuộc hoàn toàn liên quan đến mọi khoá tối tiểu, chứ không chỉ liên

quan đến một khoá tối tiểu được chọn làm khoá chính.

Trong ví dụ trên, thực thể Bán hàng đã là 1NF, ta nhận thấy đối với khoá chính {số phiếu, mã vật tư} các thuộc tính Tổng và Thanh toán phụ thuộc hàm vào thuộc tính Ngày tháng, các thuộc tính Tên hàng, Đơn giá phụ thuộc hàm vào thuộc tính Mã hàng. Ngày tháng, Mã hàng là hai thuộc tính của khóa chính. Do đó dẫn đến trùng lặp dữ liệu. Thực thể Bán hàng không là 2NF. Để thoả dạng chuẩn 2NF, ta phải tách nó thành 3 thực thể riêng biệt:

hàng hoá

Mã hàng	Tên hàng	Đơn giá
M1	Radio	1 000
M2	Máy giặt	3 000
M3	TV	4 000
M4	Video	5 000
M6	Xe đạp	1 000
M9	Máy ảnh	2 000

Ta chọn khoá chính của thực thể hàng hoá là Mã hàng

khối lượng

Ngày tháng	Mã hàng	Số lượng
210397	M1	1
210397	M3	2
210397	M6	1
220397	M2	2
230397	M1	3
230397	M4	2
230397	M9	1

Ta chọn khoá chính (nó là khoá tối tiểu) cho thực thể Khối lượng là Ngày tháng, Mã hàng

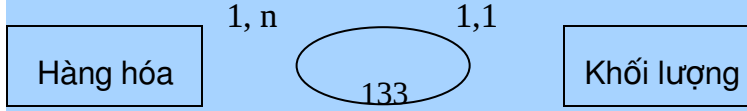
Doanh số

Ngày tháng	Tổng	Thanh toán
210397	10000	6000
220397	6000	2000
230397	15000	11000

Khoá chính là Ngày tháng

Có thể thấy thực thể Khối lượng →thực thể Hàng hoá

Ta có biểu diễn sau:



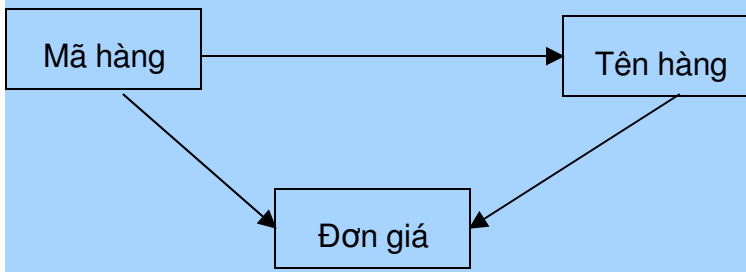
R

5.3.4. Dạng chuẩn 3 (3NF)

Một thực thể (hay quan hệ) đã là 2NF được xem là dạng chuẩn 3NF nếu tất cả các phụ thuộc hàm giữa khoá chính và các thuộc tính khác của nó đều là trực tiếp.

Hay nói cách khác, mọi thuộc tính không nằm trong khoá chính đều không phụ thuộc hàm vào một thuộc tính không phải là khóa chính. Ta có thể rút ra nhận xét: Một thực thể có nhiều khoá nhận dạng không thể thoả mãn dạng chuẩn 3NF. Mặt khác định nghĩa 3NF trong chương 2 chặt hơn vì điều kiện phụ thuộc hoàn toàn và phụ thuộc trực tiếp liên quan đến mọi khoá tối tiểu, chứ không chỉ liên quan đến một khoá tối tiểu được chọn làm khoá chính.

Trong thực thể Hàng hoá là 2NF ở trên, ta thấy trên đồ thị của các phụ thuộc hàm có hai con đường để đi từ ‘Mã hàng’ đến ‘Đơn giá’ hoặc đi qua thuộc tính ‘Tên hàng’



Điều này chứng tỏ thực thể chưa là 3NF, dẫn đến trùng lặp đơn giá của tên hàng. Để là dạng 3NF, ta tách nó thành hai thực thể riêng biệt:

Hàng

Mã hàng	Tên hàng
M1	Radio
M2	Máy giặt
M3	TV
M4	Video
M6	Xe đạp
M9	Máy ảnh

Khoá chính là Mã hàng

Mặt hàng

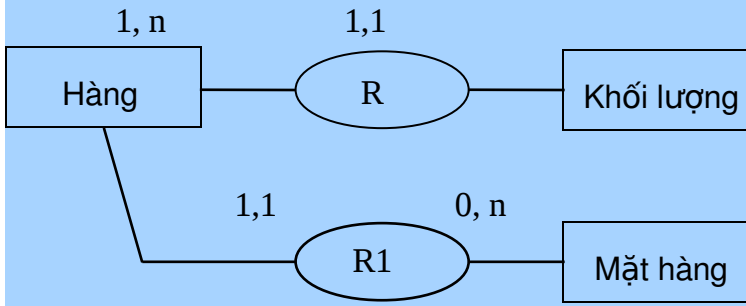
Tên hàng	Đơn giá
Radio	1000
Máy giặt	3000
TV	4000

Video	5000
Xe đạp	1000
Máy ảnh	2000

Khoá chính là Tên hàng

Có thể thấy thực thể Hàng → thực thể Mặt hàng

Tổng hợp với phần trên, ta có sơ đồ sau:



5.3.5. Dạng chuẩn của Boyce - Codd (BCNF)

Dạng chuẩn 3NF cho phép một thuộc tính thành phần của khoá chính phụ thuộc hàm vào một thuộc tính không phải là khoá

Ví dụ :

Lớp	Môn	Thầy
12	Toán	A
11	Toán	D
10	Toán	A
12	Địa	C
11	Địa	C
10	Địa	D

Thực thể này thoả dạng 3NF. Khoá chính của nó gồm các thuộc tính 'Lớp' và 'Môn'.

Nhưng do qui tắc 'Mỗi thầy chỉ dạy một môn', ta thấy có sự phụ thuộc hàm của Môn (Là một thành phần của khoá chính) vào Thầy (Là một thuộc tính bình thường):

'Thầy' $\rightarrow$'Môn'

Ta nói rằng thực thể thoả mãn dạng chuẩn Boyce-Codd (BCNF) khi tất cả các phụ thuộc hàm của nó đều thuộc dạng K $\rightarrow$ a, trong đó K là khoá chính và a là một thuộc tính bất kỳ.

Để thoả dạng BCNF, ta có thể tách thực thể trên thành hai thực thể riêng biệt như sau:

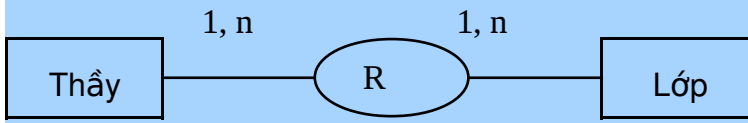
Thực thể 'Lớp':

Lớp
12
11
10

Thực thể 'Thầy':

Thầy	Môn
A	Toán
B	Toán
C	Sử Địa
D	Sử Địa

Chúng ta có biểu diễn sau:



5.3.6. Nhận xét về việc chuẩn hoá

Khi không có yêu cầu gì đặc biệt, người ta thường tìm cách chuẩn hoá mô hình dữ liệu nhằm tăng hiệu hiệu năng và giảm sơ xuất trong các giai đoạn phát triển hệ thông tin về sau.

Tuy vậy, việc chuẩn hoá không phải lúc nào cũng đạt đến mức tối đa. Thông thường chúng ta chuẩn hoá đến dạng chuẩn 2NF và 3NF.

Chương 6

Một số công đoạn xây dựng các dự án thiết kế tổng thể các hệ thống cơ sở dữ liệu hiện nay

6.1. Mô tả chung

Trong dự án thiết kế tổng thể người ta thường trình bày một số vấn đề cơ bản sau:

- Hiện trạng và tiềm lực CNTT của cơ quan chủ trì dự án.

- Hiện trạng về việc thu thập, lưu trữ và xử lí các dữ liệu liên quan hệ cơ sở dữ liệu cần xây dựng,

- Ước đoán khối lượng lưu trữ và trao đổi thông tin trong hệ cơ sở dữ liệu,

- Phân tích thiết kế hệ CSDL,

- Thiết kế hạ tầng kĩ thuật,

- Chuẩn hoá, bảo mật và an toàn thông tin,

- Tính pháp lí về quyền và nghĩa vụ trong việc thu thập, cập nhật, khai thác và bảo vệ thông tin trong hệ CSDL,

- Nội dung, đối tượng và kế hoạch triển khai công tác đào tạo,

- Tổng hợp và cân đối kinh phí

- Các biện pháp tổ chức thực hiện .

Một trong các phần quan trọng nhất của dự án là khâu phân tích thiết kế.

Thông thường để thực hiện việc phân tích thiết kế khi xây dựng dự án người ta sử dụng một số phương pháp phân tích thiết kế, ví dụ như phương pháp phân tích thiết kế có cấu trúc (Structured Analysis and Design), phương pháp GALASCI, phương pháp phân tích MCX, phương pháp phân tích MERISE (Methode pour Rassembler les Idees Sans Effort)...

Trong các phương pháp phân tích thiết kế trên MERISE có đặc tính là khi phân tích nó tách rời xử lí với dữ liệu, tổ chức theo nhiều mức, đi từ phân tích tổng thể đến chi tiết một cách tuần tự theo chiều tăng dần của mức độ phức tạp. MERISE được nhiều người xử dụng và đặc biệt tốt cho việc phân tích thiết kế cho các hệ thông tin lớn. Tại Việt Nam một số cơ quan đã được trang bị công cụ phân mềm MEGA tuân theo các chuẩn của phương pháp phân tích MERISE. Cho đến nay nhiều dự án "Thiết kế

tổng thể hệ thống cơ sở dữ liệu" đã được xây dựng trên cơ sở phương pháp phân tích MERISE.

Dựa trên phương pháp MERISE, khi phân tích thiết kế tổng thể, chúng ta tiến hành các công đoạn sau :

① Mô tả qui trình nghiệp vụ bằng lời:

Trong công đoạn này chúng ta mô tả đầy đủ, mạch lạc bằng lời qui trình nghiệp vụ của bài toán cần thiết kế. Công đoạn này thường phân biệt những yếu tố nghiệp vụ hay thay đổi với những yếu tố nghiệp vụ tương đối bất biến trong khoảng thời gian dài. Những yếu tố bất biến này thường được tập trung mô tả kĩ hơn. Trong công đoạn mô tả bằng lời này, chúng ta không chỉ mô tả những yếu tố nghiệp vụ hiện có mà còn mô tả các kiến nghị sửa đổi, các dự báo tương lai phù hợp với quá trình phát triển của hệ thống thông tin. Công đoạn này phân rã mỗi lĩnh vực nghiệp vụ thành các chức năng nghiệp vụ, và mỗi chức năng nghiệp vụ được phân rã thành các thao tác xử lí. Toàn bộ các yếu tố này đều được mô tả rõ ràng.

② Xây dựng các mô hình

Sau khi mô tả các qui trình nghiệp vụ một cách rõ ràng và mạch lạc, chúng ta tiến hành xây dựng các mô hình. Các mô hình này được phân chia theo 4 mức độ khái quát khác nhau. Đó là :

- Mức khái niệm: Mức này mô tả sự hoạt động của đơn vị theo một cấu trúc khái quát nhất. Trong mức này các chức năng hoạt động được mô tả độc lập với những bộ phận, vị trí hay nhân viên thực hiện chúng.

- Mức tổ chức: Mức này thể hiện các mục tiêu đã được khái niệm hóa lên thực tế của đơn vị trong đó có tính đến những điều kiện ràng buộc về mặt tổ chức.

- Mức lôgic: Mức này qui định các công cụ tin học mà các công cụ này được người dùng trong các thao tác xử lí.

- Mức vật lí: Mức này liên quan chặt chẽ với các trang thiết bị tin học cụ thể.

Trong mỗi mức chúng ta có 3 mô hình. Đó là mô hình trao đổi thông tin, mô hình xử lí và mô hình dữ liệu.

Cụ thể chúng ta có:

*Mô hình khái niệm trao đổi (MCC modele conceptuel de communication):

MCC phân rã lĩnh vực nghiệp vụ thành các chức năng nghiệp vụ. MCC mô tả sự trao đổi thông tin giữa các chức năng nghiệp vụ nằm trong bài toán và sự trao đổi thông tin giữa các lĩnh vực nghiệp vụ với các lĩnh vực nghiệp vụ khác và các đối tượng bên ngoài.

MCC cho ta một cái nhìn tổng thể về một lĩnh vực nghiệp vụ và các chức năng nghiệp vụ của nó cũng như các nhu cầu thông tin của nó.

*Mô hình khái niệm xử lí (MCT modele conceptuel de traitements):

MCT dùng để mô tả một lĩnh vực, qui trình, chức năng, thao tác.

MCT phân rã một chức năng nghiệp vụ thành các thao tác xử lí . Mỗi thao tác xử lí có thể được xem như một phép biến đổi thông tin. Một thao tác có thể có điều kiện khởi động là các sự kiện, các thông báo mà khi xuất hiện chúng thao tác bắt đầu được thực hiện . Trong quá trình thực hiện, thao tác cần phải truy nhập đến các thông tin đã được lưu giữ , biến đổi chúng và cập nhật lại theo một số qui luật tính toán và ràng buộc nhất định.

* Mô hình khái niệm dữ liệu (MCD modele conceptuel de donneees):

Việc mô tả dữ liệu trong mô hình MCD thông qua ngôn ngữ " Thực thể / Quan hệ " cùng với các thuộc tính của các thực thể và các quan hệ . Trên mô hình này, khóa chính, các thuộc tính chính được khai báo và lưu trữ trong hệ thống.

MCD được xây dựng cho một lĩnh vực nghiệp vụ nhằm xác định đầy đủ những dữ liệu đòi hỏi khi thực hiện các chức năng , trong đó đặc biệt là những dữ liệu cần thiết cho việc trao đổi .

* Mô hình tổ chức trao đổi (MOC):

MOC mô tả một lĩnh vực nghiệp vụ, đơn vị tổ chức.

Mô hình này mô tả các vị trí làm việc và việc luân chuyển thông tin trong đơn vị.

* Mô hình tổ chức xử lí (MOT):

MOT là mô hình tổ chức để thực hiện các thao tác của một chức năng nghiệp vụ đã được mô tả trong MCT. MOT thể hiện qui trình làm việc , trong đó nhấn mạnh tính tuần tự của các thao tác và nêu rõ những ràng buộc về thời điểm bắt đầu xử lí hay truyền thông tin. Một thao tác trong MCT có thể ứng vối nhiều thao tác trong MOT và ngược lại một thao

tác trong MOT cũng có thể ứng với nhiều thao tác trong MOT tuỳ theo các hoàn cảnh cụ thể .

* Mô hình tổ chức dữ liệu (MOD):

MOD mô tả dữ liệu cần ghi nhớ trong từng địa điểm, cho từng vị trí thực hiện.

Trong khi MCD chỉ định nghĩa các khái niệm dữ liệu, thì MOC cụ thể hóa những điều kiện có thể xảy ra để một thực thể thuộc vào mô hình.

* Mô hình lôgic trao đổi (MLC):

MLC xác định sự trao đổi giữa người với người (thông qua các mẫu biểu), giữa người với máy (thông qua giao diện) và giữa các phần mềm với nhau.

* Mô hình lôgic xử lí (MLT): Thường để mô tả công cụ tin học.

* Mô hình lôgic dữ liệu (MLD):

Nhờ MLD, chúng ta có thể chuyển các MOD sang dạng quen thuộc cho các chuyên gia tin học. Thông thường, chúng ta chuyển từ ngôn ngữ thực thể - quan hệ sang dạng biểu báo.

Các mô hình vật lí trao đổi (MPC), mô hình vật lí xử lí (MPT), mô hình vật lí dữ liệu (MPD) gắn liền với các trang thiết bị tin học cụ thể.

Các mức và mô hình của MERISE có thể tóm tắt như sau:

Mức	Trao đổi	Chức năng	Dữ liệu
Khái niệm	MCC	MCT	MCD
Tổ chức	MOC	MOT	MOD
Lôgic	MLC	MLT	MLD
Vật lí	MPC	MPT	MPD

Quan trọng nhất trong các mô hình trên là các mô hình liên quan đến dữ liệu vì nó làm nền tảng cho việc xây dựng các mô hình khác.

Trong phạm vi Giáo trình này chúng tôi trình bày việc xây dựng mô hình khái niệm dữ liệu. Đó là mô hình dữ liệu thường có trong các dự án thiết kế tổng thể các hệ thống thông tin.

Quá trình xây dựng mô hình khái niệm dữ liệu có thể được chia thành các giai đoạn sau đây:

A. Khảo sát thực tế

- Thu thập và trình bày thông tin.

B. Thiết kế mô hình dữ liệu

- Kiểm kê các dữ liệu.
- Xác định các phụ thuộc hàm.
- Xây dựng mô hình khái niệm:
- Xác định tập hợp các khoá chính.
- Nhận diện các thực thể.
- Nhận diện các quan hệ.
- Phân bổ các thuộc tính còn lại.
- Vẽ sơ đồ khái niệm dữ liệu.

C. Kiểm soát và chuẩn hoá mô hình

6.2. Khảo sát thực tế

6.2. Khảo sát thực tế

Mục tiêu của giai đoạn này là qua quá trình quan sát, phỏng vấn, tìm hiểu và phân tích, chúng ta mô tả đầy đủ hiện trạng, các bài toán nghiệp vụ và các nhu cầu của người sử dụng mà hệ thống thông tin cần phải thoả mãn. Do đó, nó không chỉ giới hạn trong việc xây dựng mô hình dữ liệu mà còn là nguồn gốc các thông tin cần thiết cho việc xây dựng mô hình xử lí.

Để đạt mục tiêu này, ta cần thu thập và trình bày tất cả những thông tin dù thuộc phương diện dữ liệu

hay chức năng có thể hữu ích cho việc thiết kế hệ thông tin về sau. Các thông tin này cần được quan sát dưới dạng: tĩnh (dữ liệu sơ cấp, tài liệu, quảng cáo, đơn vị, vị trí làm việc..), dạng động (luồng luân chuyển các thông tin, tài liệu, chu kì, thời lượng), và dạng biến đổi của chúng (thủ tục, qui tắc quản lí, công thức tính toán,..).

Các thông tin thu thập được phải đầy đủ và chính xác vì chúng là nền tảng của hệ thông tin tương lai. Nhưng cũng không nên đi quá sâu vào chi tiết và phải biết gạn bỏ những thông tin không cần thiết, để không làm chệch hướng và gây khó khăn nặng nề cho việc phân tích thiết kế.

Công việc khảo sát không chỉ tập trung hoàn toàn vào giai đoạn đầu của quá trình phân tích thiết kế, mà có thể chạy dài trong suốt quá trình này để thu thập thêm thông tin, đào sâu vấn đề hay kiểm chứng một giả thiết cùng với người sử dụng khi hệ thông tin cần xây dựng quá lớn và phức tạp, ta nên chia nó thành nhiều tiểu hệ. Mỗi tiểu hệ có thể được khảo sát, phân tích hay thiết kế độc lập với nhau, trước khi được tập hợp lại .

Để chia một hệ thông tin thành nhiều tiểu hệ, người ta thường sử dụng một trong hai phương pháp tiếp cận sau đây:

- Phương pháp 1: Các tiểu hệ độc lập được định ra, dựa trên cơ sở những bài toán , chức năng nghiệp vụ chủ yếu của tổ chức. Đôi khi dựa trên một kế hoạch thực hiện theo thứ tự ưu tiên hay để thoả những yêu cầu về thời gian.

- Phương pháp 2: Một cuộc khảo sát tổng quát sơ khởi sẽ cho phép nhận diện những tiểu hệ tương đối độc lập với nhau.

Tiếp theo đó chúng ta tiến hành thu thập thông tin về hệ thống cần xây dựng

Công việc này chủ yếu là tham khảo tài liệu và tiếp xúc với những người sử dụng, đòi hỏi những khả năng như óc quan sát, kinh nghiệm, tài giao tiếp và ứng biến... Các phương pháp gò bó, cứng nhắc sẽ chẳng đem lại kết quả mong muốn. Do đó, phần này chỉ liệt kê và phân loại các thông tin có thể gặp được trong quá trình khảo sát, xem như để trợ giúp trí nhớ.

Trong quá trình thu thập thông tin thông thường ta tập hợp các dữ liệu sau

- Dữ liệu về hệ thông tin hiện tại bao gồm các dữ liệu liên quan trực tiếp đến hệ thông tin theo mọi dạng (tĩnh, động, biến đổi) và thông tin về môi trường làm việc

- Dữ liệu về hệ thống tương lai bao gồm các nhu cầu, mong muốn cho hệ thông tin của ta. Trong các dữ liệu này ta cần phân biệt những vấn đề đã được nhận thức và phát biểu rõ ràng, những vấn đề được nhận thức nhưng chưa được công nhận, những vấn đề còn chưa nhận thức và còn đang tranh luận.

Với mỗi loại dữ liệu cần thu thập đã nêu trên, nếu cần ta còn có thể tìm hiểu thêm về một số khía cạnh khác như: số lượng, độ chính xác cần có, ai là người có trách nhiệm...

Nhận xét:

- Nếu có thể, các cuộc phỏng vấn phải được tiến hành tuần tự theo cấu trúc phân cấp của tổ chức theo từng bộ phận, lĩnh vực, chức năng hay đi từ cấp lãnh đạo qua cấp quản lí đến những người thừa hành.

- Phải luôn nhớ sao chụp mẫu các hồ sơ, tài liệu, để có được cấu trúc chính xác các thông tin làm căn bản cho việc xây dựng mô hình dữ liệu sau này.

- Cần thiết nhất là luôn phân biệt những thông tin nói về hệ thông tin đang xây dựng với những thông tin thuộc về hệ này

Các thông tin thu thập, sau khi được tổng hợp sẽ được trình bày dưới hai dạng:

a. Mô tả các bài toán nghiệp vụ, các chức năng và tổ chức của cơ quan, các nhu cầu và mong muốn của người sử dụng một cách đầy đủ, nhưng ngắn gọn và mạch lạc, bằng một ngôn ngữ thông thường, gần gũi với mọi người.

b. Minh hoạ và hệ thống hoá phần trình bày trên bằng một ngôn ngữ hình thức, thường là dưới dạng phiếu mô tả, danh sách và đồ hoạ.

Trên thực tế có nhiều phương pháp trình bày thông dụng khác nhau,

6.3. Thiết kế mô hình dữ liệu

Thiết kế một mô hình khái niệm dữ liệu là liệt kê và định nghĩa chính xác những dữ liệu có liên quan đến các chức năng, hoạt động của một tổ chức. Sau đó ta nhóm chúng lại thành thực thể và quan hệ giữa các thực thể, rồi dùng một số qui ước đã định trước để trình bày dưới dạng mô hình khái niệm.

6.3.1. Kiểm kê dữ liệu

Danh sách này chủ yếu được rút từ những thông tin thu thập được trong giai đoạn khảo sát ban đầu; tài liệu thu thập được; nhu cầu, giải thích của người sử dụng.

Có thể phân biệt hai kiểu dữ liệu:

- Loại dữ liệu xuất hiện trực tiếp trên các tài liệu, màn hình, tập tin thu thập được.

- Loại dữ liệu không xuất hiện nhưng cần thiết để chứa kết quả trung gian, các thông tin đang chờ được xử lý, hay để tính toán các dữ liệu thuộc loại thứ nhất.

Một công cụ thông dụng, hữu ích cho giai đoạn này là bảng, dùng để phân tích các tài liệu thu thập và liệt kê ra danh sách các dữ liệu. Trong bảng này, ta trình bày mỗi cột là một tài liệu và mỗi hàng là một loại dữ liệu. Tại mỗi ô giao điểm, ta đánh dấu khi loại dữ liệu có xuất hiện trên tài liệu. Nên dùng hai loại dấu khác nhau để phân biệt loại dữ liệu trực tiếp với loại được tính toán thành.

Khi xây dựng bảng này, ta nên bắt đầu bằng những tài liệu cơ bản, quan trọng nhất và chỉ cần trình bày một loại tài liệu khi nó cho phép nhận dạng ít nhất một loại dữ liệu mới.

Ví dụ: Trong một công ty có:

- Kho hàng làm nhiệm vụ lưu giữ và quản lí hàng hoá và khi cần thì đề nghị mua thêm hàng

- Phòng đặt hàng có nhiệm vụ làm đơn đặt hàng gửi cho các công ty cung cấp

- Phòng kế toán lưu bản sao đơn đặt hàng để kiểm tra hàng.

Ta dùng 1 để đánh dấu dữ liệu trực tiếp và 2 cho dữ liệu được tính toán.

Tài liệu / Loại dữ liệu	Phiếu đề nghị đặt hàng	Đơn đặt hàng	Phiếu giao hàng	Tệp tin về nhà cung cấp
Tên kho	1			
Địa chỉ kho	1			
Ngày đề nghị đặt hàng	1			
Số lượng hàng cần đặt	1			
Mã số sản phẩm	1	1	1	
Nhãn hiệu sản phẩm	1	1	1	
Mã số của công ty cung cấp		1	1	1
Tên công ty cung cấp		1	1	1

Địa chỉ công ty cung cấp		1	1	1
Đơn giá sản phẩm		1	1	1
Ngày đặt hàng		1		
Số lượng hàng đặt mua		1		
Tổng giá đơn đặt hàng		2		
Ngày giao hàng			1	
Số lượng hàng được giao			1	
Tổng giá hàng được giao			2	

Từ danh sách này, người ta cần kiểm tra bằng một số công tác thanh lọc như sau:

- Bỏ bớt các dữ liệu đồng nghĩa nhưng khác tên, chỉ giữ lại một.

Ví dụ: Mã số sản phẩm = danh mục mặt hàng

- Phân biệt các dữ liệu cùng tên nhưng khác nghĩa thành nhiều loại dữ liệu khác nhau.

Ví dụ: Giá bán của một cửa hiệu khác với giá bán của công ty sản xuất.

- Nhập chung các loại dữ liệu luôn luôn xuất hiện đồng thời với nhau trên mọi loại tài liệu thành một loại dữ liệu sơ cấp.

Ví dụ: số nhà và tên đường; ngày, tháng và năm sinh.

- Loại bỏ những loại dữ liệu có thể được xác định một cách duy nhất từ các loại dữ liệu khác, hoặc bằng công thức tính toán, hoặc do các qui luật của tổ chức

Ví dụ: Tổng giá đơn đặt hàng = Số lượng hàng* Đơn giá

Giả sử do qui luật của tổ chức, mọi đề nghị mua hàng phải được giải quyết nội trong ngày, ta suy ra:

Ngày đề nghị mua hàng = Ngày đặt hàng.

6.3.2. Định các phụ thuộc hàm giữa các dữ liệu

Từ danh sách dữ liệu đã được thanh lọc của hệ thông tin đạt được qua giai đoạn trên, ta phải định ra tất cả các phụ thuộc hàm có giữa chúng .

Cụ thể, ta phải tự đặt câu hỏi: Mỗi giá trị của một loại dữ liệu A có tương ứng với một giá trị duy nhất của loại dữ liệu B hay không?

Nếu câu trả lời là "Có" thì B phụ thuộc hàm vào A: A $\rightarrow$B

Ngoài các phụ thuộc hàm có vế trái A là một loại dữ liệu sơ cấp (gọi là phụ thuộc hàm sơ cấp) tương đối dễ xác định, ta còn phải nhận diện cả các phụ thuộc hàm trong đó vế trái A là một tập hợp của nhiều loại dữ liệu (gọi là phụ thuộc hàm đa phần). Trong trường hợp này, ta nên tự đặt câu hỏi: Cần ấn định giá trị của những loại dữ liệu nào để có thể suy ra một giá trị duy nhất của loại dữ liệu B? Các phụ thuộc hàm sẽ được trình bày dưới dạng một bảng như sau:

Loại dữ liệu Phụ thuộc hàm Phụ thuộc hàm

sơ cấp

đa phần

Mã số của công ty cung cấp .....

Tên công ty cung cấp.....
.....

Địa chỉ công ty cung cấp.....
.....

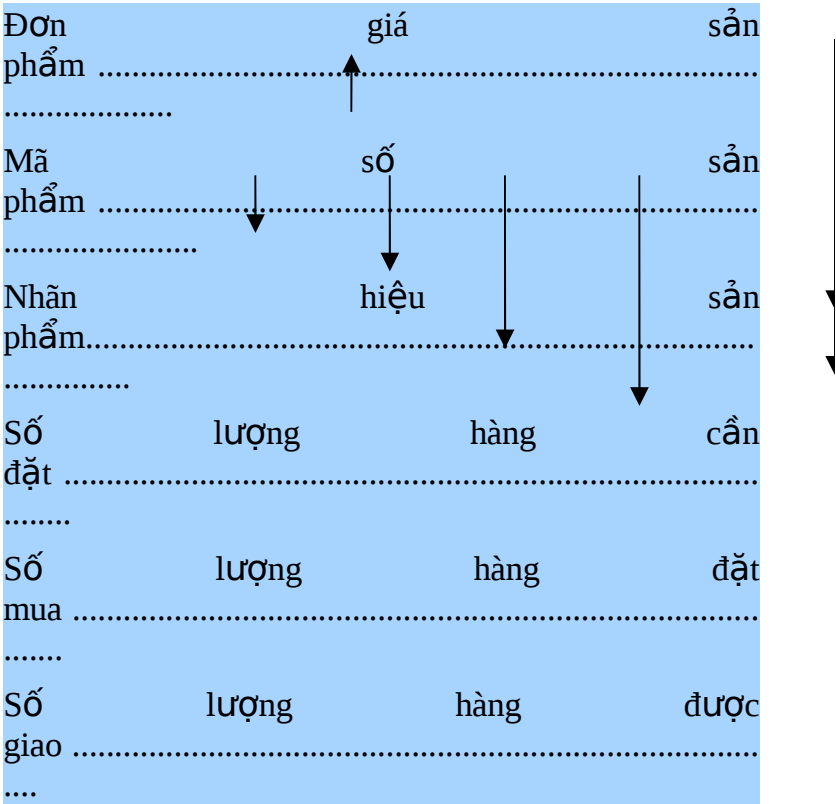
Tên kho .....
.....

Địa chỉ kho.....
.....

Ngày đề nghi đặt hàng .....
.....

Ngày đặt hàng .....
.....

Ngày giao hàng .....
.....



6.3.3. Xây dựng mô hình dữ liệu

Giai đoạn này bao gồm 5 động tác:

- Định tập hợp các khoá chính
- Nhận diện các thực thể
- Nhận diện các quan hệ

- Phân bổ các thuộc tính còn lại

- Vẽ sơ đồ khái niệm dữ liệu

6.3.3.1. Xác định tập hợp các khoá chính

Tập hợp K của những khoá chính là tập hợp tất cả những loại dữ liệu đóng vai trò nguồn (thuộc vế trái) trong ít nhất một phụ thuộc hàm.

Trong ví dụ trên ta có : K =

{'Mã số công ty cung cấp'

'Tên kho'

'Ngày đề nghi đặt hàng'

'Ngày đặt hàng'

'Ngày giao hàng'

'Mã số sản phẩm'}

6.3.3.2 Nhận diện các thực thể

Mỗi phần tử của tập hợp K sẽ là khoá chính của một thực thể

Trong ví dụ trên, ta nhận ra được 4 thực thể:

'Nhà cung cấp' với khoá chính là 'Mã số của công ty cung cấp'

'Kho' với khoá chính là 'Tên kho'

'Lịch' với khoá chính là 'Ngày'

(các thực thể 'Lịch đặt hàng', 'Lịch đề nghi mua hàng' và 'Lịch giao hàng' hoàn toàn tương đương với nhau nên được gộp thành một thực thể duy nhất là 'Lịch')

'Sản phẩm' với khoá chính là 'Mã số sản phẩm'

6.3.3.3 Nhận diện các quan hệ

Có 2 trường hợp :

a. Nếu gốc của một phụ thuộc hàm bao gồm ít nhất 2 phần tử thuộc tập hợp K thì nó tương ứng với một quan hệ N - P giữa các thực thể có khoá chính là các phần tử này.

Trong ví dụ trên, ta nhận ra được 4 quan hệ :

'Đơn giá của nhà cung cấp'

'Dòng đề nghị mua hàng'

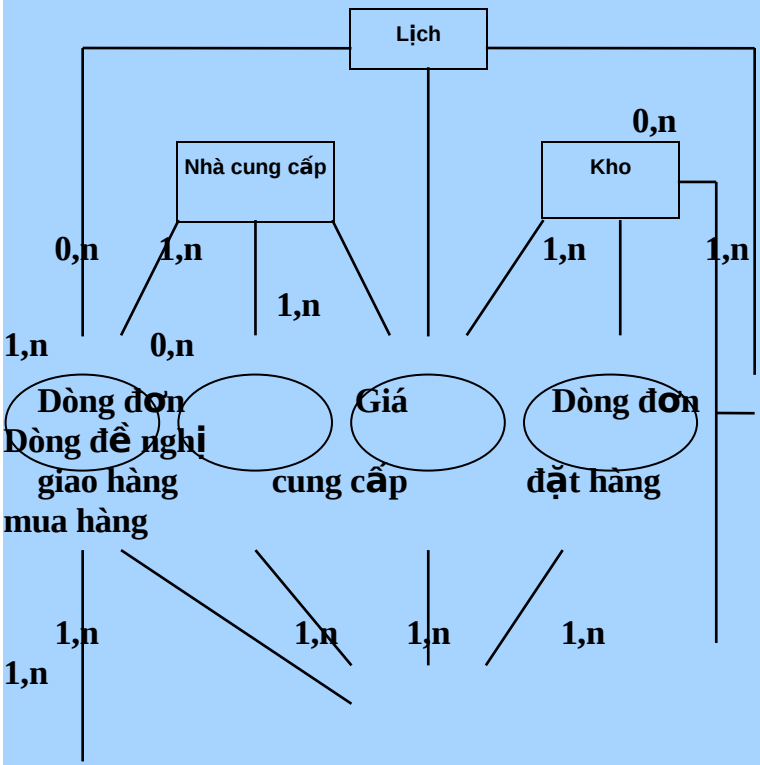
'Dòng đơn đặt hàng'

'Dòng phiếu giao hàng'

b. Một phụ thuộc hàm giữa 2 phần tử của tập hợp K xác định một quan hệ nhị nguyên kiểu 1-N giữa hai thực thể có khoá chính là các phần tử này.

6.3.3.5. Vẽ sơ đồ khái niệm dữ liệu

Từ các thực thể và quan hệ đã nhận diện, ta có thể vẽ lên một sơ đồ khái niệm dữ liệu sau:


Lịch
Nhà cung cấp
Kho
0,n
0,n
1,n
1,n
1,n
1,n
1,n
0,n
Dòng đơn
giao hàng
Giá
cung cấp
Dòng đơn
đặt hàng
Dòng đề nghị
mua hàng
1,n
1,n
1,n
1,n
1,n

6.3.4 Xây dựng mô hình dữ liệu bằng trực giác

Phương pháp phân tích hệ thống nêu trên là một công cụ hữu hiệu và chuẩn xác để xây dựng phần lớn các loại mô hình dữ liệu. Nhưng nếu áp dụng hoàn toàn trong một hệ thông tin cỡ lớn sẽ đòi hỏi nhiều thời gian và công sức. Trong thực tế, các thiết kế viên kinh nghiệm, sau khi đã nhận thức được vấn đề qua khảo sát thường chọn cách xây dựng trực tiếp một mô hình sơ khởi rồi đi thẳng vào giai đoạn sau để kiểm soát và chuẩn hoá mô hình .

Phương pháp trực giác này có ưu điểm là ít tốn thời gian và đôi khi tạo ra những mô hình đơn giản và gần thực tế hơn. Nhưng ngược lại, nó chứa nhiều rủi ro hơn.

6.4. Kiểm soát và chuẩn hoá mô hình

Để đơn giản hoá và đồng thời đảm bảo tính nhất quán của mô hình dữ liệu, ta cần kiểm soát lại mô hình vừa xây dựng bằng một số qui tắc thực tiễn sau đây:

6.4.1 Chuẩn hoá mô hình

Chú ý việc chuẩn hoá toàn bộ mô hình dữ liệu thành các dạng BCNF không phải là bắt buộc. Tuy

vậy các dạng 1FN, 2FN, 3NF nên luôn được thực hiện.

6.4.2 Tạo thêm một thực thể

Việc tạo thêm một thực thể là cần thiết khi có ít nhất một quan hệ đang được xử lý liên quan tới nó.

Việc tạo thêm một thực thể là hợp lí khi:

a) Thuộc tính sẽ được chọn làm khoá chính của thực thể này là một loại dữ liệu thông dụng trong hoạt động của tổ chức đang khảo sát.

b) Ngoài khoá chính này, quan hệ còn có chứa những thuộc tính khác .

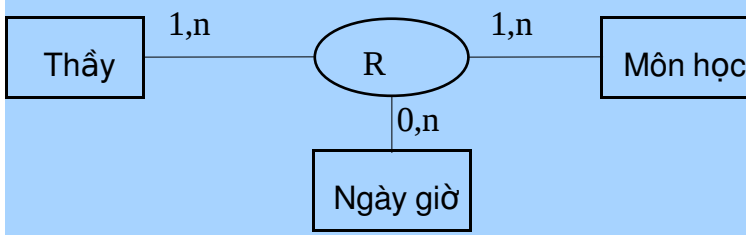
6.4.3. Biến một quan hệ thành thực thể

Một quan hệ có kích thước lớn hơn 3 nên được biến thành thực thể để đơn giản hoá.

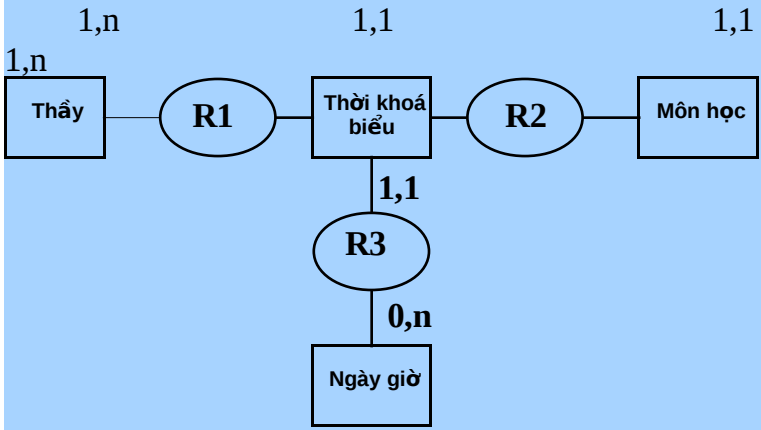
Có thể biến một quan hệ thành thực thể khi hội đủ các điều kiện sau:

- Quan hệ này có một khoá chính độc lập

- Quan hệ này tương ứng với một khái niệm quen thuộc, thông dụng trong hoạt động của tổ chức.



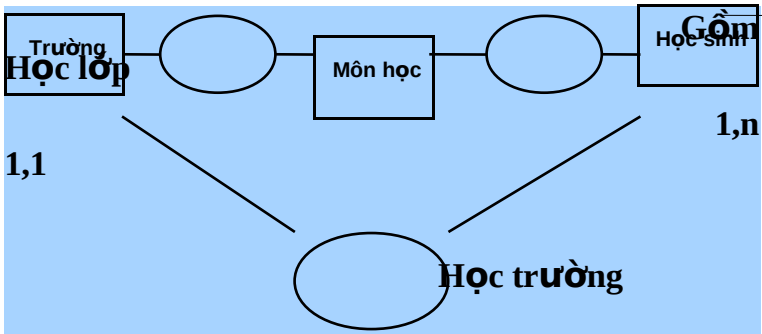
Quan hệ R được biến thành thực thể 'Thời khoá biểu':



6.4.4 Xoá một quan hệ

Một quan hệ 1 - N phải được loại bỏ khỏi mô hình dữ liệu nếu nó là tổng hợp của 2 hay nhiều quan hệ 1 - N khác.

1,n **1,1** **1,n**
1,1



Ta loại bỏ quan hệ Học trường

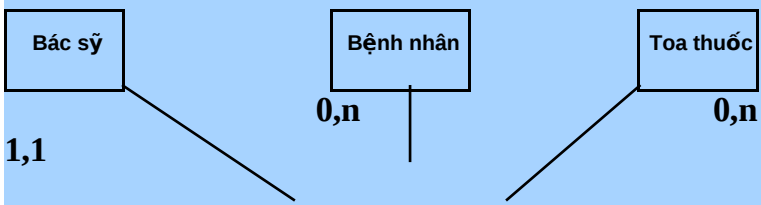
6.4.5 Phân tách một quan hệ phức tạp

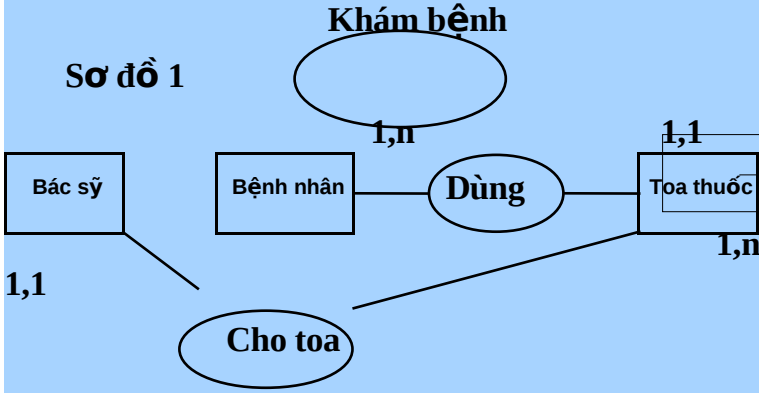
Xét một quan hệ có kích thước lớn hơn hoặc bằng 3. Quan hệ có thể được phân tách thành nhiều quan hệ khác với kích thước nhỏ hơn mà không mất thông tin nếu tồn tại ít nhất một phụ thuộc hàm giữa các thực thể cấu thành quan hệ.

6.4.5.1 Trường hợp phụ thuộc hàm ẩn

Trong trường hợp này, một trong các bản số của quan hệ bằng (1,1) hoặc (0,1). Điều này chứng tỏ sự tồn tại của một số phụ thuộc hàm ẩn.

Ví dụ:





Sơ đồ 1

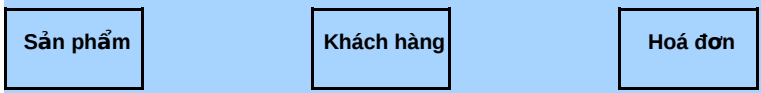
Sơ đồ 2

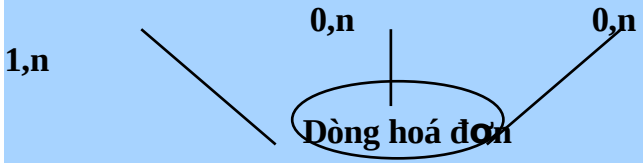
Chúng ta chọn sơ đồ 2

6.4.5.2. Trường hợp phụ thuộc hàm hiện

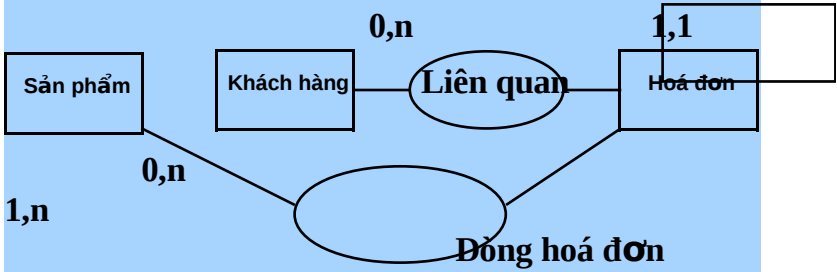
Cho một quan hệ R giữa 3 thực thể A, B, và C. Nếu tồn tại một hàm phụ thuộc A →B thì R có thể được phân thành quan hệ giữa A với B và giữa B với C.

Trong ví dụ sau đây, phụ thuộc hàm Hoá đơn → Khách hàng (mỗi hoá đơn chỉ liên quan đến một khách hàng duy nhất) cho phép ta đưa vào một quan hệ mới để diễn tả sự phụ thuộc này và đơn giản hoá mô hình.





Sơ đồ 1



Sơ đồ 2

Chúng ta chọn sơ đồ 2 đơn giản hơn.

Chương 7
Thuật toán và độ phức tạp

7.1. Khái niệm thuật toán

Nếu cho trước một bài toán, thì một cách giải bài toán được phân định ra thành một số hữu hạn bước, có kết thúc cuối cùng gọi là thuật toán.

Khi giải bài toán sẽ nẩy sinh ra các vấn đề sau:

- Độ phức tạp bài toán. Mỗi bài toán có độ phức tạp khác nhau.

- Một bài toán có nhiều thuật toán giải nó.

- Dùng nhiều ngôn ngữ lập trình để cài đặt phần mềm cho một thuật toán

- Có thể dùng nhiều cấu trúc dữ liệu cho một thuật toán.

Một số sai sót cơ bản khi giải bài toán:

- Hiểu sai bài toán.

- Tìm sai thuật toán.

- Do không hiểu ngôn ngữ lập trình nên có nhầm lẫn.

- Bản thân dữ liệu quét không hết trường hợp.

- Yêu cầu giải quyết bài toán.

- Phần mềm dễ sử dụng .

- Tốc độ tính toán nhanh .

- Bộ nhớ phù hợp .

Trong đó tính dễ sử dụng là một yêu cầu cơ bản nhất.

7.2. Độ phức tạp thuật toán

Khái niệm thuật toán chính xác liên quan đến các khái niệm máy Turing, hàm đệ qui, thuật toán Marcop, ngôn ngữ hình thức của N.Chomsky. Những khái niệm này không nằm trong khuôn khổ Giáo trình này. Chúng tôi trình bày một khái niệm quan trọng liên quan trực tiếp đến thuật toán. Đó là độ phức tạp thuật toán. Nhờ có khái niệm này chúng ta có thể đánh giá và so sánh được các thuật toán với nhau. Hay nói một cách khác, chúng ta có thể có công cụ đo, để lựa chọn một thuật toán tốt cho lời giải bài toán cần giải quyết. Thông thường chúng ta có hai loại đánh giá: Một là độ phức tạp về thời gian tính của thuật toán, hai là độ phức tạp về phạm vi bộ nhớ dùng cho thuật toán. Đối với một thuật toán, thời gian tính và phạm vi bộ nhớ cần dùng thường mâu thuẫn nhau. Có nghĩa là, nếu thời gian tính của thuật toán là ngắn thì thông thường phạm vi bộ nhớ dùng cho thuật toán đó lại lớn. Mà chúng ta lại muốn chọn một thuật toán thời gian tính thì ngắn và bộ nhớ dùng cũng nhỏ. Như vậy, trong từng trường hợp cụ thể, chúng ta sẽ quyết định chọn lựa thuật toán nào. Trong phạm vi Giáo trình này chúng ta chỉ trình bày về độ phức tạp thời gian tính. Đó là độ phức tạp thường được đề cập nhiều nhất. Đồng thời,

trong phạm vi giới hạn của giáo trình, chúng ta cũng chỉ trình bày độ phức tạp của thuật toán theo góc độ tin học.

Giả sử T là thuật toán giải quyết bài toán A. Chúng ta gọi T(n) là độ phức tạp thời gian của thuật toán T. Thông thường T(n) được biểu diễn dưới dạng sau: T(n) = O(g(n)). Trong đó hàm g(n) là cấp của T(n); n là độ dài thông tin đưa vào.

Ví dụ: $T(n) = O(n^2)$

Chúng ta hiểu f(n) = O(g(n)), nếu ∃ hằng số C và số nguyên n_0.

Sao cho:

$$\forall n \geq n_0 \text{ ta luôn có: } f(n) \leq Cg(n)$$

(Nói cách khác là g(n) là hàm chặn trên của f(n) từ một chỉ số nào đó trở đi).

Rõ ràng, trong quá trình đánh giá thuật toán, nếu có g(n) nhỏ nhất thì đó là sự đánh giá chính xác nhất.

Có thể thấy rằng, bài toán tìm g(n) nhỏ nhất khá phức tạp.

Bây giờ, chúng ta đưa ra độ phức tạp thời gian là hàm nhiều biến. Giả sử $T(n_1,... , n_k)$ là độ phức tạp thời gian của thuật toán T và $T(n_1,... , n_k) = O(g(n_1,....n_k))$. Khi đó chúng ta hiểu rằng tồn

tại các số C , $n_{01}...n_{0k}$ sao cho với mọi $n_1 \geq n_{01}$, ..., $n_k \geq n_{0k}$

$$T(n_1,...,n_k) \leq Cg(n_1,..., n_k)$$

Ví dụ: Đầu vào là R = {a_1,....., a_n},

r = {h_1,..., h_m}

Chúng ta có $T(n,m) = O(g(n,m))$

Trong trường hợp có nhiều đối số thì phức tạp thời gian được tính theo đối số có giá trị lớn nhất.

Ví dụ: $T(n,m) = O(n^2+2^m)$. **Khi đó độ phức tạp thời gian của thuật toán T là hàm số mũ.**

Việc đánh giá như trên gọi là độ phức tạp thời gian tồi nhất.

Trong thực tế có nhiều cách đánh giá độ phức tạp thời gian. Ví dụ như độ phức tạp thời gian trung bình. Độ phức tạp này gắn với nhiều độ đo khác nhau (độ đo xác suất). Giáo trình này đánh giá độ phức tạp thời gian theo cách tồi nhất (tìm g(n) chặn trên).

- Giả sử một độ phức tạp thuật toán chia làm nhiều đoạn, mỗi đoạn có độ phức tạp tương ứng là $T_1(n),...,T_q(n)$.

Khi đó chúng ta đặt $T(n) = O(\max(T_1(n),..., T_q(n))$.

- Nếu T_i là hàm nhiều biến: $T_1(n_1,...,n_m)$, ..., $T_q(n_1,...,n_m)$, thì lúc đó $T_1(n_1,..., n_m) = O(\max(T_1(n_1,..., n_m),..., T_q(n_1,..., n_m))$.

- Giả sử T là thuật toán giải quyết bài toán A.

Ta chia hai khúc T_1 và T_2 và T_2 lồng trong T_1. Khi đó $T(n) = O(T_1(n). T_2(n))$.

Ví dụ: x:= 3;

x:= x + 1.

Dễ thấy độ phức tạp T(n) = O(c) (hay O(1)).

Ví dụ: x:= 3;

For i:= 1 to n do x:= x+1

Thì: T(n) = O(cn).

Ví dụ: For i: = 1 to n do

For j: = 1 to n do x:= x+1

$T(n) = O(c.n.n) = O(c.n^2)$

Trong trường hợp thuật toán có các đoạn lồng thắt vào nhau thì độ phức tạp là tích (Thể hiện bài toán có những toán tử lặp chu trình).

- Trong thuật toán chia thành nhiều đoạn. Có đoạn lồng thắt của đoạn khác: tính tích, có đoạn rời rạc: tính max.

Thông thường người ta chia các bài toán thành ba lớp. Đó là lớp bài toán được giải quyết bằng một thuật toán có độ phức tạp là hàm mũ, lớp bài toán NP - đầy đủ và lớp bài toán được giải quyết bằng một thuật toán có độ phức tạp là hàm đa thức.

- Đối với lớp bài toán được giải bằng thuật toán là hàm mũ và lớp bài toán NP - đầy đủ (Thường gọi là các bài toán không khả thi) thực tế trong tin học các bài toán này không có khả năng thực hiện vì thời gian tính quá lớn. Khi đó, chúng ta phải tách bài toán thành các dạng riêng biệt, và cố gắng đưa nó về lớp bài toán có độ phức tạp là hàm đa thức.

- Đối với các bài toán được giải bằng thuật toán có độ phức tạp là hàm đa thức, chúng ta cố gắng giảm số mũ k xuống (gần sát tuyến tính (mũ 1)).

Để hạ k (tăng tốc độ) thông thường người ta dùng cấu trúc dữ liệu, sử dụng ngôn ngữ gần ngôn ngữ máy.

Thông thường người ta coi các thông số vào (input) bình đẳng với nhau.

Tài liệu tham khảo

[1] Armstrong W.W. Dependency Structures of Database Relationships. Information Processing 74, Holland publ. Co. (1974), 580-583.

[2] Beeri C. Bernstein P.A. Computational problems related to the design of normal form relational schemas. ACM trans on Database Syst. 4.1 (1979), 30-59

[3] Beeri C. Dowd M., Fagin R.. Staman R. On the structure of Armstrong relations for Functional Dependencies . J.ACM 31,1 (1984), 30-46.

[4] Codd E. F. A relational model for large shared data banks. Communications ACM 13 (1970), 377-387.

[5] Demetrovics J., Logical and structural Investigation of Relational Datamodel. MTA - SZTAKI Tanulmanyok, Budapest, 114 (1980), 1-97.

[6] Demetrovics J., Libkin, L. Functional dependencies in relational databases : A Lattice point of view. Discrete Aplied Mathematics 40 (1992), 155-185.

[7] Demetrovics J., Thi V.D. Some results about functional dependencies. Acta Cybernetica 8,3 (1988), 273-278.

[8] Demetrovics J., Thi V.D. Relations and minimal keys. Acta Cybernetica 8,3 (1988), 279-285.

[9] Demetrovics J., Thi V.D. On keys in the Relational Datamodel. Inform. Process Cybern. EIK 24, 10 (1988), 515 - 519

[10] Demetrovics J., Thi V.D. Algorithm for generating Armstrong relations and inferring functional dependencies in the relational datamodel. Computers and Mathematics with Applications. Great Britain. 26,4(1993), 43 - 55.

[11] Demetrovics J., Thi V.D. Some problems concerning Keys for relation Schemes and Relationals in the Relational Datamodel. Information Processing Letters. North Holland 46,4(1993),179-183

[12] Demetrovics J., Thi. V.D. Some Computational Problems Related to the functional Dependency in the Relational Datamodel. Acta Scientiarum Mathematicarum 57, 1 - 4 (1993), 627 - 628.

[13] Demetrovics J., Thi V.D. Armstrong Relation, Functional Dependencies and Strong Dependencies. Comput. and AI. (submited for publication).

[14] Demetrovics J., Thi V.D. Keys, antikeys and prime attributes. Ann. Univ. Scien. Budapest Sect. Comput. 8 (1987), 37 - 54.

[15] Demetrovics J., Thi V.D. On the Time Complexity of Algorithms Related to Boyce-Codd Normal Forms. J. Serdica, the Bulgarian Academy of Sciencies, No.19 (1993), 134 - 144.

[16] Demetrovics J., Thi V.D. Generating Armstrong Relations for Relation schemes and Inferring Functional Dependencies from Relations.

International Jounal on Information Theories and Applications, ITA-93, 1, 4 (1993), 3 - 12.

[17] Demetrovics J., Thi V.D. Some problems concerning Armstrong relations of dual schemes and

relation schemes in the relational datamodel. Acta Cybernetica 11, 1-2 (1993), 35 - 47.

[18] Demetrovics J., Thi V.D. Normal Forms and Minimal Keys in the Relational Datamodel. Acta Cybernetica Vol. 11,3 (1994), 205 - 215.

[19] Demetrovics J., Thi, V.D. Some results about normal forms for functional dependency in the relational datamodel. Discrete Aplied Mathematics 69 (1996), 61 - 74.

[20] Garey M.R., Johnson D.S Computers and Intractability: A Guide to theory of NP - Completeness. Bell Laboratories. W.H Freeman and Company. San Francisco 1979.

[21] Gottlob G. Libkin L. Investigations on Armstrong relations dependency inference, and excluded functional dependencies. Acta Cybernetica Hungary IX/4 (1990), 385 - 402.

[22] Jou J.H, Fischer P.C. The complexity of recognizing 3NF relation schemes . IPL 14 (1982), 187 - 190.

[23] * Libkin L. Direct product decompositions of lattices, closures and relation schemes. Discrete Mathematics, North-Holland, 112 (1993), 119-138.

[24] Lucchesi C.L., Osborn S.L. Candidate keys for relations. J. Comput. Syst. Scien 17,2 (1978), 270 - 279

[25] Maier D. Minimum covers in the relational database model. JACM 27,4 (1980), 664 - 674.

[26] * Mannila H., Raiha K.J. Algorithms for inferring functional dependencies from relations. Data and Knowledge Engineering, North - Holland, V. 12, No. 1 (1994), 83 - 99.

[27] * Mannila H., Raiha K.J. On the complexity of inferring functional dependencies. Discrete Applied Mathematics, North - Holland, 40 (1992), 237 - 243.

[28] * Thalheim B. The number of keys in relational and nested relational databases. Discrete Applied Mathematics, North - Holland, 40 (1992), 265 - 282.

[29] Thi V.D. Investigation on Combinatorial Characterizations Related to Functional Dependency in the Relational Datamodel. MTA-SZTAKI Tanulmanyok. Budapest, 191 (1986), 1 - 157. Ph.D. dissertation.

[30] Thi V.D. Minimal keys and Antikeys. Acta Cybernetica 7.4 (1986), 361 - 371

[31] Thi V.D. Minimal keys and Antikeys. Acta Cybernetica 7, 4 (1986), 361 - 371.

[32] Thi V.D. Strong dependencies and s-semilattices. Acta Cybernetica 7, 2 (1987), 175 - 202.

[33] Thi V.D. Logical dependencies and irredundant relations. Computers and Artificial Intelligence 7 (1988), 165 - 184.

[34] Thi V.D., Anh N.K. Weak dependencies in the relational datamodel. Acta Cybernetica 10, 1-2 (1991), 93 - 100.

[35] Thi V.D., Thanh L.T. Some remark on Functional Dependencies in the relational Datamodel J. Acta Cybernetica, Hungary Vol. 11, 4 (1994), 345 - 352.

[36] Thi V.D. On the equivalent descriptions of family of functional dependencies in the relational datamodel. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 11, 4 (1995), 40 - 50.

[37] Thi V.D. Some Computational problems related to normal forms. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 13, 1 (1997), 53 - 65.

[38] Thi V.D. On the nonkeys. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 13, 1 (1997), 11 - 15.

[39] Thi V.D. Some results about hypergraph. J. Computer Science and Cybernetic, Hanoi Vietnam, Vol 13, 2 (1997), 8 - 15.

[40] Tsou D.M., Fischer P.C. Decomposition of a relation scheme into Boyce-Codd normal form. SIGACT NEWS 14 (1982), 23 - 29.

[41] Ullman J.D. Principles of database and knowledgw base systems. Computer Science Press, Second Edition (1992).

[42] Yannakakis M., Paradimitriou C. Algebraic dependencies. J. Comp. Syst. Scien. 25 (1982), 2 - 41.

[43] Yu C.T., Johnson D.T. On the complexity of finding the set of candidate keys for a given set of functional dependencies. IPL 5, 4 (1976), 100 - 101.

[44] Zaniolo C. Analysis and design of relational schemata for database systems. Ph. D. UCLA (1976).

[45] Collongues A., Hugues J., Laroche B. MERISE - Phương pháp thiết kế hệ thống thông tin tin học hoá phục vụ quản lí doanh nghiệp (Bản dịch). Nhà xuất bản Khoa học kĩ thuật, 1994.

[46] Hồ Sĩ Khoa. Các phương pháp xây dựng các mô hình khái niệm dữ liệu

[47] Các tài liệu hướng dẫn sử dụng hệ MEGA

[48] Demetrovics J., Denev. , Pavlov R. Cơ sở toán của khoa học tính toán. Hungary, 1985.

Mục lục

Trang

Chương mở đầu 9

Chương 2
Các kiến thức cơ bản về cơ sở dữ liệu

2.1. Khái quát về mô hình dữ liệu
2.2. Các khái niệm cơ bản và hệ tiên đề Armstrong
2.3. Họ phụ thuộc hàm và các mô tả tương đương
2.4. Các thuật toán liên quan đến các khoá
2.5. Mối liên hệ giữa quan hệ Armstrong và sơ đồ quan hệ

Chương 3
Các dạng chuẩn
và các thuật toán liên quan

3.1. Các khái niệm chung

3.2. Dạng chuẩn 2 (2NF)
3.3. Dạng chuẩn 3 (3NF)
3.4. Dạng chuẩn Boyce - Codd (BCNF)
3.5. Các thuật toán liên quan
3.6. Dạng chuẩn của các hệ khoá
3.7. Ví dụ

Chương 4
Các phép toán xử lí bảng

4.1. Các phép toán cơ bản
4.2. Các phép toán khác
4.3. Các ví dụ

Chương 5
Một số áp dụng mô hình dữ liệu trong các hệ quản trị cơ sở dữ liệu (QTCSDL) hiện có

5.1. Mô tả chung
5.2. Những khái niệm cơ bản
5.3. Mối quan hệ giữa các thực thể
5.4. Các dạng chuẩn trong các hệ QTCSDL hiện có

Chương 6

Một số công đoạn xây dựng
các dự án thiết kế tổng thể các hệ thống
cơ sở dữ liệu hiện nay

6.1. Khảo sát thông tin
6.2. Thiết kế mô hình dữ liệu
6.3. Kiểm soát và chuẩn hoá mô hình

Chương 7
Thuật toán và độ phức tạp

7.1. Khái niệm thuật toán
7.2. Độ phức tạp thuật toán

Tài liệu tham khảo

Giáo Trình

Cơ Sở Dữ Liệu

LỜI MỞ ĐẦU

Giáo trình cơ sở dữ liệu này được biên soạn theo chương trình đào tạo chuyên ngành tin học ở bậc đại học và cao đẳng của Bộ Giáo Dục Đào Tạo. Giáo trình trình bày những vấn đề cốt lõi nhất của môn cơ sở dữ liệu. Các bài học được trình bày ngắn gọn, có nhiều ví dụ minh hoạ. Cuối mỗi chương đều có bài tập để sinh viên luyện tập.Cuối giáo trình còn có một số đề thi trong những năm gần đây.

Giáo trình này có thể giúp các sinh viên trong việc học môn cơ sở dữ liệu ở bậc cao đẳng, đại học cũng như trong các kỳ thi tốt nghiệp Đại Học, Cao đẳng, trong các kỳ thi liên thông. Chúng tôi mong rằng các sinh viên tự tìm hiểu trước mỗi vấn đề và kết hợp với bài giảng trên lớp của giáo viên để việc học môn này đạt hiệu quả.

Trong quá trình giảng dạy và biên soạn giáo trình này, chúng tôi đã nhận được sự động viên của các thầy trong Ban Giám Hiệu nhà trường cũng như những ý kiến của các đồng nghiệp trong khoa Điện Tử - Tin Học. Chúng tôi xin chân thành cảm ơn và hy vọng rằng giáo trình này sẽ giúp cho việc dạy và học môn cơ sở dữ liệu của trường chúng ta ngày càng tốt hơn.

TP. Hồ Chí Minh, Ngày 01 tháng 01 năm 2005

KHOA ĐIỆN TỬ -TIN HỌC

Phan Tấn Quốc

chương 1

TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

1.1.MỘT SỐ KHÁI NIỆM CƠ BẢN

1.1.1.Định nghĩa Cơ Sở Dữ Liệu (Data Base)

Cơ sở dữ liệu (CSDL) là một hệ thống các thông tin có cấu trúc được lưu trữ trên các thiết bị như băng từ, đĩa từ,… để có thể thoả mãn yêu cầu khai thác đồng thời của nhiều người sử dụng.

CSDL gắn liền với đại số, logic toán và một số lĩnh vực khác.

1.1.2.Ưu điểm của cơ sở dữ liệu

-Giảm sự trùng lắp thông tin xuống mức thấp nhất và do đó bảo đảm được tính nhất quán và toàn vẹn dữ liệu.

-Đảm bảo dữ liệu có thể truy xuất theo nhiều cách khác nhau.

-Khả năng chia sẻ thông tin cho nhiều người sử dụng.

1.1.3.Những vấn đề mà CSDL cần phải giải quyết

-Tính chủ quyền của dữ liệu

Tính chủ quyền của dữ liệu được thể hiện ở phương diện an toàn dữ liệu, khả năng biểu diễn các mối liên hệ ngữ nghĩa của dữ liệu và tính chính xác của dữ liệu. Điều này có nghĩa là người khai thác CSDL phải có nhiệm vụ cặp nhật các thông tin mới nhất của CSDL.

-Tính bảo mật và quyền khai thác thông tin của người sử dụng

Do có nhiều người được phép khai thác dữ liệu một cách đồng thời, nên cần thiết phải có một cơ chế bảo mật và phân quyền hạn khai thác CSDL. Các hệ điều hành nhiều người sử dụng hay hệ điều hành mạng cục bộ đều có cung cấp cơ chế này.

-Tranh chấp dữ liệu

Nhiều người được phép truy nhập cùng một lúc vào tài nguyên dữ liệu của CSDL với những mục đích khác nhau, do đó cần thiết phải có một cơ chế ưu tiên khi truy nhập dữ liệu. Cơ chế ưu tiên có thể được thực hiện bằng việc cấp quyền ưu tiên cho từng người khai thác.

-Đảm bảo an toàn dữ liệu khi có sự cố

Việc quản lý dữ liệu tập trung có thể làm tăng khả năng mất mát hoặc sai lệch thông tin khi có sự cố như mất điện đột xuất, hay một phần đĩa lưu trữ CSDL bị hư,… một số hệ điều hành mạng có cung cấp dịch vụ sao lưu ảnh đĩa cứng, tự động kiểm tra và khắc phục lỗi khi có sự cố. Tuy nhiên, bên cạnh dịch vụ của hệ điều hành, để đảm bảo CSDL luôn ổn định, một CSDL nhất thiết phải có một cơ chế khôi phục dữ liệu khi có các sự cố bất ngờ xảy ra.

1.1.4.Các đối tượng sử dụng CSDL

-Những người sử dụng CSDL không chuyên về lĩnh vực tin học và CSDL.

-Các chuyên viên CSDL biết khai thác CSDL Những người này có thể xây dựng các ứng dụng khác nhau, phục vụ cho các mục đích khác nhau trên CSDL.

-Những người quản trị CSDL, đó là những người hiểu biết về tin học, về các hệ quản trị CSDL và hệ thống máy tính. Họ là người tổ chức CSDL, do đó họ phải nắm rõ các vấn đề kỹ thuật về CSDL để có thể phục hồi CSDL khi có sự cố. Họ là những người cấp quyền hạn khai thác CSDL, do vậy họ có thể giải quyết được các vấn đề tranh chấp dữ liệu nếu có.

1.1.5. Hệ Quản Trị Cơ Sở Dữ Liệu (Data Base Management System)

Để giải quyết tốt những vấn đề mà cách tổ chức CSDL đặt ra như đã nói ở trên, cần thiết phải có những phần mềm chuyên dùng để khai thác chúng. Những phần mềm này được gọi là các hệ quản trị CSDL. Các hệ quản trị CSDL có nhiệm vụ hỗ trợ cho các nhà phân tích thiết kế CSDL cũng như những người khai thác CSDL. Hiện nay trên thị trường phần mềm đã có những hệ quản trị CSDL hỗ trợ được nhiều tiện ích như: MS Access, Visual Foxpro, SQL Server Oracle, …

Mỗi hệ quản trị CSDL đều được cài đặt dựa trên một mô hình dữ liệu cụ thể. Dù là dựa trên mô hình dữ liệu nào, một hệ quản trị CSDL cũng phải hội đủ các yếu tố sau:

***-Ngôn ngữ giao tiếp giữa người sử dụng và CSDL*,** bao gồm :

Ngôn ngữ mô tả dữ liệu: Để cho phép khai báo cấu trúc của CSDL, khai báo các mối liên hệ của dữ liệu và các quy tắc quản lý áp đặt lên các dữ liệu đó.

Ngôn ngữ thao tác dữ liệu: Cho phép người sử dụng có thể cập nhật dữ liệu (thêm/sửa/xoá)

Ngôn ngữ truy vấn dữ liệu: Cho phép người khai thác sử dụng để truy vấn các thông tin cần thiết trong CSDL

Ngôn ngữ quản lý dữ liệu: Cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu, khai báo bảo mật thông tin và cấp quyền hạn khai thác CSDL cho người sử dụng.,…

-Từ điển dữ liệu:

Dùng để mô tả các ánh xạ liên kết, ghi nhận các thành phần cấu trúc của CSDL, các chương trình ứng dụng, mật mã, quyền hạn sử dụng,…

-Cơ chế giải quyết vấn đề tranh chấp dữ liệu:

Mỗi hệ quản trị CSDL cũng có thể cài đặt một cơ chế riêng để giải quyết các vấn đề này. Một số biện pháp sau đây thường được sử dụng: thứ nhất: cấp quyền ưu tiên cho từng người sử dụng; thứ hai: Đánh dấu yêu cầu truy xuất dữ liệu, phân chia thời gian, người nào có yêu cầu trước thì có quyền truy xuất dữ liệu trước,…

-Hệ quản trị CSDL cũng phải có cơ chế sao lưu (backup) và phục hồi (restore) dữ liệu khi có sự cố xảy ra.

Điều này có thể thực hiện sau một thời gian nhất định hệ quản trị CSDL sẽ tự động tạo ra một bản sao CSDL, cách này hơi tốn kém, nhất là đối với CSDL lớn.

***-Hệ quản trị CSDL phải cung cấp một giao diện thân thiện*, dễ sử dụng.**

1.1.6.Các Ứng Dụng Của Cơ Sở Dữ Liệu

Hiện nay, hầu như CSDL gắn liền với mọi ứng dụng của tin học; chẳng hạn như việc quản lý hệ thống thông tin trong các cơ quan nhà nước, việc lưu trữ và xử lý thông tin trong các doanh nghiệp, trong các lĩnh vực nghiên cứu

khoa học, trong công tác giảng dạy, cũng như trong việc tổ chức thông tin đa phương tiện,…

1.2.CÁC MÔ HÌNH DỮ LIỆU

Mô hình dữ liệu là sự trừu tượng hoá môi trường thực. Mỗi loại mô hình dữ liệu đặc trưng cho một cách tiếp cận dữ liệu khác nhau của những nhà phân tích thiết kế CSDL. Mỗi loại mô hình dữ liệu đều có những ưu điểm và những mặt hạn chế của nó, nhưng vẫn có những mô hình dữ liệu nổi trội và được nhiều người quan tâm nghiên cứu.

Sau đây chúng ta sẽ điểm qua lịch sử phát triển của các mô hình dữ liệu.

Vào những năm sáu mươi, thế hệ đầu tiên của CSDL ra đời dưới dạng mô hình thực thể kết hợp, mô hình mạng và mô hình phân cấp.

Vào những năm bảy mươi, thế hệ thứ hai của CSDL ra đời. Đó là mô hình dữ liệu quan hệ do EF. Codd phát minh. Mô hình này có cấu trúc logic chặt chẽ. Đây là mô hình đã và đang được sử dụng rộng khắp trong công tác quản lý trên phạm vi toàn cầu. Việc nghiên cứu mô hình dữ liệu quan hệ nhằm vào lý thuyết chuẩn hoá các quan hệ và là một công cụ quan trọng trong việc phân tích thiết kế các hệ CSDL hiện nay. Mục đích của nghiên cứu này nhằm bỏ đi các phần tử không bình thường của quan hệ khi thực hiện các phép cập nhật, loại bỏ các phần tử dư thừa.

Sang thập kỷ tám mươi, mô hình CSDL thứ ba ra đời, đó là mô hình cơ sở dữ liệu hướng đối tượng, mô hình cơ sở dữ liệu phân tán, mô hình cơ sở dữ liệu suy diễn,…

Trong phần tiếp theo sau đây, chúng tôi sẽ trình bày về mô hình dữ liệu tiêu biểu nhất để thiết kế (bước đầu) một ứng dụng tin học đó là mô hình thực thể kết hợp. Trong các chương còn lại của giáo trình này chúng tôi sẽ trình bày về mô hình dữ liệu quan hệ.

1.3.MÔ HÌNH THỰC THỂ KẾT HỢP

Hiện nay mô hình dữ liệu quan hệ thường được dùng trong các hệ quản trị CSDL, đây là mô hình dữ liệu ở mức vật lý. Để thành lập được mô hình này, thường là phải dùng mô hình dữ liệu ở mức quan niệm để đặc tả, một trong

những mô hình ở dạng đó là mô hình thực thể kết hợp (sau đó mới dùng một số quy tắc để chuyển hệ thống từ mô hình này về mô hình dữ liệu quan hệ – các quy tắc này sẽ được nói đến trong mục 2.2).

Sau đây là các khái niệm của mô hình thực thể kết hợp.

1.3.1. Thực Thể (entity)

Thực thể là một sự vật tồn tại và phân biệt được, chẳng hạn sinh viên Nguyễn Văn Thành, lớp Cao Đẳng Tin Học 2A, môn học Cơ Sở Dữ Liệu, xe máy có biển số đăng ký 52-0549,… là các ví dụ về thực thể.

1.3.2. Thuộc tính (attribute)

Các đặc điểm riêng của thực thể gọi là các *thuộc tính.*

Chẳng hạn các thuộc tính của sinh viên Nguyễn Văn Thành là:mã số sinh viên, giới tính, ngày sinh, hộ khẩu thường trú, lớp đang theo học, …

(Trong giáo trình này, tên thuộc tính được viết bằng chữ in hoa)

1.3.3.Loại thực thể (entity type)

Là tập hợp các thực thể có cùng thuộc tính. Mỗi loại thực thể đều phải được đặt tên sao cho có ý nghĩa. Một loại thực thể được biểu diễn bằng một hình chữ nhật.

Ví dụ các sinh viên có mã sinh viên là ““02CĐTH019”, “02CĐTH519”, “02TCTH465”,… nhóm lại thành một loại thực thể, được đặt tên là Sinhvien chẳng hạn.

Tương tự trong ứng dụng quản lý điểm của sinh viên (sẽ được trình bày ngay sau đây) ta có các loại thực thể như Monhoc, Lop, Khoa,…

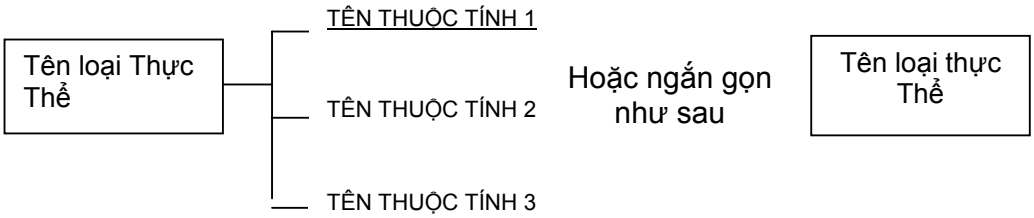
(Trong giáo trình này, tên của loại thực thể được in hoa ký tự đầu tiên, các ký tự còn lại viết thường).

1.3.4.Khoá (key)

Khoá của loại thực thể E là một hay một tập các thuộc tính của E có thể dùng để phân biệt hai thực thể bất kỳ của E.

Ví dụ khoá của loại thực thể Sinhvien là MASV, của Lớp là MALOP, của Khoa là MAKHOA, của Monhoc là MAMH,…

Cần chú ý rằng khi biểu diễn một hệ thống bằng mô hình thực thể kết hợp thì tên của các loại thực thể phải khác nhau. Trong danh sách các thuộc tính của một loại thực thể thì tập thuộc tính khoá thường được gạch dưới liền nét. Nếu một hệ thống có nhiều loại thực thể, để đơn giản hoá mô hình, người ta có thể chỉ nêu tên các loại thực thể; còn các thuộc tính của loại thực thể được liệt kê riêng.



Ví dụ 1.1:

Bài toán quản lý điểm của sinh viên được phát biểu sơ bộ như sau:

Mỗi sinh viên cần quản lý các thông tin như: họ và tên (HOTENSV),ngày tháng năm sinh(NGAYSINH), giới tính (NU), nơi sinh(NƠISINH), hộ khẩu thường trú (TINH). Mỗi sinh viên được cấp một mã số sinh viên duy nhất (MASV) để phân biệt với mọi sinh viên khác của trường, mỗi sinh viên chỉ thuộc về một lớp nào đó.

Mỗi lớp học có một mã số lớp (MALOP)duy nhất để phân biệt với tất cả các lớp học khác trong trường: có một tên gọi (TENLOP) của lớp, mỗi lớp chỉ thuộc về một khoa.

Mỗi khoa có một tên gọi (TENKHOA) và một mã số duy nhất (MAKHOA) để phân biệt với các khoa khác.

Mỗi môn học có một tên gọi (TENMH) cụ thể, được học trong một số đơn vị học trình (DONVIHT))và ứng với môn học là một mã số duy nhất (MAMH) để phân biệt với các môn học khác.

Mỗi giảng viên cần quản lý các thông tin: họ và tên(HOTENGV), cấp học vị (HOCVI), thuộc một chuyên ngành (CHUYENNGANH) và được gán cho một mã số duy nhất gọi là mã giảng viên(MAGV) để phân biệt với các giảng viên

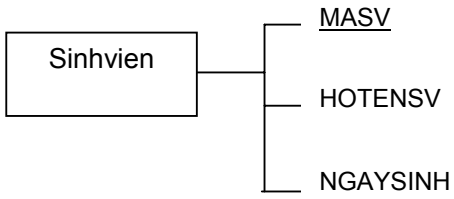
khác. Mỗi giảng viên có thể dạy nhiều môn ở nhiều khoa, nhưng chỉ thuộc về sự quản lý hành chính của một khoa.

Mỗi sinh viên với một môn học được phép thi tối đa 3 lần, mỗi lần thi (LANTHI), điểm thi (DIEMTHI).

Mỗi môn học ở mỗi lớp học chỉ phân công cho một giảng viên dạy (tất nhiên là một giảng viên thì có thể dạy nhiều môn ở một lớp).

Với bài toán trên thì các loại thực thể cần quản lý như: Sinhviên, Mônhọc, Khoa, Lớp, Giảngviên.

Ví dụ với loại thực thể Sinhviên thì cần quản lý các thuộc tính như: MASV,HOTENSV, NGAYSINH,… và ta có thể biểu diễn như sau:



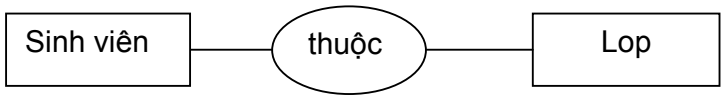
1.3.5.Mối Kết Hợp (relationship)

Mối kết hợp diễn tả sự liên hệ giữa các loại thực thể trong một ứng dụng tin học.

Ví dụ mối kết hợp giữa hai loại thực thể Sinhviên và Lop, mối kết hợp giữa Sinhviên với Mônhọc,...

Mối kết hợp được biểu diễn bằng một hình elip và hai bên là hai nhánh gắn kết với các loại thực thể (hoặc mối kết hợp) liên quan, tên mối kết hợp thường là: *thuộc, gồm , chứa,...*

Chẳng hạn giữa hai loại thực thể Lớp và Khoa có mối kết hợp “thuộc” như sau:

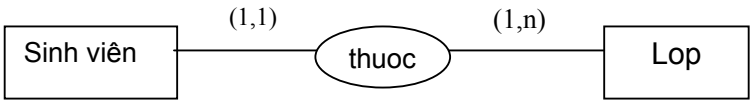


Bản số của mối kết hợp:

Bản số của một nhánh R trong mối kết hợp thể hiện số lượng các thực thể thuộc thực thể ở nhánh “bên kia” có liên hệ với một thực thể của nhánh R.

Mỗi bản số là một cặp số (min,max), chỉ số lượng tối thiểu và số lượng tối đa của thực thể khi tham gia vào mối kết hợp đó.

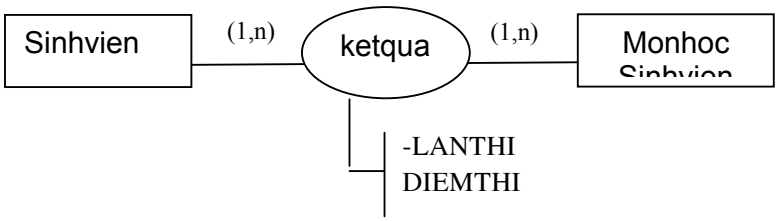
Ví dụ:



Có nghĩa là: “mỗi sinh viên thuộc một và chỉ một lớp nên bản số bên nhánh Sinhviên là (1,1), mỗi lớp có 1 đến n sinh viên nên bản số bên nhánh Lop là (1,n)”

Trong một số trường hợp đặc biệt, mối kết hợp có thể có các thuộc tính đi kèm và do đó chúng thường được đặt tên ý với nghĩa đầy đủ hơn.

Ví dụ giữa hai loại thực thể Monhoc và Sinhvien có mối kết hợp ketqua với ý nghĩa: “mỗi sinh viên ứng với mỗi lần thi của mỗi môn học có một kết quả điểm thi duy nhất”.



Khoá của mối kết hợp: là hợp của các khoá của các loại thực thể liên quan.

Chẳng hạn như thuộc tính MAGV là khoá của loại thực thể Giangvien, MALOP là thuộc tính khoá của loại thực thể Lop, MAMH là thuộc tính khoá của loại thực thể Monhoc, do đó mối kết hợp phancong (giữa các loại thực thể Giangvien,Lop,Monhoc) có khoá là {MAGV,MAMH,MALOP} - phancong là mối kết hợp 3 ngôi.

(Trong giáo trình này, tên của mối kết hợp được viết toàn bằng chữ thường).

Việc thành lập mô hình thực thể kết hợp cho một ứng dụng tin học có thể tiến hành theo các bước sau:

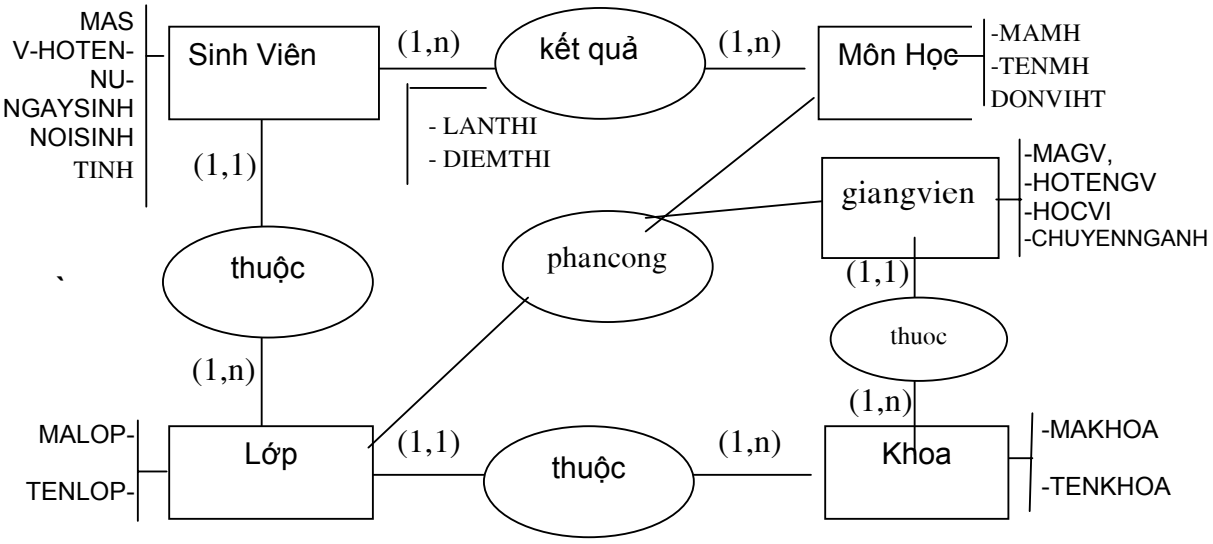
b1.Xác định danh sách các loại thực thể

b2.Xác định các mối kết hợp giữa các loại thực thể để phác thảo mô hình.

b3.Lập bản số của các mối kết hợp.

Để kết thúc chương này, chúng tôi sẽ lập mô hình thực thể kết hợp cho bài toán quản lý điểm của sinh viên đã được nêu trong ví dụ 1.1

Ví dụ 1.2:



BÀI TẬP

Dựa vào các phân tích sơ bộ dưới đây, hãy lập mô hình thực thể kết hợp (gồm loại thực thể, mối kết hợp, bản số, thuộc tính của loại thực thể, khoá của loại thực thể) cho mỗi bài toán quản lý sau:

1.1. QUẢN LÝ SỐ LƯỢNG NGÀY CÔNG CỦA CÁC NHÂN VIÊN

Để quản lý việc phân công các nhân viên tham gia vào xây dựng các công trình. Công ty xây dựng ABC tổ chức quản lý như sau:

Cùng lúc công ty có thể tham gia xây dựng nhiều công trình, mỗi công trình có một mã số công trình duy nhất (MACT), mỗi mã số công trình xác định các thông tin như: Tên gọi công trình (TENCT), địa điểm(ĐIAĐIEM), ngày công trình được cấp giấy phép xây dựng (NGAYCAPGP), ngày khởi công (NGAYKC), ngày hoàn thành (NGAYHT)

Mỗi nhân viên của công ty ABC có một mã số nhân viên(MANV) duy nhất, một mã số nhân viên xác định các thông tin như: Họ tên (HOTEN), ngày sinh(NGSINH), phái (PHAI), địa chỉ (ĐIACHI),phòng ban, …

Công ty phân công các nhân viên tham gia vào các công trình, mỗi công trình có thể được phân cho nhiều nhân viên và mỗi nhân viên cùng lúc cũng có thể tham gia vào nhiều công trình. Với mỗi công trình một nhân viên có một số lượng ngày công (SLNGAYCONG) đã tham gia vào công trình đó.

Công ty có nhiều phòng ban(Phòng kế toán, phòng kinh doanh, phòng kỹ thuật, phòng tổ chức, phòng chuyên môn, Phòng phục vụ,…). Mỗi phòng ban có một mã số phòng ban(MAPB) duy nhất, một phòng ban ứng với một tên phòng ban(TENPB).

1.2. QUẢN LÝ VIỆC MƯỢN/TRẢ SÁCH Ở MỘT THƯ VIỆN

Một thư viện tổ chức việc cho mượn sách như sau:

Mỗi quyển sách được đánh một mã sách (MASH) dùng để phân biệt với các quyển sách khác (giả sử nếu một tác phẩm có nhiều bản giống nhau hoặc có nhiều tập thì cũng xem là có mã sách khác nhau), mỗi mã sách xác định các thông tin khác như : tên sách (TENSACH), tên tác giả (TACGIA), nhà xuất bản (NHAXB), năm xuất bản (NAMXB).

Mỗi độc giả được thư viện cấp cho một thẻ thư viện, trong đó có ghi rõ mã độc giả (MAĐG), cùng với các thông tin khác như : họ tên (HOTEN), ngày sinh (NGAYSINH), địa chỉ (ĐIACHI), nghề nghiệp(NGHENGHIEP).

Cứ mỗi lượt mượn sách, độc giả phải đăng ký các quyển sách cần mượn vào một phiếu mượn, mỗi phiếu mượn có một số phiếu mượn (SOPM) khác nhau, mỗi phiếu mượn xác định các thông tin như: ngày mượn sách (NGAYMUON), mã độc giả. Các các quyển sách trong cùng một phiếu mượn không nhất thiết phải trả trong một lần. Mỗi quyển sách có thể thuộc nhiều phiếu mượn khác nhau (tất nhiên là tại các thời điểm khác nhau).

1.3. QUẢN LÝ LỊCH DẠY CỦA GIÁO VIÊN

Để quản lý lịch dạy của các giáo viên và lịch học của các lớp, một trường tổ chức như sau:

Mỗi giáo viên có một mã số giáo viên (MAGV) duy nhất, mỗi MAGV xác định các thông tin như: họ và tên giáo viên (HOTEN), số điện thoại (DTGV). Mỗi giáo viên có thể dạy nhiều môn cho nhiều khoa nhưng chỉ thuộc sự quản lý hành chánh của một khoa nào đó.

Mỗi môn học có một mã số môn học (MAMH) duy nhất, mỗi môn học xác định tên môn học(TENMH). Ứng với mỗi lớp thì mỗi môn học chỉ được phân cho một giáo viên.

Mỗi phòng học có một số phòng học (PHONG) duy nhất, mỗi phòng có một chức năng (CHUCNANG); chẳng hạn như phòng lý thuyết, phòng thực hành máy tính, phòng nghe nhìn, xưởng thực tập cơ khí,…

Mỗi khoa có một mã khoa (MAKHOA) duy nhất, mỗi khoa xác định các thông tin như: tên khoa (TENKHOA), điện thoại khoa(DTKHOA).

Mỗi lớp có một mã lớp (MALOP) duy nhất, mỗi lớp có một tên lớp (TENLOP), sĩ số lớp (SISO). Mỗi lớp có thể học nhiều môn của nhiều khoa nhưng chỉ thuộc sự quản lý hành chính của một khoa nào đó.

Hàng tuần, mỗi giáo viên phải lập lịch báo giảng cho biết giáo viên đó sẽ dạy những lớp nào, ngày nào (NGAYDAY), môn gì?, tại phòng nào, từ tiết nào (TUTIET) đến tiết nào (ĐENTIET),tựa đề bài dạy (BAIDAY), những ghi chú (GHICHU) về các tiết dạy này, đây là giờ dạy lý thuyết (LYTHUYET) hay thực hành - giả sử nếu LYTHUYET=1 thì đó là giờ dạy thực hành và nếu LYTHUYET=2 thì đó là giờ lý thuyết, một ngày có 16 tiết, sáng từ tiết 1 đến tiết 6, chiều từ tiết 7 đến tiết 12, tối từ tiết 13 đến 16.

Một số yêu cầu của hệ thống này như:: Lập lịch dạy trong tuần của các giáo viên. Tổng số dạy của các giáo viên theo từng môn cho từng lớp, ….

1.4. QUẢN LÝ HỌC VIÊN Ở MỘT TRUNG TÂM TIN HỌC

Trung tâm tin học KTCT thường xuyên mở các lớp tin học ngắn hạn và dài hạn. Mỗi lớp ngắn hạn có một hoặc nhiều môn học (chẳng hạn như lớp Tin học văn phòng thì có các môn : Word, Power Point, Excel, còn lớp lập trình Pascal thì chỉ học một môn Pascal). Các lớp dài hạn (chẳng hạn như lớp kỹ thuật viên đồ hoạ đa truyền thông, lớp kỹ thuật viên lập trình, lớp kỹ thuật viên phần cứng và mạng,…) thì có thể học nhiều học phần và mỗi học phần có thể có nhiều môn học.

Mỗi học viên có một mã học viên(MAHV) duy nhất và chỉ thuộc về một lớp duy nhất (nếu học viên cùng lúc học nhiều lớp thì ứng với mỗi lớp, học viên đó có một MAHV khác nhau). Mỗi học viên xác định họ tên (HOTEN), ngày sinh (NGAYSINH),nơi sinh (NOISINH), phái nam hay nữ (PHAI), nghề nghiệp (NGHENGHIEP) - nghề nghiệp là SINH VIÊN, GIÁO VIÊN, KỸ SƯ, HỌC SINH, BUÔN BÁN,…

Trung tâm KTCT có nhiều lớp, mỗi lớp có một mã lớp duy nhất (MALOP), mỗi lớp xác định các thông tin: tên lớp (TENLOP), thời khoá biểu, ngày khai giảng (NGAYKG), học phí (HOCPHI).

Chú ý rằng tại một thời điểm, trung tâm có thể mở nhiều lớp cho cùng một chương trình học. Với các lớp dài hạn thì ngày khai giảng được xem là ngày bắt đầu của mỗi học phần và HỌC PHÍ là học phí của mỗi học phần, với lớp ngắn hạn thì HỌC PHÍ là học phí của toàn khoá học đó.

Trung tâm có nhiều môn học, mỗi môn học có mã môn học (MAMH) duy nhất, mỗi môn học xác định tên môn học(TENMH), số tiết lý thuyết (SOTIETLT), số tiết thực hành (SOTIETTH).

Mỗi học viên ứng với mỗi môn học có một điểm thi(DIEMTHI) duy nhất. Mỗi lần đóng học phí, học viên sẽ được trung tâm giao cho một phiếu biên lai thu tiền, mỗi biên lai có một số biên lai duy nhất để quản lý.

Một số yêu cầu của hệ thống này như::Lập danh sách những học viên khai giảng khoá ngày nào đó. Lập danh sách các học viên của một lớp ? Cho biết số lượng học viên của mỗi lớp khai giảng khoá ngày nào đó ?

1.5. QUẢN LÝ COI THI TUYỂN SINH

Một hội đồng coi thi tuyển sinh có nhiều điểm thi, mỗi điểm thi được đặt tại một trường nào đó. Các điểm thi (DIEMTHISO) được đánh số là điểm thi số 1, điểm thi số 2, điểm thi số 3,…Mỗi điểm thi xác định địa chỉ (DIACHIDIEMTHI). Ví dụ: điểm thi số 1, đặt tại trường PTTH Nguyễn Thị Minh Khai, điểm thi số 2 đặt tại trường PTTH Bùi Thị Xuân,…

Mỗi thí sinh có một số báo danh (SOBD) duy nhất, mỗi số báo danh xác định các thông tin: họ và tên (HOTEN), ngày sinh (NGAYSINH), phái (PHAI), hộ khẩu thường trú (TINH), đối tượng dự thi (DOITUONG), ngành đăng ký thi, khu vực của thí sinh (KHUVUC), số hiệu phòng thi. Ví dụ: thí sinh Vũ Mạnh Cường, có số báo danh là 02978, sinh ngày 12/12/1984, phái nam, hộ khẩu thường trú tại Chợ Gạo - Tiền Giang, thuộc khu vực 1, đối tượng là 5B, đăng ký dự thi vào ngành có mã ngành là 01, thi tại phòng thi 0178, điểm thi số 1.

Mỗi ngành có một mã ngành (MANGANH) duy nhất, mỗi mã ngành xác định tên ngành (TENNGANH)

Mỗi điểm thi có nhiều phòng thi – mỗi phòng thi (PHONGTHI) được đánh số khác nhau ở tất cả các điểm thi. Trong một phòng thi, danh sách các thí sinh được sắp xếp theo thứ tự alphabet (do đó trong một phòng thi có thể có thí sinh của nhiều ngành khác nhau). Mỗi phòng thi có thêm cột ghi chú (GHICHU) - ghi thêm các thông tin cần thiết như phòng thi đó nằm tại dãy nhà nào. Ví dụ phòng thi 0060 nằm ở dãy nhà H lầu 2 - điểm thi số 1 - trường PTTH Bùi Thị Xuân.

Mỗi môn thi có một mã môn thi duy nhất (MAMT), mỗi mã môn thi biết các thông tin như : tên môn thi (TENMT), ngày thi (NGAYTHI), buổi thi (BUOITHI), thời gian làm bài thi được tính bằng phút (PHUT). Thời gian làm bài thi của các môn tối thiểu là 90 phút và tối đa là 180 phút (tuỳ theo kỳ tuyển sinh công nhân, trung cấp, cao đẳng hay đại học)

Mỗi ngành có một mã ngành, chẳng hạn ngành Công Nghệ Thông Tin có mã ngành là 01, ngành Công Nghệ Hoá Thực Phẩm có mã ngành là 10,…

Mỗi đơn vị có cán bộ tham gia vào kỳ thi có một mã đơn vị duy nhất (MADONVI), mã đơn vị xác định tên đơn vị (TENDONVI). Nếu là cán bộ, công nhân viên của trường thì đơn vị là khoa/phòng quản lý cán bộ đó, nếu là giáo viên từ các trường khác thì ghi rõ tên đơn vị đó. Chẳng hạn cán bộ Nguyễn Thanh Liêm đơn vị Khoa Công Nghệ Thông Tin, cán bộ coi thi Nguyễn Thị Tuyết Mai, đơn vị trường PTTH Ngôi Sao - Quận 1,…

Mỗi cán bộ coi thi chỉ làm việc tại một điểm thi nào đó. Mỗi cán bộ có một mã số duy nhất (MACANBO), mỗi MACANBO xác định các thông tin khác như : họ và tên (HOTENCB),

đơn vị công tác, chức vụ (CHUCVU) được phân công tại điểm thi, chẳng hạn chức vụ là điểm trưởng, điểm phó, giám sát, thư ký, cán bộ coi thi, phục vụ,… Ví dụ cán bộ Nguyen Van Thanh đơn vị Khoa Công Nghệ Thông Tin, làm nhiệm vụ thi tại điểm thi số 1, chức vụ là giám sát phòng thi.

chương 2

MÔ HÌNH DỮ LIỆU QUAN HỆ

2.1. CÁC KHÁI NIỆM CƠ BẢN

Mô hình dữ liệu quan hệ (Ralational Data Model)- gọi tắt là mô hình quan hệ, do EF.Codd đề xuất năm 1970. Nền tảng lý thuyết của nó là khái niệm lý thuyết tập hợp trên các quan hệ, tức là tập của các bộ giá trị.

Mô hình dữ liệu quan hệ là mô hình được nghiên cứu nhiều nhất, và thực tiễn đã cho thấy rằng nó có cơ sở lý thuyết vững chắc nhất. Mô hình dữ liệu này cùng với mô hình thức thể kết hợp đang được sử dụng rộng rãi trong việc phân tích và thiết kế CSDL hiện nay.

Sau đây là các khái niệm của mô hình dữ liệu quan hệ.

2.1.1.Thuộc Tính(attribte):

Thuộc tính là các đặc điểm riêng của một đối tượng (*đối tượng* được hiểu như là một loại thực thể ở mô hình thực thể kết hợp), mỗi thuộc tính có một *tên gọi* và phải thuộc về một *kiểu dữ liệu* nhất định.

Kiểu dữ liệu (data type)

Các thuộc tính được phân biệt qua tên gọi và phải thuộc một kiểu dữ liệu nhất định (số, chuỗi, ngày tháng, logic, hình ảnh,…). Kiểu dữ liệu ở đây có thể là kiểu vô hướng hoặc là kiểu có cấu trúc. Nếu thuộc tính có kiểu dữ liệu là vô hướng thì nó được gọi là thuộc tính đơn hay thuộc tính nguyên tố, nếu thuộc tính có kiểu dữ liệu có cấu trúc thì ta nói rằng nó không phải là thuộc tính nguyên tố

Chẳng hạn với sinh viên Nguyễn Văn Thành thì các thuộc tính họ và tên, mã số sinh viên thuộc kiểu chuỗi, thuộc tính ngày sinh thuộc kiểu ngày tháng, hộ khẩu thường trú kiểu chuỗi, thuộc tính hình ảnh kiểu hình ảnh,…

Miền giá trị (domain of values)

Thông thường mỗi thuộc tính chỉ chọn lấy giá trị trong một tập con của kiểu dữ liệu và tập hợp con đó gọi là miền giá trị của thuộc tính đó. Chẳng hạn thuộc tính NỮ có miền giá trị là {nam,nữ}, thuộc tính màu da có miền giá trị là

{da trắng, da vàng, da đen, da đỏ}, thuộc tính điểm thi là các số thuộc tập {0; 1 ; 2;…,10].

Lưu ý rằng nếu không lưu ý đến ngữ nghĩa thì tên của các thuộc tính thường được ký hiệu bằng các chữ cái in hoa đầu tiên trong bảng chữ cái la tinh: A,B,C,D,… Những chữ cái in hoa X,Y,Z,W,… thường dùng thay cho một nhóm nhiều thuộc tính. Đôi khi còn dùng các ký hiệu chữ cái với các chỉ số $A_1,A_2,…,A_n$ để chỉ các thuộc tính trong trường hợp tổng quát hay muốn đề cập đến số lượng các thuộc tính. Tên thuộc tính phải được đặt một cách gợi nhớ, không nên đặt tên thuộc tính quá dài (vì như thế sẽ làm cho việc viết các câu lệnh truy vấn trở nên vất vả hơn), nhưng cũng không nên đặt tên thuộc tính quá ngắn (vì nó sẽ không cho thấy ngữ nghĩa của thuộc tính), đặc biệt không đặt trùng tên hai thuộc tính mang ngữ nghĩa khác nhau thuộc hai đối tượng khác nhau.

Trong nhiều hệ quản trị cơ sở dữ liệu, người ta thường đưa thêm vào miền giá trị của các thuộc tính một giá trị đặc biệt gọi là giá trị rỗng (NULL). Tuỳ theo ngữ cảnh mà giá trị này có thể đặc trưng cho một giá trị không thể xác định được hoặc một giá trị chưa được xác định ở vào thời điểm nhập tin nhưng có thể được xác định vào một thời điểm khác.

2.1.2 Lược Đồ Quan Hệ (relation schema)

Tập tất cả các thuộc tính cần quản lý của một đối tượng cùng với các mối liên hệ giữa chúng được gọi là *lược đồ quan hệ*. Lược đồ quan hệ Q với tập thuộc tính $\{A_1,A_2,...,A_n\}$ được viết là $Q(A_1,A_2,...,A_n)$, ký hiệu $Q^+ = \{A_1,A_2,...,A_n\}$.

Chẳng hạn lược đồ quan hệ Sinhviên với các thuộc tính như đã được liệt kê trong ví dụ 1.1 được viết như sau:

Sinhvien(MASV, HOTENSV, NU, NGAYSINH, NOISINH,TINH, MALOP)

Thường thì khi thành lập một lược đồ quan hệ, người thiết kế gắn cho nó một ý nghĩa nhất định, gọi là *tân từ của lược đồ quan hệ***.** chẳng hạn tân từ của lược đồ quan hệ Sinhvien là: ”Mỗi sinh viên có mỗi MASV duy nhất. Mỗi

MASV xác định các thuộc tính còn lại của sinh viên đó như HOTENSV,NU, NGAYSINH, NOISINH,TINH,MALOP"

Khi phát biểu tân từ cho một lược đồ quan hệ, người thiết kế cần phải mô tả đầy đủ ý nghĩa để người khác tránh hiểu nhầm. Dựa vào tân từ này, người ta xác định được tập khoá, siêu khoá của lược đồ quan hệ (sẽ được trình bày trong những mục kế tiếp).

Nhiều lược đồ quan hệ cùng nằm trong một hệ thống thông tin được gọi là một *lược đồ cơ sở dữ liệu.*

Khái niệm lược đồ quan hệ ứng với khái niệm loại thực thể ở mô hình thực thể kết hợp.

2.1.3.Quan Hệ (relation)

Sự thể hiện của lược đồ quan hệ ở một thời điểm nào đó được gọi là *quan hệ*, rõ ràng là trên một lược đồ quan hệ có thể xác định nhiều quan hệ. Thường ta dùng các ký hiệu như R,S,Q để chỉ các lược đồ quan hệ, còn quan hệ thường được dùng bởi các ký hiệu là r, s,q,…

Về trực quan thì quan hệ (hay bảng quan hệ) như là một bảng hai chiều gồm các dòng và các cột.

Một quan hệ có n thuộc tính được gọi là quan hệ n ngôi.

Để chỉ quan hệ r xác định trên lược đồ quan hệ Q ta có thể viết r(Q).

2.1.4 Bộ (Tuple)

Mỗi *bộ* là những thông tin về một đối tượng thuộc một quan hệ, bộ cũng còn được gọi là mẫu tin.

Thường người ta dùng các chữ cái thường (như t,μ,…) để biểu diễn bộ trong quan hệ, chẳng hạn để nói t là một bộ của quan hệ r thì ta viết $t \in r$.

2.1.5. Siêu Khoá – Khoá (super key- key)

S là siêu khoá **(**super key*)* của Q nếu với r là quan hệ bất kỳ trên Q, t_1,t_2 là hai bộ bất kỳ thuộc r thì $t_1.S \neq t_2.S$.

Một lược đồ quan hệ có thể có một hoặc nhiều siêu khoá.

Chẳng hạn lược đồ quan hệ Sinhvien ở trên có các siêu khoá là: {MASV,HOTENSV},{MASV,HOTENSV,NU},{MASV,HOTENSV,NU,TINH },…

Siêu khoá không chứa một siêu khoá nào khác được gọi là khoá *chỉ định*, trong trường hợp lược đồ quan hệ có nhiều khoá chỉ định (hay khoá nội), thì khoá được chọn để cài đặt gọi là *khoá chính (*trong các phần sau khoá chính được gọi tắt là khoá). Chẳng hạn với lược đồ quan hệ Sinhvien trên có khoá là {MASV}. Thường các thuộc tính khoá được gạch dưới theo kiểu liền nét.

Một thuộc tính được gọi là thuộc tính khoá ngoại nếu nó không là thuộc tính khoá của một lược đồ quan hệ này nhưng lại là thuộc tính khoá của một lược đồ quan hệ khác, chẳng hạn như MALOP là khoá ngoại của lược đồ quan hệ Sinhvien. Thường các thuộc tính khoá ngoại được gạch dưới theo kiểu không liền nét.

Sinhvien(MASV, HOTENSV, NU, NGAYSINH, TINH, MALOP)

Lop(MALOP,TENLOP,MAKHOA)

Ý nghĩa thực tế của khoá là dùng để nhận diện một bộ trong một quan hệ, nghĩa là, khi cần tìm một bộ t nào đó, ta chỉ cần biết giá trị của thành phần khoá của t là đủ để dò tìm và hoàn toàn xác định được nó trong quan hệ.

Trong thực tế đối với các loại thực thể tồn tại khách quan (ví dụ: Sinh viên, Giảng viên, Nhân viên, Hàng hoá,…) người thiết kế cơ sở dữ liệu thường gán thêm cho các lược đồ quan hệ này một thuộc tính giả gọi là mã số để làm khoá (ví dụ: mã số sinh viên, mã số giảng viên, mã số nhân viên, mã số hàng hoá,…). Trong khi đó các lược đồ quan hệ biểu diễn cho sự trừu tượng hoá thường có khoá là một tổ hợp của hai hay nhiều thuộc tính của nó.

Một số hệ quản trị cơ sở dữ liệu hiện nay có tự động kiểm tra tính duy nhất trên khoá chính. Tức là nếu thêm một bộ mới q2 có giá trị khoá chính trùng với giá trị khoá chính của một bộ q1 nào đó đã có trong quan hệ thì hệ thống sẽ báo lỗi và yêu cầu nhập lại một giá trị khác.

Người ta cũng quy ước rằng:

-Trong một bộ của quan hệ các thuộc tính khoá không chứa giá trị rỗng.

-Không được phép sửa đổi giá trị thuộc tính khoá của một bộ q. Nếu muốn sửa đổi giá trị thuộc tính khoá của một bộ q, người sử dụng phải huỷ bỏ bộ q và sau đó thêm một bộ q’ với giá trị khoá đã được sửa đổi.

2.2.CHUYỂN MÔ HÌNH THỰC THỂ KẾT HỢP SANG MÔ HÌNH DỮ LIỆU QUAN HỆ

Sau đây là một số quy tắc được sử dụng trong việc chuyển đổi mô hình thực thể kết hợp sang mô hình dữ liệu quan hệ.

Quy tắc 1:

Chuyển đổi mỗi loại thực thể thành một lược đồ quan hệ, các thuộc tính của loại thực thể thành các thuộc tính của lược đồ quan hệ, thuộc tính khoá của loại thực thể là thuộc tính khoá của lược đồ quan hệ.

Chẳng hạn loại thực thể Sinhvien ở ví dụ 1.2 khi áp dụng quy tắc 1 thì sẽ được chuyển thành lược đồ quan hệ Sinhvien như sau:

Sinhvien(MASV, HOTENSV, NU, NGAYSINH, TINH,….)

Quy tắc 2:

Nếu mối kết hợp mà cả hai nhánh của nó đều có bản số max là n thì mối kết hợp này sẽ được chuyển thành một lược đồ quan hệ K’ gồm các thuộc tính của mối kết hợp K, cộng thêm các thuộc tính khoá của hai lược đồ quan hệ A, B tương ứng với hai thực thể tham gia vào mối kết hợp. Khoá của lược đồ quan hệ K’ gồm cả hai khoá của hai lược đồ quan hệ A và B.

Chẳng hạn mối kết hợp Phancong giữa ba loại thực thể Giangvien, Monhoc và Lop được chuyển thành lược đồ quan hệ Phancong và có tập khoá là {MAGV,MAMH,MALOP} như sau:

Phancong(MAGV,MAMH,MALOP)

Quy tắc 3:

Mối kết hợp mà một nhánh có bản số là n (nhánh B) và nhánh còn lại có bản số max là 1 (nhánh A) thì loại bỏ mối kết hợp này khỏi mô hình thực thể kết hợp và thêm các thuộc tính khoá của lược đồ tương ứng với loại thực thể ở nhánh B vào lược đồ tương ứng với loại thực thể ở nhánh A (khoá của B sẽ thành khoá ngoại của A). Nếu mối kết hợp có các thuộc tính thì những thuộc tính này cũng được thêm vào lược đồ quan hệ tương ứng với loại thực thể ở nhánh A.

Chẳng hạn mối kết hợp ***thuộc*** giữa hai loại thực thể Sinhvien và Lop nên lược đồ quan hệ Sinhvien được sửa thành như sau:

Sinhvien(MASV,HOTENSV,NU,NGAYSINH, TINH,MALOP)

Quy tắc 4:

Nếu mối kết hợp mà cả hai nhánh đều có bản số max là 1 thì áp dụng quy tắc 3 cho một trong hai nhánh tuỳ chọn.

Ví dụ 2.1:

Sau đây là mô hình dữ liệu quan hệ được chuyển từ mô hình thực thể kết hợp ở ví dụ 1.2.

Sinhvien(MASV, HOTENSV, NU, NGAYSINH, NOISINH,TINH, MALOP)

Lop(MALOP,TENLOP,MAKHOA)

Khoa(MAKHOA,TENKHOA)

Monhoc(MAMH,TENMH,DONVIHT)

Giangvien(MAGV,HOTENGV,HOCVI,CHUYENNGANH,MAKHOA)

Ketqua(MASV, MAMH, LANTHI,DIEMTHI)

Phancong(MALOP,MAMH,MAGV)

2.3. NGÔN NGỮ ĐẠI SỐ QUAN HỆ

2.3.1. Phép Hợp 2 quan hệ(Union)

Ta nói hai quan hệ r_1 và r_2 là tương thích nếu chúng được định nghĩa trên cùng một lược đồ quan hệ.

Cho hai quan hệ tương thích r_1 và r_2 . Hợp của hai quan hệ r_1 và r_2 ký hiệu là $r_1 + r_2$ là một quan hệ trên lược đồ quan hệ Q gồm các phần tử thuộc r_1 hoặc thuộc r_2, tức là:

$r_1 + r_2 = \{t / t \in r_1$ hoặc $t \in r_2\}$

Ví dụ 2.2

r_1

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	d3
a4	b4	c4	d4

r_2

A	B	C	D
x1	y1	z1	v1
a2	b2	c2	d2
x3	y3	z3	v3

Khi đó nội dung của quan hệ $r_1 + r_2$ là:

A	B	C	D
a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	d3
a4	b4	c4	d4
x1	y1	z1	v1
x3	y3	z3	v3

Do thứ tự trước/sau của các bộ trong các quan hệ là không quan trọng nên ta có:

$\forall r_1, r_2$ thì $r_1 + r_2 = r_2 + r_1$

$\forall r$ thì $r + r = r$

Một cách tổng quát có thể lấy hợp của n quan hệ tương thích: cho n quan hệ tương thích $r_1,r_2,\ldots,r_n$

Hợp của n quan hệ $r_1,r_2,\ldots,r_n$ là một quan hệ $r_1 + r_2 + \ldots + r_n$ gồm các phần tử thuộc r_1 hoặc thuộc r_2 … hoặc thuộc r_n

2.3.2. Phép Giao 2 quan hệ (Intersection)

Cho lược đồ quan hệ $Q(A_1,A_2,..,A_n)$. r_1 và r_2 là hai quan hệ tương thích trên Q.

Giao của hai quan hệ r_1 và r_2 ký hiệu là $r_1 * r_2$ là một quan hệ trên Q gồm các phần tử vừa thuộc r_1 vừa thuộc r_2.

Vậy: $r_1 * r_2 = \{ t / t \in r_1 \text{ và } t \in r_2\}$

Chẳng hạn với ví dụ 2.2 ở trên thì $r_1 * r_2$ là:

A	B	C	D
a2	b2	c2	d2

2.3.3.Phép Trừ 2 quan hệ (Minus)

Cho hai quan hệ tương thích r_1 và r_2 có tập thuộc tính $Q(A_1,A_2,..,A_n)$. Hiệu của r_1 cho r_2 ký hiệu là $r_1 – r_2$ là một quan hệ trên Q gồm các phần tử chỉ thuộc r_1 mà không thuộc r_2, nghĩa là $r_1 - r_2 = \{t \in r1 \text{ và } t \notin r2\}$

Chẳng hạn với ví dụ 2.2. thì $r_1 - r_2$ là:

A	B	C	D
a1	b1	c1	d1
a3	b3	c3	d3
a4	b4	c4	d4

2.3.4.Tích Decac của 2 quan hệ Cartesian Product)

Cho hai lược đồ quan hệ

$Q_1(A_1,A_2,..,A_n)$

$Q_2(B_1,B_2,..,B_m)$

Giả sử r_1 , r_2 là hai quan hệ trên Q_1,Q_2 tương ứng. Tích Descartes (decac) của r_1 và r_2 ký hiệu là r_1 x r_2 là quan hệ trên lược đồ quan hệ có tập thuộc tính $Q= Q_1 \cup Q_2$.

Vậy quan hệ r_1 x r_2 là quan hệ trên lược đồ:

$Q = Q_1 \cup Q_2 = \{ A_1,A_2,..,A_n,B_1,B_2,..,B_m \}$ với

r_1 x $r_2 = \{(t_1,t_2) : t_1 \in r_1, t_2 \in r_2 \}$

Ví dụ 2.3. cho r_1 và r_2 là

r_1

A	B	C
6	5	4
7	5	5

r_2

E	F	H
1	5	9
4	6	8
7	5	3

Thì kết quả r_1 x r_2 như sau:

A	B	C	E	F	H
6	5	4	1	5	9
6	5	4	4	6	8
6	5	4	7	5	3
7	5	5	1	5	9
7	5	5	4	6	8
7	5	5	7	5	3

2.3.5.phép chia 2 quan hệ:

cho 2 lược đồ quan hệ

$Q_1(A_1,A_2,..,A_n)$

$Q_2(B_{1,}B_{2,}..,B_m)$

r là quan hệ xác định trên Q_1; s là quan hệ xác định trên Q_2 (n>m và s khác rỗng), có m thuộc tính chung (giống nhau về mặt ngữ nghĩa, hoặc các thuộc tính có thể so sánh được) giữa r và s. phép chia 2 quan hệ r và s ký hiệu r ÷ s , là một quan hệ q có n - m thuộc tính được định nghĩa như sau:

$q= r \div s=\{t/ \forall u \in s, (t,u) \in r\}$

Ví dụ 2.4:

r

A	**B**	**C**	**D**
a	b	c	d
a	b	e	f
b	c	e	f
c	d	c	d
c	d	e	f
a	b	d	e

s

C	**D**
c	d
e	f

r ÷ s

A	**B**
a	b
c	d

2.3.6.Phép Chiếu (projection)

Cho lược đồ quan hệ $Q(A_{1,}A_{2,}..,A_n)$, r là quan hệ trên Q và $X \subseteq Q^+$.

Phép chiếu của r lên tập thuộc tính X, ký hiệu là r[X] (hoặc r.X) sẽ tạo thành lược đồ quan hệ r', trong đó tập thuộc tính của r' chính là X và quan hệ r' được trích từ r bằng cách chỉ lấy các thuộc tính có trong X.

Phép chiếu chính là phép rút trích dữ liệu theo cột. Chẳng hạn với r_1 ở ví dụ 2.2 thì khi đó ta có quan hệ con của r_1 chiếu lên X={A,C} là:

$r_1[X]$

A	**C**
a1	c1
a2	c2
a3	c3
a4	c4

2.3.7.Phép Chọn (Selection)

Cho lược đồ quan hệ $Q(A_1,A_2,..,A_n)$, r là một quan hệ trên lược đồ quan hệ Q. X là một tập con của Q^+ và E là một mệnh đề logic được phát biểu trên tập X. Phần tử $t \in r$ thoả mãn điều kiện E ký hiệu là t(E). Phép chọn từ quan hệ r theo điều kiện E (ký hiệu là r : E)sẽ tạo thành một quan hệ mới ký hiệu là r(E), trong đó r(E) = {t: $t \in r$ và t(E)}

Phép chọn chính là phép rút trích dữ liệu theo dòng. Chẳng hạn với r_2 ở ví dụ 2.3 và điều kiện E là: "F >= 6" thì kết quả $r_2(E)$ hay r2: "F >= 6" có nội dung là

E	F	H
4	6	8

2.3.8. Phép θ - Kết

Cho hai lược đồ quan hệ Q_1 và Q_2 như sau

$Q_1(A_1,A_2,..,A_n)$

$Q_2(B_1,B_2,..,B_m)$

r và s lần lượt là hai quan hệ trên Q_1 và Q_2.

A_i và B_j lần lượt là thuộc tính của Q_1 , Q_2 sao cho $MGT(A_I)= MGT(B_J)$. θ là một trong các phép so sánh (=, <, >, $\leq$, $\geq$, $\neq$) trên $MGT(A_I)$.

$A_i \theta B_j$

Phép θ kết giữa r và s theo điều kiện $A_i \theta B_j$ ký hiệu là r |><| s là một quan hệ trên lược đồ quan hệ có tập thuộc tính là $Q_1 \cup Q_2$. gồm những bộ thuộc tích Descartes của r và s sao Ai θ Bj.

$A_i \theta B_j$

r |><| s = {t_{12} / $\exists$ $t_1 \in r_1$, $\exists$ $t_2 \in r_2$ sao cho $t_{12}.Q_1^{+.}= t_1$; $t_{12}.Q_2^{+.}= t_2$; $t_{12}A_i \theta t_{12}.B_j$ }

Ví dụ 2.5 Cho hai quan hệ r_1 và r_2 như sau:

r_1

A	B	C
6	5	4
7	5	5
4	2	6

r_2

E	F	H
1	5	9
4	6	8
7	5	3

A_i là thuộc tính B, B_j là thuộc tính F và θ là phép so sánh ">=". Ta được kết quả là quan hệ sau:

A	B	C	E	F	H
6	5	4	1	5	9
6	5	4	7	5	3
7	5	5	1	5	9
7	5	5	7	5	3

2.3.9.Phép Kết Tự Nhiên (natural join)

Nếu θ được sử dụng trong phép kết trên là phép so sánh bằng (=) thì gọi là phép *kết bằng*. Hơn nữa nếu $A_l \equiv B_j$ thì phép kết bằng này được gọi là phép *kết tự nhiên*. Phép kết tự nhiên là phép kết thường dùng nhất trong thực tế.

Ngôn ngữ với các phép toán trên gọi là ngôn ngữ đại số quan hệ.

Sau đây là một ví dụ về ngôn ngữ đại số quan hệ.

Ví dụ 2.6

Cho lược đồ CSDL dùng để quản lý điểm sinh viên được mô tả như ở ví dụ 2.1. Hãy thực hiện các yêu cầu sau bằng ngôn ngữ đại số quan hệ:

1.Lập danh sách các sinh viên lớp có mã lớp là CDTH2A, danh sách cần MASV,HOTENSV

2.Lập danh sách sinh viên nữ và có mã khoa là "CNTT", danh sách cần MASV, HOTENSV.

3.Lập bảng điểm thi lần 1 của tất cả các môn cho sinh viên lớp CDTH2A, danh sách cần MASV, HOTENSV, TENMH, DIEMTHI.

4.Lập phiếu điểm thi lần 1 các môn cho sinh viên có MASV="00CDTH189". danh sách cần MAMH,TENMH, DONVIHT, DIEMTHI.

Giải:

1. Sinhvien: MALOP="CDTH2A" [MASV,HOTENSV]

2. (Sinhvien|><|$_{\text{MALOP}}$ Lop: NU and MAKHOA="CNTT") [MASV,HOTENSV]

MASV MAMH

3. (((Sinhvien |><| Ketqua) |><| Monhoc): MALOP = “CDTH2A” and LANTHI=1) [MASV,HOTENSV,TENMH, DIEMTHI]

MAMH

4. (Ketqua |><| Monhoc : MASV=‘00CDTH189' and LANTHI=1) [MAMH,TENMH,DONVIHT,DIEMTHI]

BÀI TẬP

2.1. Hãy lập mô hình dữ liệu quan hệ cho các bài toán quản lý 1.1, 1.2, 1.3,1.4, 1.5. Hãy xác định khoá cho từng lược đồ cho mỗi bài toán trên.

2.2. Cho lược đồ cơ sở dữ liệu

Sinhvien(MASV, HOTENSV, NU, NGAYSINH, NOISINH,TINH,MALOP)

Lop(MALOP,TENLOP, MAKHOA)

Khoa(MAKHOA,TENKHOA)

Monhoc(MAMH,TENMH,DONVIHT)

Giangvien(MAGV,HOTENGV,HOCVI,CHUYENNGANH,MAKHOA)

Ketqua(MASV, MAMH, LANTHI, DIEMTHI)

Phancong(MALOP,MAMH,MAGV)

Thực hiện các yêu cầu sau bằng ngôn ngữ đại số quan hệ:

a.Lập danh sách những sinh viên có hộ khẩu thường trú ở tỉnh “LONG AN”, danh sách cần các thông tin: MASV, HOTENSV, NGAYSINH, TENLOP

b.Lập danh sách các sinh viên của lớp có MALOP là CDTH2A, danh sách cần các thông tin: MASV, HOTENSV, NGAYSINH, TINH.

c.Lập danh sách các giảng viên có cấp học vị là THAC SY của khoa có MAKHOA là “CNTT”, danh sách cần: MAGV,HOTENGV, CHUYENNGANH.

d.Lập bảng điểm thi lần 1 môn học “869” cho tất cả sinh viên thuộc hai lớp có MALOP là “CĐTH2A” và “CĐTH2B”, danh sách cần: MASV,HOTENSV,DIEMTHI.

e.Lập danh sách các giảng viên đã dạy lớp CĐTH2A, danh sách cần các thông tin: MAGV, HOTENGV,TENKHOA,HOCVI,TENMH.

f.Lập danh sách các môn mà lớp CDTH2A đã học, danh sách cần các thông tin: MAMH,TENMH,DONVIHT,HOTENGV.

g.Lập danh sách những giảng viên đã dạy sinh viên có MASV là “00CDTH189“, danh sách cần MAGV,HOTENGV,HOCVI,CHUYENNGANH,TENKHOA,TENMH

h.Lập danh sách các sinh viên có mã khoa “CNTT” có điểm thi lần 1 môn học “869” lớn hơn hoặc bằng 8, danh sách cần MASV, HOTENSV, DIEMTHI, TENLOP.

chương 3

NGÔN NGỮ TRUY VẤN DỮ LIỆU

(Structured Query Language)

3.1. Mở ĐầU

Mỗi hệ quản trị CSDL đều phải có ngôn ngữ giao tiếp giữa người sử dụng với cơ sở dữ liệu. Ngôn ngữ giao tiếp CSDL gồm các loại sau:

Ngôn ngữ mô tả dữ liệu (Data Definition Language –DDL): Cho phép khai báo cấu trúc các bảng của CSDL, khai báo các mối liên hệ của dữ liệu (relatíonship) và các quy tắc áp đặt lên các dữ liệu đó.

Ngôn ngữ thao tác dữ liệu (Data Manipullation Language- DML) cho phép người sử dụng có thể thêm (insert), xoá (delete), sửa (update) dữ liệu trong CSDL.

*Ngôn ngữ truy vấn dữ liệu (*hay ngôn ngữ hỏi đáp có cấu trúc(Structured Query Language-SQL)): Cho phép người sử dụng khai thác CSDL để truy vấn các thông tin cần thiết trong CSDL.

Ngôn ngữ quản lý dữ liệu (Data Control Language- DCL): Cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu, khai báo bảo mật thông tin và cấp quyền khai thác CSDL cho người sử dụng.

Những năm 1975-1976, IBM lần đầu tiên đưa ra hệ quản trị CSDL kiểu quan hệ mang tên SYSTEM–R với ngôn ngữ giao tiếp CSDL là SEQUEL (Structured English Query Language). Năm 1976 ngôn ngữ SEQUEL được cải tiến thành SEQUEL-2, khoảng năm 1978-1979 SEQUEL-2 được cải tiến và đổi tên thành ngôn ngữ truy vấn có cấu trúc (Structured Query Language). Cuối năm 1979 hệ quản trị CSDL được cải tiến thành SYSTEM-R*. Năm 1986 viện tiêu chuẩn quốc gia Mỹ (American National Standards Institute –ANSI) đã công nhận và chuẩn hoá ngôn ngữ SQL và sau đó tổ chức tiêu chuẩn thế giới (International Standards Organization -ISO) cũng đã công nhân ngôn ngữ này. Đó là chuẩn SQL-86. tới này SQL đã qua 3 lần chuẩn hoá (1989,1992,1996) để

mở rộng các phép toán và tăng cường khả năng bảo mật và tính toàn vẹn dữ liệu.

Trong chương này chúng ta chỉ nghiên cứu về ngôn ngữ SQL.

Ngôn ngữ truy vấn SQL có tập lệnh khá phong phú để thao tác trên cơ sở dữ liệu. Chẳng hạn lệnh create để tạo các bảng quan hệ, lệnh update để cập nhật dữ liệu, lệnh delete để xoá dữ liệu, lệnh insert để thêm dữ liệu,… Trong chương này, chúng tôi chỉ trình bày với bạn đọc câu lệnh quan trọng nhất của SQL đó là câu lệnh hỏi - tìm kiếm dữ liệu SELECT. Kết quả của lệnh select là một quan hệ, quan hệ kết quả này có thể kết xuất ra màn hình, máy in, hoặc là trên các thiết bị lưu trữ thông tin khác. Để đơn giản trong cách trình bày, ta xem quan hệ để thực hiện câu truy vấn là quan hệ nguồn và quan hệ kết quả của truy vấn là quan hệ đích.

Mỗi câu lệnh SQL có thể được viết trên nhiều dòng và kết thúc lệnh bởi dấu chấm phẩy (;), tuy nhiên từ khoá, tên hàm, tên thuộc tính, tên bảng, tên đối tượng thì không được phép viết tách xuống hàng. Trong vận dụng thực tế, từ khoá, tên thuộc tính, tên bảng, tên đối tượng được viết in hoa hay chữ thường là như nhau.

Cú pháp tổng quát của câu lệnh select như sau:

```
select      distinct /*/danh sách thuộc tính/ <biểu thức>,...
from        <danh sách các quan hệ>
where           <biểu thức điều kiện>
group by    <danh sách thuộc tính>
having      <điều kiện nhóm>
order by    <danh sách các thuộc tính [desc]>
```

Trong đó:

<biểu thức> (expression) là sự kết hợp một cách hợp lệ giữa các thuộc tính, các toán tử và các hàm. Sau đây sẽ là các toán tử và hàm thông dụng nhất. (cần chú ý rằng cách sử dụng các toán tử và các hàm này còn tuỳ thuộc vào câu lệnh SELECT của ngôn ngữ được sử dụng).

Các toán tử số học:

^ (luỹ thừa),*(nhân),/ (chia), mod (phần dư), +(cộng), - (trừ)

Các toán tử luận lý:

not(phủ định), and(phép hội), or (phép tuyển)

Các toán tử tập hợp:

In (danh sách các giá trị), LIKE, NOT LIKE, union(phép hợp), intersect (phép giao), minus(phép trừ)

Các toán tử so sánh :

=, <>, >, <, >=, <=

các hàm xử lý ngày tháng

date()

Trả về ngày tháng năm của hệ thống

time()

Trả về giờ phút giây của hệ thống

day(biểu thức ngày)

Trả về một trị số từ 1 đến 31 của biểu thức ngày

month(biểu thức ngày)

Trả về một số từ 1 đến 12 - là tháng của biểu thức ngày

year(biểu thức ngày)

Trả về năm của biểu thức ngày

len(biểu thức chuỗi)

Trả về chiều dài của chuỗi

Các hàm tính toán theo nhóm

sum <thuộc tính>tính tổng giá trị của các bộ theo thuộc tính đã chỉ ra.

max<thuộc tính>:cho biết giá trị lớn nhất của các bộ theo thuộc tính đã chỉ ra.

min<thuộc tính>:cho biết giá trị nhỏ nhất của các bộ theo thuộc tính đã chỉ ra.

avg<thuộc tính>:Cho biết giá trị trung bình của các bộ theo thuộc tính đã chỉ ra.

count */ thuộc tính/ distinct <thuộc tính>

count *: đếm tất cả các bộ

count<thuộc tính>:chỉ đếm những bộ mà giá trị của thuộc tính là khác NULL

count distinct <thuộc tính>

Chỉ đếm những bộ mà giá trị của thuộc tính là khác NULL. hơn nữa, những bộ mà giá trị trùng nhau trên thuộc tính chỉ được đếm là một (đại diện cho cả nhóm).

Sau đây ta sẽ lần lượt tìm hiểu kỹ hơn các mệnh đề của câu lệnh SELECT

Để minh họa cho các ví dụ trong chương này, chúng ta sẽ dùng lại lược đồ cơ sở dữ liệu đã được đề cập trong chương 2.

Sinhvien(MASV, HOTENSV, NU, NGAYSINH, NOISINH,TINH, MALOP)

Lop(MALOP,TENLOP,MAKHOA)

Khoa(MAKHOA,TENKHOA)

Monhoc(MAMH,TENMH,DONVIHT)

Giangvien(MAGV,HOTENGV,HOCVI,CHUYENNGANH,MAKHOA)

Ketqua(MASV, MAMH, LANTHI,DIEMTHI)

Phancong(MALOP,MAMH,MAGV)

3.2.TÌM THÔNG TIN TỪ CÁC CỘT CỦA BẢNG - MỆNH ĐỀ SELECT

select distinct /*/danh sách thuộc tính/ <biểu thức>,...

from <danh sách các quan hệ>

-Những thuộc tính được liệt kê trong mệnh đề select sẽ là các thuộc tính có trong quan hệ đích.

-Ký hiệu * theo sau từ khóa select dùng để chỉ tất cả các thuộc tính của quan hệ nguồn sẽ là thuộc tính của quan hệ đích. Danh sách các thuộc tính cách nhau bởi dấu phẩy và thứ tự này cũng là thứ tự của các thuộc tính trong quan hệ đích.

-Mệnh đề from:

Những quan hệ liên quan đến câu truy vấn được liệt kê sau mệnh đề from, các quan hệ này cách nhau bởi dấu phẩy, thứ tự của các quan hệ được chỉ ra ở đây là không quan trọng.

Cần chú ý rằng khi mệnh đề From chỉ ra từ hai quan hệ trở lên, nếu có một thuộc tính ở mệnh đề select là thuộc tính của nhiều hơn một quan hệ thì cần phải chỉ rõ thuộc tính đó thuộc về quan hệ nào theo cú pháp *tênquanhệ.tênthuộctính* (sinh viên thường mắc lỗi này khi thực hành với câu lệnh truy vấn SQL)

(Do sinh viên khi học môn này chưa học SQL server, nên nếu khi thực hành bài tập chương này với Access thì cuối mỗi dòng không có dấu chấm phẩy – trừ dòng cuối cùng, ký tự đại diện cho một nhóm ký tự là dấu sao(*) , còn nếu thực hành với Visual Foxpro thì cuối mỗi dòng có dấu chấm phẩy – trừ dòng cuối cùng)

Ví dụ 3.1:

Lập danh sách sinh viên gồm MASV, HOTENSV, NU, NGAYSINH, TINH, MALOP

```
select MASV, HOTENSV, NU, NGAYSINH, TINH, MALOP
from sinhvien;
```

Khi cần lấy thông tin về tất cả các cột của bảng, chúng ta có thể sử dụng dấu sao (*) thay cho việc liệt kê các tên cột của bảng. Nếu áp dụng cách viết này thì câu lệnh trên tương đương với câu lệnh sau:

```
SELECT *
FROM Sinhvien;
```

(tất nhiên cú pháp này chỉ được sử dụng khi câu truy vấn chỉ liên quan đến một quan hệ)

Ví dụ 3.2:

Lập danh sách bao gồm các thông tin về giảng viên như mã số giảng viên, họ và tên giảng viên, học vị, chuyên ngành.

```
SELECT MAGV,HOTENGV,HOCVI, CHUYENNGANH
FROM giangvien;
```

Câu lệnh tìm kiếm thông tin từ các cột của bảng ở trên là cài đặt của phép chiếu trên bốn thuộc tính MAGV,HOTENGV,HOCVI,CHUYENNGANH của quan hệ Giangvien.

Nếu chúng ta muốn đặt tên khác cho tên của các cột của bảng (còn gọi là bí danh- ALIAS), việc này được thực hiện bằng cách thêm từ khóa AS và theo sau là một tên mới. Nếu tên có chứa các ký tự đặc biệt và/hoặc khoảng trắng thì viết tên đó trong cặp dấu ngoặc vuông ([ ]).

Chẳng hạn ví dụ 3.2 có thể viết lại là:

```
SELECT  MAGV AS [MÃ SỐ GIẢNG VIÊN] ,HOTENGV AS [HỌ VÀ
TÊN]  HOCVI  [TRÌNH ĐỘ] CHUYENNGANH AS  [ CHUYÊN NGÀNH]
FROM        Giangvien;
```

Câu lệnh SELECT không chỉ thực hiện việc trích thông tin từ các cột đơn lẽ của bảng mà còn có thể thực hiện các tính toán theo công thức hay biểu thức bất kỳ dựa trên giá trị của các cột trên từng bản ghi của bảng.

Từ khóa DISTINCT nhằm loại bỏ bớt các bộ trùng nhau trong bảng kết quả của lệnh truy vấn (chỉ giữ lại một bộ đại diện cho các bộ giống nhau)

Ví dụ 3.3:

Hãy cho biết các giảng viên của trường thuộc những chuyên ngành nào?

```
SELECT      DISTINCT CHUYENNGANH
FROM            Giangvien;
```

kết quả của câu lệnh này là tất cả những chuyên ngành mà các giảng trong trường có thể đảm nhận(tất nhiên mỗi chuyên ngành chỉ xuất hiện một lần trong kết quả truy vấn được).

3.3.CHỌN CÁC DÒNG CỦA BẢNG – MỆNH ĐỀ WHERE

```
SELECT      DISTINCT /*/danh sách thuộc tính/ <biểu thức>,...
FROM            <danh sách các quan hệ>
WHERE       <biểu thức điều kiện>
```

Trong đó <biểu thức điều kiện> có giá trị là hoặc đúng (true) hoặc sai (false). Đây là sự cài đặt của phép chọn trong ngôn ngữ đại số quan hệ.

Nếu điều kiện này chỉ liên quan đến một quan hệ thì gọi là *điều kiện chọn*, nếu điều kiện liên quan đến từ hai quan hệ trở lên thì gọi là *điều kiện kết.* Các điều kiện chọn và điều kiện kết có thể phối hợp với nhau bởi các toán tử logic (and,or,not) để tạo nên những biểu thức logic phức tạp hơn. Cần chú ý rằng thứ tự của các điều kiện ở đây là quan trọng: Nếu có thể thì nên thực hiện *điều kiện chọn* trước khi thực hiện *điều kiện kết* (đây là vấn đề tối ưu hoá câu truy vấn, chúng tôi không đi sâu về vấn đề này[3]).

Sau đây là một số ví dụ cho phép chọn.

Ví dụ 3.4:

Lập danh sách những môn học có số đơn vị học trình >=4. Danh sách cần MAMH, TENMH,DONVIHT.

```
SELECT MAMH, TENMH,DONVIHT
FROM        monhoc
WHERE DONVIHT >=4;
```

Ví dụ 3.5:

Lập danh sách các sinh viên có mã lớp là CDTH2A, CDTH2B, CDTH2C.

```
SELECT *
FROM        sinhvien
WHERE malop=”CDTH2A” or  malop =””CDTH2B” or malop = ””CDTH2C”;
```

Cũng có thể viết cách khác như sau:

```
SELECT *
FROM sinhvien
WHERE        malop in (“CDTH2A”,”CDTH2B”,”CDTH2C”);
```

Ví dụ 3.6:

Lập danh sách những sinh viên của lớp CDTH2A có điểm thi lần 1 môn CSDL là 6,8

```
Selete masv,diemthi
From ketqua
Where diemthi>=6 and diemthi<=8 and mamh=”csdl”;
```

Hoặc có thể viết cách khác

SELECT masv,diemthi

FROM ketqua

WHERE diemthi between 6 and 8 and mamh=”csdl”

Toán tử so sánh tương đối : like

Mẫu so sánh trong phép toán like là một giá trị kiểu text, đó là một dãy ký tự bất kỳ, trong đó có hai ký tự có ý nghĩa đặc biệt sau đây:

? đại diện cho một ký tự bất kỳ tại vị trí có dấu chấm hỏi

% đại diện cho một nhóm ký tự bất kỳ tại vị trí đó

Ví dụ 3.7:

Lập danh sách các sinh viên có họ là Nguyễn đang theo học tại lớp có mã lớp là CDTH2A

SELECT *

FROM sinhvien

WHERE malop=”CDTH2A” and HOTENSV like “Nguyễn%”;

3.4.SẮP XẾP CÁC DÒNG CỦA BẢNG - MỆNH ĐỀ ORDER BY

Quan hệ đích có thể được sắp xếp tăng/giảm theo một (hoặc nhiều) thuộc tính nào đó bằng cách sử dụng mệnh đề ORDER BY <danh sách thuộc tính> (độ ưu tiên giảm dần tư trái sang phải), từ khóa DESC (DESCending) được dùng nếu muốn sắp xếp giảm dần, nếu không có DESC, mặc định CSDL sẽ được sắp xếp tăng dần ASC (ASCending) theo các thuộc tính đã chỉ ra.

Nghĩa là danh sách các lớp được sắp xếp theo cột mã khoa, nếu cột mã khoa trùng nhau thì sắp xếp theo cột số học viên

Sau đây là vấn đề truy vấn thông tin từ nhiều bảng dữ liệu

Ví dụ 3.8:

Lập danh sách các lớp có mã khoa là “CNTT” danh sách cần các thông tin MALOP,TENLOP, TENKHOA

SELECT MALOP,TENLOP,TENKHOA

FROM lop,khoa

WHERE makhoa=“CNTT” and

lop.makhoa=khoa.makhoa;

Ví dụ 3.9:

Lập danh sách các sinh viên lớp CDTH2A có điểm thi môn học có mã môn học là “869” lớn hơn hay bằng 8.0

```
SELECT Sinhvien.MASV,HOTENSV,NU,NGAYSINH,DIEMTHI
FROM Sinhvien,ketqua
WHERE   malop=”CDTH2A” and
        MAMH=”869” and
        Sinhvien.MASV=Ketqua.MASV and
        DIEMTHI>=8.0;
```

Cần chú ý rằng do thuộc tính MASV xuất hiện ở cả hai quan hệ Sinhvien và ketqua, nên khi liệt kê nó ở mệnh đề SELECT cần chỉ rõ ra nó thuộc quan hệ nào ? Tuy nhiên sinh viên cũng cần chú ý rằng: nếu ta không ghi Sinhvien.MASV mà ghi là ketqua.MASV thì kết quả vẫn đúng.

3.5. CÂU LỆNH TRUY VẤN LỒNG NHAU

Là những câu lệnh mà trong thành phần WHERE có chứa thêm một câu lệnh Select khác nữa. Câu lệnh này thường gặp khi dữ liệu cần thiết phải duyệt qua nhiều lần. Đây là một trong những vấn đề khó khăn nhất khi truy vấn dữ liệu

Ví dụ 3.10:

Lập danh sách những sinh viên lớp CDTH2A có điểm thi lần 1 môn học CSDL cao nhất.

Với câu lệnh nàynếu dùng các ngôn ngữ lập trình không có ngôn ngữ hỏi cấu trúc thì thật là dài dòng (đầu tiên phải tìm cho ra số điểm lớn nhất thoã mãn điều kiện trên, sau đó phải duyệt dữ liệu thêm một lần nữa để chọn ra những bộ thoã đề bài)

```
SELECT  sinhvien.MASV,HOTENSV,NU,NGAYSINH,DIEMTHI
FROM    sinhvien,ketqua
WHERE   MAMH='CSDL' AND
        Lanthi=1
        sinhvien.MASV=Ketqua.MASV AND
```

```
DIEMTHI>=ALL(SELECT DIEMTHI
                FROM ketqua,sinhvien
                WHERE MAMH='CSDL' AND Lanthi=1 and
                sinhvien.MASV=Ketqua.MASV);
```

Tiếp theo sau đây chúng tôi sẽ giới thiệu với bạn đọc thêm một ví dụ về câu lệnh truy vấn lồng nhau:

Ví dụ 3.11:

Lập danh sách những giảng viên cùng khoa với giảng viên NGUYEN VAN THANH ?

Giải:

```
select *
from giangvien
where makhoa  in
      (select makhoa
      from giangvien
      where Hotengv=”NGUYEN VAN THANH”);
```

kết quả của câu hỏi con được sử dụng trong phép so sánh với một giá trị khác trong biểu thức điều kiện của câu hỏi bao nó. Các phép so sánh có dạng

<phep so sánh>[<lượng từ>](select - câu hỏi con)

trong đó phép so sánh có thể là phép so sánh số học hoặc phép so sánh trên tập hợp (chúng tôi đã đề cập ở phần 3.1)

<lượng từ > có thể là ALL,ANY (hoặc SOME). Phép so sánh bằng ANYcó thể được thay tương đương bằng phép toán IN, phép so sánh <>ALL có thể thay tương đương bằng phép toán NOT IN.

3.6.GOM NHÓM DỮ LIỆU– MỆNH ĐỀ GROUP BY

Khi cần tính toán trên các bộ theo một nhóm nào đó - theo một thuộc tính nào đó, thì ta dùng mệnh đề GROUP BY, chẳng hạn cần tính điểm trung bình chung tất cả các môn học cho tất cả các sinh viên, hay là cần tính số

lượng sinh viên cho mỗi lớp, mỗi khoa, đếm số lượng sinh viên nữ của mỗi khoa, đếm số lượng sinh viên của mỗi tỉnh,…

Mệnh đề GROUP BY <thuộc tính> dùng để phân nhóm dữ liệu. những bộ của bảng có cùng giá trị trên các thuộc tính này sẽ tạo thành một nhóm.

Ví dụ 3.12:

Lập bảng điểm trung bình lần 1 các môn học của các sinh viên của lớp có mã lớp là CDTH2A. Danh sách cần: MASV, HOTENSV,DIEMTB((trong đó DIEMTB là thuộc tính tự đặt).

```
SELECT KETQUA.MASV, HOTENSV,AVG(DIEMTHI) AS DIEMTB
FROM SINHVIEN,KETQUA
WHERE MALOP=”CDTH2A” AND LANTHI=1 AND
      SINHVIEN.MASV=KETQUA.MASV
GROUP BY KETQUA.MASV, HOTENSV
```

Mệnh đề HAVING <điều kiện trên nhóm>

Nếu cần kiểm tra điều kiện của một nhóm thì dùng mệnh đề Having , chẳng hạn như cho biết những sinh viên nào có điểm trung bình các môn ≥ 8, những khoa nào có nhiều hơn 100 sinh viên nữ,…

Lưu ý những thuộc tính có tham gia vào mệnh đề GROUP BY để phân nhóm phải được liệt kê trong danh sách thuộc tính theo sau từ khóa SELECT.

Mệnh đề HAVING <điều kiện trên nhóm> được sử dụng như là phép chọn phối hợp với việc phân nhóm dữ liệu.

Ví dụ 3.13:

Giống như ở ví dụ 3.11 nhưng có thêm điều kiện là điểm trung bình các môn đã thi lớn hơn hoặc bằng 8.0.

```
SELECT KETQUA.MASV, HOTENSV,AVG(DIEMTHI) AS DIEMTB
FROM SINHVIEN,KETQUA,LOP
WHERE MALOP=”CDTH2A” AND LANTHI=1 AND
      SINHVIEN.MASV=KETQUA.MASV
GROUP BY KETQUA.MASV, HOTENSV
HAVING AVG(DIEMTHI)>=8.0;
```

Trong một lệnh truy vấn tổng hợp, ngoại trừ thành phần SELECT bắt buộc phải đặt lên đầu, thứ tự của các thành phần khác là tùy ý. Thứ tự dịch một lệnh truy vấn tổng hợp là như sau:

FROM → WHERE → GROUP BY → HAVING → SELECT→ ORDER BY

Suy cho cùng, các chương trình quản lý cũng là việc kết xuất các báo cáo từ các quan hệ, mà SQL cho phép tạo ra những quan hệ này một cách tiện lợi. Vì thế hiểu và vận dụng tốt lệnh truy vấn dữ liệu là một việc làm cực kỳ hiệu quả!

BÀI TẬP

3.1 Dựa vào lược đồ CSDL

Congtrinh(MACT,TENCT,ĐIAĐIEM,NGAYCAPGP ,NGAYKC,NGAYHT)

Nhanvien(MANV,HOTEN,NGAYSINH,PHAI,ĐIACHI,MAPB)

Phongban(MAPB,TENPB)

Phancong(MACT,MANV,SLNGAYCONG)

Hãy thực hiện các câu hỏi sau bằng SQL

a.Danh sách những nhân viên có tham gia vào công trình có mã công trình(MACT) là X.Yêu cầu các thông tin: MANV,HOTEN, SLNGAYCONG, trong đó MANV được sắp tăng dần

b.Đếm số lượng ngày công của mỗi công trình. Yêu cầu các thông tin: MACT, TENCT, TONGNGAYCONG (TONGNGAYCONG là thuộc tính tự đặt)

c.Danh sách những nhân viên có sinh nhật trong tháng 08. yêu cầu các thông tin: MANV, TENNV, NGAYSINH, ĐIACHI,TENPB, sắp xếp quan hệ kết quả theo thứ tự tuổi giảm dần.

d.Đếm số lượng nhân viên của mỗi phòng ban. Yêu cầu các thông tin: MAPB, TENPB, SOLUONG. (SOLUONG là thuộc tính tự đặt.)

3.2. Dựa vào lược đồ cơ sở dữ liệu

Giaovien(MAGV,HOTEN, MAKHOA)

Monhoc(MAMH,TENMH)

Phonghoc(PHONG,CHUCNANG)

Khoa(MAKHOA,TENKHOA)

Lop(MALOP,TENLOP, MAKHOA)

Lichday(MAGV,MAMH,PHONG,MALOP,NGAYDAY,TUTIET,ĐENTIET, BAIDAY, LYTHUYET, GHICHU)

Hãy thực hiện các câu hỏi sau bằng SQL

a.Xem lịch báo giảng tuần từ ngày 08/09/2003 đến ngày 14/09/2003 của giáo viên có MAGV (mã giáo viên) là TH3A040. Yêu cầu: MAGV,HOTEN, TENLOP,TENMH,PHONG, NGAYDAY, TUTIET, ĐENTIET, BAIDAY, GHICHU)

b.Xem lịch báo giảng ngày 08/09/2003 của các giáo viên có mã khoa là CNTT. Yêu cầu:MAGV,HOTEN,TENLOP,TENMH,PHONG, NGAYDAY, TUTIET, ĐENTIET,BAIDAY, GHICHU)

c.Cho biết số lượng giáo viên (SOLUONGGV) của mỗi khoa, kết quả cần sắp xếp tăng dần theo cột tên khoa. yêu cầu: TENKHOA ,SOLUONGGV (SOLUONGGV là thuộc tính tự đặt)

3.3.Hàng năm, Trường X tổ chức kỳ thi giỏi nghề cho các học sinh- sinh viên của trường, mỗi thí sinh sẽ thi hai môn (chẳng hạn các thi sinh thi giỏi nghề công nghệ thông tin thi hai môn là Visual Basic và Cơ Sở Dữ Liệu).

Giả sử lược đồ cơ sở dữ liệu của bài toán quản lý các kỳ thi trên được cho như sau:

THISINH(MASV,HOTEN, NGAYSINH, MALOP)

LOP(MALOP,TENLOP,MAKHOA)

KHOA(MAKHOA,TENKHOA,ĐIENTHOAI)

MONTHI(MAMT,TENMONTHI)

KETQUA(MASV,MAMT,ĐIEMTHI)

(Phần giải thích các thuộc tính: HOTEN (họ tên thí sinh), NGAYSINH (ngày sinh), MALOP (mã lớp),MASV(mã sinh viên), TENLOP(tên lớp),MAKHOA(mã khoa), TENKHOA (tên khoa),ĐIENTHOAI(số điện thoại khoa), MAMT (mã môn thi), TENMONTHI (tên môn thi), ĐIEMTHI (điểm thi)).

Dựa vào lược đồ cơ sở dữ liệu trên, hãy thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

a.Hãy cho biết số lượng thí sinh của mỗi khoa đăng ký thi giỏi nghề, cần sắp xếp kết quả theo chiều tăng dần của cột TENKHOA.

b.Lập danh sách những thí sinh đạt danh hiệu giỏi nghề

(Thí sinh đạt danh hiệu giỏi nghề nếu thí sinh không có môn thi nào điểm dưới 8).

c.Lập danh sách những thí sinh nhỏ tuổi nhất có mã khoa là “CNTT” dự thi giỏi nghề.

3.4. Cho Lược đồ cơ sở dữ liệu quản lý nhân viên của một công ty như sau:

Nhanvien(MANV,HOTEN, NU,NGAYSINH,LUONG,MAPB, MACV)

Mỗi nhân viên có một mã nhân viên (MANV) duy nhất, mỗi mã nhân viên xác định họ và tên nhân viên (HOTEN), giới tính (NU), lương (LUONG), mã phòng ban (MAPB), mã chức vụ (MACV).

Phongban(MAPB,TENPB,TRUSO,MANVPHUTRACH,KINHPHI,DOANHTHU)

Mỗi phòng ban có tên gọi phòng ban(TENPB), địa điểm đặt trụ sở (TRUSO), mã nhân viên phụ trách(MANVPHUTRACH), kinh phí hoạt động (KINHPHI), và doanh thu(DOANHTHU)

Chucvu(MACV,TENCV,LUONGTHAPNHAT,LUONGCAONHAT)

Mỗi chức vụ co tên gọi chức vụ (TENCV), mức lương tối thiểu(LUONGTHAPNHAT), mức lương tối đa (LUONGCAONHAT).

Hãy biểu diễn các câu hỏi sau bằng SQL

a.Lập danh sách gồm các thông tin về các phòng ban trong công ty như: mã số phòng ban, tên phòng ban, địa điểm trụ sở, mã số người phụ trách, kinh phí hoạt động, doanh thu.

b.Lập danh sách những nhân viên sinh nhật trong tháng 10

c.Lập danh sách gồm các thông tin mã số nhân viên, họ và tên và lương cả năm của các nhân viên (giả sử rằng luơng cả năm =12*lương)

d.Lập những phòng ban có kinh phí hoạt động cao nhất.

e.Lập danh sách nhân viên của phòng ban có mã số phòng ban là 40.

f Lập danh sách nhân viên của phòng có mã số phòng ban 10,30,50.

g.Lập danh sách các nhân viên có lương tháng từ 2.500.000 đến 4.000.000

h.Tìm những nhân viên có tuổi cao nhất thuộc phòng ban có MAPB là 10

i.Lập danh sách các nhân viên của phòng 10,30,50. kết quả in ra theo thứ tự tăng dần của mã phòng nếu trùng mã phòng thì sắp xếp giảm dần theo mức lương.

k.Lập danh sách các nhân viên phòng 10,30,50, chỉ in ra những người là lãnh đạo của mỗi phòng ban này.

l.lập danh sách gồm mã phòng mà người có mức lương cao nhất của phòng lớn hơn hoặc bằng 4.000.000

m.Lập mã phòng ban, tên phòng ban, họ và tên của lãnh đạo phòng tương ứng.

n.Lập danh sách những người làm việc cùng phòng với ông Nguyen Van Thanh

o.Lập biết mã số nhân viên, họ và tên, mức lương của người lãnh đạo ông Nguyen Van Thanh.

p.Lập danh sách nhân viên có mức lương lớn hơn hay bằng mức lương cao nhất của phòng ông Nguyen Van Thanh.

q.Cho biết mã số nhân viên, họ và tên , tổng số nhân viên, mức lương cao nhất, mức lương thấp nhất, mức lương trung bình của từng phòng ban.

r.Cho biết các nhân viên có mức lương cao nhất của các phòng ban.

s.Cho biết số lượng nhân viên của mỗi phòng ban.

chương 4

RÀNG BUỘC TOÀN VẸN

(Integrity Constraint)

4.1 RÀNG BUỘC TOÀN VẸN

4.1.1 Khái Niệm Ràng Buộc Toàn Vẹn

Trong mỗi CSDL luôn tồn tại nhiều mối liên hệ giữa các thuộc tính, giữa các bộ; sự liên hệ này có thể xảy ra trong cùng một quan hệ hoặc trong các quan hệ của một lược đồ CSDL. Các mối liên hệ này là những điều kiện bất biến mà tất cả các bộ của những quan hệ có liên quan trong CSDL đều phải thoả mãn ở mọi thời điểm. Những điều kiện bất biến đó được gọi là *ràng buộc toàn vẹn.*. Trong thực tế ràng buộc toàn vẹn là các quy tắc quản lý được áp đặt trên các đối tượng của thế giới thực. Chẳng hạn mỗi sinh viên phải có một mã sinh viên duy nhất, hai thí sinh dự thi vào một trường phải có số báo danh khác nhau, một sinh viên dự thi một môn học không quá 3 lần,…

Nhiệm vụ của người phân tích thiết kế là phải phát hiện càng đầy đủ các ràng buộc toàn vẹn càng tốt và mô tả chúng một cách chính xác trong hồ sơ phân tích thiết kế - đó là một việc làm rất quan trọng. Ràng buộc toàn vẹn được xem như là một công cụ để diễn đạt ngữ nghĩa của CSDL. Một CSDL được thiết kế cồng kềnh nhưng nó thể hiện được đầy đủ ngữ nghĩa của thực tế vẫn có giá trị cao hơn rất nhiều so với một cách thiết kế gọn nhẹ nhưng nghèo nàn về ngữ nghĩa vì thiếu các ràng buộc toàn vẹn của CSDL.

Công việc kiểm tra ràng buộc toàn vẹn thường được tiến hành vào thời điểm cập nhật dữ liệu (thêm, sửa, xoá). Những ràng buộc toàn vẹn phát sinh phải cần được ghi nhận và xử lý một cách tường minh (thường là bởi một hàm chuẩn hoặc một đoạn chương trình).

Ràng buộc toàn vẹn và kiểm tra sự vi phạm ràng buộc toàn vẹn là hai trong số những vấn đề quan trọng trong quá trình phân tích thiết kế cơ sở dữ liệu, nếu không quan tâm đúng mức đến những vấn đề trên, thì có thể dẫn đến

những hậu quả nghiêm trọng về tính an toàn và toàn vẹn dữ liệu , đặc biệt là đối với những cơ sở dữ liệu lớn.

4.1.2 Các Yếu Tố Của Ràng Buộc Toàn Vẹn

Mỗi ràng buộc toàn vẹn có bốn yếu tố: điều kiện, bối cảnh, bảng tầm ảnh hưởng và hành động phải cần thực hiện khi phát hiện có ràng buộc toàn vẹn bị vi phạm:

4.1.2.1.Điều kiện

Điều kiện của ràng buộc toàn vẹn là sự mô tả, và biểu diễn hình thức nội dung của nó

Điều kiện của một ràng buộc toàn vẹn R có thể được biểu diễn bằng ngôn ngữ tự nhiên, ngôn ngữ đại số quan hệ, ngôn ngữ mã giả, ngôn ngữ truy vấn SQL,… ngoài ra điều kiện của ràng buộc toàn vẹn cũng có thể được biểu diễn bằng phụ thuộc hàm (khái niệm phụ thuộc hàm sẽ được đề cập trong chương 5)

Sau đây là một số ràng buộc toàn vẹn trên lược đồ CSDL quản lý sinh viên .

Mỗi lớp học phải có một mã số duy nhất để phân biệt với các lớp học khác trong trường.

Mỗi lớp học phải thuộc về một khoa của trường.

Mỗi sinh viên có một mã số sinh viên duy nhất, không trùng với bất cứ sinh viên nào trong trường.

Mỗi học viên phải đăng ký vào một lớp học trong trường.

Mỗi học viên chỉ được thi tối đa 3 lần cho mỗi môn học.

Tổng số học viên của một lớp phải lớn hơn hoặc bằng số lượng đếm được của một lớp tại một thời điểm nào đó.

4.1.2.2.Bối cảnh

Bối cảnh của ràng buộc toàn vẹn là những quan hệ mà ràng buộc đó có hiệu lực hay nói một cách khác, đó là những quan hệ cần phải được kiểm tra khi tiến hành cập nhật dữ liệu. Bối cảnh của một ràng buộc toàn vẹn có thể là một hoặc nhiều quan hệ.

Chẳng hạn với ràng buộc toàn vẹn R trên thì bối cảnh của nó là quan hệ Sinhvien

4.1.2.3.Bảng tầm ảnh hưởng

Trong quá trình phân tích thiết kế một CSDL, người phân tích cần lập bảng tầm ảnh hưởng cho một ràng buộc toàn vẹn nhằm xác định thời điểm cần phải tiến hành kiểm tra khi tiến hành cập nhật dữ liệu.

Thời điểm cần phải kiểm tra ràng buộc toàn vẹn chính là thời điểm cập nhật dữ liệu.

Một bảng tầm ảnh hưởng của một ràng buộc toàn vẹn có dạng sau:

Tên RBTV	Thêm(T)	Sửa(S)	Xoá(X)
r_1	+		
r_2		-	
r_3			-(*)
r_n			

Bảng này chứa toàn các ký hiệu + , – hoặc -(*).

Chẳng hạn + tại (dòng r_1, cột Thêm) thì có nghĩa là khi thêm một bộ vào quan hệ r_1 thì RBTV bị vi phạm.

Dấu - Tại ô (dòng r_2, cột sửa) thì có nghĩa là khi sửa một bộ trên quan hệ r_2 thì RBTV không bị vi phạm.

,…

Quy ước:

-Không được sửa thuộc tính khoá.

-Nếu không bị vi phạm do không được phép sửa đổi thì ký hiệu là -(*).

***4.1.2.4.*Hành động cần phải có khi phát hiện có RBTV bị vi phạm:**

khi một ràng buộc toàn vẹn bị vi phạm, cần có những hành động thích hợp. Thông thường có 2 giải pháp:

Thứ nhất: Đưa ra thông báo và yêu cầu sửa chữa dữ liệu của các thuộc tính cho phù hợp với quy tắc đảm bảo tính nhất quán dữ liệu. Thông báo phải

đầy đủ và phải thân thiện với người sử dụng. Giải pháp này là phù hợp cho việc xử lý thời gian thực.

Thứ hai:Từ chối thao tác cập nhật. Giải pháp này là phù hợp đối với việc xử lý theo lô. Việc từ chối cũng phải được lưu lại bằng những thông báo đầy đủ, rõ ràng vì sao thao tác bị từ chối và cần phải sữa lại những dữ liệu nào ?

Khóa nội, khoá ngoại, giá trị NOT NULL là những ràng buộc toàn vẹn miền giá trị của các thuộc tính. Những ràng buộc toàn vẹn này là những ràng buộc toàn vẹn đơn giản trong CSDL.

Các hệ quản trị cơ sở dữ liệu thường có các cơ chế tự động kiểm tra các ràng buộc toàn vẹn về miền giá trị của khoa nội, khoá ngoại, giá trị NOT NULL.

Việc kiểm tra ràng buộc toàn vẹn có thể tiến hành vào những thời điểm sau đây.

Thứ nhất: Kiểm tra ngay sau khi thực hiện một thao tác cặp nhật CSDL. Thao tác cặp nhật chỉ được xem là hợp lệ nếu như nó không vi phạm bất cứ một ràng buộc toàn vẹn nào , nghĩa là nó không làm mất tính toàn vẹn của CSDL Nếu vi phạm ràng buộc toàn vẹn , thao tác cặp nhật bị coi là không hợp lệ và sẽ bị hệ thống huỷ bỏ (hoặc có một xử lý thích hợp nào đó)

Thứ hai: Kiểm tra định kỳ hay đột xuất, nghĩa là việc kiểm tra ràng buộc toàn vẹn được tiến hành độc lập với thao tác cặp nhật dữ liệu. Đối với những trường hợp vi phạm ràng buộc toàn vẹn , hệ thống có những xử lý ngầm định hoặc yêu cầu người sử dụng xử lý những sai sót một cách tường minh.

4.2. PHÂN LOẠI RÀNG BUỘC TOÀN VẸN

Trong quá trình phân tích thiết kế CSDL, người phân tích phải phát hiện tất cả các ràng buộc toàn vẹn tiềm ẩn trong CSDL đó. Việc phân loại các ràng buộc toàn vẹn là rất có ích, nó nhằm giúp cho người phân tích có được một định hướng để phát hiện các ràng buộc toàn vẹn, tránh bỏ sót. Các ràng buộc toàn vẹn có thể được chia làm hai loại chính như sau:

Thứ nhất: Ràng buộc toàn vẹn có phạm vi là một quan hệ bao gồm :Ràng buộc toàn vẹn miền giá trị, ràng buộc toàn vẹn liên thuộc tính, ràng buộc toàn vẹn liên bộ.

Thứ hai: Ràng buộc toàn vẹn có phạm vi là nhiều quan hệ bao gồm :Ràng buộc toàn vẹn phụ thuộc tồn tại, ràng buộc toàn vẹn liên bộ - liên quan hệ, ràng buộc toàn vẹn liên thuộc tính - liên quan hệ.

Để minh hoạ cho phần lý thuyết của chương này, chúng ta xét ví dụ sau đây:

Ví dụ 4.1

Cho một CSDL C dùng để quản lý việc đặt hàng và giao hàng của một công ty. Lược đồ CSDL C gồm các lược đồ quan hệ như sau:

Q_1: Khach (MAKH, TENKH, DIACHIKH, DIENTHOAI)

Tân từ:

Mỗi khách hàng có một mã khách hàng (MAKH) duy nhất, mỗi MAKH xác định tên khách hàng (TENKH), địa chỉ (DIACHIKH), số điện thoại (DIENTHOAI).

Q_2: Hang(MAHANG,TENHANG,QUYCACH, DVTINH)

Tân từ:

Mỗi mặt hàng có một mã hàng (MAHANG) duy nhất, mỗi MAHANG xác định tên hàng (TENHANG), quy cách hàng (QUYCACH), đơn vị tính (DVTINH).

Q_3: Dathang(SODH,MAHANG, SLDAT, NGAYDH, MAKH)

Tân từ:

Mỗi mã số đặt hàng (SODH) xác định một ngày đặt hàng (NGAYDH) và mã khách hàng tương ứng (MAKH). Biết mã số đặt hàng và mã mặt hàng thì biết được số lượng đặt hàng(SLDAT). Mõi khách hàng trong một ngày có thể có nhiều lần đặt hàng

Q_4: Hoadon(SOHD, NGAYLAP, SODH, TRIGIAHD, NGAYXUAT)

Tân từ:

Mỗi hoá đơn tổng hợp có một mã số duy nhất là SOHD, mỗi hoá đơn bán hàng có thể gồm nhiều mặt hàng. Mỗi hoá đơn xác định ngày lập hoá đơn

(NGAYLAP), ứng với số đặt hàng nào (SODH). Giả sử rằng hoá đơn bán hàng theo yêu cầu của chỉ một đơn đặt hàng có mã số là SỌDH và ngược lại , mỗi đơn đặt hàng chỉ được giải quyết chỉ trong một hoá đơn. Do điều kiện khách quan có thể công ty không giao đầy đủ các mặt hàng cũng như số lượng từng mặt hàng như yêu cầu trong đơn đặt hàng nhưng không bao giờ giao vượt ngoài yêu cầu. Mỗi hóa đơn xác định một trị giá của nhưng các mặt hàng trong hoá đơn (TRIGIAHD) và một ngày xuất kho giao hàng cho khách (NGAYXUAT)

Q_5: Chitiethd (SOHD, MAHANG, GIABAN, SLBAN)

Tân từ:

Mỗi SOHD, MAHANG xác định giá bán (GIABAN) và số lượng bán (SLBAN) của một mặt hàng trong một hoá đơn.

Q_6: Phieuthu(SOPT, NGAYTHU, MAKH, SOTIEN)

Tân từ:

Mỗi phiếu thu có một số phiếu thu (SOPT) duy nhất, mỗi SOPT xác định một ngày thu (NGAYTHU) của một khách hàng có mã khách hàng là MAKH và số tiền thu là SOTIEN. Mỗi khách hàng trong một ngày có thể có nhiều số phiếu thu.

4.2.1.Ràng buộc toàn vẹn có bối cảnh là một quan hệ

4.2.1.1.Ràng Buộc Toàn Vẹn liên bộ:

+Ràng buộc toàn vẹn về khoá chính:

Đây là một trường hợp dặc biệt của Ràng Buộc toàn Vẹn liên bộ, RBTV này rất phổ biến và thường được các hệ quản trị CSDL tự động kiểm tra.

Ví dụ 4.2:

Với r là một quan hệ trên lược đồ quan hệ Khach ta có ràng buộc toàn vẹn sau:

R_1: $\forall t_1, t_2 \in r$

$t_1.\text{MAKH} \neq t_2.\text{MAKH}$

Cuối $\forall$

R_1	Thêm	Sửa	Xoá
Khach	+	-	-

+Ràng buộc toàn vẹn về tính duy nhất

Ví dụ: mỗi phòng ban phải có một tên gọi duy nhất

+Ngoài ra nhiều khi ta còn gặp những RBTV khác chẳng hạn như RBTV sau trong quan hệ sau đây.

Ví dụ:KETQUA(MASV,MAMH,LANTHI,DIEM)

Mỗi sinh viên chỉ được đăng thi mỗi môn tối đa là 3 lần.

4.2.1.2.Ràng Buộc Toàn Vẹn Về Miền Giá Trị

Ràng buộc toàn vẹn có liên quan đến miền giá trị của các thuộc tính trong một quan hệ. Ràng buộc này thường gặp. Thông thường các hệ quản trị CSDL đã tự động kiểm tra (một số) ràng buộc loại này.

Ví dụ 4.3:

Với r là một quan hệ của Hoadon ta có ràng buộc toàn vẹn sau

R_3: $\forall$ t $\in$ r

t.TRIGIAHD > 0

Cuối $\forall$

R_3	Thêm	Sửa	Xoá
Hoadon	+	+	-

4.2.1.3.Ràng Buộc Toàn Vẹn Liên Thuộc Tính

Ràng buộc toàn vẹn liên thuộc tính (một quan hệ) là mối liên hệ giữa các thuộc tính trong một lược đồ quan hệ.

Ví dụ 4.4

Với r là một quan hệ của Hoadon ta có ràng buộc toàn vẹn sau:

R_4: $\forall$ t $\in$ r

t.NGAYLAP <= t.NGAYXUAT

Cuối $\forall$

R_4	Thêm	Sửa	Xoá
Hoadon	+	+	-

4.2.2.Ràng buộc toàn vẹn có bối cảnh là nhiều quan hệ

4.2.2.1.Ràng Buộc Toàn Vẹn Về Khoá Ngoại:

Ràng buộc toàn vẹn về khoá ngoại còn được gọi là ràng buộc toàn vẹn phụ thuộc tồn tại. Cũng giống như ràng buộc toàn vẹn về khoá nội, loại ràng buộc toàn vẹn này rất phổ biến trong các CSDL.

Ví dụ 4.5

R_2. dathang[MAKH] $\subseteq$ khach[MAKH]

R_2	Thêm	Sửa	Xoá
dathang	+	+	-
Khach	-	-	+

4.2.2.2.Ràng Buộc Toàn Vẹn Liên Thuộc Tính Liên Quan Hệ

Ràng buộc loại này là mối liên hệ giữa các thuộc tính trong nhiều lược đồ quan hệ.

Ví dụ 4.6

Với r,s lần lượt là quan hệ của Dathang và Hoadon. Ta có ràng buộc toàn vẹn R_5 như sau:

R_5: $\forall t_1 \in r, t_2 \in s$

Nếu t_1.SODH=t_2.SODH thì

t_1.NGAYDH <= t_2.NGAYXUAT

Cuối $\forall$

R_5	Thêm	Sửa	Xoá
Dathang	+	-	-
Hoandon	+	+	-

4.2.2.3.Ràng Buộc Toàn Vẹn Liên Bộ Liên Quan Hệ

Ràng buộc loại này là mối liên hệ giữa các bộ trong một lược đồ cơ sở dữ liệu. Chẳng hạn như tổng số tiền phải trả trong mỗi hoá đơn (chitiethd) phải bằng TRỊ GIÁ HOÁ ĐƠN của hoá đơn đó trong quan hệ Hoadon. Hoặc số lượng học viên trong một lớp phải bằng SOHOCVIEN của lớp đó.

Ngoài ra còn có một số loại RBTV khác như :RBTV về thuộc tính tổng hợp, RBTV do tồn tại chu trình ,RBTV về giá trị thuộc tính theo thời gian.

BÀI TẬP

4.1. Việc tổ chức kỳ thi tốt nghiệp của một khoa như sau:

Mỗi thí sinh có một Mã số sinh viên duy nhất (MASV), mỗi MASV xác định được các thông tin: họ và tên (HOTEN), ngày sinh (NGAYSINH), nơi sinh, nữ,phái, dân tộc.

Mỗi lớp có một mã lớp (MALOP) duy nhất , mỗi mã lớp xác định các thông tin: tên lớp (TENLOP), mỗi lớp chỉ thuộc sự quản lý của một khoa nào đó. Mỗi khoa có một mã khoa duy nhất (MAKHOA), mỗi mã khoa xác định tên khoa (TENKHOA).

Mỗi thí sinh đều phải dự thi tốt nghiệp ba môn. Mỗi môn thi có một mã môn thi (MAMT) duy nhất, mỗi mã môn thi xác định các thông tin: tên môn thi (TENMT), thời gian làm bài – được tính bằng phút (PHUT), ngày thi (NGAYTHI), buổi thi (BUOITHI), môn thi này là môn lý thuyết hay thực hành (LYTHUYET). Chú ý rằng, nếu một môn học được cho thi ở nhiều hệ thì được đặt MAMT khác nhau (chẳng hạn cả trung cấp và cao đẳng ngành công nghệ thông tin đều thi môn Cơ Sở Dữ Liệu), để diễn tả điều này, mỗi mã môn học cần phải được ghi chú (GHICHU) để cho biết môn thi đó dành cho khối nào trung cấp, hay cao đẳng). Mỗi thí sinh ứng với một môn thi có một điểm thi (DIEMTHI) duy nhất, điểm thi được chấm theo thang điểm 10 và có lấy điểm lẻ đến 0.5. Một thí sinh được coi là đậu tốt nghiệp nếu điểm thi của tất cả các môn của thí sinh đó đều lớn hơn hoặc bằng 5.

Trong một phòng thi có thể có thí sinh của nhiều lớp. Trong một kỳ thi, mỗi thí sinh có thể thi tại những phòng thi (PHONGTHI) khác nhau, chẳng hạn một thí sinh thi tốt nghiệp ba môn là Cơ sở dữ liệu, Lập trình C và Visual Basic thì môn Cơ Sở Dữ Liệu và Lập Trình C thi tại phòng A3.4, còn môn thực hành Visual Basic thi tại phòng máy H6.1

Qua phân tích sơ bộ trên, ta có thể lập một lược đồ cơ sở dữ liệu như sau:

THISINH(MASV,HOTEN,NGAYSINH,MALOP)

LOP(MALOP,TENLOP)

MONTHI(MAMT,TENMT, LYTHUYET,PHUT,NGAYTHI,BUOITHI,GHICHU)

KETQUA(MASV,MAMT,DIEMTHI)

a. Tìm khoá cho mỗi lược đồ quan hệ trên.

b.Hãy phát biểu các ràng buộc toàn có trong cơ sở dữ liệu trên.

4.2. Cho lược đồ cơ sở dữ liệu (đã được phân tích ở Ví dụ 2.1)

Hãy phát biểu các ràng buộc toàn có trong lược đồ cơ sở dữ liệu trên.

4.3.Cho lược đồ cơ sở dữ liệu ở bài tập 4.1. Thực hiện các yêu cầu sau bằng ngôn ngữ SQL:

a.Lập bảng điểm môn thi có mã môn thi là “CSDL02” cho tất các thí sinh có mã lớp là “CDTH2A”. danh sách cần MASV, HOTEN, NGAYSINH, DIEMTHI và được sắp xếp tăng dần theo MASV.

b.Hãy thống kê xem mỗi môn thi có bao nhiêu thí sinh có điểm thi lớn hơn hay bằng 5? Danh sách cần: MAMT,TENMT,GHICHU,SOLUONG trong đó số lượng (SOLUONG) là thuộc tính tự đặt.

c.Lập danh sách những thí sinh đậu tốt nghiệp (theo tiêu chuẩn đã phân tích ở trên), danh sách cần: MASV,HOTEN,NGAYSINH,DIEMTONG, trong đó DIEMTONG bằng tổng điểm thi của 3 môn thi, DIEMTONG là thuộc tính tự đặt.

d.Nếu cần mở rộng bài toán theo hai hướng; Thứ nhất là quản lý kỳ thi tốt nghiệp cho tất cả các khoa trong toàn trường, Thứ hai là quản lý thông tin về phòng thi (PHONGTHI) của mỗi thí sinh, thì lược đồ cơ sở dữ liệu trên cần phải được điều chỉnh như thế nào ?

e.Hãy phát biểu các ràng buộc toàn có trong lược đồ cơ sở dữ liệu trên.

4.4.Hãy tìm các ràng buộc toàn vẹn có trong mỗi lược đồ cơ sở dữ liệu ở các bài tập 3.1. đến 3.4.

chương 5

LÝ THUYẾT THIẾT KẾ CƠ SỞ DỮ LIỆU

5.1. CÁC VấN Đề GặP PHải KHI Tổ CHứC Dữ LIệU:

Trước khi bàn về cách thiết kế một cơ sở dữ liệu tốt, chúng ta hãy phân tích xem tại sao trong một số lược đồ quan hệ lại tồn tại những vấn đề rắc rối. Chẳng hạn cho lược đồ quan hệ:

Thi(MASV,HOTEN,MONHỌC,DIEMTHI)

và sau đây là một quan hệ trên lược đồ quan hệ Thi

MASV	HOTEN	MONHOC	DIEMTHI
00CDTH189	Nguyễn Văn Thành	Cấu Trúc Dữ Liệu	7
00CDTH189	Nguyễn Văn Thành	Cơ Sở Dữ Liệu	9
00CDTH211	Trần Thu Hà	Kỹ Thuật Lập Trình	5
00CDTH189	Nguyễn Văn Thành	Kỹ Thuật Lập Trình	8

Quan hệ này ghi kết quả điểm thi các môn của các sinh viên. Chúng ta có thể nhận thấy một số vấn đề nảy sinh sau:

1)Dư thừa (redundancy): Họ tên của các sinh viên được lặp lại mỗi lần cho mỗi môn thi.

2)Mưu thuẫn tiềm ẩn (potentia inconsistancyl hay bất thường khi cập nhật. Do hậu quả của dư thừa, chúng ta có thể cập nhật họ tên của một sinh viên trong một bộ nào đó nhưng vẫn để lại họ tên cũ trong những bộ khác. Vì vậy chúng ta có thể không có một họ tên duy nhất đối với mỗi sinh viên như chúng ta mong muốn.

3)Bất thường khi chèn (insertion anomaly). Chúng ta không thể biết họ tên của một sinh viên nếu hiện tại sinh viên đó không dự thi môn nào.

4)Bất thường khi xoá (deletion anomaly). Ngược lại với vấn đề 3) là vấn đề chúng ta có thể xoá tất cả các môn thi của một sinh viên, vô ý làm mất dấu vết để tìm ra họ tên của sinh viên này.

Những vấn đề nêu trên sẽ được giải quyết nếu chúng ta phân rã lược đồ quan hệ Diemthi thành hai lược đồ quan hệ:

Sinhvien(MASV,HOTEN)

Ketqua(MASV,MONHỌC,DIEMTHI)

Lúc này lược đồ quan hệ Sinhvien cho biết họ tên của mỗi sinh viên chỉ xuất hiện đúng một lần; do vậy không có dư thừa. Ngoài ra chúng ta cũng có thể nhập họ tên của một sinh viên dù hiện tại sinh viên đó chưa có kết quả thi môn nào. Tuy nhiên lúc này ta nhận thấy rằng để tìm danh sách họ tên của các sinh viên ứng với môn thi cơ sở dữ liệu thì chúng ta phải thực hiện một phép kết nối, còn với một quan hệ duy nhất Thi chúng ta có thể dễ dàng trả lời bằng cách thực hiện một phép chọn rồi một phép chiếu. Làm sao để đưa được một lược đồ cơ sở dữ liệu chưa tốt về một lược đồ cơ sở dữ liệu tốt hơn ? chương này và chương tới nhằm giải quyết vấn đề này.

5.2. PHỤ THUỘC HÀM

Phụ thuộc hàm (functional dependancy) là một công cụ dùng để biểu diễn một cách hình thức các ràng buộc toàn vẹn. Phương pháp biểu diễn này có rất nhiều ưu điểm, và đây là một công cực kỳ quan trọng, gắn chặt với lý thuyết thiết kế cơ sở dữ liệu.

Trong chương này chúng ta sẽ tìm hiểu về lý thuyết thiết kế cơ sở dữ liệu quan hệ, mà bắt đầu là phụ thuộc hàm và một số ứng dụng trong việc giải quyết các bài toán như: tìm khoá, tìm phủ tối thiểu, xác định dạng chuẩn. Trong chương tới chúng ta sẽ tiếp tục tìm hiểu về cách thức chuẩn hoá một cơ sở dữ liệu.

5.2.1 Định Nghĩa Phụ Thuộc Hàm

Cho lược đồ quan hệ $Q\{A_1,A_2,\ldots,A_n\}$. X,Y là hai tập con khác rỗng của Q^+. Ta nói X xác định Y (hay Y phụ thuộc hàm vào X) nếu với r là một quan hệ nào đó trên Q, $\forall\ t_1,t_2 \in r$ mà $t_1.X = t_2.X \Rightarrow t_1.Y = t_2.Y$ (nghĩa là không thể tồn tại hai bộ trong r giống nhau ở các thuộc tính trong tập X mà lại khác nhau ở một hay nhiều thuộc tính nào đó trong tập Y). Khi đó ta ký hiệu là $X \rightarrow Y$.

Chẳng hạn như phụ thuộc hàm của thuộc tính họ tên của sinh viên (HOTENSV) vào mã số sinh viên (MASV) và ta có thể diễn tả bằng phụ thuộc hàm:

MASV→ HOTENSV

Phụ thuộc hàm X → X được gọi là *phụ thuộc hàm hiển nhiên*. người ta thường dùng F để chỉ tập các phụ thuộc hàm định nghĩa trên Q. Vì Q hữu hạn nên F cũng hữu hạn, ta có thể đánh số các phụ thuộc hàm của F là $f_1,f_2,..,f_m$.

Quy ước: chỉ cần mô tả các phụ thuộc hàm không hiển nhiên trong tập F, các phụ thuộc hàm hiển nhiên được ngầm hiểu là đã có trong F.

Ví dụ 5.1:

Cho lược dồ quan hệ Q(ABCDE), r là quan hệ xác định trên Q được cho như sau:

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b2	c2	d2	e1
a2	b1	c3	d3	e1
a2	b1	c4	d3	e1
a3	b2	c3	d1	e1

Những phụ thuộc hàm nào sau đây thoả r ?

A → D; AB → D; E → A; A → E;

Giải:

AB → D; A → E;

5.2.2 Cách Xác Định Phụ Thuộc Hàm Cho Lược Đồ Quan Hệ

Cách duy nhất để xác định đúng các phụ thuộc thích hợp cho một lược đồ quan hệ là xem xét nội dung tân từ của lược đồ quan hệ đó.

Chẳng hạn với lược đồ cơ sở dữ liệu đã cho trong ví dụ 2.1, thì phụ thuộc hàm ứng với từng lược đồ quan hệ được xác định như sau:

MASV → HOTENSV, NU, NGAYSINH, MALOP, TINH

MALOP → TENLOP,MAKHOA

MAKHOA → TENKHOA

MAMH → TENMH, DONVIHT

MASV, MAMH,LANTHI → DIEMTHI

…..

5.2.3 Một Số Tính Chất Của Phụ Thuộc Hàm - *hệ luật dẫn Armstrong*

Để có thể xác định được các phụ thuộc hàm khác từ tập phụ thuộc hàm đã có, ta dùng hệ tiên đề Armstrong (1974), gồm các luật sau:

với $X,Y,Z,W \subseteq Q^+$

1.*Luật phản xạ* (reflexivity)

$$X \supseteq Y \Rightarrow X \rightarrow Y$$

Quy tắc này đưa ra những phụ thuộc hàm hiển nhiên (phụ thuộc hàm tầm thường), đó là những phụ thuộc hàm mà vế trái bao hàm cả vế phải. Những phụ thuộc hàm hiển nhiên đều đúng trong mọi quan hệ.

2.*Luật tăng trưởng*(augmentation)

$$X \rightarrow Y \Rightarrow XZ \rightarrow YZ$$

3.*Luật bắc cầu*(transitivity)

$$X \rightarrow Y,\ Y \rightarrow Z \Rightarrow X \rightarrow Z$$

Các quy tắc suy rộng:

4.*Luật hợp* (the union rule)

$$\text{Cho } X \rightarrow Y,\ X \rightarrow Z \Rightarrow X \rightarrow YZ$$

5.*Luật bắc cầu giả* (the pseudotransitivity rule)

$$\text{Cho } X \rightarrow Y, WY \rightarrow Z \Rightarrow XW \rightarrow Z$$

6.*Luật phân rã* (the decomposition rule):

$$\text{Cho } X \rightarrow Y,\ Z \subseteq Y \Rightarrow X \rightarrow Z$$

5.3 BAO ĐÓNG CỦA TẬP PHỤ THUỘC HÀM VÀ BAO ĐÓNG CỦA TẬP THUỘC TÍNH

5.3.1. Bao Đóng Của Tập Phụ Thuộc Hàm F

Bao đóng (closure) của tập phụ thuộc hàm F (ký hiệu là F^+) là tập hợp tất cả các phụ thuộc hàm có thể suy ra từ F dựa vào các tiên đề Armstrong. Rõ ràng $F \subseteq F^+$

Ví dụ 5.2

Cho lược đồ quan hệ Q(ABCDEGH) và F được cho như sau:

F = {B $\rightarrow$ A; DA$\rightarrow$ CE; D $\rightarrow$ H; GH$\rightarrow$ C; AC$\rightarrow$ D }

Khi đó F^+ ={B$\rightarrow$ A; DA$\rightarrow$ CE; D $\rightarrow$ H; GH$\rightarrow$ C; AC$\rightarrow$ D ;

BC $\rightarrow$ AC; BC $\rightarrow$ D; DA $\rightarrow$ AH; DG $\rightarrow$ C;BC $\rightarrow$ AD;….}

(Lưu ý rằng, nếu mỗi thuộc tính được biểu diễn bằng một ký tự thì danh sách các thuộc tính có hoặc không có dấu phẩy đều được, còn giữa các phụ thuộc hàm phải có dấu chấm phẩy)

Các tính chất của tập F^+

1. *Tính phản xạ*:

   Với mọi tập phụ thuộc hàm F^+ ta luôn có F $\subseteq$ F^+

2. *Tính đơn điệu*:

   Nếu F $\subseteq$ G thì $F^+ \subseteq G^+$

3. *Tính luỹ đẳng:*

   Với mọi tập phụ thuộc hàm F ta luôn luôn có $F^{++} = F^+$.

5.3.2.Bao Đóng Của Tập Thuộc Tính X

Cho lược đồ quan hệ Q. giả sử F là tập các phụ thuộc hàm trong Q, X $\subseteq$ Q^+.

Bao đóng của tập thuộc tính X đối với F ký hiệu là X^+ (hoặc X_F^+) là tập tất cả các thuộc tính A $\in$ Q^+ được suy ra từ X dựa vào các phụ thuộc hàm trong F và hệ tiên đề Armstrong, nghĩa là:

X^+ = {A : A $\in$ Q^+ và X $\rightarrow$ A $\in$ F^+}

Ví dụ 5.3

Cho lược đồ quan hệ Q(ABCDEGH) và tập phụ thuộc hàm F

F = {B $\rightarrow$ A; DA$\rightarrow$ CE; D $\rightarrow$ H; GH$\rightarrow$ C; AC$\rightarrow$ D }

Hãy tính:

B^+; H^+;BC^+

Giải

Khi đó B^+ = BA ; (do có phụ thuộc hàm B $\rightarrow$ A)

H^+ = H. (do có phụ thuộc hàm H $\rightarrow$ H)

BC^+= BCADEH. (do có các phụ thuộc hàm:B $\rightarrow$ A;AC$\rightarrow$D;DA$\rightarrow$ CE; D $\rightarrow$ H)

Tương tự như tập bao đóng của tập phụ thuộc hàm F^+, tập bao đóng X^+ cũng chứa các phần tử của tập X, tức là $X \subseteq X^+$.

Các tính chất của bao đóng của tập thuộc tính X^+

Nếu X,Y là các tập con của tập thuộc tính Q thì ta có các tính chất sau đây:

1. *Tính phản xạ*: $X \subseteq X^+$
2. *Tính đơn điệu*: Nếu $X \subseteq Y$ thì $X^+ \subseteq Y^+$
3. *Tính luỹ đẳng*: $X^{++} = X^+$
4. $(XY)^+ \supseteq X^+Y^+$
5. $(X^+Y)^+ = (XY^+)^+ = (X^+Y^+)^+$
6. **$X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X^+$**
7. $X \rightarrow Y \Leftrightarrow Y^+ \subseteq X^+$

(Để giáo trình không bị ảnh hưởng quá nặng về lý thuyết toán, chúng tôi không muốn đi sâu về các khái niệm F^+, X^+ cũng như việc chứng minh các tính chất của F^+, X^+, Bạn đọc có thể tham khảo thêm ở tài liệu tham khảo [2])

5.3.3.Bài Toán Thành Viên

Qua phần trên ta nhận thấy X^+ được định nghĩa thông qua F^+. Vấn đề nảy sinh khi nghiên cứu lý thuyết CSDL là: Cho trước tập các phụ thuộc hàm F và một phụ thuộc hàm f, bài toán kiểm tra có hay không $f \in F^+$ gọi là *bài toán thành viên.*

Để giải quyết bài toán bài toán thành viên thật sự không đơn giản; vì mặc dù F là rất nhỏ nhưng F^+ thì có thể rất lớn. Tuy nhiên ta có thể giải bằng cách tính X^+ và so sánh X^+ với tập Y. Dựa vào tính chất **$X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X^+$**, ta có ngay câu trả lời $X \rightarrow Y \in F^+$ hay không ? Như vậy thay vì giải bài toán thành viên ta đưa về giải bài toán *tìm bao đóng của tập thuộc tính*.

5.3.4.Thuật Toán Tìm Bao Đóng Của Một Tập Thuộc Tính

Thuật toán 5.1

Thuật toán tìm bao đóng với độ phức tạp $O(N^2)$, với N là số lượng thuộc tính của lược đồ quan hệ Q.

Dữ Liệu Vào Q, F, $X \subseteq Q^+$

Dữ Liệu Ra X^+

Bước 1: Đặt $X^+ = X$

Bước 2: Temp = X^+

$\forall$ f $U \rightarrow V \in F$

if ($U \subseteq X^+$)

$X^+ = X^+ \cup V$.

F= F – f;

Bước 3: if (X^+=Temp)

"X^+ chính là kết quả cần tìm "

Dừng thuật toán

else

trở lại *Bước* 2:

Ví dụ 5.4:

Cho lược đồ quan hệ Q(ABCDEGH) và tập phụ thuộc hàm F

F = { f_1: B $\rightarrow$ A;

f_2: DA $\rightarrow$ CE;

f_3: D $\rightarrow$ H;

f_4: GH $\rightarrow$ C;

f_5: AC $\rightarrow$ D}

Tìm bao đóng của các tập X = {AC} dựa trên F.

Giải:

X^+ = AC

Do f_1, f_2, f_3, f_4 không thoả. f_5 thoả :

X^+=ACD

Lập lại bước 2. f_1 không thoả, f_2 thoả:

X^+=ACDE,

f_3 thoả :

X^+=ACDEH

Đến đây rõ ràng không có phụ thuộc hàm nào làm thay đổi X^+ nữa, thuật toán dừng lại và kết quả X^+ = ACDEH

Chú ý rằng bạn đọc hãy nắm thật kỹ thuật toán này – nó mở đầu cho một loạt ứng dụng quan trọng về sau. Tiếp theo, chúng tôi nêu lên một thuật toán tìm bao đóng với độ phức tạp tuyến tính để các bạn tham khảo.

Thuật toán 5.2

Thuật toán tìm bao đóng với độ phức tạp tuyến tính[3]

Bước 1:

Xây dựng mảng một chiều COUNT

Với COUNT(i) là số thuộc tính vế trái của phụ thuộc hàm thứ i

Bước 2:

Xây dựng mảng LIST với

LIST(A) = {$X \rightarrow Y$} $\in$ F, A $\in$ X}

(lưu chỉ số phụ thuộc hàm)

Bước 3:

$X^+ = X$

Bước 4:

Mọi thuộc tính A $\in$ X^+

Giảm COUNT($X \rightarrow Y$} đi một nếu A $\in$ X

Nếu COUNT{$X \rightarrow Y$} = 0 thì $X^+ = X^+ \cup Y$

Quay lại duyệt thuộc tính kế tiếp trong X^+ cho đến khi nào duyệt hết mọi phần tử của X^+ thì dừng lại. Kết quả X^+ là bao đóng cần tìm.

5.4. KHOÁ CỦA LƯỢC ĐỒ QUAN HỆ - MỘT SỐ THUẬT TOÁN TÌM KHOÁ

5.4.1.Định Nghĩa Khoá Của Quan Hệ (relation key)

Cho quan hệ $Q(A_1,A_2,\ldots,A_n)$ được xác định bởi tập thuộc tính Q^+ và tập phụ thuộc hàm F định nghĩa trên Q, cho K $\subseteq$ Q^+.

K là một khoá của Q nếu thoả đồng thời cả hai điều kiện sau:

1. $K \rightarrow Q^+ \in F^+$ (hay $K_F^+ = Q^+$)

(K chỉ thoả điều kiện 1 thì được gọi là siêu khoá)

2. Không tồn tại $K' \subset K$ sao cho $K'^{+} = Q^{+}$

Một lược đồ quan hệ có thể có nhiều siêu khoá, nhiều khoá.

5.5.2.Thuật Toán Tìm Một Khoá Của Một Lược Đồ Quan Hệ Q

Thuật toán 5.3

$K = Q^{+}$;

While $A \in K$ do

if $(K - A)^{+} = Q^{+}$ then $K = K - A$

K còn lại chính là một khoá cần tìm.

Nếu muốn tìm các khoá khác (nếu có) của lược đồ quan hệ, ta có thể thay đổi thứ tự loại bỏ các phần tử của K.

Ví dụ 5.7

Cho lược đồ quan hệ Q(ABC) và tập phụ thuộc hàm

F={ $A \rightarrow B$;

$A \rightarrow C$;

$B \rightarrow A$}

Hãy tìm một khóa của Q.

Giải:

K={A,B,C}

Loại thuộc tính A, do $(K-A)^{+} = Q^{+}$ nên K={B,C}

thuộc tính B không loại được do $(K - B)^{+} \neq Q^{+}$ nên K={B,C}

Loại thuộc tính C, do $(K-C)^{+} = Q^{+}$ nên K={B}.

Vậy một khóa của Q là B.

5.4.3. Thuật Toán Tìm Tất Cả Các Khoá Của Một Lược Đồ Quan Hệ

Thuật toán 5.4 (thuật toán cơ bản)

Bước 1:Xác định tất cả các tập con của Q

Để xác định tất cả các tập con của một lược đồ quan hệ $Q(A_1,A_2,\ldots,A_n)$ ta lần lượt duyệt tất cả 2^n-1 tập hợp con khác rỗng của Q^{+} (n là số thuộc tính của lược đồ quan hệ Q), kết quả tìm được giả sử là các tập thuộc tính: $S=\{X_1, X_2, \ldots,X_{2n-1}\}$

Bước 2: Tính X_i^{+}

Bước 3: Nếu $X_i^+ = Q^+$ thì Xi là siêu khoá

Nếu một tập con X_i (i = 1..,2_{n-1}) của Q^+ có bao đóng đúng bằng Q^+ thì tập con dó (theo định nghĩa trên) là một siêu khoá của Q.

Giả sử sau bước này có m siêu khoá: S = {S_1,S_2,…,S_m}

Bước 4

Xây dựng tập chứa tất cả các khoá của Q từ tập S

Xét mọi S_i,S_j con của S ($i \neq j$), nếu $S_i \subset S_j$ thì ta loại S_j (i,j=1..m), kết quả còn lại chính là tập tất cả các khoá cần tìm.

Ví dụ 5.8

Tìm tất cả các khoá của lược đồ quan hệ Q và tập phụ thuộc hàm F được cho như sau:

Q(A,B,C);

F={ A$\rightarrow$ B;

A $\rightarrow$ C;

B $\rightarrow$ A}

X_i	X_i^+	Siêu khóa	khóa
A	Q^+	A	A
B	Q^+	B	B
C	C		
AB	Q^+	AB	
AC	Q^+	AC	
BC	Q^+	BC	
ABC	Q^+	ABC	

Vậy lược đồ quan hệ Q có hai khoá là: {A} và {B}

Thuật toán trên thì dễ hiểu, dễ cài đặt, tuy nhiên nếu với n khá lớn thì phép duyệt để tìm ra tập tất cả các tập con của tập Q^+ là điều không hiệu quả, do vậy cần thu hẹp không gian duyệt. Chúng ta sẽ nghiên cứu thuật toán cải tiến theo hướng giảm số thuộc tính của tập cần duyệt.

Chú ý rằng thuật toán này tìm được tất cả các siêu khóa, tất cả các khóa

Thuật toán 5.5 (thuật toán cải tiến)

Trước khi đi vào thuật toán cải tiến, ta cần đưa thêm một số khái niệm sau:

-Tập nguồn(TN) chứa tất cả các thuộc tính có xuất hiện ở vế trái và không xuất hiện ở vế phải của tập phụ thuộc hàm. Những thuộc tính không tham gia vào bất kỳ một phụ thuộc hàm nào thì cũng đưa vào tập nguồn.

-Tập đích chứa tất cả các thuộc tính có xuất hiện ở vế phải và không xuất hiện ở vế trái của tập phụ thuộc hàm.

-Tập trung gian(TG) chứa tất cả các thuộc tính vừa tham gia vào vế trái vừa tham gia vào vế phải.

Dữ liệu vào: Lược đồ quan hệ phổ quát Q và tập phụ thuộc dữ liệu F

Dữ liệu ra: Tất cả các khoá của quan hệ

Bước 0. Tìm tập thuộc tính nguồn(TN), tập thuộc tính trung gian(TG)

Tìm tất cả các tập con của tập trung gian gọi là X_i (bằng phương pháp duyệt nhị phân)

if tập trung gian=$\varnothing$ then

Tập Khoá = Tập nguồn ;kết thúc

Ngược lại

Qua bước 1

Bước 1Tìm tất cả các tập con của tập trung gian: Xi

: S= ϕ

$\forall$ Xi $\in$ tập trung gian

if (Tập nguồn $\cup$ Xi)$^+$ = Q^+ then

S = S $\cup$ { Tập nguồn $\cup$ Xi}

{S là tập các siêu khoá cần tìm}

Bước 2: Tính TN $\cup$ Xi

Bước 3: Tính (TN $\cup$ Xi)$^+$

Bước 4: Nếu $X_i^+ = Q^+$ thì Xi là siêu khoá

Nếu một tập con TN $\cup$ Xi có bao đóng đúng bằng Q^+ thì TN $\cup$ Xi là một siêu khoá của Q.

Giả sử sau bước này có m siêu khoá: $S = \{S_1,S_2,\ldots,S_m\}$

Bước 5

Xây dựng tập chứa tất cả các khoá của Q từ tập S

Xét mọi S_i,S_j con của S ($i \neq j$), nếu $S_i \subset S_j$ thì ta loại S_j (i,j=1..m), kết quả còn lại chính là tập tất cả các khoá cần tìm.

Ví dụ 5.9 (Giải lại bài tập ở ví dụ 5.8)

Ap dụng thuật toán cải tiến ta có lời giải như sau:

TN ={ ϕ} ; TG ={A,B}

Gọi X_i là các tập con của tập TG:

X_i	(TN$\cup$ Xi)	(TN$\cup$ Xi)$^+$	Siêu khoá	khoá
ϕ	ϕ	ϕ		
A	A	Q^+	A	A
B	B	Q^+	B	B
AB	AB	Q^+	AB	

Vậy quan hệ trên có hai khoá là : [A] và [B]

Chú ý :

Thuật toán cải tiến này tìm được tất cả các khoá, nhưng không chắc tìm ra tất cả các siêu khoá.

5.5. PHỦ TỐI THIỂU (minimal cover)

5.5.1. Tập Phụ Thuộc Hàm Tương Đương (equivalent functional dependancy)

Cho F và G là hai tập phụ thuộc hàm, ta nói F và G tương đương (hay F phủ G hoặc G phủ F) và ký hiệu là $F^+ = G^+$ nếu và chỉ nếu mỗi phụ thuộc hàm thuộc F đều thuộc G^+ và mỗi phụ thuộc hàm thuộc G đều thuộc F^+.

Chẳng hạn cho lược đồ quan hệ Q(ABCDEGH), thì hai tập phụ thuộc hàm F và G (xác định trên Q) là tương đương.

F = {B → A; DA→ CE; D → H; GH→ C; AC→ D; DG → C}

G={B→ A; DA→ CE; D → H; GH→ C; AC→ D ;BC → AC;

BC → D; DA → AH; AC → DEH}

Bạn đọc hãy kiểm chứng lại ví dụ nhận xét này bằng cách sử dụng định nghĩa về tập phụ thuộc hàm tương đương và tính chất $X \to Y \in F^+ \Leftrightarrow Y \subseteq X^+$)

Ví dụ 5.5:

Chẳng hạn hai tập phụ thuộc hàm sau là tương đương:

Q(A,B,C)

F={ A→B; A→C; B→A; C→A; B→C}

G={ A→B; C→A; B→C}

(việc chứng minh xem như bài tập dành cho bạn đọc)

***5.5.2.P*hủ Tối Thiểu**

Để có thể phục vụ quá trình thiết kế cơ sở dữ liệu, cần đưa thêm khái niệm *tập phụ thuộc hàm tối thiểu.*

Bổ đề

Mỗi tập các phụ thuộc hàm F đều được phủ bởi tập các phụ thuộc hàm G mà vế phải của các phụ thuộc hàm G chỉ gồm một thuộc tính.

Định nghĩa

F được gọi là một tập phụ thuộc hàm tối thiểu nếu F thoả đồng thời ba điều kiện sau:

Điều kiện a) Vế phải của F chỉ có một thuộc tính.

Điều kiện b) Không $\exists$ f: $X \to A \in F$ và $Z \subset X$ mà:

$$F^+ = (F - (X \to A) \cup (Z \to A))^+$$

Điều kiện c) Không $\exists$ $X \to A \in F$ mà:

$$F^+ = (F - (X \to A))^+$$

Trong đó vế phải của mỗi phụ thuộc hàm ở điều kiện a) chỉ có một thuộc tính, nên bảo đảm không có thuộc tính nào ở vế phải là dư thừa. điều kiện b) bảo đảm không có một thuộc tính nào tham gia vế trái của phụ thuộc hàm là dư thừa. điều kiện c)bảo đảm cho tập F không có một phụ thuộc hàm nào là dư thừa.

Chú ý rằng một tập phụ thuộc hàm luôn tìm ra ít nhất một phủ tối thiểu và nếu thứ tự các phụ thuộc hàm trong tập F là khác nhau thì có thể sẽ thu được những phủ tối thiểu khác nhau.

5.5.3.Thuật Toán Tìm Phủ Tối Thiểu

Thuật toán 5.6

Dữ liệu vào : Lược đồ quan hệ ban đầu Q và tập phụ thuộc hàm F, số lượng

phụ thuộc hàm trong F là m.

Dữ liệu ra : Tập phụ thuộc hàm tối thiểu của F

Bước 1:

Tách vế phải mỗi phụ thuộc hàm trong F sao cho vế phải của mỗi phụ thuộc hàm chỉ chứa một thuộc tính (điều này luôn thực hiện được do bổ đề trên)

$\forall$ f: $X \rightarrow Y \in F$

$\forall$ $A \in Y$

$g = X \rightarrow A$

$F = F \cup g$

$m = m + 1$

Cuối $\forall$

Cuối $\forall$

Bước 2. Tìm tập phụ thuộc hàm đầy đủ bằng cách loại bỏ các thuộc tính dư thừa ở vế trái của từng phụ thuộc hàm.

$\forall$ f $X \rightarrow A \in F$

$\forall$ $B \in X$

$X' = X - B$

If $(X' \rightarrow A \in F^+)$ $X = X'$

Cuối $\forall$

Cuối $\forall$

Chú ý:

Việc tìm tất cả các tập $X' \subseteq X$ theo thuật toán trên hoàn toàn thay thế được việc tìm X' cách tìm các tập con của X.

Bước 3. Loại bỏ các phụ thuộc hàm dư thừa trong F.

$\forall$ f $\in$ F

G = F – f {loại f ra khỏi F. và lưu { F – f} vào G }

If ($F^+ = G^+$) {gọi thủ tục kiểm tra F, G tương đương ở dưới}

F = G {cập nhật lại F mới}

Cuối $\forall$

Ví dụ 5.6

Cho lược đồ quan hệ Q và tập phụ thuộc F như sau:

Q(ABCD)

F={ AB→CD;

B→C;

C→D}

Hãy tìm phủ tối thiểu của F.

Giải:

kết quả của bước 1 là:

F={ AB→C; AB→D; B→C; C→D}

kết quả của bước 2 là:

F={ B→C; B→D; B→C; C→D}

kết quả của bước 3 cho phủ tối thiểu:

Q(ABCD)

F={ B→C; C→D }

5.6.DẠNG CHUẨN CỦA LƯỢC ĐỒ QUAN HỆ

Khi thiết kế một hệ thống thông tin, thì việc lập lược đồ CSDL đạt đến một tiêu chuẩn nào đó là một việc làm quan trọng. Chất lượng của hệ thống thông tin phụ thuộc rất nhiều vào lược đồ CSDL này. Việc xác định chuẩn cho một lược đồ quan hệ có liên quan mật thiết với thuật toán tìm khoá. Có thể

khẳng định rằng thuật toán tìm khoá là một trong những thuật toán quan trọng của lý thuyết thiết kế cơ sở dữ liệu.

Chất lượng thiết kế của một lược đồ CSDL có thể được đánh giá dựa trên nhiều tiêu chuẩn trong đó sự trùng lắp thông tin và chi phí kiểm tra các ràng buộc toàn vẹn là hai tiêu chuẩn quan trọng. Sau đây là một số dạng chuẩn để đánh giá mức độ tốt/xấu của một lược đồ cơ sở dữ liệu.

Trước hết, chúng ta cùng tìm hiểu một số khái niệm liên quan.

5.6.1.Một Số Khái Niệm Liên Quan Đến Các Dạng Chuẩn

Thuộc tính khoá/không khoá

A là một thuộc tính khoá nếu A có tham gia vào bất kỳ một khoá nào của quan hệ, ngược lại A gọi là thuộc tính không khoá.

Ví dụ 5.10

Cho lược đồ quan hệ Q(ABC) và tập phụ thuộc hàm

F={ A$\rightarrow$ B;

A $\rightarrow$ C;

B $\rightarrow$ A}

Có hai khóa là A và B. khi đó thuộc tính khoá là A, B; thuộc tính không khóa là: C.

Thuộc tính phụ thuộc đầy đủ- phụ thuộc hàm đầy đủ.

A là một thuộc tính phụ thuộc đầy đủ vào tập thuộc tính X nếu X $\rightarrow$A là một phụ thuộc hàm đầy đủ (tức là không tồn tại X' $\subset$ X sao cho X' $\rightarrow$ A $\in$ F^+)

Ví dụ 5.10

Cho lược đồ quan hệ Q(ABC) và tập phụ thuộc hàm

F={ A $\rightarrow$ B

A$\rightarrow$ C;

AB $\rightarrow$ C

}

thì A $\rightarrow$;B A $\rightarrow$ C là các phụ thuộc hàm đầy đủ. Phụ thuộc hàm AB $\rightarrow$ C không là phụ thuộc hàm đầy đủ vì có A $\rightarrow$ C.

Chú ý rằng, một phụ thuộc hàm mà vế trái chỉ có một thuộc tính là phụ thuộc hàm đầy đủ.

5.6.2.Dạng Chuẩn Một (First Normal Form)

Lược đồ quan hệ Q được gọi là đạt dạng chuẩn 1 (1NF) nếu và chỉ nếu toàn bộ các thuộc tính của Q đều mang giá trị đơn.

Chẳng hạn xét *quan hệ*

MASV	HOTEN	MONHOC	DIEMTHI
00CDTH189	Nguyễn Văn Thành	*Kỹ Thuật Lập Trình*	*8*
		Cơ Sở Dữ Liệu	*9*
		Cấu Trúc Dữ Liệu	*7*
00CDTH211	Trần Thu Hà	*Kỹ Thuật Lập Trình*	*5*

Lược đồ quan hệ này không đạt dạng chuẩn 1 vì các thuộc tính MONHOC, DIEMTHI không mang giá trị đơn (chẳng hạn sinh viên Nguyễn Văn Thành có thuộc tính môn học là *Kỹ Thuật Lập Trình, Cơ Sở Dữ Liệu, Cấu Trúc Dữ Liệu.*

Ta hoàn toàn có thể đưa quan hệ trên về dạng chuẩn 1 như sau:

MASV	*HOTEN*	*MONHOC*	*DIEMTHI*
00CDTH189	Nguyễn Văn Thành	*Kỹ Thuật Lập Trình*	*9*
00CDTH189	Nguyễn Văn Thành	*Cơ Sở Dữ Liệu*	*8*
00CDTH189	Nguyễn Văn Thành	*Cấu Trúc Dữ Liệu*	*7*
00CDTH211	Trần Thu Hà	*Kỹ Thuật Lập Trình*	*5*

Chú ý rằng nếu ta không nói gì thêm, thì lược đồ quan hệ đang xét ít nhất là đạt dạng chuẩn 1.

5.6.3.Dạng Chuẩn 2 (second normal form)

Một lược đồ quan hệ Q đạt dạng chuẩn 2 nếu Q đạt dạng chuẩn 1 và tất cả các thuộc tính không khoá của Q đều phụ thuộc đầy đủ vào khoá.

Nếu một lược đồ quan hệ không đạt chuẩn 2 thì ta nói nó đạt dạng chuẩn1.

Chẳng hạn xét lược đồ quan hệ

Q(A,B,C,D) và

F={ $AB \rightarrow C,D$;

$B \rightarrow D$;

$C \rightarrow A\}$

Khoá là {A,B} và {B,C}. Do đó D là thuộc tính không khoá, $A,B \rightarrow D$ không là phụ thuộc hàm đầy đủ vì có $B \rightarrow D$.

Vậy Q đạt chuẩn 1.

Ví dụ 5.11:

Xác định dạng chuẩn của lược đồ quan hệ sau.

Q(GMVNHP)

F={G→N; G→H; G→P; M→V; NHP→M}

Dễ thấy khoá của Q là G.

Thuộc tính không khoá là M,V,N,H,P.

Do các phụ thuộc hàm $G \rightarrow M$; $G \rightarrow V$; $G \rightarrow N$; $G \rightarrow H$; $G \rightarrow P$ là các phụ thuộc hàm đầy đủ, nên lược đồ quan hệ Q đạt dạng chuẩn 2

Hệ quả:

-Q đạt 2NF nếu Q là 1NF và tập thuộc tính không khoá của Q bằng rỗng.

-Nếu khoá của quan hệ có một thuộc tính thì quan hệ đó ít nhất đạt chuẩn 2.

Ví dụ 5.12:

Q(ABCDEH)

$F=\{A \rightarrow E; C \rightarrow D; E \rightarrow DH\}$

Dễ thấy khoá của Q là K={ABC}

D là thuộc tính không khoá. và $C \rightarrow D$, vì C là tập con thực sự của khoá nên Q không đạt dạng chuẩn 2.

Dạng Chuẩn 3 (third normal form)

Một lược đồ quan hệ Q đạt dạng chuẩn 3 nếu mọi phụ thuộc hàm $X \rightarrow A \in F^+$ (F là tập phụ thuộc không hiển nhiên định nghĩa trên Q, A là thuộc tính đơn, X là tập thuộc tính con của tập Q^+), thì một trong hai điều kiện sau được thoả:

Hoặc X là một siêu khoá của Q

Hoặc A là một thuộc tính khoá

Nhận xét:

Nếu Q đạt chuẩn 3 thì Q đạt chuẩn 2

Ví dụ 5.13

Cho lược đồ quan hệ Q(ABCD)

F=[AB $\rightarrow$ C ; D $\rightarrow$ B C $\rightarrow$ ABD]

K1=[AB]; K2=[AD];K3=[C]

là các khoá, vậy Q không có thuộc tính không khoá nên Q đạt chuẩn 3

Hệ quả

Nếu lược đồ quan hệ Q,F mà Q không có thuộc tính không khoá thì Q đạt chuẩn 3.

Ví dụ 5.14

Xác định dạng chuẩn của lược đồ quan hệ sau.

Q(NGPM)

F={NGP$\rightarrow$M;

M$\rightarrow$P}

Dễ thấy các khoá của Q là {NGP}, {NGM}

NGP $\rightarrow$ M có vế trái là siêu khoá

M $\rightarrow$ P có vế phải là thuộc tính khoá.

Nên Q đạt chuẩn 3.

5.6.5.Dạng Chuẩn BC (Boyce Codd normal form)

Một lược đồ quan hệ Q ở dạng chuẩn BC nếu với mỗi phụ thuộc hàm không hiển nhiên X $\rightarrow$ A $\in$ F thì X là một siêu khoá của Q.

*Nhận xét***:** Nếu Q đạt chuẩn BC thì Q đạt chuẩn 3

Ví dụ 5.15

Xác định dạng chuẩn của lược đồ quan hệ sau.

Q(ACDEIB)

F={ACD$\rightarrow$EBI;

CE$\rightarrow$AD}

Dễ thấy Q có hai khoá là: ACD và CE. Các phụ thuộc hàm của F đều có vế trái là siêu khoá, nên Q đạt dạng chuẩn BC.

ĐỊNH LÝ : Các lớp dạng chuẩn của một lược đồ quan hệ có quan hệ lồng nhau: nghĩa là lớp sau nằm trọn trong lớp trước. BCNF $\subset$ 3NF $\subset$ 2NF $\subset$ 1NF

Ví dụ 5.16

Chẳng hạn cho lược đồ quan hệ Q(ABCD) và F = [AB $\rightarrow$ C; D $\rightarrow$ B; C$\rightarrow$ ABD] thì Q đạt chuẩn 3NF nhưng không là BCNF

Nếu F = [B $\rightarrow$ D, A $\rightarrow$ C, C $\rightarrow$ ABD] thì Q đạt dạng chuẩn 2NF nhưng không là 3 NF.

Dạng chuẩn của một lược đồ cơ sở dữ liệu là dạng chuẩn thấp nhất của các lược đồ quan hệ con.

Chú ý: Các dạng chuẩn cao hơn như dạng chuẩn bốn (với phụ thuộc đa trị), dạng chuẩn năm (với phụ thuộc chiếu kết) có thể xem các tài liệu tham khảo đã chỉ ra.

BÀI TậP

5.1. a) Cho lược đồ quan hệ Q(ABCD), r là một quan hệ trên Q.

r

A	B	C	D
a1	b1	c1	d1
a1	b2	c1	d1
a1	b3	c2	d1
a2	b2	c2	d2

phụ thuộc hàm nào sau đây không thoả r

a) $D \rightarrow A$;

b) $A,C \rightarrow D$;

c) $CD \rightarrow A$;

d) $D \rightarrow B$;

b.Cho lược đồ quan hệ Q(ABCD), r là quan trên Q được cho như sau:

A	**B**	**C**	**D**
a1	b1	c1	d2
a3	b1	c2	d1
a1	b1	c2	d2

Những phụ thuộc hàm nào sau đây thoả r ?

$AB \rightarrow D$; $C \rightarrow B$; $B \rightarrow C$; $BC \rightarrow A$; $BD \rightarrow A$.

c.Cho lược đồ quan hệ Q(ABCD), r là quan hệ được cho như sau:

A	**B**	**C**	**D**
x	u	x	y
y	x	z	x
z	y	y	y
y	z	w	z

Những phụ thuộc hàm nào sau đây không thoả r ?

$A \rightarrow B$; $A \rightarrow C$; $B \rightarrow A$; $C \rightarrow D$; $D \rightarrow C$; $D \rightarrow A$

5.2. a.Cho lược đồ quan hệ Q(ABCD) và tập phụ thuộc hàm

F = { $A \rightarrow B$;

$BC \rightarrow D$}.

Những phụ thuộc hàm nào sau đây thuộc F^+ ?

$C \rightarrow D$; $A \rightarrow D$; $AD \rightarrow C$; $AC \rightarrow D$; $BC \rightarrow A$; $B \rightarrow CD$.

b.Cho lược đồ quan hệ Q(ABCDEGH) và tập phụ thuộc hàm

F ={ $AB \rightarrow C$;

$B \rightarrow D$;

$CD \rightarrow E$;

$CE \rightarrow GH$;

$G \rightarrow A$}

Những phụ thuộc hàm nào sau đây không thuộc vào F^+ ?

$AB \rightarrow E$; $AB \rightarrow GH$; $CGH \rightarrow E$; $CB \rightarrow E$; $GB \rightarrow E$.

c)Cho lược đồ quan hệ Q,F như sau:

với Q=(ABCD)

F=[$A \rightarrow B$;

$A \rightarrow C$].

Trong các phụ thuộc hàm sau, những phụ thuộc hàm được suy ra từ F ?

$A \rightarrow D$; $C \rightarrow D$; $AB \rightarrow B$; $BC \rightarrow A$; $A \rightarrow BC$

5.3. Cho lược đồ quan hệ Q(ABCD) và tập phụ thuộc hàm

F={ $A \rightarrow D$;

$D \rightarrow A$;

$AB \rightarrow C$}

a.Tính AC^+

b.Chứng minh $BD \rightarrow C \in F^+$

5.4. a)Q(ABCDEG)

Cho F={$AB \rightarrow C$;

C → A;

BC → D;

ACD → B;

D → EG;

BE → C ;

CG → BD;

CE → AG}

X=[BD], X^+=*?*

Y=[CG], Y^+=*?*

b)Cho lược đồ quan hệ Q và tập phụ thuộc hàm F

F={ AB → E;

AG → I;

BE → I;

E → G ;

GI → H }

Chứng minh rằng AB → GH.

c.Tương tự cho tập phụ thuộc hàm

F = { AB → C;

B → D;

CD → E;

CE → GH;

G → A}

Chứng minh rằng AB → E; AB → G

d.Q(ABCDEGH)

F = {B → A; DA→ CE; D → H; GH→ C; AC→ D }

Hãy tìm một khoá của Q ?

5.5. Hãy tìm tất cả các khoá cho lược đồ quan hệ sau:

Q(BROKER,OFFICE,STOCK,QUANTITY,INVESTOR,DIVIDENT)

F={

STOCK→DIVIDENT

INVESTOR → BROKER

INVESTOR, STOCK → QUANTITY

BROKER → OFFICE }

5.6.Q(A,B,C,D)

F=[AB → C;

D → B;

C → ABD]

Hãy tìm tất cả các khoá của Q

5.7. Cho lược đồ quan hệ Q(MSCD,MSSV,CD,HG) và tập phụ thuộc F như sau:

F=[MSCD→CD;

CD→MSCD;

CD,MSSV→HG;

MSCD,HG→MSSV;

CD,HG→MSSV;

MSCD,MSSV→HG}

Hãy tìm phủ tối thiểu của F.

5.8 Xác định phủ tối thiểu của tập phụ thuộc hàm sau:

Q(ABCDEG)

F = {AB → C;

C → A;

BC → D;

ACD → B;

D → EG;

BE → C;

CG → BD;

CE → AG}

5.9. Các nhận xét sau đúng (Đ) hay sai (S) ? (kẻ bảng sau và ghi Đ hoặc S cho mỗi câu trên)

a	b	c	d	e	f	g	H

a.Cho Q và F={AB → C; A → B} thì Q đạt dạng chuẩn 1.

b.Một lược đồ quan hệ Q luôn tìm được ít nhất một khoá.

c.Nếu $XY \rightarrow Z$ thì $X \rightarrow Z$ và $Y \rightarrow Z$.

d.Các thuộc tính không tham gia vào vế phải của bất kỳ phụ thuộc hàm nào thì phải là thuộc tính tham gia vào khoá.

e.Nếu $X \rightarrow Y$ và $YZ \rightarrow W$ thì $XZ \rightarrow W$

f.Nếu Q đạt dạng chuẩn một và khoá của Q chỉ có một thuộc tính thì Q đạt dạng chuẩn ba.

g.Một tập phụ thuộc hàm F có thể có nhiều tập phủ tối thiểu.

h.Nếu $X \rightarrow Y$ và $U \rightarrow V$ thì $XU \rightarrow YV$.

5.10 a.Cho Q(ABCD) và

F = {$AB \rightarrow C$; $D \rightarrow B$; $C \rightarrow ABD$}.

Hãy kiểm tra xem $AB \rightarrow D$ có thuộc F^+ hay không ?

Hãy tìm tất cả các khoá của lược đồ quan hệ Q. Xác định dạng chuẩn của Q.

b.Cho Q(A,B,C,D)

và F={$C \rightarrow A$; $A \rightarrow C$; $AD \rightarrow B$; $BC \rightarrow D$; $AB \rightarrow D$; $CD \rightarrow B$ }

Hãy tìm phủ tối thiểu của F.

5.15. Cho biết dạng chuẩn của các lược đồ quan hệ sau:

a.Q(ABCDEG);

F=[$A \rightarrow BC$, $C \rightarrow DE$, $E \rightarrow G$]

b.Q(ABCDEGH);

F=[$C \rightarrow AB$, $D \rightarrow E$, $B \rightarrow G$]

c.Q(ABCDEGH);

F=[$A \rightarrow BC$. $D \rightarrow E$, $H \rightarrow G$]

d.Q(ABCDEG);

F=[$AB \rightarrow C$; $C \rightarrow B$; $ABD \rightarrow E$;$G \rightarrow A$]

e.Q(ABCDEGHI);

F=[$AC \rightarrow B$; $BI \rightarrow ACD$; $ABC \rightarrow D$; $H \rightarrow I$; $ACE \rightarrow BCG$, $CG \rightarrow AE$]

PHỤ LỤC(MỘT SỐ ĐỀ KIỂM TRA, ĐỀ THI MÔN CSDL)

TRƯỜNG CĐ KỸ THUẬT CAO THẮNG

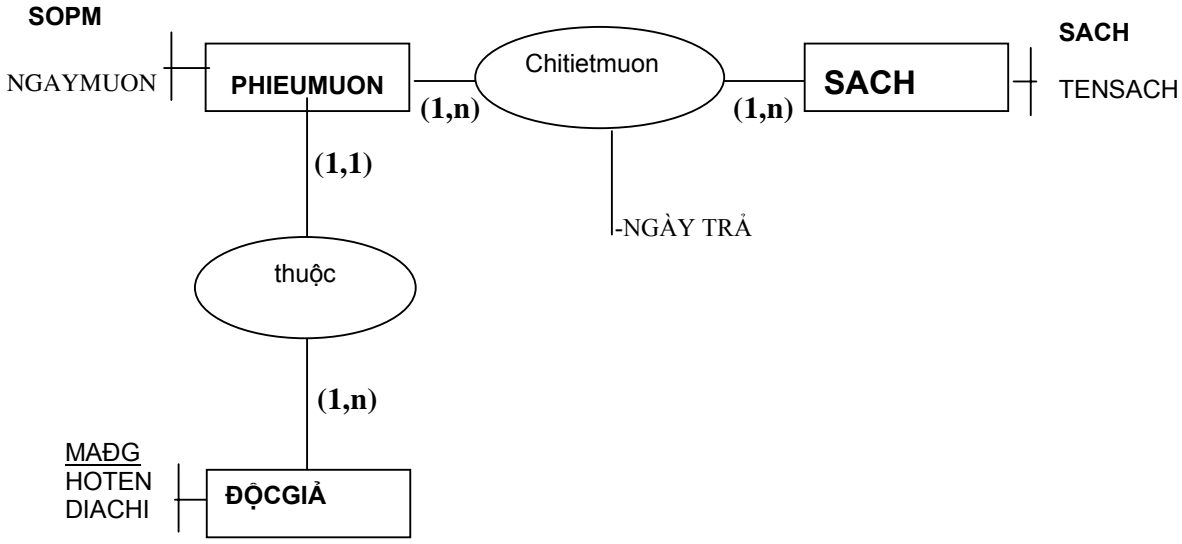
KHOA ĐIỆN TỬ - TIN HỌC

BÀI KIỂM TRA HỆ SỐ 2 - MÔN CƠ SỞ DỮ LIỆU **(bài thứ nhất)**

(Thời gian 45 phút)

CÂU 1:(3 điểm)

Chuyển mô hình thực thể kết hợp sau sang mô hình dữ liệu quan hệ. Tìm khóa cho mỗi lược đồ quan hệ con.



CÂU 2:(3 điểm)

Cho lược đồ cơ sở dữ liệu với 3 lược đồ quan hệ sau:

Nhanvien(MANV,HOTEN,NGAYSINH,ĐIACHI, MAPB,MACV)

Mỗi nhân viên có một mã nhân viên (MANV) duy nhất, mỗi MANV xác định các thông tin: họ và tên (HOTEN), ngày sinh (NGAYSINH) - NGÀY SINH là dạng ngày tháng năm, địa chỉ (ĐIACHI), mã phòng ban(MAPB) và mã chức vụ (MACV).

Phongban (MAPB,TENPB)

Mỗi phòng ban có một mã phòng ban (MAPB) duy nhất, mỗi MAPB xác định tên phòng ban (TENPB).

Chucvu(MACV,TENCV,PHUCAP)

Mỗi chức vụ có một mã chức vụ (MACV) duy nhất, mỗi MACV xác định tên chức vụ (TENCV), phu cấp chức vụ(PHUCAP)

Câu 1.Tìm khóa cho mỗi lược đồ quan hệ trên.

Câu 2:Dựa vào lược đồ cơ sở dữ liệu trên, hãy thực hiện các yêu cầu sau bằng ngôn ngữ đại số quan hệ

a.Lập danh sách các nhân viên có MAPB là “KT”. Danh sách cần MANV, HOTEN,NGAYSINH,ĐIACHI.

b.Lập danh sách các nhân viên có phụ cấp chức vụ. Danh sách cần các thông tin: MANV, HOTEN, ĐIACHI, TENPB, TENCV.

CÂU 3:(4 điểm)

a.(1,0 điểm)

Cho lược đồ quan hệ Q(ABCD), r và s là hai quan hệ được cho như sau:

r

A	B	C	D
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1

s

A	B	C	D
2	1	1	1
2	2	1	1
1	1	1	0
x	y	z	v

Tìm r - s

Tính r + s

Tính r - s

b.(1,0 điểm)

Cho hai lược đồ quan hệ Q1(ABC) và Q2(DEF), r và s là hai quan hệ được cho như sau:

r

A	**B**	**C**
1	2	3
a	b	c
x	y	z

s

D	**E**	**F**
1	e	f
a	e	f
5	6	7

A = D

Tìm r |><| s

. c.(1,0 điểm)

Cho hai lược đồ quan hệ Q1(ABC) và Q2(DE), r và s là hai quan hệ được cho như sau

r s

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

B > D

Tìm r |><| s

d.(1,0 điểm)

Cho quan hệ về khả năng lái các loại máy bay của các phi công:

KHANANG(SỐ HIỆU PHI CÔNG, SỐ HIỆU MÁY BAY)

SỐ HIỆU PHI CÔNG	SỐ HIỆU MÁY BAY
32	102
30	101
30	103
32	103
33	100
30	102
31	102
30	100
31	100

Hãy cho biết các phi công có khả năng lái được cả 3 loại máy bay:

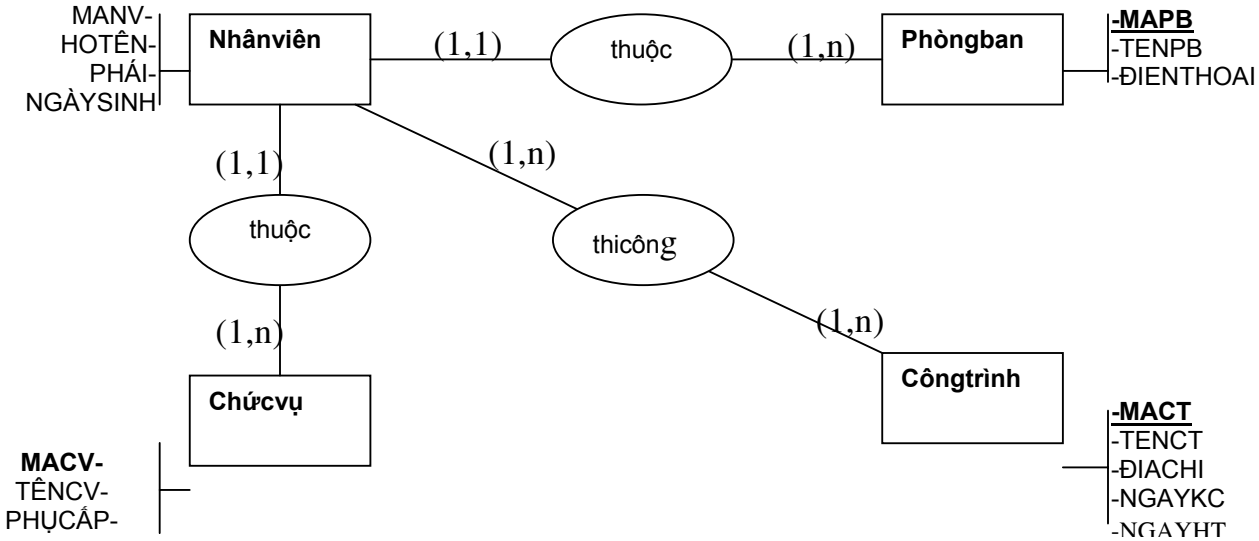
100,101,103 ? Đây là kết quả của phép toán quan hệ nào ?

Hết

TRƯỜNG CĐ KỸ THUẬTCAO THẮNG
KHOA ĐT-TH

ĐỀ KIỂM TRA GIỮA KỲ
MÔN CƠ SỞ DỮ LIỆU
thời gian làm bài 60 phút

CÂU I: (2,0đ)



CÂU II: (3,0đ)

1.Cho hai lược đồ quan hệ Q_1(ABC) và Q_2(DEF), r và s là hai quan hệ được cho như sau:

r

A	**B**	**C**
2	5	7
4	2	6
1	5	9

s

D	**E**	**F**
2	4	3
1	5	7
5	2	3

a.Tính r |><|$_{A=D}$ s

b.Tính r |><|$_{B > F}$ s

2.Cho hai lược đồ quan hệ Q_1(ABCD) và Q_2(CD), r và s là hai quan hệ được cho như sau:

r

A	**B**	**C**	**D**
3	4	5	6
1	2	3	4
1	2	5	6

s

C	**D**
3	4
5	6

2 3 5 6

1 2 4 5

Tính r ÷ s

CÂU III (5,0đ)

Một phòng giáo dục quận muốn lập một hệ thống thông tin để quản lý việc làm thi tốt nghiệp phổ thông cơ sở (lớp 9). Công việc làm thi được tổ chức như sau:

Lãnh đạo phòng giáo dục thành lập nhiều hội đồng thi (mỗi hội đồng thi gồm một trường hoặc một số trường gần nhau). Mỗi hội đồng thi có một mã số duy nhất (MAHĐT), một hội đồng thi xác định tên hội đồng thi(TENHĐT), họ tên chủ tịch hội đồng(TENCT), địa chỉ (ĐCHĐT),điện thoại(ĐTHĐT).

Mỗi hội đồng thi được bố trí cho một số phòng thi, mỗi phòng thi có một số hiệu phòng(SOPT) duy nhất, một phòng thi xác định địa chỉ phòng thi (ĐCPT). Số hiệu phòng thi được đánh số khác nhau ở tất cả các hội đồng thi.

Giáo viên của các trường trực thuộc phòng được điều động đến các hội đồng để coi thi, mỗi trường có thể có hoặc không có thí sinh dự thi, mỗi trường có một mã trường duy nhất (MATR), mỗi mã trường xác định tên trường(TENTR),địa chỉ (ĐCTR), loại hình đào tạo (LHĐT) (Công lập, chuyên, bán công, dân lập,…). Các giáo viên của một trường có thể làm việc tại nhiều hội đồng thi. Một giáo viên có một mã giáo viên(MAGV), một mã giáo viên xác định tên giáo viên (TENGV), chuyên môn giảng dạy (CHUYENMON), chức danh trong hội đồng thi(CHUCDANH).

Các thí sinh dự thi có một số báo danh duy nhất(SOBD), mỗi số báo danh xác định tên thí sinh(TENTS), ngày sinh (NGSINH), giới tính (PHAI), mỗi thí sinh được xếp thi tại một phòng thi nhất định cho tất cả các môn, mỗi thí sinh có thể có chứng chỉ nghề (CCNGHE) hoặc không (thuộc tính CCNGHE kiểu chuỗi, CCNGHE=”x” nếu thí sinh có chứng chỉ nghề và CCNGHE bằng rỗng nếu thí sinh không có chứng chỉ nghề).Thí sinh của cùng một trường chỉ dự thi tại một hội đồng thi.

Mỗi môn thi có một mã môn thi duy nhất (MAMT), mỗi mã môn thi xác định tên môn thi(TENMT), buổi thi (BUOI), ngày thi (NGAY). Giả sử toàn bộ các thí sinh trong hội đồng thi đó đều thi chung một số môn do sở giáo dục quy định (có thể thay đổi tuỳ theo năm). Mỗi môn thi

được tổ chức trong một buổi của một ngày nào đó. Ứng với mỗi môn thi một thí sinh có một điểm thi duy nhất(ĐIEMTHI)

Dựa vào phân tích ở trên, giả sử ta có lược đồ cơ sở dữ liệu sau:

HĐ(MAHĐT,TENHĐT, TENCT, ĐCHĐT,ĐTHĐT)

PT(SOPT,ĐCPT,MAHĐT)

TS(SOBD, TENTS,NGSINH,PHAI,CCNGHE, MATR,SOPT)

MT(MAMT,TENMT,BUOI,NGAY)

GV(MAGV,TENGV,CHUYENMON,CHUCDANH,MAHĐT,MATR)

TR(MATR,TENTR,ĐCTR,LHĐT)

KQ(SOBD,MAMT,ĐIEMTHI)

YÊU CẦU

1.Hãy xác định khóa cho mỗi lược đồ quan hệ trên.

2.Dựa vào lược đồ cơ sở dữ liệu trên, hãy thực hiện các yêu cầu sau bằng SQL.

a. Danh sách các thí sinh thi tại phòng thi có số hiệu phòng thi (SOPT) là “100”
   Yêu cầu các thông tin: SOBD,TENTS,NGSINH,TENTR
b. Kết quả của môn thi có mã môn thi (MAMT) là “T” của tất cả các thí sinh có mã trường (MATR) là “NTMK”, kết quả được sắp theo chiều giảm dần của điểm thi (ĐIEMTHI).
   Yêu cầu các thông tin:SOBD,TENTS, ĐIEMTHI
c. Tổng số thí sinh có chứng chỉ nghề(CCNGHE) của mỗi trường, thông tin cần được sắp theo chiều tăng dần của TENTR.
   Yêu cầu các thông tin: MATR, TENTR, SOLUONGCC, trong đó SOLUONGCC là thuộc tính tự đặt.

Hết

(Sinh viên không được sử dụng tài liệu

Cán bộ coi thi không giải thích)

TRƯỜNG CĐ KỸ THUẬTCAO THẮNG
KHOA ĐT-TH

ĐỀ KIỂM CHƯƠNG 3
MÔN CƠ SỞ DỮ LIỆU
thời gian làm bài 90 phút

HỌ VÀ TÊN SV: MASV:	ĐIỂM

CÂU	*KẾT QUẢ*	*CÂU*	*KẾT QUẢ*
1		26	
2		27	
3		28	
4		29	
5		30	
6		31	
7		32	
8		33	
9		34	
10		35	
11		36	
12		37	
13		38	
14		39	
15		40	
16		41	
17		42	
18		43	
19		44	
20		45	
21		46	
22		47	
23		48	
24		49	
25		50	

1. Consider a relation Q(A,B,C,D) with the following functional dependencies:

   $A \rightarrow B$ and $B \rightarrow C$

   The number of superkeys of Q is:

   (a) 2

   (b) 3

   (c) 4

   (d) 5

   (e) None of the above

2. Which of the following functional dependencies must be FALSE ?

   a) $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

   b) $X \rightarrow Y; YW \rightarrow Z \Rightarrow XW \rightarrow Z.$

   c) $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$

   d) $X \rightarrow Y; Z \subseteq Y \Rightarrow X \rightarrow Z$

   e) None of the above

3. Consider relation S(B, G, U, N, A) with the FD's: $BG \rightarrow U$, $G \rightarrow N$, $NA \rightarrow B$

What are all the keys of S ?

   a) {B, G} and {N, A}

   b) {U, A}

   c) {G, A}

   d) {B,N}

   e) {B, G, U, N}

   f) {B, G, A} and {G, N, A}

4. Which one of the following is correct?

   (a) All FDs must involve the attributes of the same relation.

   (b) All FDs must involve only a single attribute on the left side.

   (c) All FDs must involve only a single attribute on the right side.

   (d) All FDs must involve only single attributes on both sides.

   (e) None of the above

5. Assume the relation R(A, B, C ,D, E) is in at least 3NF. Which of the following functional dependencies must be FALSE?

   a. A, B → C

   b. A, B → D

   c. C, D → E

   d. None of the above

6. SQL provides a number of special aggregate functions. Which one of the following is not included in SQL?

   (a) SUM

   (b) MAX

   (c) MIN

   (d) COUNT

   (e) MEDIAN

7. Which of the following finds all groups meeting stated conditions?

   (a) Select

   (b) Where

   (c) Find

   (d) Having

8. The _____ clause is used to restrict the groups returned by a query.

   (a) FROM

   (b) WHERE

   (c) HAVING

   (d) GROUP BY

9. To get all the customers from Hawaii sorted together, which of the following would be used?

   (a) Order By

   (b) Group By

   (c) Having

   (d) Sort

10. The ____ set operator will show all rows common to both tables A and B.

    (a) intersection

    (b) union

    (c) difference

    (d) product

11. Using the product operator, if table A has 2 rows and table B has 4 rows, the number of rows in the product of these two tables is:

    (a) 4.

    (b) 8.

    (c) 16.

    (d) 20.

12. Suppose relation R(A,B,C,D,E) has the following functional dependencies:

    $A \rightarrow B$

    $B \rightarrow C$

    $BC \rightarrow A$

    $A \rightarrow D$

    $D \rightarrow E$

Which of the following is not a key?

    (a) A

    (b) E

    (c) B

    (d) D

    (e) (b) and (d)

13. Consider a relation Q with five attributes L V O B Y. You are given the follwoing dependencies: $L \rightarrow V$; $VO \rightarrow Y$ and $YB \rightarrow L$.

    The number of superkeys of Q is: (not superkeys)

    (a).2

    (b).3

    (c),4

    (d).5

14. Use the following tables and data,All of the fields are Integers.The table names are Q

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

How many records does the following SQL example return?

SELECT * FROM Q WHERE A>=5 OR D>=7;

a) 1

b) 2

c) 3

d) 4

15. How many records does the following SQL example return?

SELECT sum(A) AS S FROM Q GROUP BY A;

a)	b)	c)	d)
S	S	S	S
11	1	4	None of the above
	1	7	
	2	6	
	2	3	
	5	5	

16. The SQL below will return one value. What is it?

SELECT sum(A) AS S FROM Q GROUP BY A HAVING count(A)>1;

a)	b)	c)	d)
S	S	S	S
2	9	2	None of the above
4		2	

5

17. How many records does the following SQL example return?

SELECT count(*) FROM Q GROUP BY A, B;

a) 2

b) 3

c) 4

d) 6

e) None of the above

18. Use the following tables and data..All of the fields are Integers.The table names are R and S

R

A	B
6	7
4	2

S

C	D	E
8	4	2
3	5	7
6	2	9

How many records does the following SQL example return?

SELECT COUNT(*) FROM R, S WHERE R.A=S.D OR R.B=S.E;

a) 1

b) 2

c) 3

d) 4

19. How many records does the following SQL example return?

SELECT COUNT(*) FROM R, S WHERE R.A=S.D;

a) 0

b) 1

c) 2

d) 4

20. Which of the following statements contains an error?

(a) SELECT * FROM emp WHERE empid = 493945;

(b) SELECT empid FROM emp WHERE empid= 493945;

(c) SELECT empid FROM emp;

(d) SELECT empid WHERE empid = 56949 AND lastname = 'SMITH';

21. Which of the following statements will return the names of the products with Product ID 10, 11, or 42?

(a) SELECT ProductName FROM products WHERE ProductID IN (10,11,42)

(b) SELECT ProductName FROM products WHERE ProductID IN 10 OR 11 OR 42

(c) SELECT ProductName FROM products WHERE ProductID = (10,11,42)

(d) SELECT ProductName FROM products WHERE ProductID IS (10,11,42)

(e) None of the above

22. Which of the following commands will return the list of product names sorted in ascending alphabetic order?

(a) SELECT ProductName FROM products ORDER BY ProductName DESC

(b) SELECT ProductName FROM products ORDER BY ProductName ASC

(c) SELECT ProductName FROM products SORTED BY ProductName ASC

(d) SELECT ProductName FROM products SORTED BY ProductName DESC

(e) None of the above

23. Which of the following will return a list of every product ID currently listed in the order_details table where each product ID is listed only once?

(a) SELECT DISTINCT ProductID FROM order_details

(b) SELECT ProductID FROM order_details ONLY ONCE

(c) SELECT ProductID FROM order_details

(d) SELECT UNIQUE ProductID FROM order_details

(e) None of the above

24. In the instance of the relation R(A,O,T,V,U) shown below, which of the following functional dependencies hold ?

A O T V U

1 2 3 4 5

1 4 3 4 5

1 2 4 4 1

a) A O $\rightarrow$ T; V U $\rightarrow$ A

b) O $\rightarrow$ V; A V $\rightarrow$ U

c) T $\rightarrow$ O; A V $\rightarrow$ U

d) A O $\rightarrow$ T

e) O $\rightarrow$ V; V U $\rightarrow$ A

25. Which of the following statements contains an error?

(a) SELECT cid, sum (qty) from orders group by cid having sum(dollars) > 2000;

(b) SELECT aid, avg (qty) from orders group by aid;

(c) SELECT cid, sum (dollars) from orders;

(d) SELECT count (*) from orders;

26. Which code lists employees by descending order of salary

(a) SELECT * FROM EMPLOYEES SORT BY SALARY DESCENDING;

(b) SELECT * FROM EMPLOYEES IN ORDER OF SALARY;

(c) SELECT * FROM EMPLOYEES ORDER BY SALARY DESC;

(d) SELECT * FROM EMPLOYEES ORDER BY SALARY;

27. In order to perform a join, which criteria must be true?

(a) The two tables must have only one column exact same columns.

(b) The tables in the join need to have common rows.

(c) The two tables must both have primary keys

(d) The two tables must have a common column.

28. Consider the follow attributes and functional dependencies:

A B C

AB$\rightarrow$ C

C $\rightarrow$ A

List all keys (not superkeys):

29. What will result from the following SQL Select statement?

Select min(product_description)

from product_v;

(a) The minimum value of product_description will be displayed.

(b) An error message will be generated.

(c) The first product description alphabetically in product_v will be shown.

(d) none of the above

30. The following two SQL statements will produce the same results:

Select last_name, first_name

from customer

where credit_limit > 99 and credit_limit < 10001;

Select last_name, first_name

from customer

where credit_limit between 100 and 10000;

a.TRUE

b.FALSE

31. The following query will execute without errors:

select customer.customer_name, salesman.sales_quota

from customer

where customer.salesman_id =

(select salesman_id

from salesman

where lname = 'SMITH');

a.TRUE

b.FALSE

32. Table TT {J , D , C , V , N , G } and a set of functional dependencies

J,C,N $\rightarrow$ V,G

D $\rightarrow$ C,V,G

J $\rightarrow$ D,C,G

The closure of {D } is:

a) J D C V N G

b) J D C V G

c) J C V G

d) D C V N

e) D C V G

33. Table TT {O , M , Z , I , Q } and a set of functional dependencies

O $\rightarrow$ I,Q

I $\rightarrow$ M,Z

O,Z $\rightarrow$ M

Table TT is:

a) 1 NF

b) 2 NF

c) 3 NF

34. For which task in SQL would you use an IN clause

(a) To query the database for unknown values

(b) To query the database for a range of values

(c) To query the database for a character pattern

(d) To query the database for values in a specified list

35. A Cartesian product is

(a) A group function

(b) Produced as a result of a join select statement with no where clause

(c) The result of fuzzy logic

(d) A special feature of Oracle Server

36. A type of query that is placed within a WHERE or HAVING clause of another query is called a:

(a) master query.

(b) subquery.

(c) superquery.

(d) multi-query.

37. In order to perform a join, which criteria must be true?

(a) The two tables must have the exact same columns.

(b) The tables in the join need to have common rows.

(c) The two tables must both have primary keys

(d) The two tables must have at least one common column.

38. An attribute(s) which uniquely determines a tuple within a relation is called a:

(a) Candidate key

(b) Foreign key

(c) Primary key

(d) All of the above

(e) A and C

39. Table Q(A,B,C) and a set of functional dependencies

$AB \rightarrow C$;

$C \rightarrow A$;

Table Q is:

a) 1 NF

b) 2 NF

c) 3 NF

d) BCNF

40. Table Q(A,B,C,D) and a set of functional dependencies

$AB \rightarrow C$;

$D \rightarrow B$;

$C \rightarrow AD$

Table Q is:

a) 1 NF

e) 2 NF

f) 3 NF

g) BCNF

41. The following table and functional dependencies exhibits what type of dependency?

Table(A, B, C)

A → C

A → B

B → C

(a) Partial Dependence

(b) Transitive Dependence

(c) Full Dependence

(d) A and B

(e) None of the above

42. In the instance of the relation R(A,B,C,D,E) shown below, which of the following functional dependencies (FD's) hold? Briefly justify your answer.

A	B	C	D	E
1	2	3	4	5
1	4	3	4	5
1	2	4	4	1

I. AB → C II. B → D III. DE → B

(a) I only

(b) II only

(c) I and III only

(d) II and III only

43. Table Q{A , B , C , D} and a set of functional dependencies

A → B, C

D → C

The closure of {A D } is:

a) A D

b) A D B C

c) B C

d) None of the above

44. Consider the relation student(sno, sname, cname, cno) where (sno, cno) or (sname, cname) are candidate keys. There are functional dependencies within the keys.

The highest normal form whose requirements this relation satisfies is:

(a) 1NF

(b) 2NF

(c) 3NF

(d) BCNF

45. Let R be a relation with attributes (B,I,N,R,U,L) and let the following functional dependencies hold.

$B \rightarrow I$

$B \rightarrow N$

$N\ R \rightarrow U$

$N\ R \rightarrow L$

$I \rightarrow U$

Given the above functional dependencies, which of the following functional dependencies does not hold:

a) $N\ R \rightarrow U\ L$

b) $B\ R \rightarrow L$

c) $B \rightarrow U$

d) $I \rightarrow N\ R$

46. Suppose relation R(S,G,F,Y,N) has the following functional dependencies:

$S \rightarrow G$

$G \rightarrow F$

$G\ F \rightarrow S$

$S \rightarrow Y$

$N \rightarrow S$

$Y \rightarrow N$

Which of the following is not a key?

a) N b) G,F

c) S d) Y

47. Table TT {A , N , H , K , J , O , X } and a set of functional dependencies

$A, N \rightarrow O, X$

J → A , N, H

H, O → K, X

H, K, X → O

A, N, X → J

The closure of {A N } is:

e) A H K X J O

f) A N H K X J

g) A N H X J O

h) A N H K X J O

48. Consider the follow attributes and functional dependencies:

A B C D E F H

A→ D

AE → H

DF → BC

E → C

H → E

List all keys (not superkeys):

49. (Đề 48) Consider the decomposition into 3 relations: (AD) (EC) (ABEFH). Is this decomposition in

(a) BCNF

(b) 3NF

(c) 1NF

(d) None of the above

50. Consider a relation R(A,B,C) with the following functional dependencies:

A → B;B → C and B → A

The number of superkeys of R is:

(a) 2

(b) 3

(c) 5

(d) 6

TRƯỜNG CĐ KỸ THUẬTCAO THẮNG **ĐỀ THI MÔN CƠ SỞ DỮ LIỆU**
KHOA ĐT-TH thời gian làm bài 90 phút

CÂU I:(5đ)

Sau đây là một số lược đồ quan hệ được trích từ bài toán quản lý tuyển sinh:

ĐIEMTHI(ĐIEMTHISO,ĐIACHIDIEMTHI)

Một hội đồng coi thi tuyển sinh có nhiều điểm thi, mỗi điểm thi được đặt tại một trường nào đó. Các điểm thi (ĐIEMTHISO) được đánh số là điểm thi số 1, điểm thi số 2, điểm thi số 3,…Mỗi điểm thi xác định địa chỉ (ĐIACHIDIEMTHI).

THISINH(SOBD,HOTEN,NGAYSINH, PHAI, ĐIACHI, MANGANH, PHONGTHI)

Mỗi thí sinh có một số báo danh (SOBD) duy nhất, mỗi số báo danh xác định các thông tin: họ và tên (HOTEN), ngày sinh (NGAYSINH), phái (PHAI), địa chỉ thường trú (ĐIACHI), mã ngành đăng ký thi(MANGANH), số hiệu phòng thi(PHONGTHI).

NGANH(MANGANH,TENNGANH)

Mỗi ngành có một mã ngành (MANGANH) duy nhất, mỗi mã ngành xác định tên ngành (TENNGANH), chẳng hạn ngành Công Nghệ Thông Tin có mã ngành là 01, ngành Công Nghệ Hoá Thực Phẩm có mã ngành là 10,…

PHONG(PHONGTHI,ĐIEMTHISO)

Mỗi điểm thi có nhiều phòng thi (PHONGTHI) được đánh số hiệu khác nhau ở tất cả các điểm thi (trong một phòng thi có thể có các thí sinh của nhiều ngành khác nhau)

1.Xác định khoá cho mỗi lược đồ quan hệ trên (1 điểm)

2.Hãy xác định các ràng buộc toàn vẹn có trong lược đồ cơ sở dữ liệu trên (mỗi loại cho một ví dụ) (1 điểm)

3.Thực hiện các yêu cầu sau bằng SQL (3 điểm)

a.Lập danh sách các thí sinh đăng ký dự thi có số hiệu phòng là “0061”, danh sách cần: SOBD,HOTEN,TENNGANH và được sắp tăng dần theo cột SOBD.

b.Danh sách các thí sinh đã đăng ký thi vào ngành có mã ngành là ”01”, danh sách cần: SOBD, HOTEN, NGAYSINH, PHONGTHI, ĐIACHIDIEMTHI và được sắp tăng dần theo cột SOBD.

c.Hãy thống kê xem mỗi ngành có bao nhiêu thí sinh đã đăng ký thi, danh sách cần: MANGANH,TENNGANH, SOLUONG, trong đó số lượng(SOLUONG) là thuộc tính tự đặt..

CÂU II: (2đ)

1.Cho lược đồ quan hệ Q(ABCD), r và s là hai quan hệ được cho như sau:

r

A	B	C	D
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1

s

A	B	C	D
2	1	1	1
2	2	1	1
1	1	1	0
x	y	z	v

Tìm r - s

r * s

2.Cho hai lược đồ quan hệ Q1(ABC) và Q2(DEF), r và s là hai quan hệ được cho như sau:

r

A	B	C
1	2	3
a	b	c
x	y	z

s

D	E	F
1	e	f
a	e	f
5	6	7

Tìm r |><|$_{A=D}$ s

3.Cho hai lược đồ quan hệ Q1(ABC) và Q2(DE), r và s là hai quan hệ được cho như sau

r

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

s

D	E
3	1
6	2

Tìm r |><|$_{B>D}$ s

4.Cho hai lược đồ quan hệ Q_1(ABCD) và Q_2(CD), r và s là hai quan hệ được cho như sau:

r

A	**B**	**C**	**D**
a	b	c	d

s

C	**D**
c	d

a b e f e f

b c e f

c d e f

a b d e

Tính r ÷ s

CÂU III (1đ)

1)Cho lược đồ quan hệ Q(ABCDE), r là một quan hệ được cho như sau:

A	B	C	D	E
1	2	3	4	5
1	4	3	4	5
1	2	4	4	1

Những phụ thuộc hàm nào sau đây thoả r ?

$C \rightarrow B$; $AD \rightarrow E$; $B \rightarrow D$; $AB \rightarrow C$. $AC \rightarrow D$

2.Cho lược đồ quan hệ Q(ABCD) và tập phụ thuộc hàm F = {$A \rightarrow B$; $BC \rightarrow D$}.

Những phụ thuộc hàm nào sau đây thuộc F^+ ?

$C \rightarrow D$; $A \rightarrow D$; $AD \rightarrow C$; $AC \rightarrow D$; $BC \rightarrow A$; $B \rightarrow CD$.

CÂU IV (2đ)

1.Cho lược đồ quan hệ Q và tập phụ thuộc hàm F, $K \subseteq Q^+$. Hãy nêu điều kiện để K là khoá của Q.

2.Cho lược đồ quan hệ Q(ABCD) và tập phụ thuộc hàm F={$A \rightarrow D$; $D \rightarrow A$; $AB \rightarrow C$}

a.Tính AC^+

b.Tìm tất cả các khoá của Q.

c. Q đạt dạng chuẩn nào ? giải thích.

Hết

(Sinh viên không được sử dụng tài liệu

cán bộ coi thi không giải thích gì thêm

TÀI LIỆU THAM KHẢO

[1]JEFFREY D. ULLMAN

Principles of database and knowledge base systems

[2].Nguyễn Bá Tường

cơ sở dữ liệu- Lý thuyết và thực hành

[3].NGUYỄN ĐĂNG TỴ- ĐỖ PHÚC

Cơ sở dữ liệu

MỤC LỤC

LỜI NÓI ĐẦU 1

chương 1 **TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU**

1.1. MỘT SỐ KHÁI NIỆM CƠ BẢN 2

1.1.1. Định nghĩa cơ sở dữ liệu 2

1.1.2. Ưu điểm của cơ sở dữ liệu 2

1.1.3. Các đối tượng sử dụng CSDL: 2

1.1.3. Những vấn đề mà CSDL cần phải giải quyết 2

1.1.5. Hệ quản trị cơ sở dữ liệu 3

1.1.6. Các ứng dụng của cơ sở dữ liệu 4

1.2. CÁC MÔ HÌNH CƠ SỞ DỮ LIỆU 5

1.3. MÔ HÌNH THỰC THỂ KẾT HỢP 5

1.3.1. Thực thể 6

1.3.2. Thuộc tính 6

1.3.3. Loại thực thể 6

1.3.4. Khóa 6

1.3.5 Mối kết hợp 8

BÀI TẬP 11

chương 2 **MÔ HÌNH CƠ SỞ DỮ LIỆU QUAN HỆ**

2.1. CÁC KHÁI NIỆM CƠ BẢN 16

2.1.1.Thuộc Tính(attribte): 16

2.1.2 Lược đồ quan hệ (Relation schema) 17

2.1.3.Quan hệ (Relation) 18

2.1.4 Bộ (Tuple) 18

2.1.5.Siêu khoá - Khoá chính 18

2.2. CHUYỂN TỪ MÔ HÌNH THỰC THỂ KẾT HỢP SANG MÔ HÌNH DỮ LIỆU QUAN HỆ 20

2.3. CÁC PHÉP TOÁN ĐẠI SỐ TRÊN CÁC QUAN HỆ)

2.3.1 Phép hợp (Union) 21

2.3.2 Phép giao (Intersection) 22

2.3.3.Phép trừ (Minus) 22
2.3.4.Tích Descartes (Cartesian Product) 23
2.3.5.Phép chia hai quan hệ 23
2.3.6.Phép chiếu(Projection) 24
2.3.7.Phép chọn (Selection) 25
2.3.8.Phép θ - kết 25
2.3.9.Phép kết tự nhiên 26
BÀI TẬP 28
chương 3 **NGÔN NGỮ TRUY VẤN DỮ LIỆU**
3.1.MỞ ĐẦU 29
3.2.TÌM THÔNG TIN TỪ CÁC CỘT CỦA BẢNG - MỆNH ĐỀ SELECT 32
3.3.CHỌN CÁC DÒNG CỦA BẢNG – MỆNH ĐỀ WHERE 34
3.4.THỨ TỰ THỂ HIỆN CÁC BẢN GHI - MỆNH ĐỀ ORDER BY 36
3.5. CÂU LỆNH SQL LỒNG NHAU 37
3.6.GOM NHÓM DỮ LIỆU– MỆNH ĐỀ GROUP BY 38
BÀI TẬP 41
chương 4 **RÀNG BUỘC TOÀN VẸN**
4.1 RÀNG BUỘC TOÀN VẸN 45
4.1.1 Khái niệm ràng buộc toàn vẹn 45
4.1.2 Các yếu tố của ràng buộc toàn vẹn 46
4.1.2.1.Điều kiện 46
4.1.2.2.Bối cảnh 46
4.1.2.3.Bảng tầm ảnh hưởng 47
4.1.2.4.Hành động 47
4.2. PHÂN LOạI RÀNG BUỘC TOÀN VẸN 48
4.2.1.Ràng buộc toàn vẹn có bối cảnh là một quan hệ 50
4.2.1.1.Ràng buộc toàn vẹn liên bộ 50
4.2.1.2.ràng buộc toàn vẹn về miền giá trị 51
4.2.1.3.Ràng Buộc Toàn Vẹn Liên Thuộc Tính 51
4.2.2.Ràng buộc toàn vẹn có bối cảnh là nhiều quan hệ 51

4.2.2.1.Ràng Buộc Toàn Vẹn Về Khoá Ngoại: 51
4.2.2.2.Ràng Buộc Toàn Vẹn Liên Thuộc Tính Liên Quan Hệ 52
4.2.2.3.Ràng Buộc Toàn Vẹn Liên Bộ Liên Quan Hệ 52
BÀI TẬP 53
chương 5 **LÝ THUYẾT THIẾT KẾ CƠ SỞ DỮ LIỆU**
5.1. CÁC VấN Đề GặP PHải KHI Tổ CHứC Dữ LiệU: 55
5.2. PHụ THUỘC HÀM 56
5.2.1 Định nghĩa phụ thuộc hàm 56
5.2.2 Cách xác định phụ thuộc hàm cho lược đồ quan hệ 57
5.2.3 Một số tính chất của phụ thuộc hàm -*hệ luật dẫn armstrong:* *58*
5.3 BAO ĐÓNG CỦA TẬP PHỤ THUỘC HÀM F VÀ BAO ĐÓNG CỦA TẬP THUỘC TÍNH X 59
5.3.1.Bao đóng của tập phụ thuộc hàm F 59
5.3.2.Bao đóng của tập thuộc tính X 60
5.3.3.Bài toán thành viên 60
5.3.4.Thuật toán tìm bao đóng của một tập thuộc tính (X) 60
5.4. KHOÁ CỦA LƯỢC ĐỒ QUAN HỆ - MỘT SỐ THUẬT TOÁN TÌM KHOÁ 62
5.4.1.Định nghĩa 62
5.4.2.Thuật toán tìm một khoá của một lược đồ quan hệ Q 63
5.4.3.Thuật toán tìm tất cả các khoá của một lược đồ quan hệ 63
5.5. PHỦ TỐI THIỂU 66
5.5.2.Tập phụ thuộc hàm tương đương 66
5.5.1.Phủ tối thiểu 67
5.5.3.Thuật toán tìm phủ tối thiểu của một tập phụ thuộc hàm 68
5.6. DẠNG CHUẨN CỦA LƯỢC ĐỒ QUAN HỆ 69
5.6.1.Một số khái niệm liên quan đến các dạng chuẩn 70
5.6.2.Dạng chuẩn 1 71
5.6.3.Dạng chuẩn 2 71
5.6.4.Dạng chuẩn 3 72
5.6.5.Dạng chuẩn BC 74

BÀI TẬP 75
BÀI TẬP THỰC HÀNH 80
CÁC BỘ ĐỀ THI KIỂM TRA MÔN CSDL 87
TÀI LIỆU THAM KHẢO 110
MỤC LỤC 111

GIÁO TRÌNH
LÝ THUYẾT CƠ SỞ DỮ LIỆU

GIÁO VIÊN: NGÔ THÀNH LONG **BỘ MÔN:** HỆ THỐNG THÔNG TIN **KHOA:** CÔNG NGHỆ THÔNG TIN

CHƯƠNG I

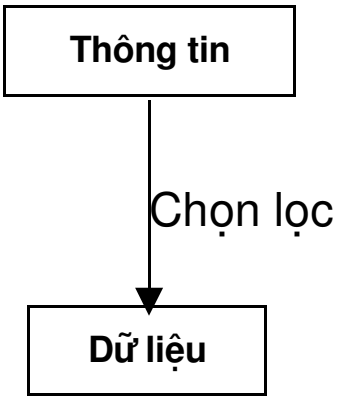
TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

Nội dung chi tiết

- Giới thiệu
- Quá trình phát triển
- Một số đặc tính của CSDL
- Người sử dụng CSDL
- Kiến trúc của HQT CSDL
- Các tính năng của HQT CSDL
- Các khái niệm
- Ngôn ngữ CSDL

I. Giới thiệu

- Ví dụ
  - Kinh doanh
  - Ngân hàng và tài chính
  - Giáo dục
  - Hành chính
  - Giải trí
  - …

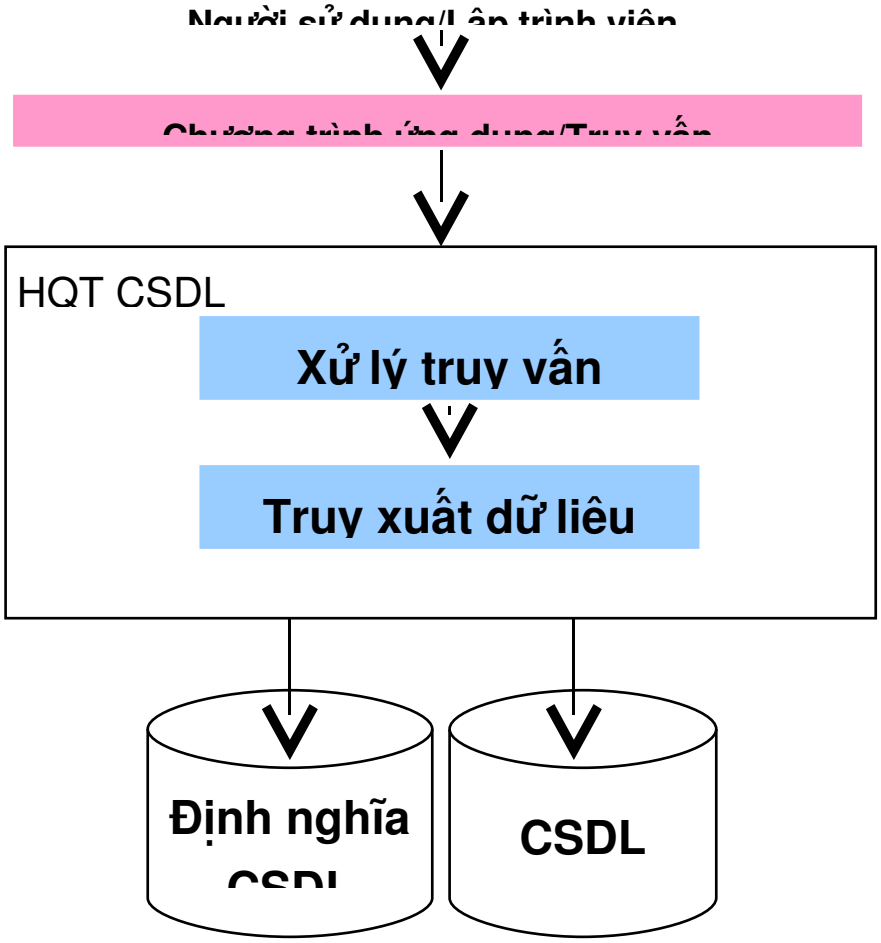


- Dữ liệu (Data)
  - Một mô tả hình thức về thông tin và hoạt động
  + Tên, địa chỉ, số điện thoại của khách hàng
  + Báo cáo doanh thu
  + Đăng ký học phần

- Cơ sở dữ liệu (Database)
  - Một tập hợp có cấu trúc của những dữ liệu có liên quan với nhau được lưu trữ trong máy tính
    - Danh sách sinh viên
    - Niên giám điện thoại
    - Danh mục các đề án
  - Một CSDL biểu diễn một phần của thế giới thực (thế giới thu nhỏ)
  - CSDL được thiết kế, xây dựng, và lưu trữ với một mục đích xác định, phục vụ cho một số ứng dụng và người dùng
  - Tập ngẫu nhiên của các dữ liệu không thể xem là một CSDL
- Hệ quản trị CSDL (Database Management System)
  - Tập hợp các chương trình cho phép người sử dụng tạo ra và duy trì CSDL
  - Một phần mềm hệ thống cho phép định nghĩa, xây dựng và xử lý dữ liệu

- Định nghĩa – khai báo bộ khung dữ liệu cùng với các mô tả chi tiết về dữ liệu
- Xây dựng – lưu trữ dữ liệu lên bộ nhớ phụ
- Xử lý – truy vấn, cập nhật và phát sinh báo cáo

- Hệ CSDL (Database System)



Một ví dụ về CSDL

NHANVIEN	HONV	TENLOT	TENNV	MANV	NGSINH	MA_NQL	PHG
	Tran	Hong	Quang	987987987	03/09/1969	987654321	4
	Nguyen	Thanh	Tung	333445555	12/08/1955	888665555	5
	Nguyen	Manh	Hung	666884444	09/15/1962	333445555	5
	Tran	Thanh	Tam	453453453	07/31/1972	333445555	5

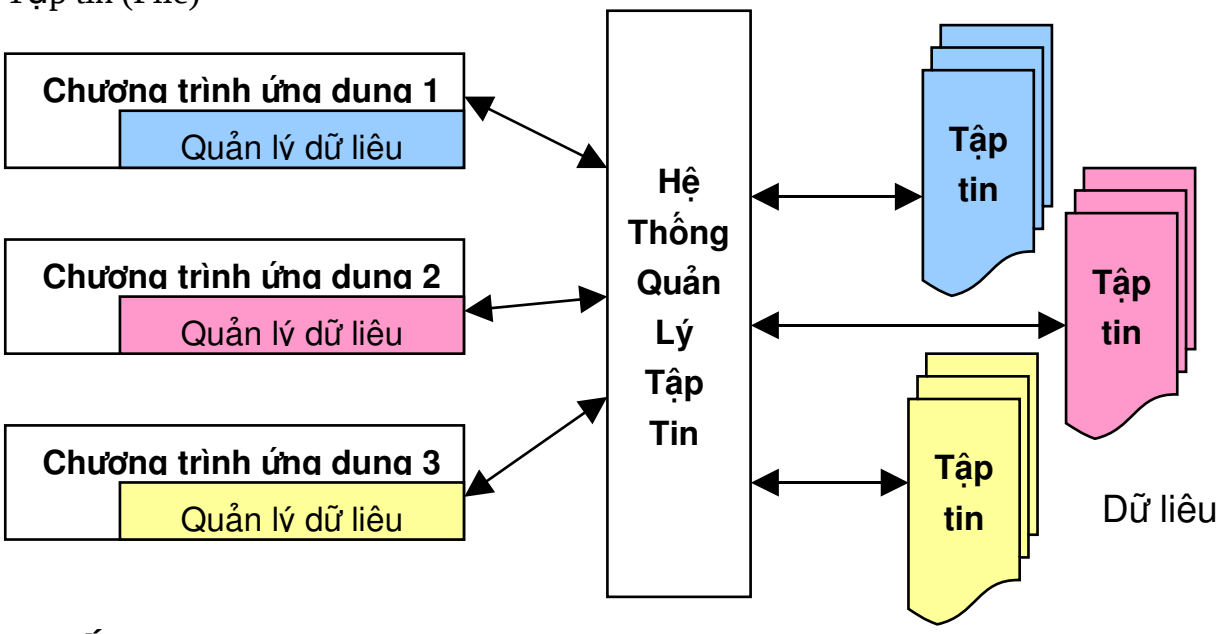
DEAN	TENDA	MADA	DDIEM_DA	PHONG
	San pham X	1	VUNG TAU	5
	San pham Y	2	NHA TRANG	5
	San pham Z	3	TP HCM	5
	Tin hoc hoa	10	HA NOI	4

PHANCONG	MA_NVIEN	SODA	THOIGIAN
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0

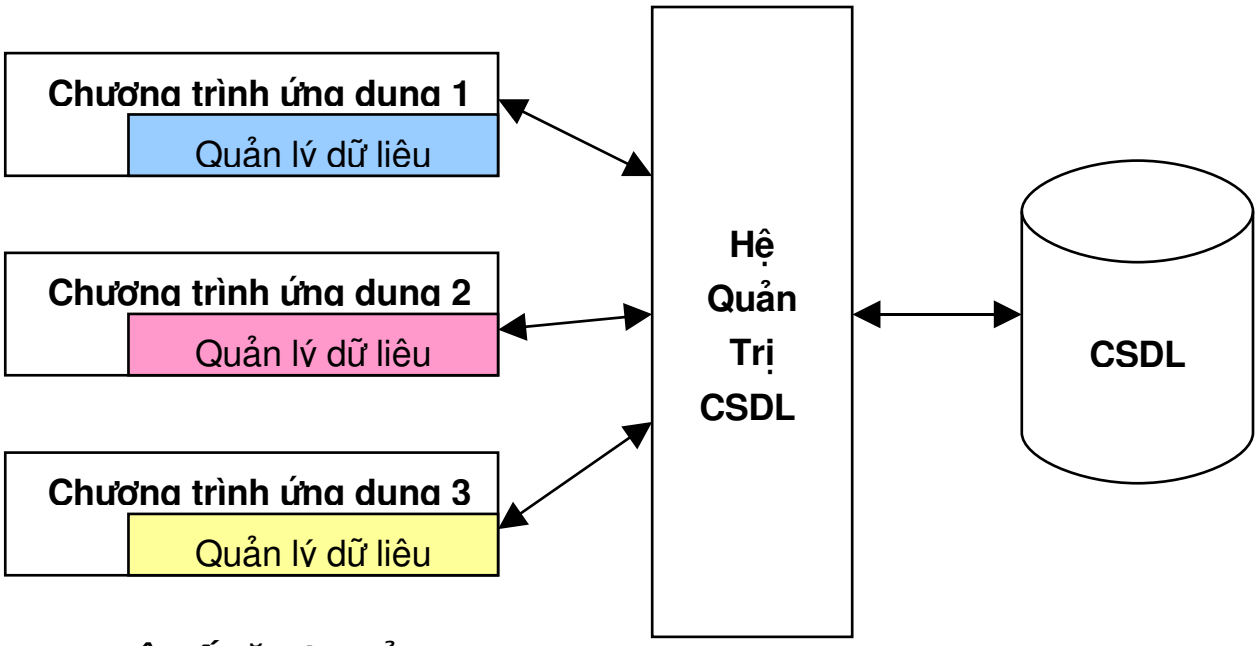
- Quản lý đề án của một công ty
  - Định nghĩa CSDL
    - Cấu trúc bảng, bao gồm các thành phần dữ liệu và kiểu dữ liệu tương ứng
  - Xây dựng CSDL
    - Đưa dữ liệu vào các bảng
  - Xử lý CSDL
    - Thực hiện các truy vấn: "Cho biết những nhân viên thuộc phòng 5"
    - Thực hiện các phép cập nhật: "Chuyển nhân viên Nguyễn Thanh Tùng sang phòng số 1"

II. Quá trình phát triển

Tập tin (File)



- ***Hạn chế***
  - Dữ liệu bị trùng lắp và dư thừa
  - Thiếu tính nhất quán giữa các dữ liệu
  - Khó khăn trong việc truy xuất
  - Việc chia sẻ dữ liệu bị hạn chế
  - Khó khôi phục
- Cơ sở dữ liệu (Database)



III. Một số đặc tính của CSDL

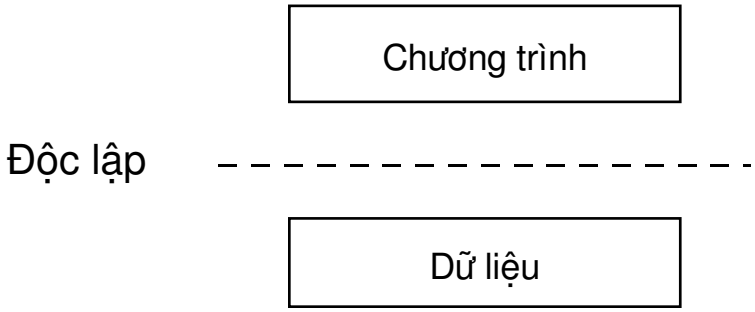
1. Tính tự mô tả

- Hệ CSDL không chỉ chứa bản thân CSDL mà còn chứa định nghĩa đầy đủ (mô tả) của CSDL
- Các định nghĩa được lưu trữ trong catalog
  - Chứa các thông tin về cấu trúc tập tin, kiểu và dạng thức lưu trữ của mỗi thành phần dữ liệu và những ràng buộc dữ liệu
- Dữ liệu trong catalog gọi là meta-data (data of data)

- Các CTƯD có thể truy xuất đến nhiều CSDL nhờ thông tin cấu trúc được lưu trữ trong catalog

2. Tính độc lập

- Vì định nghĩa về cấu trúc CSDL được lưu trữ trong catalog nên khi có thay đổi nhỏ về cấu trúc ta ít phải sửa lại chương trình



3. Tính trừu tượng

- Hệ CSDL cho phép trình bày dữ liệu ở một mức trừu tượng cho phép, nhằm che bớt những chi tiết lưu trữ thật của dữ liệu
- Trừu tượng hóa dữ liệu
  - Mô hình dữ liệu

- Đối tượng
- Thuộc tính của đối tượng
- Mối liên hệ

4. Tính nhất quán

- Lưu trữ dữ liệu thống nhất
  - Tránh được tình trạng trùng lặp thông tin
- Có cơ chế điều khiển truy xuất dữ liệu hợp lý
  - Tránh được việc tranh chấp dữ liệu
  - Bảo đảm dữ liệu luôn đúng tại mọi thời điểm

5. Các cách nhìn dữ liệu

- Hệ CSDL cho phép nhiều người dùng thao tác lên cùng một CSDL
- Mỗi người đòi hỏi một cách nhìn (view) khác nhau về CSDL
- Một view là
  - Một phần của CSDL hoặc
  - Dữ liệu tổng hợp từ CSDL

IV. Người sử dụng CSDL

1. Quản trị viên

- Có trách nhiệm quản lý hệ CSDL
  - Cấp quyền truy cập CSDL
  - Điều phối và giám sát việc sử dụng CSDL

2. Thiết kế viên

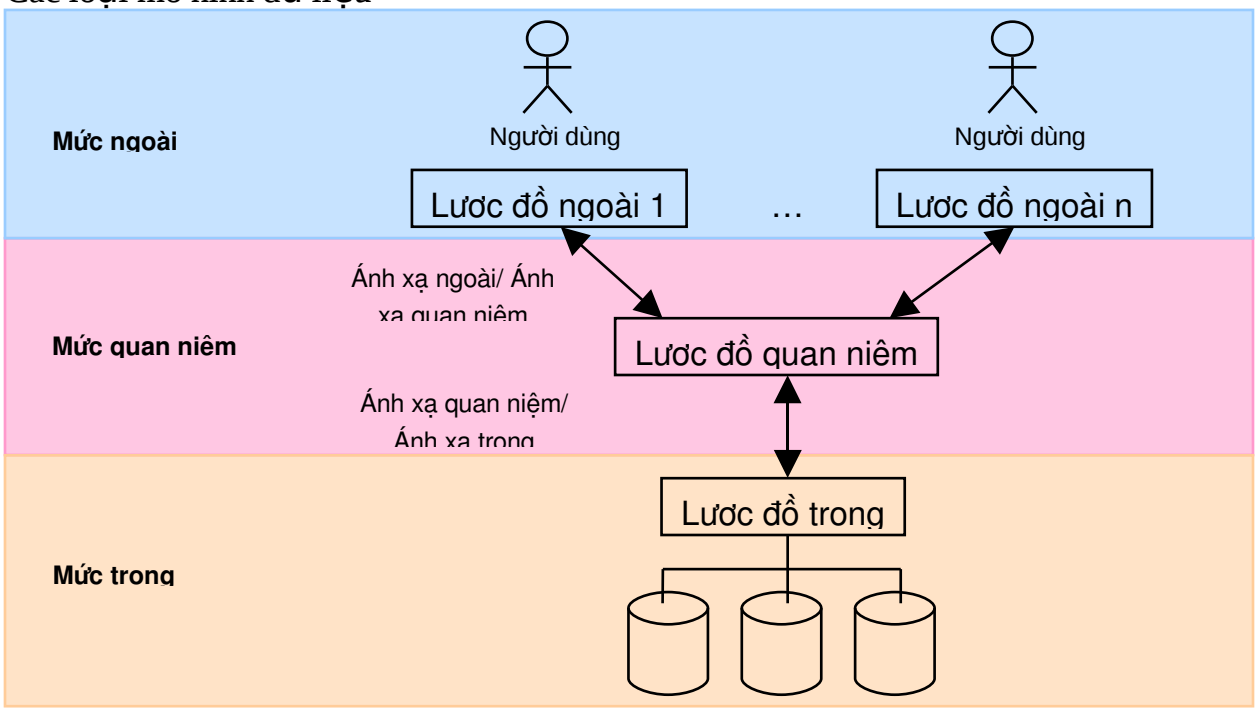
- Chịu trách nhiệm về
  - Lựa chọn cấu trúc phù hợp để lưu trữ dữ liệu
  - Quyết định những dữ liệu nào cần được lưu trữ
- Liên hệ với người dùng để nắm bắt được những yêu cầu và đưa ra một thiết kế CSDL thỏa yêu cầu này
- Có thể là 1 nhóm các DBA quản lý các CSDL sau khi việc thiết kế hoàn tất

3. Người dùng cuối

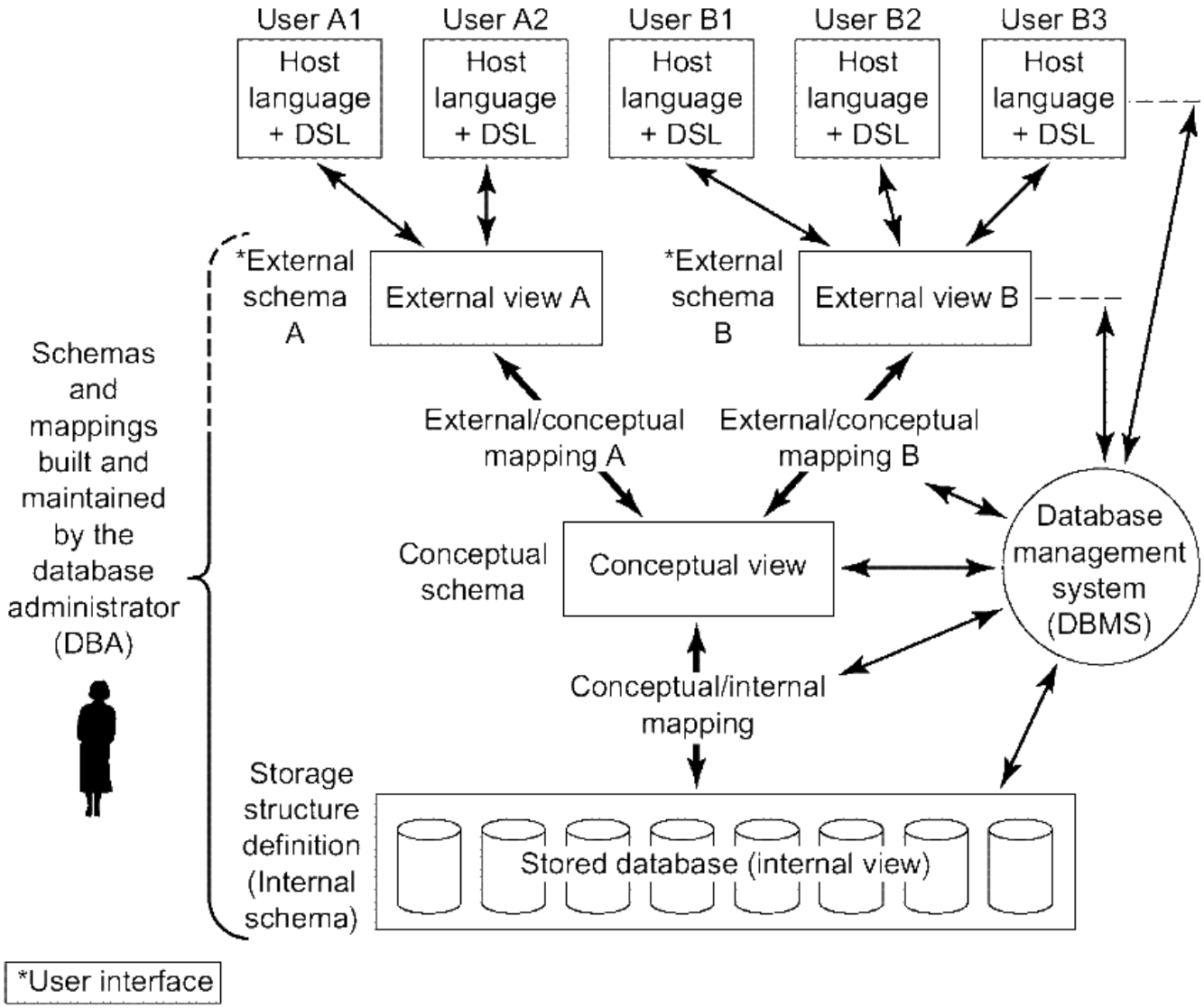
- Người ít sử dụng
  - Ít khi truy cập CSDL, nhưng cần những thông tin khác nhau trong mỗi lần truy cập và dùng những câu truy vấn phức tạp
  - Người quản lý
- Người sử dụng thường xuyên
  - Thường xuyên truy vấn và cập nhật CSDL nhờ vào một số các chức năng đã được xây dựng sẵn
  - Nhân viên
- Người sử dụng đặc biệt
  - Thông thạo về HQT CSDL, tự xây dựng những truy vấn phức tạp cho công việc
  - Kỹ sư, nhà khoa học, người phân tích,…

V. Kiến trúc của HQT CSDL

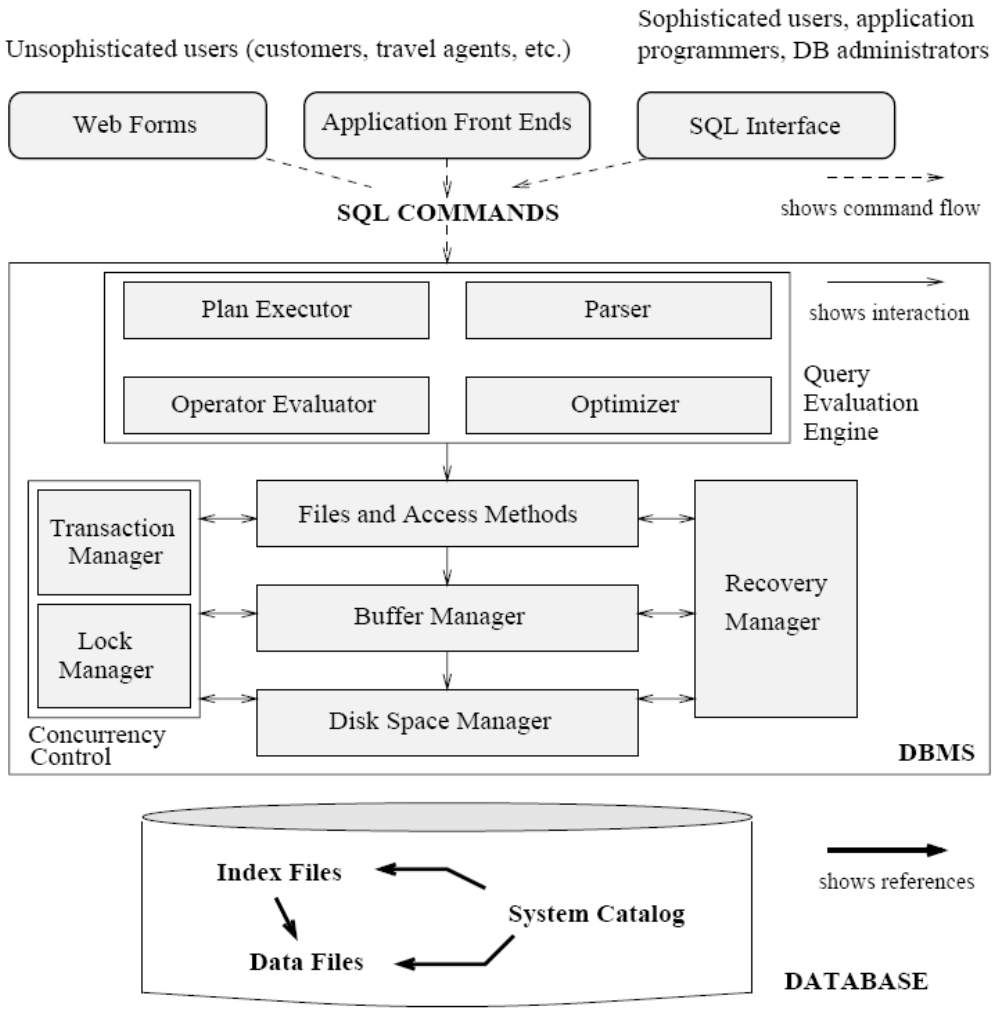
Các loại mô hình dữ liệu



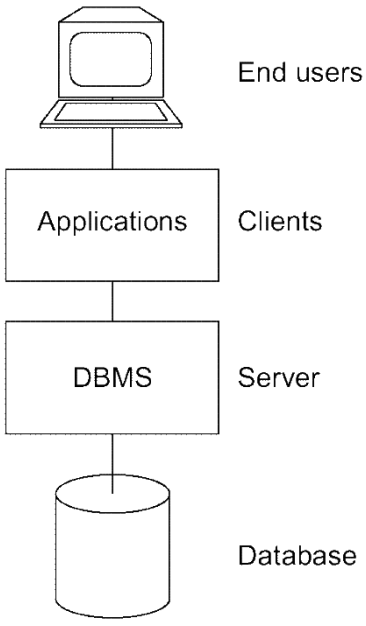
- Mức trong (lược đồ trong)
  - Mô tả cấu trúc lưu trữ vật lý CSDL
- Mức quan niệm (lược đồ quan niệm)
  - Mô tả cấu trúc của toàn thể CSDL cho 1 cộng đồng người sử dụng, gồm thực thể, kiểu dữ liệu, mối liên hệ và ràng buộc
  - Che bớt các chi tiết của cấu trúc lưu trữ vật lý
- Mức ngoài (lược đồ ngoài)
  - Còn gọi là mức khung nhìn (view)
  - Mô tả một phần của CSDL mà 1 nhóm người dùng quan tâm đến và che dấu phần còn lại của CSDL đối với nhóm người dùng đó



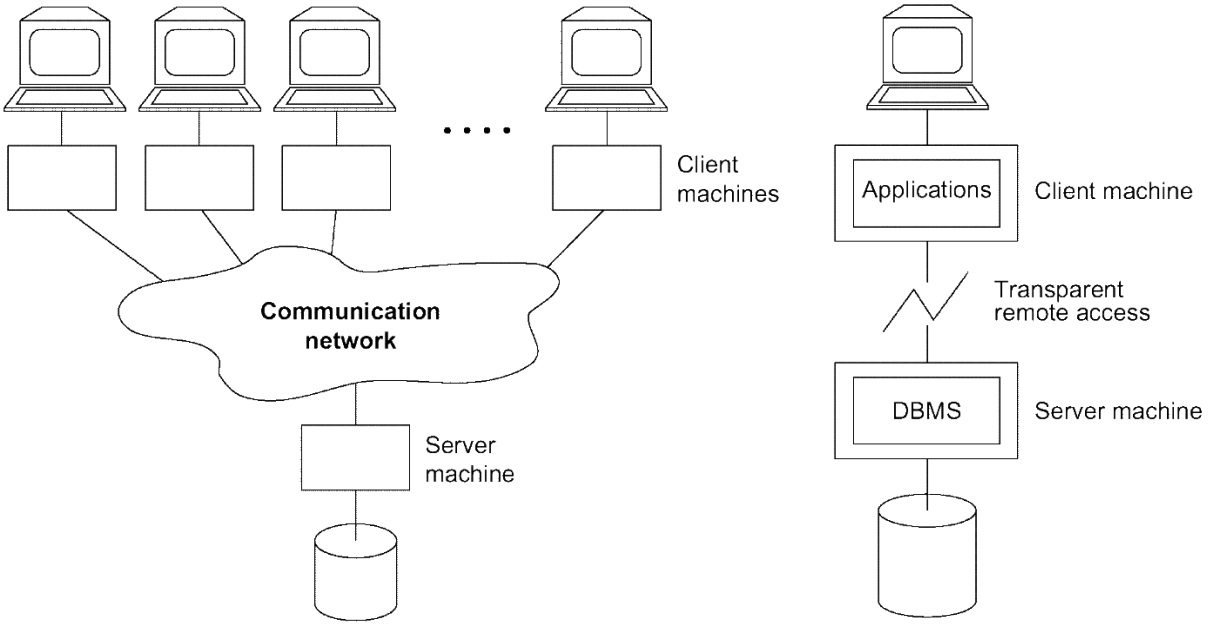
- Độc lập dữ liệu
  - Độc lập logic
    - Khả năng thay đổi lược đồ quan niệm mà không thay đổi lược đồ ngoài hoặc các chương trình ứng dụng
  - Độc lập vật lý
    - Khả năng thay đổi lược đồ trong mà không làm thay đổi lược đồ quan niệm cũng như lược đồ ngoài



1. Kiến trúc Client/Server



2. Kiến trúc phân tán



VI. Các tính năng của HQT CSDL

- Kiểm soát được tính dư thừa của dữ liệu: Tích hợp các nhu cầu dữ liệu của người dùng để xây dựng một CSDL thống nhất
- Chia sẻ dữ liệu: Trong môi trường đa người dùng, các HQT phải cho phép truy xuất dữ liệu đồng thời
- Hạn chế những truy cập không cho phép: Từng người dùng và nhóm người dùng có một tài khoản và mật mã để truy xuất dữ liệu
- Cung cấp nhiều giao diện: HQT cung cấp ngôn ngữ giữa CSDL và người dùng

*Đảm bảo các ràng buộc toàn vẹn:

- RBTV (Integrity Constraints) là những qui định cần được thỏa mãn để đảm bảo dữ liệu luôn phản ánh đúng ngữ nghĩa của thế giới thực.
- Một số ràng buộc có thể được khai báo với HQT và HQT sẽ tự động kiểm tra.
- Một số ràng buộc khác được kiểm tra nhờ chương trình ứng dụng

* Khả năng sao lưu dự phòng khi gặp sự cố

- Có khả năng khôi phục dữ liệu khi có sự hư hỏng về phần cứng hoặc phần mềm.

- Các tính năng khác
  - Chuẩn hóa
    - Cho phép DBA định nghĩa và bắt buộc áp dụng một chuẩn thống nhất cho mọi người dùng
  - Uyển chuyển
    - Khi nhu cầu công việc thay đổi, cấu trúc CSDL rất có thể thay đổi, HQT cho phép thêm hoặc mở rộng cấu trúc mà không làm ảnh hưởng đến chương trình ứng dụng

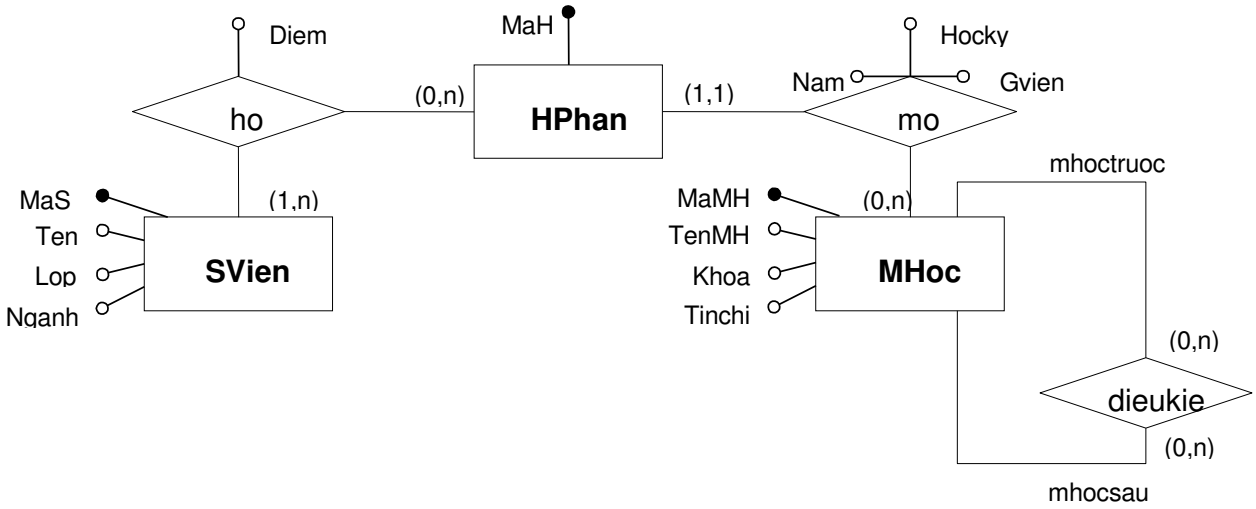
- Giảm thời gian phát triển ứng dụng
- Tính khả dụng
  - Khi có một sự thay đổi lên CSDL, tất cả người dùng đều thấy được

VII. Các KN

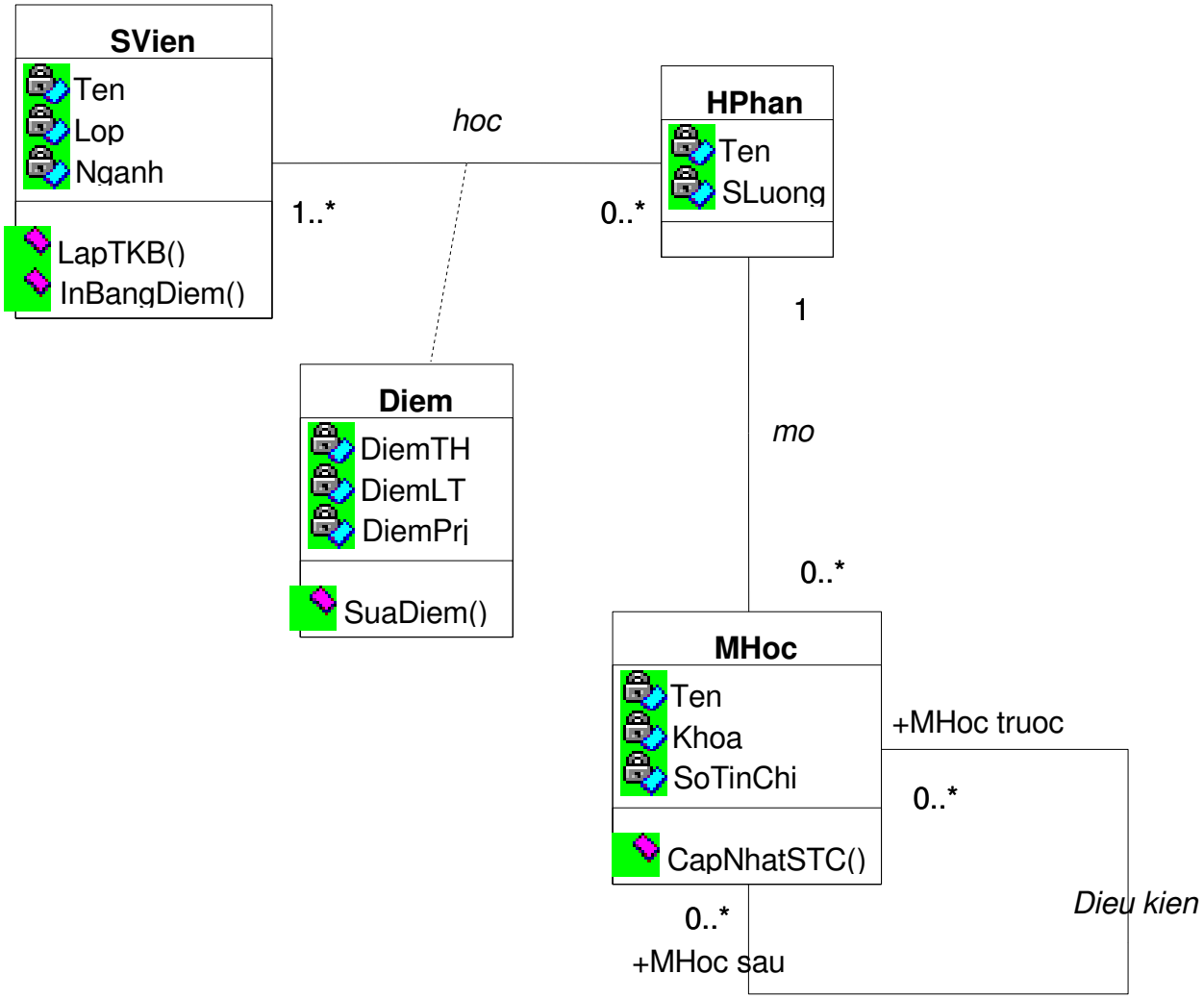
1. Mô hình dữ liệu

- Mô hình dữ liệu (Data Model) bao gồm
  - Các khái niệm biểu diễn dữ liệu
  - Các phép toán xử lý dữ liệu
- Mô hình mức cao
  - Cung cấp các khái niệm gần gũi với người dùng
  - Mô hình phải tự nhiên và giàu ngữ nghĩa
  - VD: mô hình thực thể kết hợp (ER), mô hình đối tượng…
- Mô hình cài đặt
  - Đưa ra các khái niệm người dùng có thể hiểu được nhưng không quá xa với cách dữ liệu được tổ chức thật sự trên máy tính
  - VD: mô hình quan hệ, mô hình mạng, mô hình phân cấp
- Mô hình mức thấp (mô hình vật lý)
  - Đưa ra các khái niệm mô tả chi tiết về cách thức dữ liệu được lưu trữ trong máy tính

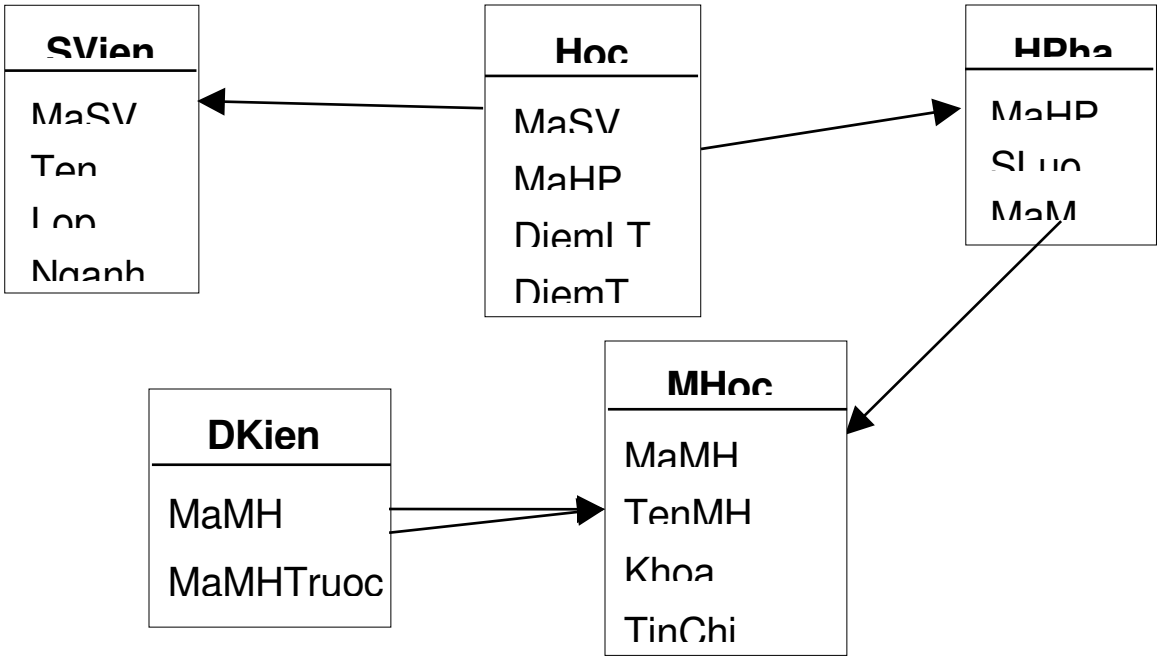
* Ví dụ mô hình ER



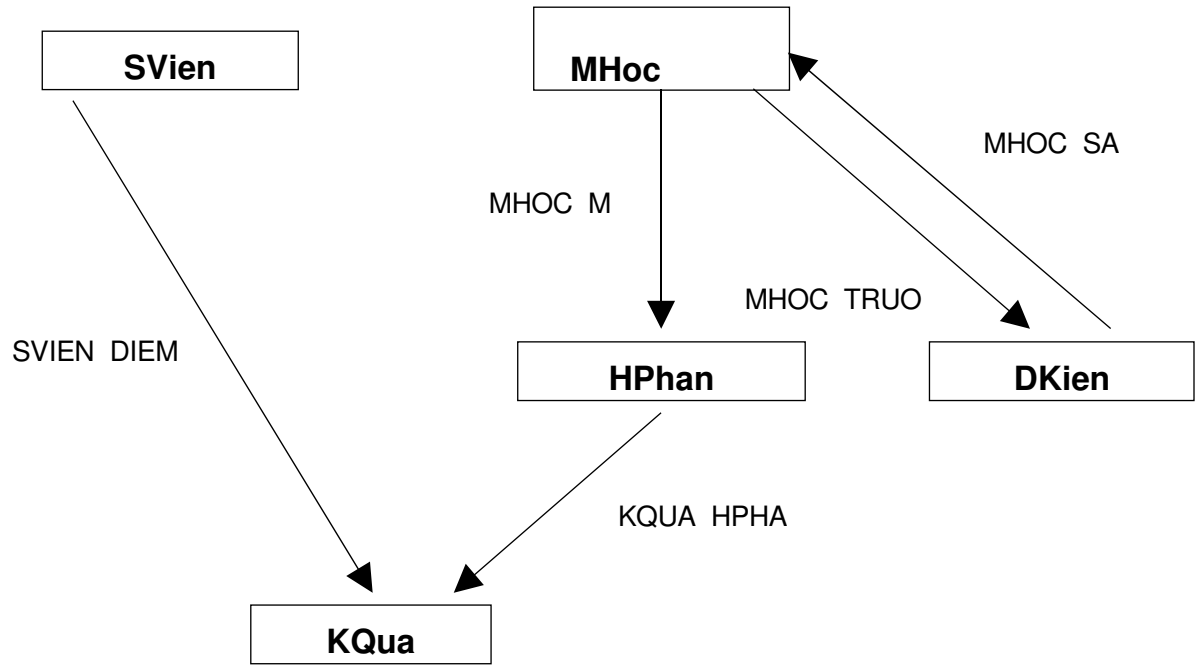
* Ví dụ mô hình đối tượng



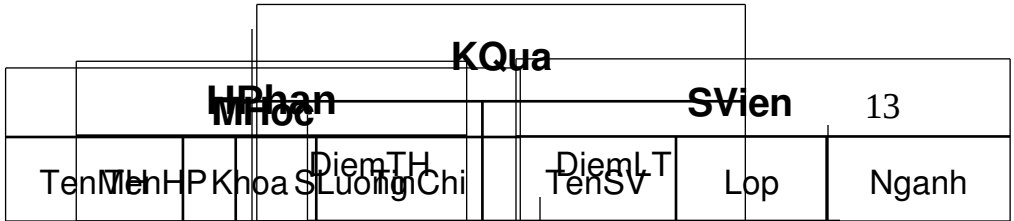
* Ví dụ mô hình quan hệ



* Ví dụ mô hình mạng

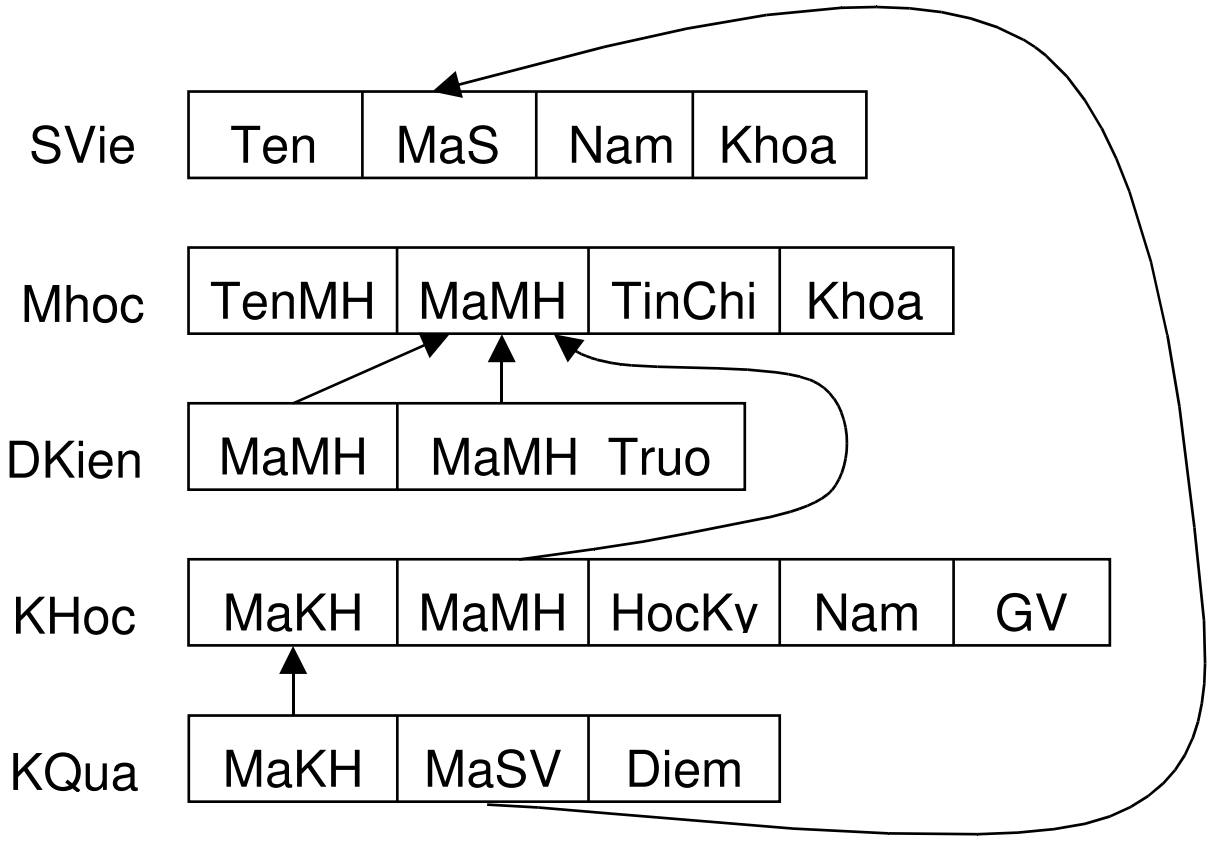


* Ví dụ mô hình phân cấp



Mức 3:

* Lược đồ
  - Lược đồ CSDL (Database Schema)
    - Là các mô tả về cấu trúc và ràng buộc trên CSDL



*Thể hiện

- Thể hiện CSDL (Database Instance)
  - Là dữ liệu hiện thời được lưu trữ trong CSDL ở một thời điểm nào đó
  - Tình trạng của CSDL

Mhoc	TenMH	MaMH	TinChi	Khoa
	Nhap mon tin hoc	COSC1310	4	CNTT
	Cau truc du lieu	COSC3320	4	CNTT
	Toan roi rac	MATH2410	3	TOAN
	Co so du lieu	COSC3380	3	CNTT

KQua	MaSV	MaKH	Diem
	17	112	8
	17	119	6
	8	85	10
	8	92	9
	8	102	8
	8	135	10

SVien	Ten	MaSV	Nam	Khoa
	Son	17	1	CNTT
	Bao	8	2	CNTT

DKien	MaMH	MaMH Truoc
	COSC3380	COSC3320
	COSC3380	MATH2410
	COSC3320	COSC3380

VIII. Ngôn ngữ CSDL

- Ngôn ngữ định nghĩa dữ liệu
  (DDL – Data Definition Language)
    - Xác định ra lược đồ quan niệm

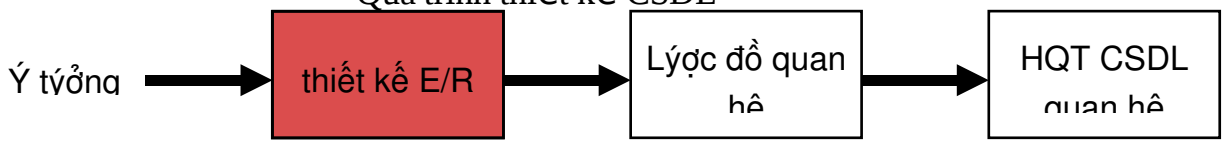
- Ví dụ
  CREATE TABLE employees (
  id INTEGER PRIMARY KEY,
  first_name CHAR(50) null,
  last_name CHAR(75) not null,
  date_of_birth DATE null);
- Ngôn ngữ thao tác dữ liệu
  (DML – Data Manipulation Language)
    - Cho phép truy xuất, thêm, xóa, sửa dữ liệu
    - Mức cao (phi thủ tục)
    - Mức thấp (thủ tục)

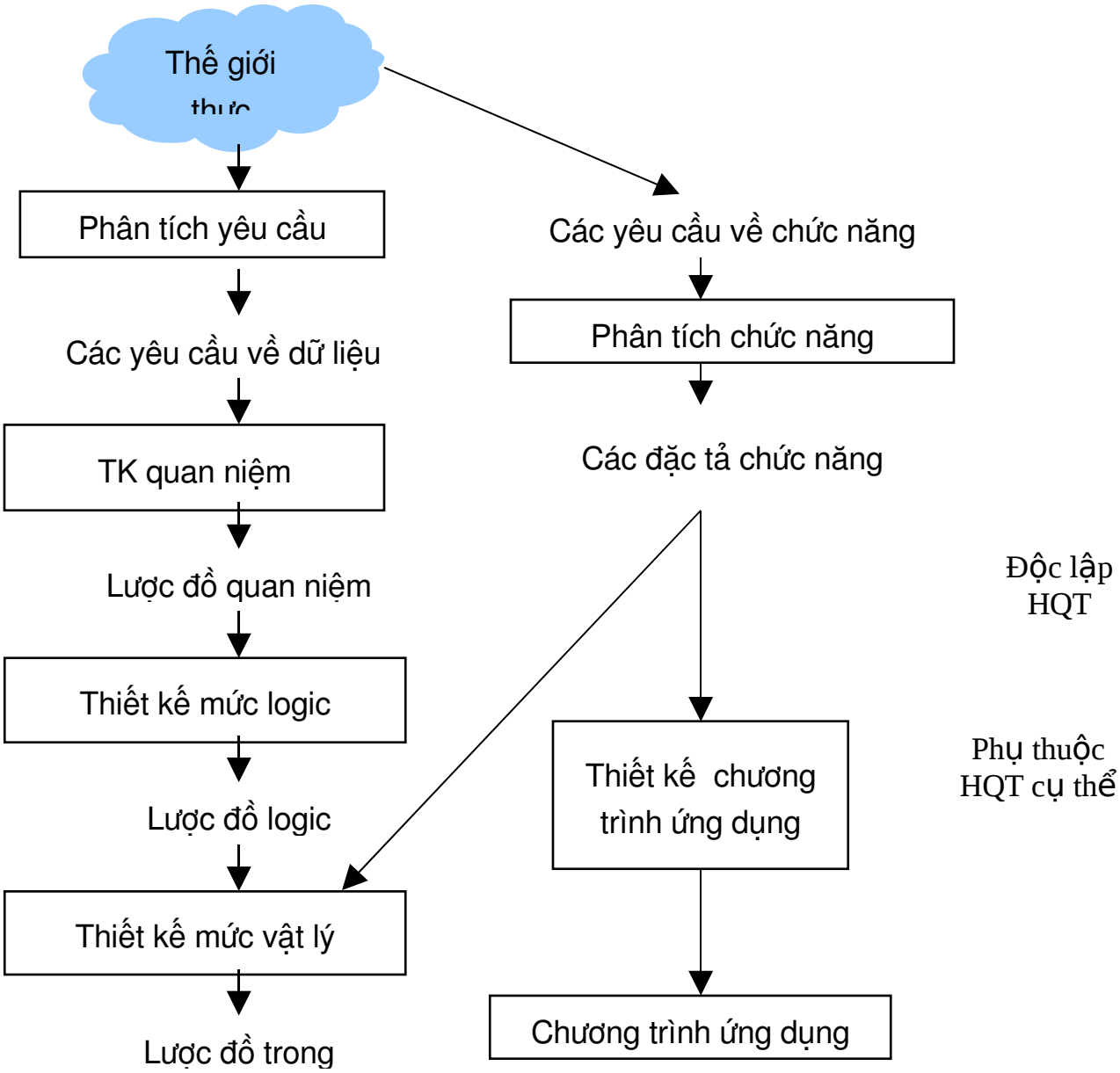
- Ví dụ
  - Các câu lệnh trong SQL: SELECT, INSERT, UPDATE, và DELETE.
  - SELECT id, last_name FROM employees
- Ngôn ngữ điều khiển giao dịch
  (Transaction Control Language - TCL)
  - Đảm bảo tính toàn vẹn dữ liệu khi thực hiện các tác vụ có sự thay đổi dữ liệu
  - Các câu lệnh SQL tương ứng:
    - COMMIT, ROLLBACK, và SAVEPOINT.

- Ngôn ngữ điều khiển dữ liệu
  (Data Control Language - DCL)
  - Cung cấp các tính năng bảo vệ cho các đối tượng của CSDL
  - Các câu lệnh SQL tương ứng:
    - GRANT và REVOKE.

Mô hình liên kết thực thể (Entity-Relationship)

I. Quá trình thiết kế CSDL





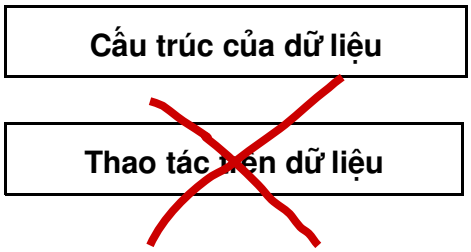
II. Mô hình thực thể - liên kết

- Được dùng để thiết kế CSDL ở mức quan niệm
- Biểu diễn trừu tượng cấu trúc của CSDL

- Lược đồ thực thể- liên kết (Entity-Relationship Diagram)
  - Tập thực thể (Entity Sets)
  - Thuộc tính (Attributes)
  - Mối quan hệ (Relationship)

1. Tập thực thể

- Một thực thể là một đối tượng của thế giới thực. Thực thể được mô tả bởi một tập các thuộc tính
- Tập hợp các thực thể giống nhau tạo thành 1 tập thực thể
- Chú ý
  - Thực thể (Entity)
  - Đối tượng (Object)
  - Tập thực thể (Entity set)
  - Lớp đối tượng (Class of objects)
- Ví dụ "Quản lý đề án công ty"
  - Một nhân viên là một thực thể
  - Tập hợp các nhân viên là tập thực thể
  - Một đề án là một thực thể
  - Tập hợp các đề án là tập thực thể
  - Một phòng ban là một thực thể
  - Tập hợp các phòng ban là tập thực thể



2. Thuộc tính
- Là tập các giá trị có thể gán cho thuộc tính đối với mỗi thực thể riêng biệt
- Miền giá trị của thuộc tính (domain)
  - Kiểu chuỗi (string)
  - Kiểu số nguyên (integer)
  - Kiểu số thực …
- Ví dụ tập thực thể NHANVIEN có các thuộc tính
  - Họ tên (hoten: string[20])
  - Ngày sinh (ns: date)
  - Điểm TB (DTB:float)
  - …

Thuộc tính (tính chất)
- Loại thuộc tính
  - Thuộc tính đơn – không thể tách nhỏ ra được
  - Thuộc tính phức hợp – có thể tách ra thành các thành phần nhỏ hơn
- Loại giá trị của thuộc tính
  - Đơn trị: các thuộc tính có giá trị duy nhất cho một thực thể (VD: số CMND, …)
  - Đa trị: các thuộc tính có một tập giá trị cho cùng một thực thể (VD: bằng cấp, …)
  - Suy diễn được (năm sinh ← → tuổi)
- Tất cả các thực thể nằm trong tập thực thể có cùng tập thuộc tính
- Mỗi thực thể đều được phân biệt bởi một thuộc tính khóa
- Mỗi thuộc tính đều có miền giá trị tương ứng với nó
- Ví dụ tập thực thể NHANVIEN có các thuộc tính
  - Mã NV (MaNV: integer)

- Họ tên (Hoten: string[50])
- Ngày sinh (ns:date)
- Địa chỉ (diachi:string[100])
- Quê quán (quequan:string[30])
- Hệ số lương (hsluong:float)
- Hệ số phụ cấp (hsphucap:float)
- Tổng lương (tongluong:float)

- Loại thuộc tính? Miền giá trị? Và loại giá trị của tt?

3. Mối quan hệ

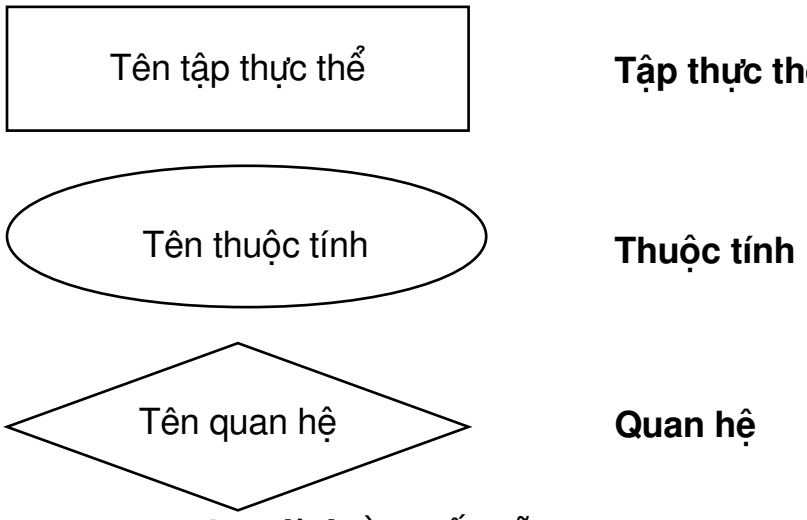
- Quan hệ: Là sự liên kết giữa 2 hay nhiều tập thực thể

- Ví dụ giữa tập thực thể NHANVIEN và PHONGBAN có các liên kết
  - Một nhân viên thuộc một phòng ban nào đó
  - Một phòng ban có một nhân viên làm trưởng phòng

- Tập các quan hệ: là tập hợp các mối quan hệ giống nhau

III. Lược đồ ER

- Là đồ thị biểu diễn các tập thực thể, thuộc tính và mối quan hệ
  - Đỉnh



Tập thực thể

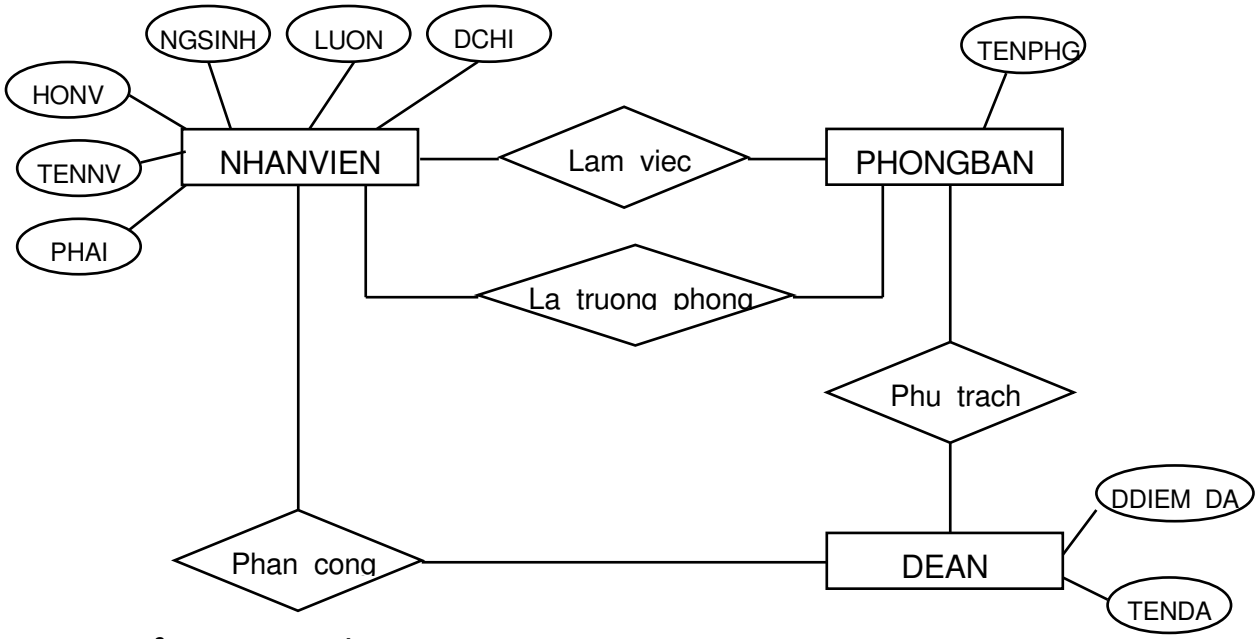
Thuộc tính

Quan hệ

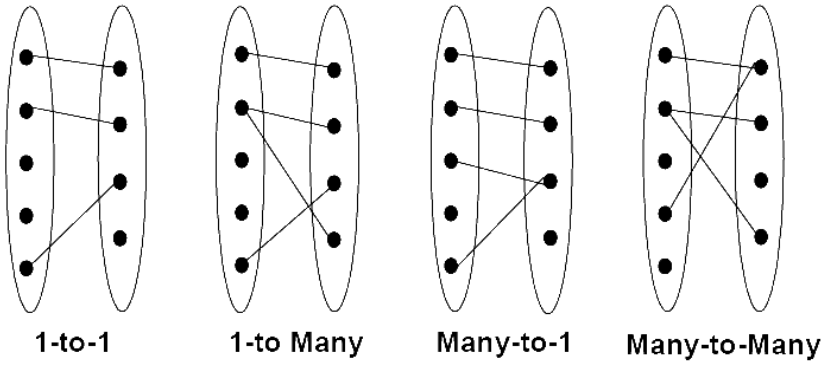
  - Cung là đường nối giữa
    - Tập thực thể và thuộc tính
    - Mối quan hệ và tập thực thể

Ví dụ lược đồ ER

- Kiểu liên kết



- Thể hiện liên kết



**** Thể hiện của lược đồ ER***

- Một CSDL được mô tả bởi lược đồ ER sẽ chứa đựng <u>những dữ liệu cụ thể</u> gọi là thể hiện CSDL
  - Mỗi tập thực thể sẽ có tập hợp hữu hạn các thực thể
    - Giả sử tập thực thể NHANVIEN có các thực thể như NV1, NV2, …NVn
  - Mỗi thực thể sẽ có 1 giá trị cụ thể tại mỗi thuộc tính
    - NV1 có TENNV=“Tung”, NS=“08/12/1955”, GT=“‘Nam”
    - NV2 có TENNV= “Hang”, NS=“07/19/1966”, GT=“Nu”

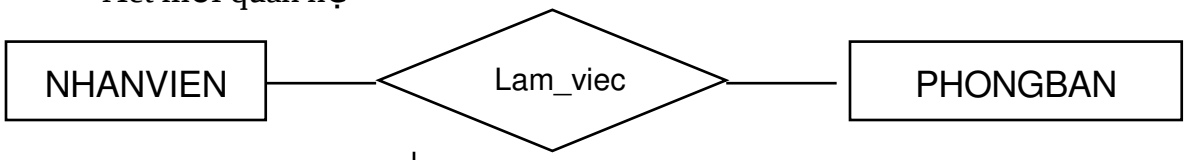
- Chú ý
  - Không lưu trữ lược đồ ER trong CSDL
    - Khái niệm trừu tượng
  - Lược đồ ER chỉ giúp ta *<u>thiết kế CSDL</u>* trước khi chuyển các quan hệ và dữ liệu xuống mức vật lý

**** Ràng buộc trên kiểu liên kết***

- Thể hiện CSDL còn chứa các mối quan hệ cụ thể

- Cho mối quan hệ R kết nối n tập thực thể E1, E2, …, En
- Thể hiện của R là tập hữu hạn các danh sách (e1, e2, …, en)
- Trong đó ei là các giá trị được chọn từ các tập thực thể Ei

• Xét mối quan hệ

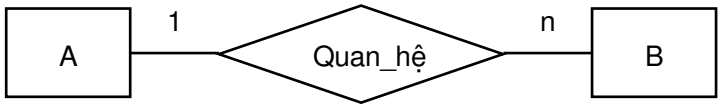


NHANVIEN	PHONGBAN
Tung	Nghien cuu
Hang	Dieu hanh
Vinh	Quan ly

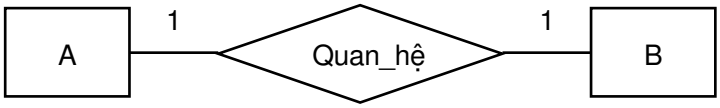
(Tung, Nghien cuu)
(Hang, Dieu hanh)
(Vinh, Quan ly)

• Xét mối quan hệ nhị phân R (binary relationship) giữa 2 tập thực thể A và B, ràng buộc liên kết bao gồm

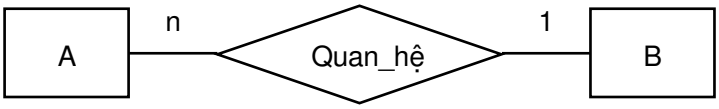
– Một-Nhiều



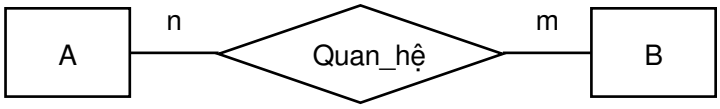
– Một-Một

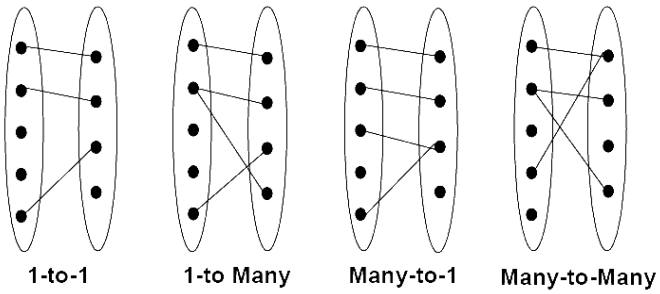


– Nhiều-Một

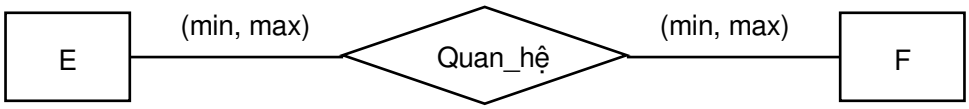


– Nhiều-Nhiều

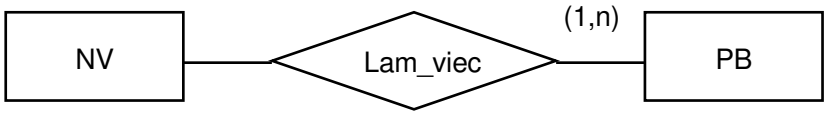




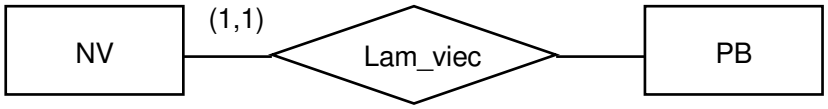
- (min, max) chỉ định mỗi thực thể e ∈ E tham gia ít nhất và nhiều nhất vào thể hiện của R



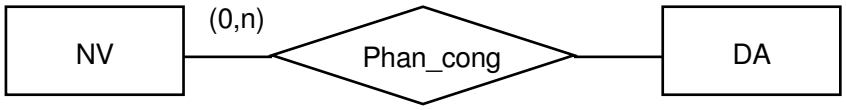
  - (0,1) – không hoặc 1
  - (1,1) – duy nhất 1
  - (0,n) – không hoặc nhiều
  - (1,n) – một hoặc nhiều
- Ví dụ
  - Một phòng ban có nhiều nhân viên



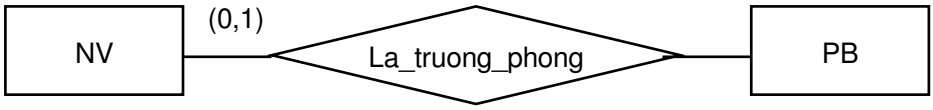
  - Một nhân viên chỉ thuộc 1 phòng ban



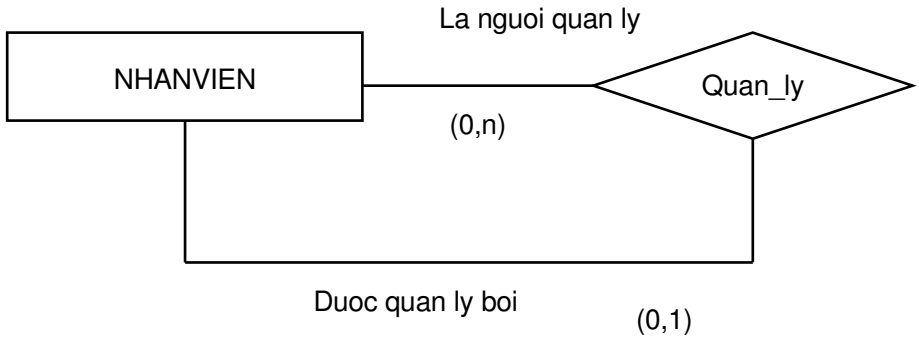
  - Một nhân viên có thể được phân công vào nhiều đề án hoặc không được phân công vào đề án nào



  - Một nhân viên có thể là trưởng phòng của 1 phòng ban nào đó

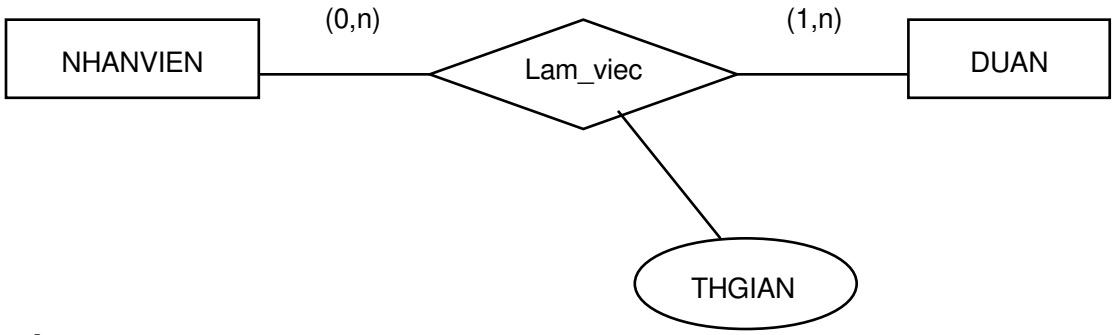


- Một loại thực thể có thể tham gia nhiều lần vào một quan hệ với nhiều vai trò khác nhau



**** Thuộc tính trên mối quan hệ***

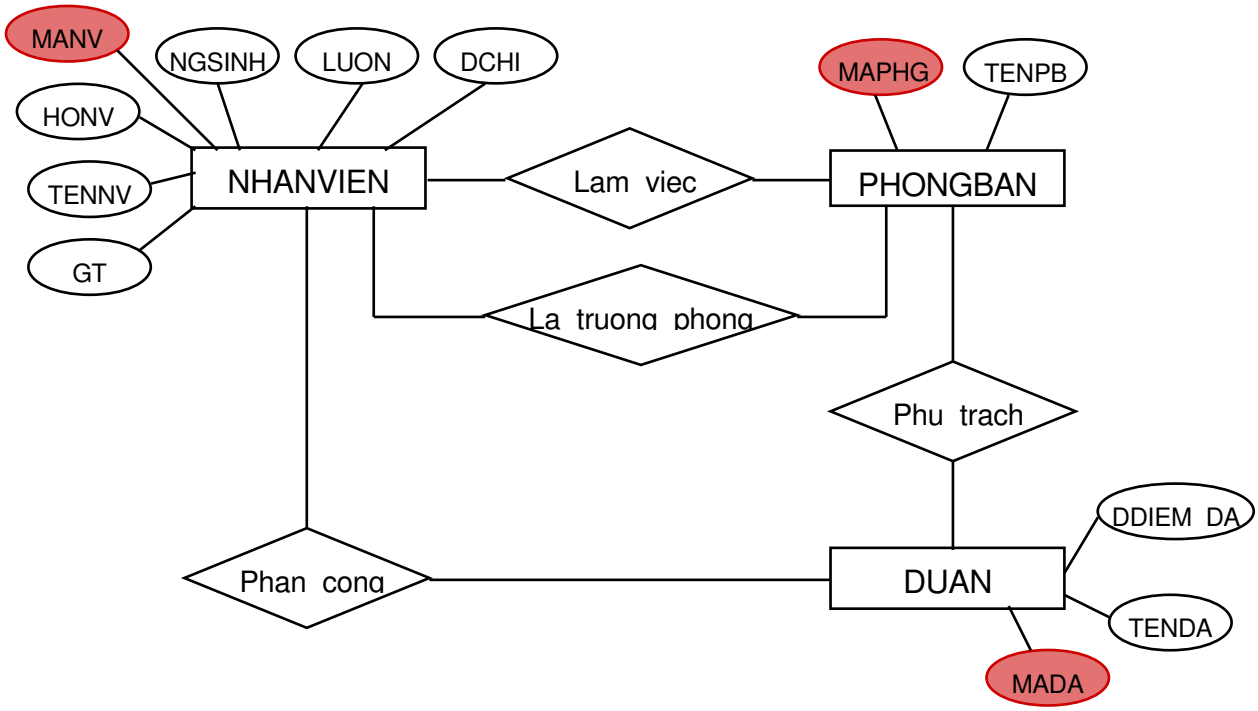
- Thuộc tính trên mối quan hệ mô tả tính chất cho mối quan hệ đó
- Thuộc tính này không thể gắn liền với những thực thể tham gia vào mối quan hệ



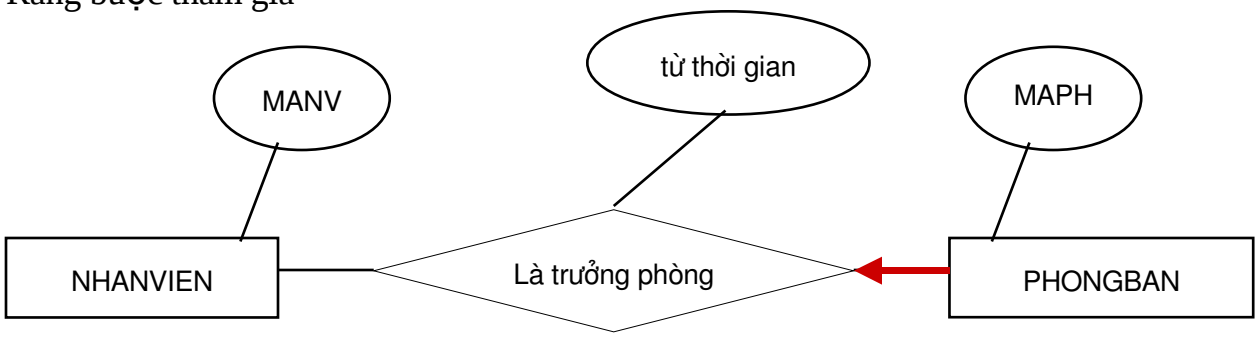
**** Thuộc tính khóa***

- Các thực thể trong tập thực thể cần phải được phân biệt
- Khóa K của tập thực thể E là một hay nhiều thuộc tính sao cho
  - Lấy ra 2 thực thể bất kỳ e1, và e2 trong E
  - Thì e1 và e2 không thể có các giá trị giống nhau tại các thuộc tính trong K
- Chú ý
  - Mỗi tập thực thể phải có 1 khóa
  - Một khóa có thể có 1 hay nhiều thuộc tính
  - Có thể có nhiều khóa trong 1 tập thực thể, ta sẽ chọn ra 1 khóa làm khóa chính cho tập thực thể đó.

Ví dụ thuộc tính khóa



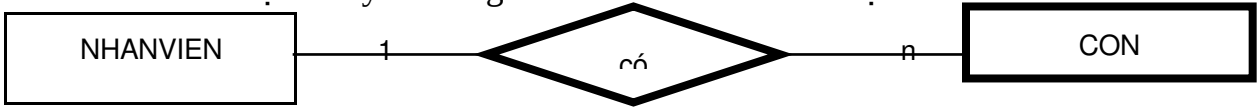
Ràng buộc tham gia



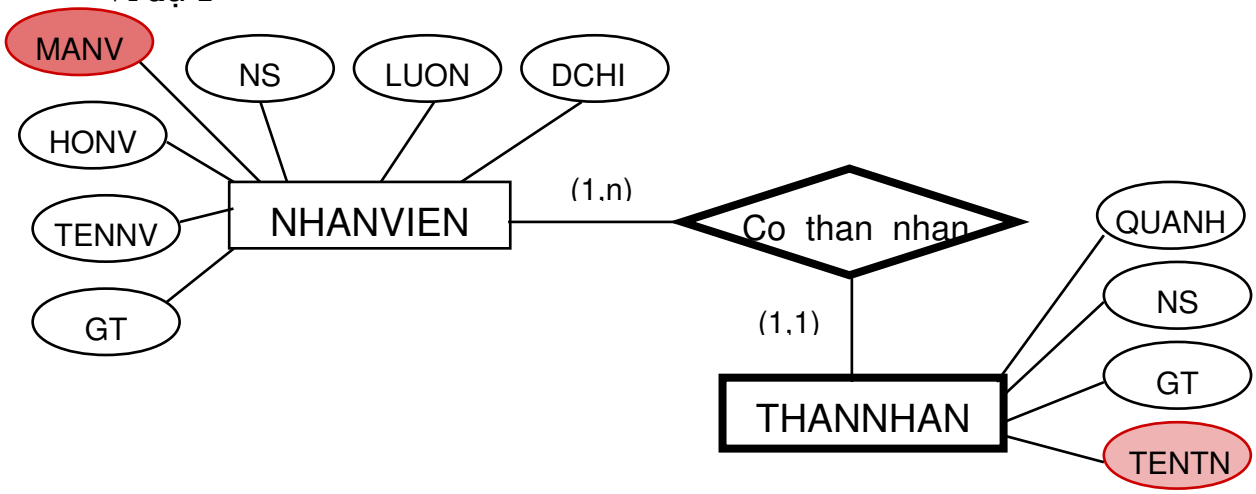
- Xét ví dụ trên
  - Có phải phòng nào cũng có trưởng phòng không?
    - Nếu có → đó là ràng buộc tham gia giữa thực thể NHANVIEN và PHONGBAN.
    - Tham gia toàn bộ vào liên kết
  - Có phải nhân viên nào cũng là trưởng phòng?
    - Sai → *tham gia bộ phận* vào liên kết
- Biểu diễn
  - ← hoặc =

**** Tập thực thể yếu***

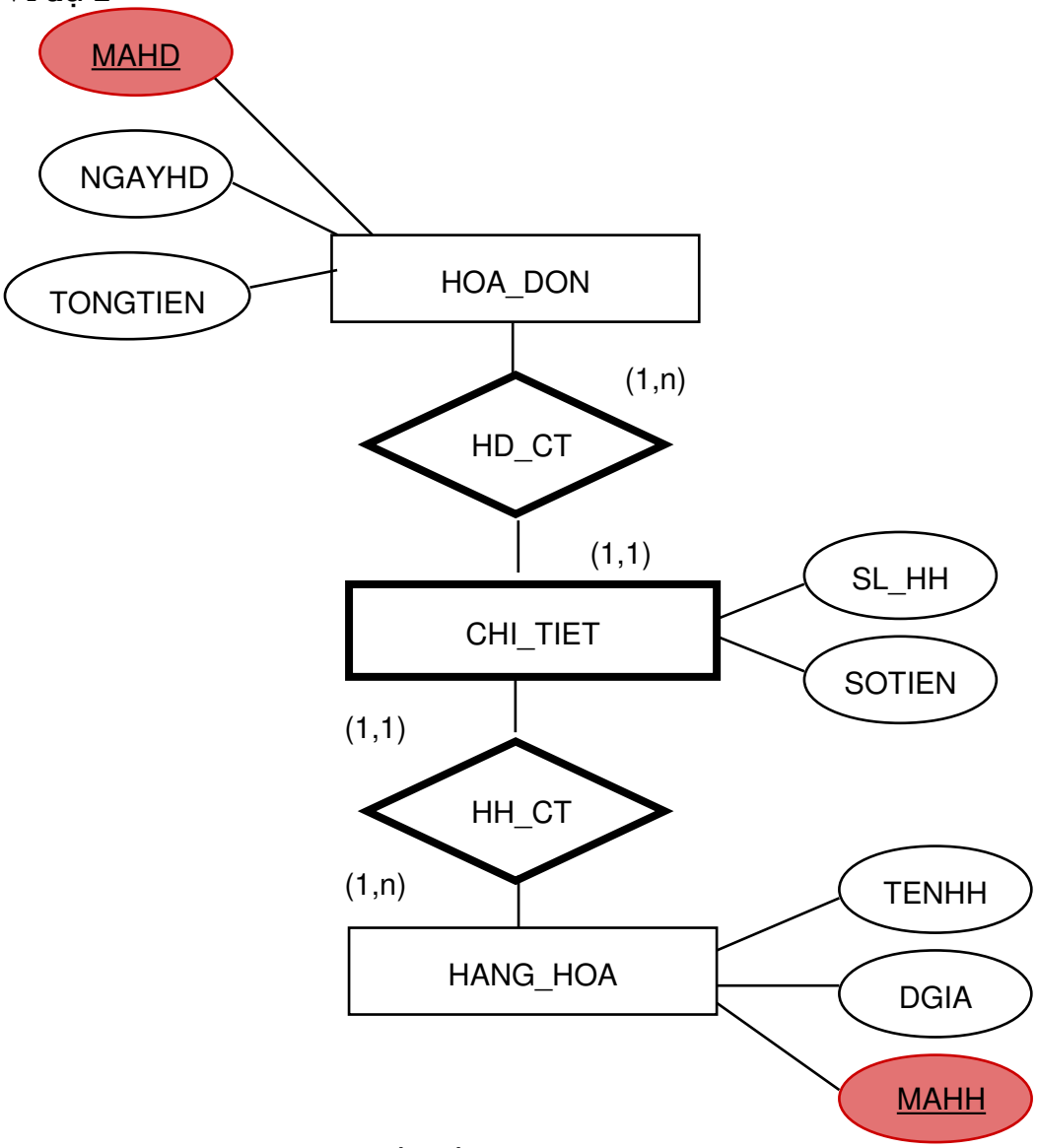
- Là thực thể mà khóa có được từ những thuộc tính của tập thực thể khác
- Thực thể yếu (weak entity set) phải tham gia vào mối quan hệ mà trong đó có một tập thực thể chính (*kiểu thực thể chủ*)
- Mô tả kiểu thực thể yếu bằng hình thoi và hình chữ nhật nét đôi.



- Ví dụ 1

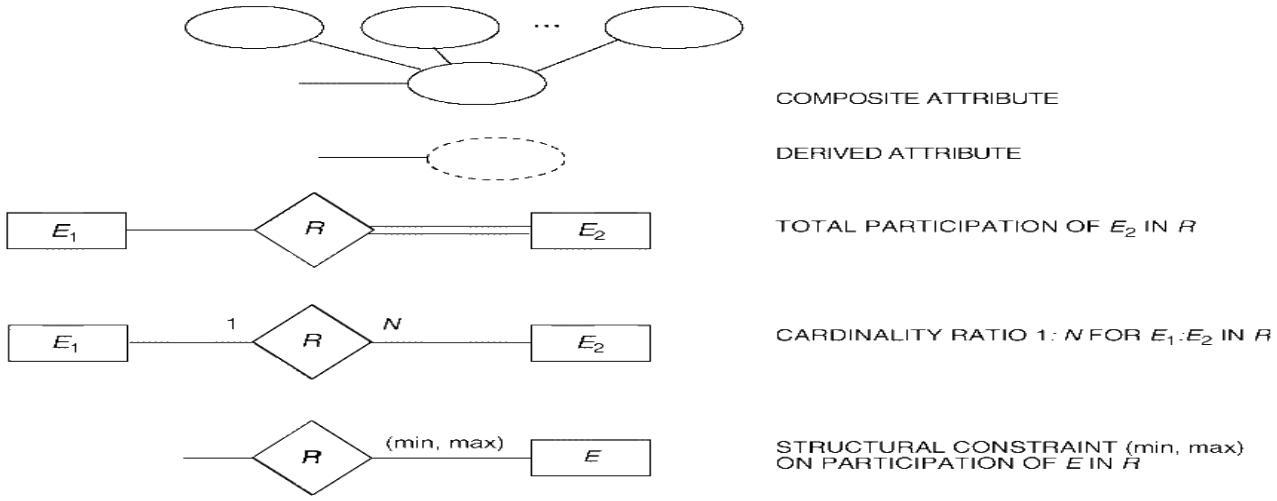


- Ví dụ 2



II. Thiết kế

Các ký hiệu



**** Các bước thiết kế***

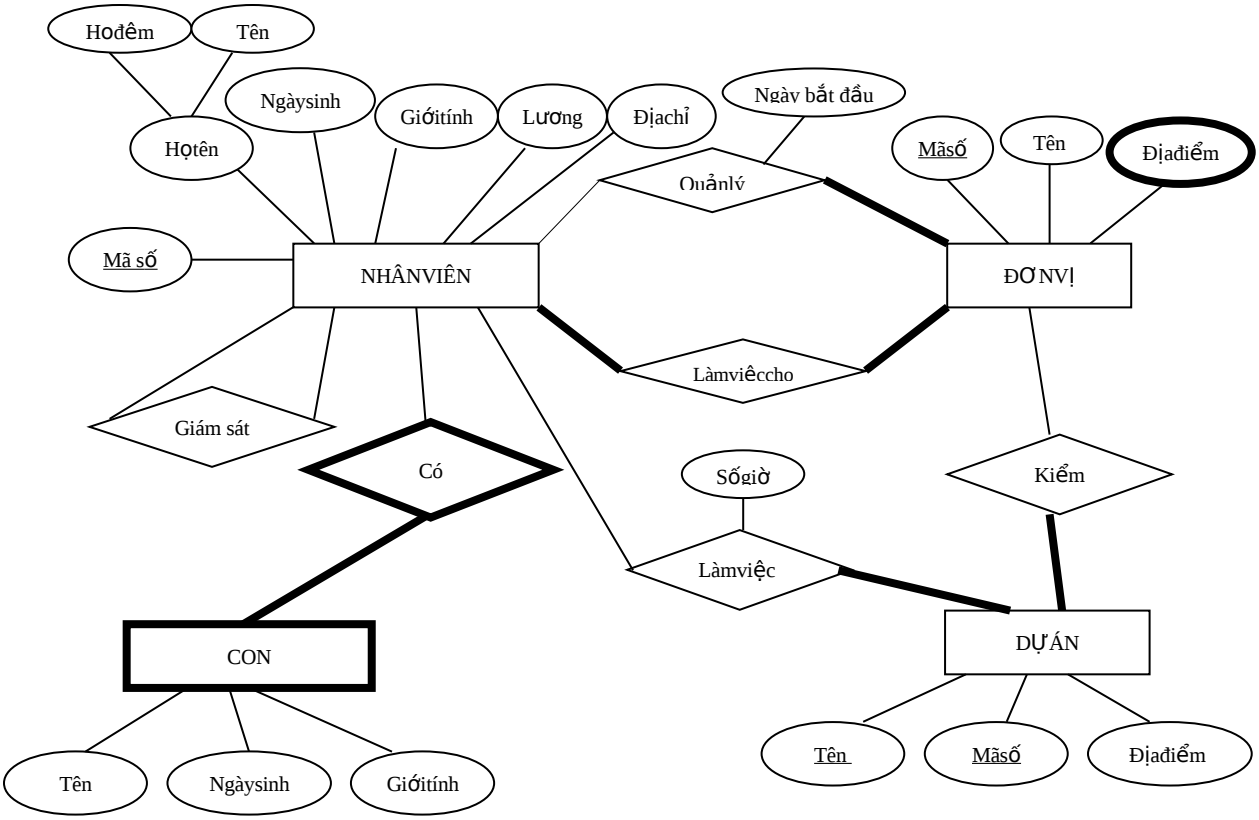
- Xác định tập thực thể
- Xác định mối quan hệ
- Xác định thuộc tính và gắn thuộc tính cho tập thực thể và mối quan hệ
- Quyết định miền giá trị cho thuộc tính
- Quyết định thuộc tính khóa
- Quyết định (min, max) cho mối quan hệ

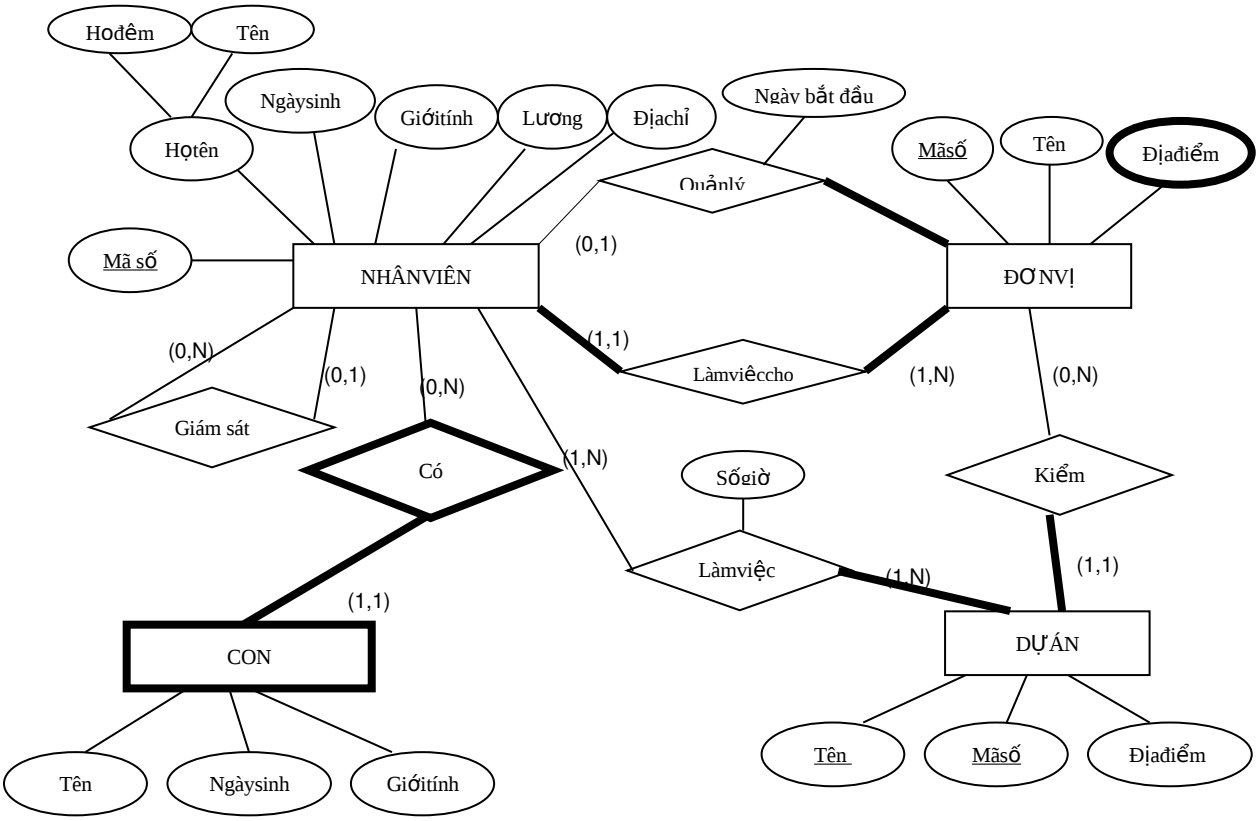
****Qui tắc thiết kế***

- Chính xác
- Tránh trùng lặp
- Dễ hiểu
- Chọn đúng mối quan hệ
- Chọn đúng kiểu thuộc tính

**** Ví dụ 'Quản lý đề án công ty'***

- CSDL đề án công ty theo dõi các thông tin liên quan đến nhân viên, phòng ban và đề án
  - Cty có nhiều đơn vị, mỗi đơn vị có tên duy nhất, mã đơn vị duy nhất, một trưởng phòng và ngày nhận chức. Mỗi đơn vị có thể ở nhiều địa điểm khác nhau.
  - Dự án có tên duy nhất, mã duy nhất, do 1 một phòng ban chủ trì và được triển khai ở 1 địa điểm.
  - Nhân viên có mã số, tên, địa chỉ, ngày sinh, giới tính và lương. Mỗi nhân viên làm việc ở 1 phòng ban, tham gia vào các đề án với số giờ làm việc khác nhau. Mỗi nhân viên đều có một người quản lý trực tiếp.
  - Một nhân viên có thể có những người con được hưởng bảo hiểm theo nhân viên. Mỗi người con của nhân viên có tên, giới tính, ngày sinh.


Hođệm
Tên
Ngàysinh
Giớitính
Lương
Địachỉ
Ngày bắt đầu
Họtên
Mãsố
Tên
Địađiểm
Quảnlý
Mã số
NHÂNVIÊN
ĐƠNVỊ
Làmviêccho
Giám sát
Có
Sốgiờ
Kiểm
Làmviệc
CON
DỰÁN
Tên
Ngàysinh
Giớitính
Tên
Mãsố
Địađiểm


Hođệm
Tên
Ngàysinh
Giớitính
Lương
Địachỉ
Ngày bắt đầu
Họtên
Mãsố
Tên
Địađiểm
Quảnlý
Mã số
NHÂNVIÊN
(0,1)
ĐƠNVỊ
(1,1)
Làmviêccho
(0,N)
(0,1)
(0,N)
(1,N)
(0,N)
Giám sát
Có
(1,N)
Sốgiờ
Kiểm
Làmviệc
(1,N)
(1,1)
(1,1)
CON
DỰÁN
Tên
Ngàysinh
Giớitính
Tên
Mãsố
Địađiểm

Bài tập về nhà

- Bài tập
  - Hoàn chỉnh lược đồ ER cho ví dụ "Quản lý đề án công ty"
  - Các bài tập 1 và 2 trong chương 2
    - Xây dựng mô hình ER cho CSDL TRƯỜNG
    - Xây dựng mô hình ER cho CSDL THƯ VIỆN

BT 1

- Hãy xây dựng lược đồ ER cho CSDL "TRƯỜNG", dựa trên các ghi chép sau:
  - Trường được chia thành các trường con: Trường KHTN, Trường KHXH, Trường Công nghệ,…. Mỗi trường có một hiệu trưởng quản lý. Mỗi hiệu trưởng quản lý một trường.
  - Mỗi trường có nhiều khoa. Chẳng hạn, trường KHTN có các khoa Toán, Lý, Hoá,… Mỗi một khoa chỉ thuộc về một trường. Thông tin về Khoa gồm Mã khoa, tên khoa, địa chỉ, số điện thoại, tên trường.
  - Mỗi Khoa cung cấp nhiều môn học. Mỗi môn học gồm có Tên môn học, mã số, số đơn vị học trình, trình độ, tên Khoa.
  - Mỗi môn học có thể có nhiều học phần.Mỗi học phần được lưu giữ bằng các thông tin: Mã học phần, Tên môn học, Tên giáo viên dạy, học kỳ.
  - Mỗi khoa có nhiều giáo viên làm việc, nhưng mỗi giáo viên chỉ làm việc cho một khoa. Mỗi một khoa có một chủ nhiệm khoa, đó là một giáo viên.
  - Mỗi giáo viên có thể dạy nhiều nhất là 4 học phần và cũng có thể không dạy học phần nào.
  - Mỗi sinh viên phải học nhiều học phần.
  - Mỗi một khoa có nhiều sinh viên, mỗi sinh viên chỉ thuộc về một khoa. Thông tin về mỗi sinh viên gồm: Mã sinh viên, Họ tên, địa chỉ, ngày sinh, giới tính, Lớp, Tên Khoa và chế độ đào tạo.
  - Mỗi sinh viên có một người giám sát (giáo viên chủ nhiệm), người đó là một giáo viên.
  - Sau mỗi học kỳ sẽ có một danh sách điểm để phân loại. Nó gồm các thông tin: Mã sinh viên, mã học phần, điểm bằng chữ, điểm bằng số.

BT 2

- Hãy xây dựng lược đồ ER cho CSDL "THƯ VIỆN", dựa trên các ghi chép sau:
  - Thư viện được chia ra thành các nhánh. Thông tin về mỗi nhánh gồm có Mã nhánh, Tên nhánh và Địa chỉ.
  - Mỗi cuốn sách trong thư viện có các thông tin về Mã sách, Tên sách Nhà xuất bản và Tác giả…
  - Một tác giả có thể viết nhiều cuốn sách. Một cuốn sách có thể có nhiều tác giả viết.

- Một nhà xuất bản xuất bản nhiều cuốn sách. Một cuốn sách do một nhà xuất bản xuất bản. Thông tin về Nhà xuất bản gồm có Tên, Địachỉ và Sốđiệnthoại.
- Một cuốn sách có thể có nhiều bản sao được lưu trữ tại các nhánh. Thông tin về bản sao sách gồm Mã sách, số các bản sao.
- Thư viện có những người mượn sách. Thông tin về những người mượn sách gồm có Số thẻ, Họ tên, Địa chỉ và Số điện thoại.
- Sách được cho các người mượn mượn tại các nhánh. Thông tin về một lần mượn gồm có Ngày mượn và ngày trả.

Chương 2 Mô hình dữ liệu quan hệ

NỘI DUNG

2.1 Các khái niệm cơ bản

* Quan hệ

- Các thông tin lưu trữ trong CSDL được tổ chức thành bảng (table) 2 chiều gọi là quan hệ

1 cột là 1 thuộc tính của nhân viên

TENNV	HONV	NS	DIACHI	GT	LUONG	PHG
Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	5
Hang	Bui	07/19/1968	332 NTH Q1	Nu	25000	4
Nhu	Le	06/20/1951	291 HVH QPN	Nu	43000	4
Hung	Nguyen	09/15/1962	Ba Ria VT	Nam	38000	5

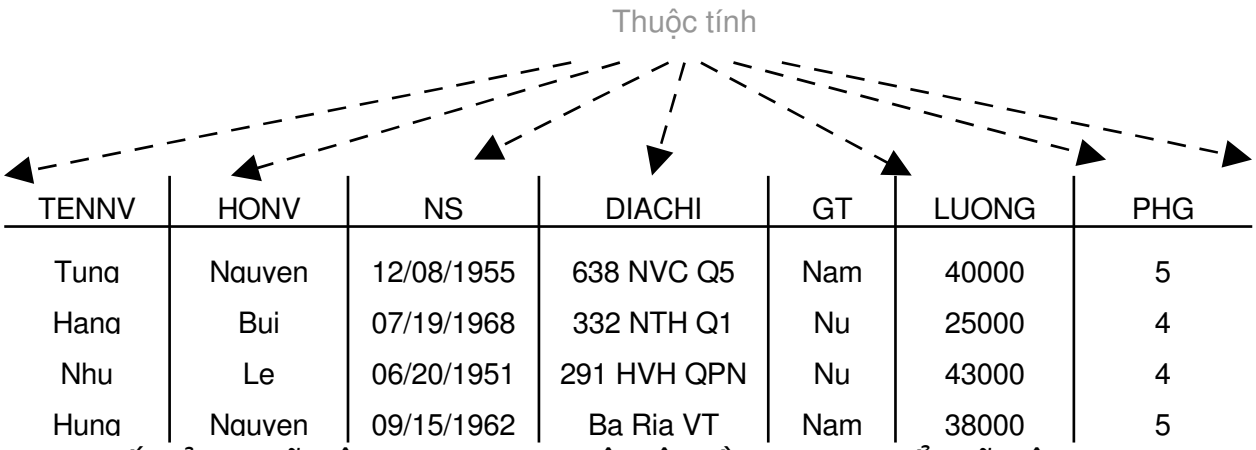
1 dòng là 1 nhân viên

Tên quan hệ là NHANVIEN

- Quan hệ gồm
  - Tên
  - Tập hợp các cột
    - Cố định
    - Được đặt tên
    - Có kiểu dữ liệu
  - Tập hợp các dòng
    - Thay đổi theo thời gian
- Một dòng ~ Một thực thể
- Quan hệ ~ Tập thực thể

**** Thuộc tính***

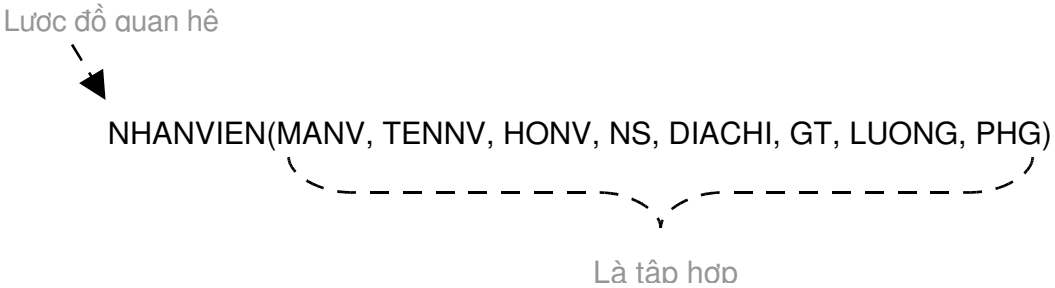
- Tên các cột của quan hệ
- Mô tả ý nghĩa cho các giá trị tại cột đó



- Tất cả các dữ liệu trong cùng 1 một cột đều có cùng kiểu dữ liệu

*** Lược đồ**

- Lược đồ quan hệ
  - Tên của quan hệ
  - Tên của tập thuộc tính



- Lược đồ CSDL
  - Gồm nhiều lược đồ quan hệ

NHANVIEN(MANV, TENNV, HONV, NS, DIACHI, GT, LUONG, PHG)

PHONGBAN(MAPHG, TENPHG, TRPHG, NG_NHANCHUC)

DIADIEM_PHG(MAPHG, DIADIEM)

THANNHAN(MA_NVIEN, TENTN, GT, NS, QUANHE)

DEAN(TENDA, MADA, DDIEM_DA, PHONG)

PHANCONG(MADA, MANV, THOIGIAN)

- ***Bộ***
  - Là các dòng của quan hệ (trừ dòng tiêu đề - tên của các thuộc tính)
  - Thể hiện dữ liệu cụ thể của các thuộc tính trong quan hệ

<Tung, Nguyen, 12/08/1955, 638 NVC, Q5, Nam, 40000, 5>

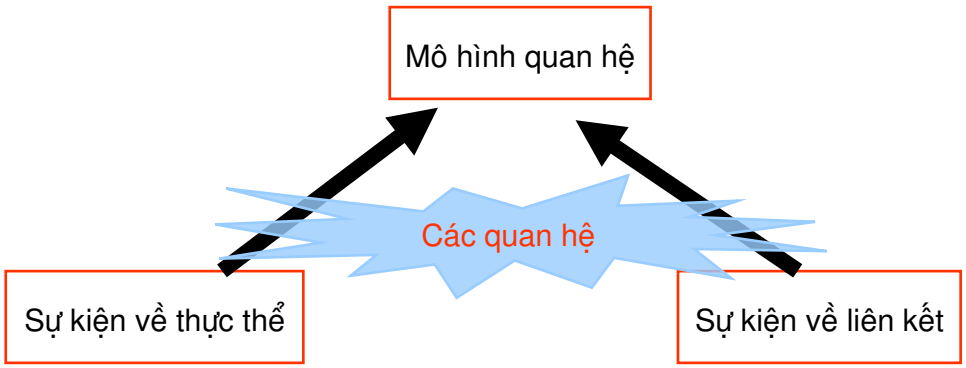
Dữ liệu cụ thể của thuộc tính

* Miền giá trị
  - Là tập các giá trị nguyên tố gắn liền với một thuộc tính
    - Kiểu dữ liệu cơ sở
      - Chuỗi ký tự (string)
      - Số (integer)
    - Các kiểu dữ liệu phức tạp
      - Tập hợp (set)
      - Danh sách (list)
      - Mảng (array)
      - Bản ghi (record)
  - Ví dụ
    - TENNV: string
    - LUONG: integer

* Định nghĩa hình thức
  - Lược đồ quan hệ
    - Cho A1, A2, …, An là các thuộc tính
    - Có các miền giá trị D1, D2, …, Dn tương ứng
    - Ký hiệu R(A1:D1, A2:D2, …, An:Dn) là một lược đồ quan hệ
    - Bậc của lược đồ quan hệ là số lượng thuộc tính trong lược đồ
    - NHANVIEN(MANV:integer, TENNV:string, HONV:string, NGSINH:date, DCHI:string, GT:string, LUONG:integer, DONVI:integer)
      - NHANVIEN là một lược đồ bậc 8 mô tả đối tượng nhân viên
      - MANV là một thuộc tính có miền giá trị là số nguyên
      - TENNV là một thuộc tính có miền giá trị là chuỗi ký tự
  - Quan hệ (hay thể hiện quan hệ)
    - Một quan hệ r của lược đồ quan hệ R(A1, A2, …, An), ký hiệu r(R), là một tập các bộ r = {t1, t2, …, tk}
    - Trong đó mỗi ti là 1 danh sách có thứ tự của n giá trị ti=<v1, v2, …, vn>
      - Mỗi vj là một phần tử của miền giá trị DOM(Aj) hoặc giá trị rỗng

	TENNV	HONV	NGSINH	DCHI	PHAI	LUONG	PHG
t_1	Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	5
t_2	Hang	Bui	07/19/1968	332 NTH Q1	Nu	25000	4
t_3	Nhu	Le	06/20/1951	291 HVH QPN	Nu	43000	4
t_4	Hung	Nguyen	09/15/1962	null	Nam	38000	5

* Thể hiện Mô hình quan hệ



*Tóm tắt các ký hiệu

- Lược đồ quan hệ R bậc n
  - R(A1, A2, …, An)
- Tập thuộc tính của R
  - R+
- Quan hệ (thể hiện quan hệ)
  - R, S, P, Q
- Bộ
  - t, u, v
- Miền giá trị của thuộc tính A
  - DOM(A) hay MGT(A)
- Giá trị tại thuộc tính A của bộ thứ t
  - t.A hay t[A]

2.2. Đại số quan hệ

* Giới thiệu

- Xét một số xử lý trên quan hệ NHANVIEN
  - Thêm mới một nhân viên
  - Chuyển nhân viên có tên là “Tùng” sang phòng số 1
  - Cho biết họ tên và ngày sinh các nhân viên có lương thấp hơn 50000

TENNV	HONV	NS	DCHI	GT	LUONG	PHONG
Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	5
Hang	Bui	07/19/1968	332 NTH Q1	Nu	25000	4
Nhu	Le	06/20/1951	291 HVH QPN	Nu	43000	4
Hung	Nguyen	09/15/1962	Ba Ria VT	Nam	38000	5
Quang	Pham	11/10/1937	450 TV HN	Nam	55000	1

- Có 2 loại xử lý
  - Làm thay đổi dữ liệu (cập nhật)
    - Thêm mới, xóa và sửa
  - Không làm thay đổi dữ liệu (rút trích)
    - Truy vấn (query)
- Thực hiện các xử lý
  - Đại số quan hệ (Relational Algebra)
    - Biểu diễn câu truy vấn dưới dạng biểu thức
  - Phép tính quan hệ (Relational Calculus)
    - Biểu diễn kết quả
  - SQL (Structured Query Language)

* Các thao tác cập nhật

- Nội dung của CSDL có thể được cập nhật bằng các thao tác
  - Thêm (insertion)
  - Xóa (deletion)
  - Sửa (updating)
- Các thao tác cập nhật được diễn đạt thông qua phép toán gán

$$R_{new} \leftarrow \text{các phép toán trên } R_{old}$$

* Thao tác thêm

$$R_{new} \leftarrow R_{old} \cup E$$

- Được diễn đạt
  - R là quan hệ
  - E là một bộ mới cần thêm vào
- Vi phạm toàn vẹn
  - Ràng buộc miền
  - Ràng buộc khóa
  - Ràng buộc tham chiếu
- Ví dụ
  - Phân công nhân viên có mã 123 làm thêm đề án mã số 20 với số giờ là 10

$$PHANCONG \leftarrow PHANCONG \cup (\text{'123'}, 20, 10)$$

* Thao tác xóa

$$R_{new} \leftarrow R_{old} - E$$

- Được diễn đạt
  - R là quan hệ
  - E là một biểu thức ĐSQH
- Ràng buộc toàn vẹn
  - Ràng buộc tham chiếu: được tham chiếu
  - Xử lý:
    - Loại bỏ phép xóa, lan truyền, sửa đổi giá trị
- Ví dụ
  - Xóa các phân công đề án của nhân viên 123456789

* Thao tác sửa

$$R_{new} \leftarrow \pi_{F1, F2, \ldots, Fn} (R_{old})$$

- Được diễn đạt
  - R là quan hệ
  - Fi là biểu thức tính toán cho ra giá trị mới của thuộc tính
- Ràng buộc toàn vẹn
  - Ràng buộc miền
  - Với khóa chính = xóa, chèn
  - Khóa ngoài: đảm bảo tham chiếu đúng giá trị
- Ví dụ
  - Tăng thời gian làm việc cho tất cả nhân viên lên 1.5 lần
  - Chuyển nhân viên "Tùng" từ phòng Nghiên cứu sang phòng Kỹ thuật

* Đại số quan hệ

- Nhắc lại
- Đại số
  - Toán tử (operator)
  - Toán hạng (operand)
- Trong số học
  - Toán tử: +, -, *, /
  - Toán hạng - biến (variables): x, y, z
  - Hằng (constant)
  - Biểu thức
    - (x+7) / (y-3)
    - (x+y)*z and/or (x+7) / (y-3)

II. Đại số quan hệ

- Biến là các <u>quan hệ</u>
  - Tập hợp (set)
- Toán tử là các phép toán (operations)

  - Dựa trên lý thuyết tập hợp
    - Hội $\cup$ (union)
    - Giao $\cap$ (intersec)
    - Trừ $-$ (difference)
  - Rút trích 1 phần của quan hệ
    - Chọn σ (selection)
    - Chiếu π (projection)
  - Kết hợp các quan hệ
    - Tích Đề-các $\times$ (Cartesian product)
    - Nối (join)
  - Đổi tên ρ
- Hằng số là thể hiện của quan hệ
- Biểu thức
  - Được gọi là câu truy vấn
  - Là chuỗi các phép toán đại số quan hệ
  - Kết quả trả về là một thể hiện của quan hệ

III. Phép toán tập hợp

- Quan hệ là tập hợp các bộ
  - Phép hợp $R \cup S$
  - Phép giao $R \cap S$
  - Phép trừ $R - S$
- Tính khả hợp (Tương thích đồng nhất - Union Compatibility)
  - Hai lược đồ quan hệ R(A1, A2, …, An) và S(B1, B2, …, Bn) là <u>khả hợp</u> nếu
    - Cùng bậc n
    - Và có DOM(Ai)=DOM(Bi) , $1 \leq i \leq n$
- Kết quả của $\cup$, $\cap$, và $-$ là một <u>quan hệ</u> có cùng tên thuộc tính với quan hệ đầu tiên (R)
- Ví dụ

NHANVIEN	TENNV	NS	GT
	Tung	12/08/1955	Nam
	Hang	07/19/1968	Nu
	Nhu	06/20/1951	Nu
	Hung	09/15/1962	Nam

THANNHAN	TENTN	NS_TN	GT_TN
	Trinh	04/05/1986	Nu
	Khang	10/25/1983	Nam
	Phuong	05/03/1958	Nu
	Minh	02/28/1942	Nam
	Chau	12/30/1988	Nu

Bậc n=3 DOM(TENNV) = DOM(TENTN)
DOM(NS) = DOM(NS_TN) DOM(GT) =
DOM(GT_TN)

1. Phép hợp
   - Cho 2 quan hệ R và S khả hợp
   - Phép hợp của R và S
     - Ký hiệu $R \cup S$
     - Là một quan hệ gồm các bộ thuộc R hoặc thuộc S, hoặc cả hai (các bộ trùng lắp sẽ bị bỏ)

$$R \cup S = \{ t / t \in R \vee t \in S \}$$

   - Ví dụ

R	A	B
	α	1
	α	2
	β	1

S	A	B
	α	2
	β	3

2. Phép giao
   - Cho 2 quan hệ R và S khả hợp
   - Phép giao của R và S
     - Ký hiệu $R \cap S$
     - Là một quan hệ gồm các bộ thuộc R đồng thời thuộc S

$$R \cap S = \{ t / t \in R \wedge t \in S \}$$

   - Ví dụ

R	A	B
	α	1
	α	2
	β	3

S	A	B
	α	2
	β	3

3. Phép trừ

- Cho 2 quan hệ R và S khả hợp
- Phép giao của R và S
  - Ký hiệu R –S
  - Là một quan hệ gồm các bộ thuộc R và không thuộc S

$$R - S = \{ t / t \in R \wedge t \notin S \}$$

- Ví dụ

R	A	B
	α	1
	α	2
	β	1

S	A	B
	α	2
	β	3

4. Các tính chất
   - Giao hoán

$$R \cup S = S \cup R$$
$$R \cap S = S \cap R$$

   - Kết hợp

$$R \cup (S \cup T) = (R \cup S) \cup T$$
$$R \cap (S \cap T) = (R \cap S) \cap T$$

III. PHÉP CHỌN

- Được dùng để lấy ra các bộ của quan hệ R
- Các bộ được chọn phải thỏa mãn <u>điều kiện chọn P</u>
- Ký hiệu
- P là biểu thức gồm các mệnh đề có dạng
  - <tên thuộc tính> <phép so sánh> <hằng số>
  - <tên thuộc tính> <phép so sánh> <tên thuộc tính>
    - <phép so sánh> gồm $<$, $>$, $\leq$, $\geq$, $\neq$, $=$
    - Các mệnh đề được nối lại nhờ các phép $\wedge$, $\vee$, $\neg$
- Kết quả trả về là một <u>quan hệ</u>
  - Có cùng danh sách thuộc tính với R
  - Có số bộ luôn ít hơn hoặc *bằng* số bộ của R
- Ví dụ

$$\sigma_{(A=B) \wedge (D>5)} (R)$$

R	A	B	C	D
	α	α	1	7
	α	β	5	7
	β	β	12	3
	β	β	23	10

- Phép chọn có tính giao hoán

$$\sigma_{p1}(\sigma_{p2}(R)) = \sigma_{p2}(\sigma_{p1}(R))$$

- Kết hợp nhiều phép chọn thành 1 phép chọn

$$\sigma_{p1}(\sigma_{p2}(R)) = \sigma_{p1\wedge p2}(R))$$

Ví dụ 1

- Cho biết các nhân viên ở phòng số 4
  - Quan hệ: NHANVIEN
  - Thuộc tính: PHG
  - Điều kiện: PHG=4

$$\sigma_{PHG=4}(NHANVIEN)$$

Ví dụ 2

- Tìm các nhân viên có lương trên 2.5tr ở phòng 4 hoặc các nhân viên có lương trên 3tr ở phòng 5
  - Quan hệ: NHANVIEN
  - Thuộc tính: LUONG, PHG
  - Điều kiện:
    - LUONG>2500000 và PHG=4 hoặc
    - LUONG>3000000 và PHG=5

$$\sigma_{(PHG=4 \text{ AND } LUONG>2.5tr) OR (PHG=5 \text{ AND } LUONG>3tr)}(NHANVIEN)$$

IV. Phép chiếu

- Được dùng để lấy ra một vài cột của quan hệ R
- Ký hiệu

$$\pi_{A1, A2, \ldots, Ak}(R)$$

- Kết quả trả về là một quan hệ
  - Có k thuộc tính

– Có số bộ luôn **ít hơn** hoặc bằng số bộ của R

- Ví dụ

R	A	B	C
	α	10	1
	α	20	1
	β	30	1
	β	40	2

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

- Phép chiếu không có tính giao hoán

$$\pi_{X,Y}(R) = \pi_X(\pi_Y(R))$$

$$\pi_{A1, A2, ..., An}(\pi_{A1, A2, ..., Am}(R)) = \pi_{A1, A2, ..., An}(R), n \leq m$$

Ví dụ 3

- Cho biết họ tên và lương của các nhân viên
  - Quan hệ: NHANVIEN
  - Thuộc tính: HONV, TENNV, LUONG

$$\pi_{HONV, TENNV, LUONG}(NHANVIEN)$$

Ví dụ 4

Cho biết mã nhân viên có tham gia đề án hoặc có thân nhân

$$\pi_{MANV}(DEAN)$$

$$\pi_{MANV}(THANNHAN)$$

$$\pi_{MANV}(DEAN) \cup \pi_{MANV}(THANNHAN)$$

Ví dụ 5

Cho biết mã nhân viên có người thân và có tham gia đề án

V. Chuỗi các phép toán

- Kết hợp các phép toán đại số quan hệ
  - Lồng các biểu thức lại với nhau

$\pi_{A1, A2, ..., Ak}(\sigma_P(R))$ $\sigma_P(\pi_{A1, A2, ..., Ak}(R))$

  - Thực hiện từng phép toán một
    - B1 $\sigma_P(R)$
    - B2 $\pi_{A1, A2, ..., Ak}$(Quan hệ kết quả ở B1)

Cần đặt tên cho quan hệ

1. Phép gán

- Được sử dụng để nhận lấy kết quả trả về của một phép toán
  - Thường là kết quả trung gian trong chuỗi các phép toán
- Ký hiệu ←
- Ví dụ
  - B1 $S \leftarrow \sigma_P(R)$
  - B2 $KQ \leftarrow \pi_{A1, A2, ..., Ak}(S)$

2. Phép đổi tên

- Được dùng để đổi tên
  - Quan hệ

Xét quan hệ R(B, C, D)

$\rho_S(R)$: (đọc là rho) Đổi tên quan hệ R thành S

  - Thuộc tính

Đổi tên quan hệ R thành S và thuộc tính B thành X

$\rho_{X, C, D}(R)$: Đổi tên thuộc tính B thành X

$\rho_{S(X,C,D)}(R)$

Ví dụ 7

- Cho biết họ và tên nhân viên làm việc ở phòng số 4
  - Quan hệ: NHANVIEN
  - Thuộc tính: HONV, TENNV
  - Điều kiện: PHG=4
- C1: $\pi_{HONV, TENNV}(\sigma_{PHG=4}(NHANVIEN))$
- C2: $NV\_P4 \leftarrow \sigma_{PHG=4}(NHANVIEN)$

$KQ \leftarrow \pi_{HONV, TENNV}(NV\_P4)$

$KQ(HO, TEN) \leftarrow \pi_{HONV, TENNV}(NV\_P4)$

$\rho_{KQ(HO, TEN)}(\pi_{HONV, TENNV}(NV\_P4))$

VI. Phép tích Đề các

- Được dùng để kết hợp các bộ của các quan hệ lại với nhau
- Ký hiệu
- Kết quả trả về là một quan hệ Q

- Mỗi bộ của Q là tổ hợp giữa 1 bộ trong R và 1 bộ trong S
- Nếu R có u bộ và S có v bộ thì Q sẽ có u × v bộ
- Nếu R có n thuộc tính và S có m thuộc tính thì Q sẽ có (n + m) thuộc tính ($R^+ \cap Q^+ \neq \varnothing$)

• Ví dụ

R	A	B
	α	1
	β	2

S	B	C	D
	α	10	+
	β	10	+
	β	20	-
	γ	10	-

R × S

unambiguous

R	A	B
	α	1
	β	2

S	B	C	D
	α	10	+
	β	10	+
	β	20	-
	γ	10	-

R × S	A	R.B	S.B	C	D
	α	1	α	10	+
	α	1	β	10	+
	α	1	β	20	-
	α	1	γ	10	-
	β	2	α	10	+
	β	2	β	10	+
	β	2	β	20	-
	β	2	γ	10	-

Thông thường theo sau phép tích Đề-các là phép chọn

$\sigma_{A=S.B}(R \times S)$

$R \times S$

A	R.B	S.B	C	D
α	1	α	10	+
α	1	β	10	+
α	1	β	20	-
α	1	γ	10	-
β	2	α	10	+
β	2	β	10	+
β	2	β	20	-
β	2	γ	10	-

A	R.B	S.B	C	D
α	1	α	10	+
β	2	β	10	+
β	2	β	20	-

Ví dụ 8

- Với mỗi phòng ban, cho biết thông tin của người trưởng phòng
  - Quan hệ: PHONGBAN, NHANVIEN
  - Thuộc tính: TRPHG, MAPHG, TENNV, HONV, …

TENPHG	MAPHG	TRPHG	NG_NHANCHUC
Nghien cuu	5	333445555	05/22/1988
Dieu hanh	4	987987987	01/01/1995
Quan ly	1	888665555	06/19/1981

TENPHG	MAPHG	TRPHG	NG_NHANCHU	MANV	TENNV	HONV	…
Nghien cuu	5	333445555	05/22/1988	333445555	Tung	Nguyen	…
Dieu hanh	4	987987987	01/01/1995	987987987	Hung	Nguyen	…
Quan ly	1	888665555	06/19/1981	888665555	Vinh	Pham	…

MANV	TENNV	HONV	NS	DCHI	GT	LUONG	PHG
333445555	Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	5
999887777	Hang	Bui	07/19/1968	332 NTH Q1	Nu	25000	4
987654321	Nhu	Le	06/20/1951	291 HVH	Nu	43000	4
987987987	Hung	Nguyen	09/15/1962	Ba Ria VT	Nam	38000	5

$\sigma_{TRPHG=MANV}$ (PHONGBAN × NHANVIEN)

- B1: Tích Đề-các PHONGBAN và NHANVIEN

$$PB\_NV \leftarrow (NHANVIEN \times PHONGBAN)$$

- B2: Chọn ra những bộ thỏa TRPHG=MANV

$$KQ \leftarrow \sigma_{TRPHG=MANV}(PB\_NV)$$

Ví dụ 9

- Cho biết các phòng ban có cùng địa điểm với phòng số 5
  - Quan hệ: DIADIEM_PHG
  - Thuộc tính: DIADIEM, MAPHG
  - Điều kiện: MAPHG=5

Phòng 5 có tập hợp những địa điểm nào?

MAPHG	DIADIEM
1	TP HCM
4	HA NOI
5	VUNGTAU
5	NHATRANG
5	TP HCM

Phòng nào có địa điểm nằm trong trong tập hợp đó?

MAPHG	DIADIEM
1	TP HCM
4	HA NOI
5	VUNGTAU
5	NHATRANG
5	TP HCM

- B1: Tìm các địa điểm của phòng 5

$$DD\_P5(DD) \leftarrow \pi_{DIADIEM}(\sigma_{MAPHG=5}(DIADIEM\_PHG))$$

- B2: Lấy ra các phòng có cùng địa điểm với DD_P5

$$R1 \leftarrow \sigma_{MAPHG \neq 5}(DIADIEM\_PHG)$$

$$R2 \leftarrow \sigma_{DIADIEM=DD}(R1 \times DD\_P5)$$

$$KQ \leftarrow \pi_{MAPHG}(R2)$$

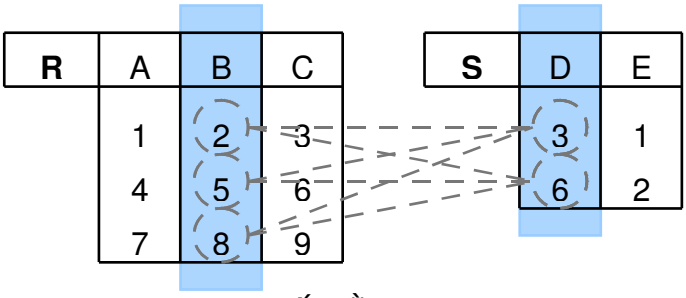
VI. Phép nối

- Được dùng để tổ hợp 2 bộ có liên quan từ 2 quan hệ thành 1 bộ
- Ký hiệu R S
  - R(A1, A2, …, An) và S(B1, B2, …, Bm)
- Kết quả của phép nối là một quan hệ Q
  - Có n + m thuộc tính Q(A1, A2, …, An, B1, B2, …, Bm)

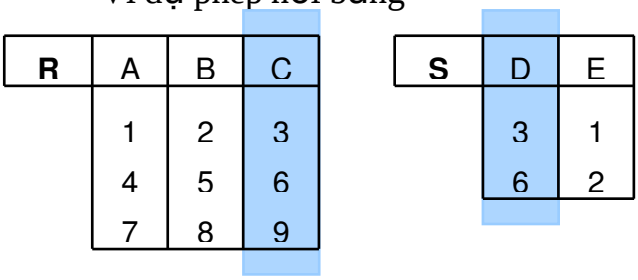
- Mỗi bộ của Q là tổ hợp của 2 bộ trong R và S, thỏa mãn một số điều kiện nối nào đó
  - Có dạng Ai θ Bj
  - Ai là thuộc tính của R, Bj là thuộc tính của S
  - Ai và Bj có cùng miền giá trị
  - θ là phép so sánh ≠ , = , <, >, ≤ , ≥

- Phân loại
  - Nối theta (theta join) là phép nối có điều kiện
    - Ký hiệu R C S
    - C gọi là điều kiện nối trên thuộc tính
  - Nối bằng (equi join) khi C là điều kiện so sánh bằng
  - Nối tự nhiên (natural join)
    - Ký hiệu R S hay R * S
    - R+ ∩ S+ ≠ ∅
    - Kết quả của phép nối bằng bỏ bớt đi 1 cột giống nhau
  - Ví dụ phép nối theta

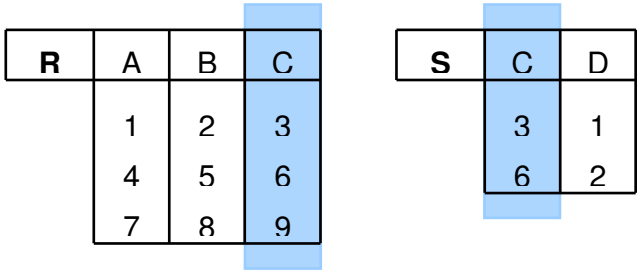
$R \bowtie_{\_\ \_} S$



- Ví dụ phép nối bằng



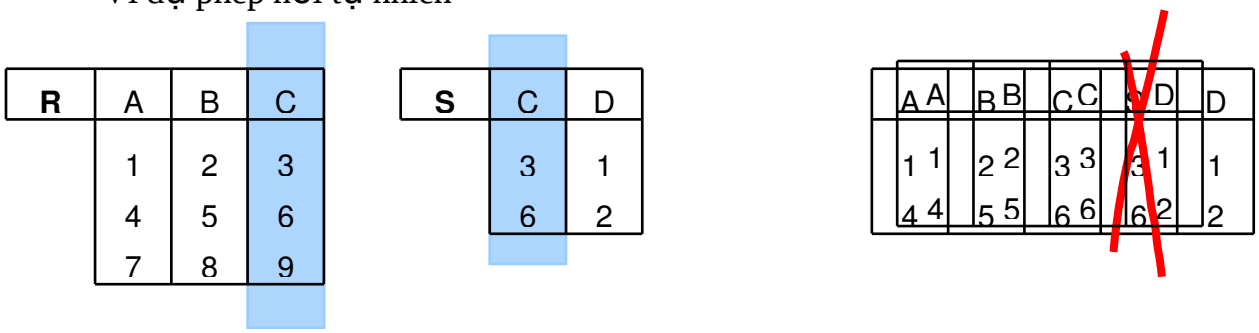
$R \bowtie_{\_\ \_} S$



$R \bowtie_{\_\ \_\ \_} S$

$R \bowtie S$

- Ví dụ phép nối tự nhiên



Ví dụ 10

- Cho biết nhân viên có lương hơn lương của nhân viên 'Tùng'
  - Quan hệ: NHANVIEN
  - Thuộc tính: LUONG

  B1: $R(L\_TUNG) \leftarrow \pi_{LUONG}(\sigma_{TENNV='Tung'}(NHANVIEN))$

  B2: $KQ \leftarrow NHANVIEN \ _{LUONG>L\_TUNG}\ R$

Ví dụ 11

- Với mỗi nhân viên, hãy cho biết thông tin của phòng ban mà họ đang làm việc
  - Quan hệ: NHANVIEN, PHONGBAN

Ví dụ 12

- Với mỗi phòng ban hãy cho biết các địa điểm của phòng ban đó
  - Quan hệ: PHONGBAN, DDIEM_PHG

Ví dụ 13

- Với mỗi phòng ban hãy cho biết thông tin của người trưởng phòng
  - Quan hệ: PHONGBAN, NHANVIEN

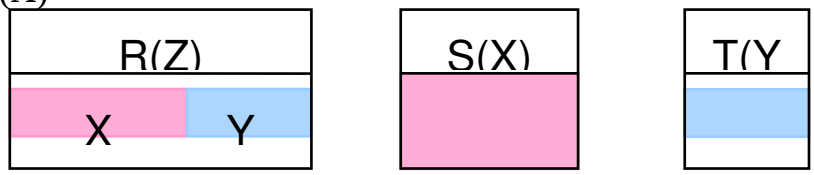
* Tập đầy đủ các phép toán ĐSQH

- Tập các phép toán $\sigma, \pi, \times, \neg, \cup$ được gọi là tập đầy đủ các phép toán ĐSQH
  - Nghĩa là các phép toán có thể được biểu diễn qua chúng
  - Ví dụ
    - $R \cap S = R \cup S - ((R - S) \cup (S - R))$
    - $R \ \ CS = \sigma C(R \times S)$

VII. Phép chia

- Được dùng để lấy ra một số bộ trong quan hệ R sao cho thỏa với <u>tất cả</u> các bộ trong quan hệ S
- Ký hiệu R ÷ S
  - R(Z) và S(X)
    - Z là tập thuộc tính của R, X là tập thuộc tính của S
    - $X \subseteq Z$
- Kết quả của phép chia là một quan hệ T(Y)
  - Với Y=Z-X

– Có t là một bộ của T nếu <u>với mọi bộ</u> tS∈ S, tồn tại bộ tR∈ R thỏa 2 điều kiện
  - tR(Y) = t
  - tR(X) = tS(X)



- Ví dụ

R	A	B	C	D	E
	α	a	α	a	1
	α	a	γ	a	1
	α	a	γ	b	1
	β	a	γ	a	1
	β	a	γ	b	3
	γ	a	γ	a	1
	γ	a	γ	b	1
	γ	a	β	b	1

S	D	E
	a	1
	b	1

R ÷ S

Ví dụ 16
- Cho biết mã nhân viên tham gia tất cả các đề án
  – Quan hệ: PHANCONG, DEAN
  – Thuộc tính: MANV

  B1:

$$DA \leftarrow \pi_{MADA}(DEAN)$$

  B2:

$$NV\_DEAN \leftarrow \pi_{MANV,\ MADA}(PHANCONG)$$

  B3:

$$MA\_NV \leftarrow \pi_{MANV}(NV\_DEAN \div DA)$$

Ví dụ 17
- Cho biết mã nhân viên tham gia tất cả các đề án do phòng số 4 phụ trách
  – Quan hệ: NHANVIEN, PHANCONG, DEAN

– Thuộc tính: MANV
– Điều kiện: PHONG=4

B1:

$$P4\_DA \leftarrow \pi_{MADA}(\sigma_{PHG=4}(DEAN))$$

$$NV\_DA \leftarrow \pi_{MANV, MADA}(PHANCONG)$$

B2:

B3:

$$MA\_NV \leftarrow \pi_{MANV}(NV\_DA \div P4\_DA)$$

* Hàm kết hợp

- Nhận vào tập hợp các giá trị và trả về một <u>giá trị đơn</u>
  – AVG
  – MIN
  – MAX
  – SUM
  – COUNT
- Ví dụ

R	A	B
	1	2
	3	4
	1	2
	1	2

SUM(B) = 10
AVG(A) = 1.5
MIN(A) = 1
MAX(B) = 4
COUNT(A) = 4

Phép gom nhóm

- Được dùng để phân chia quan hệ thành nhiều nhóm dựa trên điều kiện gom nhóm nào đó
- Ký hiệu

  – E là biểu thức ĐSQH
  – G1, G2, …, Gn là các thuộc tính gom nhóm
  – F1, F2, …, Fn là các hàm
  – A1, A2, …, An là các thuộc tính tính toán trong hàm F

Ví dụ 18

- Tính số lượng nhân viên và lương trung bình của cả công ty

$$\mathfrak{I}_{COUNT(), AVERAGE(LUONG)}(NHANVIEN)$$

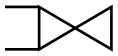
Ví dụ 19

- Tính số lượng nhân viên và lương trung bình của từng phòng ban

$$\mathcal{J}_{MAPGH\ COUNT(),\ AVERAGE(LUONG)}(NHANVIEN)$$

Phép nối ngoài

- Mở rộng phép nối để tránh mất mát thông tin
  - Thực hiện phép nối
  - Lấy thêm các bộ không thỏa điều kiện nối

- Có 3 hình thức
  - Nối ngoài trái 
  - Nối ngoài phải
  - Nối ngoài đầy đủ

Ví dụ 20

- Cho biết họ tên nhân viên và tên phòng ban mà họ phụ trách nếu có
  - Quan hệ: NHANVIEN, PHONGBAN
  - Thuộc tính: TENNV, TENPH

$$R1 \leftarrow NHANVIEN \bowtie_{PHG=MAPHG} PHONGBAN$$

$$KQ \leftarrow \pi_{HONV,TENNV,\ TENPHG}(R1)$$

TENNV	HONV	TENPHG
Tung	Nguyen	Nghien cuu
Hang	Bui	null
Nhu	Le	null
Vinh	Pham	Quan ly

CHƯƠNG III. PHỤ THUỘC HÀM VÀ CHUẨN HÓA CSDL QUAN HỆ

I. Giới hạn của lược đồ ER

- Cung cấp một tập các hướng dẫn → không đưa tới một lược đồ CSDL duy nhất
- Không đưa ra cách đánh giá giữa các lược đồ khác nhau
- → Lý thuyết về chuẩn hóa CSDL quan hệ cung cấp kỹ thuật để phân tích và chuyển hóa từ lược đồ ER sang lược đồ quan hệ

II. Sự dư thừa

- Sự phụ thuộc giữa các thuộc tính gây ra sự dư thừa
  - Ví dụ:

- Điểm các môn học → Điểm trung bình → xếp loại

TENPHG	MAPHG	TRPHG	NG_NHANCHU	MANV	TENNV	HONV	...
Nghien cuu	5	333445555	05/22/1988	333445555	Tung	Nguyen	...
Dieu hanh	4	987987987	01/01/1995	987987987	Hung	Nguyen	...
Quan ly	1	888665555	06/19/1981	888665555	Vinh	Pham	...

- Thuộc tính đa trị trong lược đồ ER → nhiều bộ số liệu trong lược đồ quan hệ
- Ví dụ:
  NHANVIEN(TENNV, HONV, NS,DCHI,GT,LUONG, BANGCAP)

TENNV	HONV	NS	DCHI	GT	LUONG	BANGCA
Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	Trung hoc
Nhu	Le	06/20/1951	291 HVH	Nu	43000	Trung hoc
Nhu	Le	06/20/1951	291 HVH	Nu	43000	Đai hoc
Hung	Nguyen	09/15/1962	Ba Ria VT	Nam	38000	Thac sỹ

- Sự dư thừa → sự dị thường
  - Thao tác sửa đổi: cập nhật tất cả các giá trị liên quan
  - Thao tác xóa: người cuối cùng của đơn vị → mất thông tin về đơn vị
  - Thao tác chèn

TENPHG	MAPHG	TRPHG	NG_NHANCHU	MANV	TENNV	HONV	...
Nghien cuu	5	333445555	05/22/1988	333445555	Tung	Nguyen	...
Dieu hanh	4	987987987	01/01/1995	987987987	Hung	Nguyen	...
Quan ly	1	888665555	06/19/1981	888665555	Vinh	Pham	...

- Các giá trị không xác định
  - Đặt thuộc tính Trưởng phòng vào quan hệ NHANVIEN thay vì vào quan hệ PHONGBAN
- Các bộ giả
  - Sử dụng các phép nối
- Một số quy tắc
  - NT1: Rõ ràng về mặt ngữ nghĩa, tránh các phụ thuộc giữa các thuộc tính với nhau
  - NT2: Tránh sự trùng lặp về nội dung →đảm bảo tránh được các dị thường khi thao tác cập nhật dữ liệu
    - Phải có một số thao tác khi thêm mới và cập nhật vào lược đồ quan hệ, cũng như có thể gây sai hỏng trong trường hợp xóa bỏ các bộ
  - NT3: Tránh đặt các thuộc tính có nhiều giá trị Null
    - Khó thực hiện các phép nối và kết hợp

- NT4: Thiết kế các lược đồ quan hệ sao cho chúng có thể được nối với điều kiện bằng trên các thuộc tính là khoá chính hoặc khoá ngoài theo cách đảm bảo không sinh ra các bộ "giả"
    - Gây lỗi khi thực hiện các phép kết nối

III. Phụ thuộc hàm (Functional Dependency)

- Lý thuyết về chuẩn hóa
    - Các phân tích để đưa ra lược đồ thực thể liên kết cần phải được sửa chữa ở các bước tiếp theo
    - Vấn đề nêu ở slide trên sẽ được giải quyết nếu có một phương pháp phân tích thích hợp

➔ lý thuyết chuẩn hóa (dựa trên phụ thuộc hàm, …) sẽ là nền tảng cơ sở để thực hiện việc phân tích và chuẩn hóa lược đồ ER

- Quan hệ R được định nghĩa trên tập thuộc tính U = { A1, A2, ..., An}. A, B Ì U là 2 tập con của tập thuộc tính U. Nếu tồn tại một ánh xạ f: A ® B thì ta nói rằng A xác định hàm B, hay B phụ thuộc hàm vào A, và ký hiệu là A ® B.
- Quan hệ R (A, B, C) có phụ thuộc hàm A xác định B (ký hiệu là A ® B) nếu: " r, r' ∈ Q, sao cho r.A = r'.A thì r.B = r'.B
- A ® B được gọi là phụ thuộc hàm hiển nhiên nếu B Í A .
- A ® B được gọi là phụ thuộc hàm nguyên tố, hoặc B được gọi là phụ thuộc hàm đầy đủ (fully functional dependence) vào A nếu "A' ⊂ A đều không có A'→B .

Ví dụ

- Quan hệ HÓAĐƠN (Số-hóa-đơn, Số_chủng_loại_mặt_hàng, Tổng-trị-giá) có các phụ thuộc hàm sau:
    f1: Số-hóa-đơn →Số_chủng_loại_mặt_hàng;
    f2: Số-hóa-đơn →Tổng-trị-giá;
- CHITIẾT_HĐ (Số-hóa-đơn, Mã-hàng, Số-lượng-đặt, Đơn-giá, Trị-giá) có các phụ thuộc hàm sau:
    f1: Số-hóa-đơn, Mã-hàng →Số-lượng-đặt.
    f2: Số-hóa-đơn, Mã-hàng →Đơn-giá.
    f3: Số-hóa-đơn, Mã-hàng → Trị-giá.
    f4: Số-lượng-đạt, Đơn-giá → Trị-giá.

* Phụ thuộc hàm

- Gọi F là tập các phụ thuộc hàm đối với lược đồ quan hệ R định nghĩa trên tập thuộc tính U và X →Y là một phụ thuộc hàm; X,Y Í U. Ta nói rằng X →Y được suy diễn lôgic từ F nếu R thỏa các phụ thuộc hàm của F thì cũng thỏa X→Y và ký hiệu là:
  F ½= X →Y.

IV. Phụ thuộc thứ nguyên (Dimensional dependencies)

- Quan hệ R được cung cấp một phụ thuộc thứ nguyên, kí hiệu $X \to n \to C$ với $X \subseteq R+$ và C là một phần tử của R+, n là số nguyên dương nếu có tương ứng mỗi giá trị $x \in X$ nhiều nhất n giá trị C trong R.
- Ví dụ:
  - CONTRACT(NOCON, EMPLOYEE…)
  - NOCON $\to 10 \to$EMPLOYEE.

- ***Các tính chất của phụ thuộc hàm***

- **A1. Tính phản xạ**
  X → X, hay tổng quát hơn nếu $Y \subset X$ thì X → Y
- **A2. Tính bắc cầu**: X → Y và Y → Z thì X → Z.
- **A3. Tính mở rộng hai vế**
  X → Y thì XZ → YZ. (Mở rộng hai vế Z)
- **A4. Tính tựa bắc cầu**: X → Y và YZ → W thì XZ → W
- **A5. Tính mở rộng trái thu hẹp phải**
  X → Y thì XZ → Y – W
- **A6. Tính cộng đầy đủ**: X → Y và Z → W thì XZ → YW
- **A7. Tính tích lũy**: X → Y và Y → ZW thì X → YZW

Chứng minh A2:
$t.X = t'.X \Rightarrow t.Y = t'.Y$
$t.Y = t'.Y \Rightarrow t.Z = t'.Z$
$\Rightarrow t.X = t'.X$ thì $t.Z = t'.Z \Leftrightarrow X \to Z$

V. Hệ tiên đề Armstrong

Hệ A bao gồm các tính chất {A1, A2, A3} của phụ thuộc hàm được gọi là hệ tiên đề Armstrong của lớp các phụ thuộc hàm.
Các tính chất còn lại ({A4, A5, A6, A7}) đều được suy ra từ hệ tiên đề Armstrong.

Chứng minh tính chất A4.
X → Y theo tính mở rộng hai vế
XZ → YZ
Và YZ → W
Theo tính bắc cầu
XZ → W

- **Phép suy dẫn theo hệ tiên đề Armstrong**

PTH f được suy dân theo hệ tiên đề Armstrong là được chứng minh thông qua tiên đề Armstrong. Ký hiệu F |= f.

- **Phép suy dẫn theo quan hệ**

PTH f suy dẫn được từ tập PTH F theo quan hệ (hoặc PTH f được suy dẫn theo quan hệ từ tập PTH F) ký hiệu F |- f, nếu với mọi quan hệ r trên lược đồ R mà F thõa mãn quan hệ đó thì f cũng thỏa mãn r.

- **Định lý 2.1**:

Cho tập PTH F và một PTH f trên R khi đó ta có
F |- f khi và chỉ khi F |= f.

Chứng minh định lý 2.1:

1. F |= f, thì f là một PTH trên R, điều này có nghĩa là f thỏa mãn mọi r thuộc R.
2. F |- f, thì F |= f. Điều này tương đương việc f không suy dẫn được từ PTH F theo hệ tiên đề Armstrong thì cũng không suy dẫn được theo quan hệ.

Định lý 2.2:

Hệ tiên đề Armstrong là đúng đắn và đầy đủ

Chứng minh:

1. Tính đúng đắn của hệ tiên đề Armstrong:

Tính đúng đắn của A1, A2, A3

Bổ đề 2.1:

Giả sử $X \subseteq R$, nếu gọi X+ là tập các thuộc tính A của R mà $F \models X \rightarrow A$ thì với mọi tập $Y \subseteq R$, $F \models X \rightarrow Y \Leftrightarrow Y \subseteq X+$.

a. Chứng minh chiều thuận

Ta có $F \models X \rightarrow Y$. Giả sử Y={A, B, C, ...} theo tính mở rộng trái thu hẹp phải:

$F \models X \rightarrow A$, nên $A \in X+$.

$F \models X \rightarrow B$, nên $B \in X+$.

$F \models X \rightarrow C$, nên $C \in X+$.

..., vậy {A, B, C, ...} = $Y \subset X+$.

b. Chứng minh điều ngược lại

$Y \subset X+$. Theo định nghĩa tập X+ thì mọi $A \in Y$ ta có $F \models X \rightarrow A$, vậy theo tính chất cộng đầy đủ ta có $F \models X \rightarrow Y$.

Chứng minh tính đầy đủ của hệ tiên đề Armstrong

Giả sử f: $X \rightarrow Y$ là một PTH trên R nhưng không suy dẫn được từ tập PTH F theo hệ tiên đề Armstrong. Ta xây dựng được một quan hệ r trên R mà trên đó PTH F thỏa mãn nhưng f: $X \rightarrow Y$ không thỏa mãn.

Xét quan hệ r trên R có hai phần tử t1, t2:

Chia tập R thành hai nhóm thuộc tập X+ và nhóm $R \setminus X+$. t2 chứa toàn giá trị 1, t1 chứa toàn giá trị 1 trong nhưng thuộc tính thuộc X+ và 0 trong những thuộc tính còn lại.

- Chứng minh rằng t1, t2 thỏa mãn mọi PTH của F.

Nếu $W \rightarrow V$ của F không thỏa mãn r thì $W \subseteq X+$ nếu không sẽ không thỏa mãn t1.W = t2.W và $V \not\subset X+$. Nếu không thì t1.V = t2.V và thỏa mãn $W \rightarrow V$. Vậy ít nhất $A \in V$ mà $A \notin X+$.

Chứng minh tính đầy đủ của hệ tiên đề Armstrong

Vì $W \subseteq X+$ nên $X \rightarrow W$ mà $W \rightarrow V$ suy ra $X \rightarrow V$ suy ra $X \rightarrow A$ mà $A \notin X+$ vô lý nên r thỏa mãn mọi f thuộc F.

- f: $X \rightarrow Y$ thỏa mãn r nên $X, Y \subseteq X+$ nếu không sẽ không thỏa mãn t1.X =t2.X hoặc t1.Y=t2.Y. điều này lại suy ra $X \rightarrow Y$ (Hệ quả 2.1). Điều này lại trái với giả thiết vậy f sẽ không thỏa mãn trên r. Vậy f không thuộc F.

VI. Bao đóng F+ của tập PTH F

4. Bao đóng F+ của tập PTH F

Tập PTH f được suy dẫn từ F được gọi là bao đóng của tập PTH F, ký hiệu F+.
Ví dụ:
R={A, B, C, D}
F={A → B, B → C, A →D, B → D}
F+ = {A → B, B → C, A → C, A → D, B → D, A → BD, A → BCD, A → BC, A → CD, B → CD}

Các tính chất của F+

a. Tính phản xạ: $F \subseteq F+$

b. Tính đơn điệu: $F \subseteq G \Rightarrow F+ \subseteq G+$

c. Tính lũy đẳng: F++=F+

- ***Bao đóng X+***

- Định nghĩa bao đóng X+

Cho lược đồ quan hệ R = {A1, ..., An}. Giả sử F là tập PTH trên R. X là tập con của tập thuộc tính R.
Bao đóng X đối với F, ký hiệu X+ (X+ F để chỉ bao đóng lấy theo tập F) là tập thuộc tính A của R mà X → A được suy dẫn từ tập F.
$X+ = \{A: A \in R \text{ và } X \rightarrow A \in F+\}$
Ví dụ:
R = {A, B, C, D, E, G}
F = {A → C, A → EG, B → D, G → E}
X = {A, B}
Y = {C, D, G}
X+ = {A, B, C, D, E, G}
Y+ = {C, D, E, G}

- ***Tính chất của bao đóng X+***

1. Tính phản xạ: $X \subseteq X+$

2. Tính đơn điệu: $X \subseteq Y \Rightarrow X+ \subseteq Y+$.
3. Tính lũy đẳng: X++ = X+
4. Bao đóng tổng chứa tổng các bao đóng: $X+Y+ \subseteq (XY)+$
5. (X+Y)+ = (XY+)+ = (X+Y+)+ = (XY)+
6. $X \rightarrow Y \Rightarrow Y \subseteq X+$
7. $X \rightarrow Y \Rightarrow Y+ \subseteq X+$
8. $X \rightarrow X+$ và $X+ \rightarrow X$
9. $X+ = Y+ \Leftrightarrow X \rightarrow Y$ và $Y \rightarrow X$.

- ***Thuật toán tìm bao đóng X+***

- **Bài toán thành viên**

Vấn đề được đưa ra ở đây là: Cho trước một tập PTH F có hay không một khẳng định $f \in F+$. Để giải quyết bài toán này người ta sử dụng tính chất 6 của tập bao đóng hay bổ đề 2.1: $X \rightarrow Y \in F+ \Leftrightarrow Y \subset X+$.
Do vậy chỉ cần tìm được X+ ta sẽ giải quyết được bài toán X → Y có thuộc F+.

- **Thuật toán tìm bao đóng X+**

Thuật toán tìm bao đóng X+ của Beeri và Bernstein
Cho R = {A1, ..., An}. T là tập PTH trên R. X là tập thuộc tính.
Ta xây dựng tập X0, .., Xk như sau:
X0=X
X(i+1) = XiZi với Zi={A: A $\notin$ Xi và Xi $\rightarrow$ A $\in$ F+} i= 1, 2, ...
Tập X0, X1, ... là tập tăng dần và tập R là hữu hạn nên sau hữu hạn bước thuật toán phải kết thúc. Tồn tại Xk = Xk+1 = ... Chính Xk là tập X+.

Thuật toán

Input: Lược đồ quan hệ R
Tập PTH F, Tập thuộc tính X
Output: Tập X+
Begin
Y:=X
Repeat
Z:=$\emptyset$
For each A in R do
If (A $\notin$ Y and Y $\rightarrow$ A $\in$ F+) then Z= Z $\cup$ A;
Y : = Y $\cup$ Z;
Until Z = $\emptyset$;
X+ = Y
End;

Ví dụ

Cho R = {A, B, C, D, E, G}
Cho tập PTH F ={AB $\rightarrow$ C, C $\rightarrow$ A, BC $\rightarrow$ D, ACD $\rightarrow$ B, D $\rightarrow$ EG, BE $\rightarrow$ C, CG $\rightarrow$ DB, CE $\rightarrow$ AG}
X = {B, D}
X0 = {B, D}, Z0 = {E, G} (D $\rightarrow$ EG)
X1 = {B, D, E, G}, Z1 = {C} (BE $\rightarrow$ C)
X2 = {B, C, D, E, G}, Z2 ={A} (C $\rightarrow$ A)
X3 ={A, B, C, D, E, G} Z3=$\emptyset$
X+ = X3

Chứng minh tính đúng đắn của thuật toán

Chứng minh: X+ $\subset$ Xk và Xk $\subset$ X+.
a. X+ $\subset$ Xk
Thật vậy lấy A $\in$ X+. Như trên ta thấy X+=XZ với Z = {A: A $\notin$ X và X $\rightarrow$ A $\in$ F+}
Nếu A $\in$ X thì A $\in$ Xk vì X $\subset$ Xk.
Nếu A $\in$ Z thì theo định nghĩa các tập Zi, tồn tại một chỉ số i để A $\in$ Zi vậy A $\in$ Xk $\Rightarrow$X+ $\subset$ Xk.
b. Xk $\subset$ X+
X0 $\rightarrow$ X1 $\rightarrow$ … $\rightarrow$ Xk => X $\rightarrow$ Xk => Xk $\subset$ X+

VI. Sơ đồ quan hệ

- Khái niệm sơ đồ quan hệ:

Cho R là một quan hệ trên tập thuộc tính U+, ta nói R thuộc sơ đồ quan hệ α=<U+, F> nếu R thoả mãn tất cả các phụ thuộc hàm của tập F

- Sơ đồ quan hệ là một lược đồ quan hệ và tập phụ thuộc hàm trên nó

- ***Khóa và siêu khóa***

- Cho một quan hệ R trên tập thuộc tính U+ và X$\subset$U+. X được gọi là siêu khoá của quan hệ R nếu R thoả mãn phụ thuộc hàm X $\rightarrow$U+.
- Ví dụ:
  - U+ là một siêu khoá vì U+ $\rightarrow$U+ thoả mãn trên mọi quan hệ R có tập thuộc tính là U+.
  - Quan hệ KQHocTap(MaSV, MaMH, Diem)
    - X1=MaSV, MaMH, Diem
    - X2=MaSV, MaMH
    - X1, X2 là hai siêu khoá của quan hệ QKHocTap
  - Siêu khoá tối thiểu là siêu khoá mà nếu bỏ đi một thuộc tính thì nó không còn là siêu khoá.
  - Một siêu khoá của quan hệ R được gọi là khoá của R nếu nó là siêu khoá tối thiểu.
  - Vậy X (X$\subseteq$U+) là khoá của R $\Rightarrow$R:X$\rightarrow$U+ và $\forall$A$\in$ X thì R không thoả phụ thuộc hàm X-A $\rightarrow$U+
- Ví dụ:
  - X2 là hai siêu khoá của quan hệ QKHocTap

- ***Khoá và siêu khoá của sơ đồ quan hệ***

- *Siêu khoá:* Cho sơ đồ quan hệ α=<U+, F> và X$\subseteq$U+. X được gọi là siêu khoá của lược đồ quan hệ α nếu X là siêu khoá của mọi quan hệ thuộc lược đồ α.
- *Khoá:* Cho lược đồ quan hệ α=<U+, F> và X$\subseteq$U+. X được gọi là khoá của lược đồ quan hệ α nếu X là siêu khoá tối thiểu của lược đồ α.
- ***Ví dụ 1:*** Cho lược đồ quan hệ:
  - α=<U+ ={MaSV, MaMH, Diem}, F={MaSV,MaMH$\rightarrow$Diem}>
  - X=MaSV, MaMH là siêu khoá của lược đồ quan hệ α=<U+, F>. Vì với một quan hệ R bất kỳ thuộc lược đồ thì tập thuộc tính của R, R+=U+={MaSV, MaMH, Diem} và X $\rightarrow$U+.
  - Ngoài ra X là siêu khoá tối thiểu, nên X là khoá của lược đồ quan hệ.
- ***Chú ý:*** X là một siêu khoá chỉ có một thuộc tính thì nó chính là khoá.
- ***Ví dụ 2:*** Cho lược đồ quan hệ α=<U+ =ABCD, F={AB $\rightarrow$CD, B $\rightarrow$AC} Vậy B $\rightarrow$ABCD, $\rightarrow$ B là khoá.

- ***Suy dẫn theo tiên đề***

- Cho F={ X1$\rightarrow$Y1 …Xm$\rightarrow$Ym } là tập các phụ thuộc hàm trên tập thuộc tính U+ và X$\rightarrow$Y là một phụ thuộc hàm.

- Ta nói X→Y có thể suy dẫn theo tiên đề từ F, ký hiệu F |= X→Y nếu ta có thể nhận được X→Y từ các phụ thuộc hàm của F bằng cách sử dụng các tiên đề.
- *Ví dụ*: F={A →B, C →D}. Chứng minh F |= AC →BD. Ta có:

$$\left.\begin{array}{l} A \to B \Rightarrow AC \to BC(fd2) \\ C \to D \Rightarrow BC \to BD(fd2) \end{array}\right\} \Rightarrow AC \to BD \text{ (fd3)}$$

- ***Suy dẫn theo logic***
- Ta nói X→Y có thể suy dẫn theo logic từ F, ký hiệu F |- X→Y, nếu với mỗi quan hệ tuỳ ý, R thoả các phụ thuộc hàm của F thì R cũng thoả X→Y.
- Ví dụ:F={A →B, C →D}. Chứng minh F |- AC →BD
  - Giả sử có t và t' tuỳ ý của R và giả sử rằng t[AC]=t'[AC]. Ta cần chứng tỏ t[BD]=t'[BD]. Thật vậy

$$\left.\begin{array}{l} \left.\begin{array}{l} t[A] = t'[A] \\ A \to B \end{array}\right\} \Rightarrow t[B] = t'[B] \\ \left.\begin{array}{l} t[C] = t'[C] \\ C \to D \end{array}\right\} \Rightarrow t[D] = t'[D] \end{array}\right\} \Rightarrow t[BD] = t'[BD] \Rightarrow đpcm.$$

- ***Suy dẫn theo tiên đề***
- **Bổ đề:** F|=X→Y ⇔ Y⊆ .
- *Chứng minh*
  - Giả sử: Có F|=X→Y , Y=A1A2…Ak.
  - Do A1A2…Ak →Ai (i tuỳ ý:) theo tính phản xạ fd1. Sử dụng tính bắc cầu fd3 ta có F|=X→Ai.
  - Theo định nghĩa của X+F:
  - Ai ∈ X+F. Vì i tuỳ ý nên Y⊆ X+F.
  - Ngược lại, giả sử có Y⊆. Y=A1A2…Ak.

$$\left.\begin{array}{l} A_1 \in X^+ \Rightarrow Theo\ đ/n\ X^+ F \models X \to A_1 \\ \ldots \\ A_k \in X^+ \Rightarrow Theo\ đ/n\ X^+ F \models X \to A_k \end{array}\right\} \Rightarrow F \models X \to Y$$

- ***Tương đương suy dẫn tiên đề và logic***
- Định lý: F |-X→Y ⇔ F |=X→Y

VII. Khóa của sơ đồ

- Định lý: Cho sơ đồ α=<U+, F> và tập thuộc tính X⊆U+.

– X là siêu khoá của α khi và chỉ khi

$$X_F^+ = U^+$$

– X là khoá của α khi và chỉ khi

$$\begin{cases} X_F^+ = U^+ \\ \forall A \in X : (X - A)^+ \neq U^+ \end{cases}$$

- ***Thuật toán tìm khóa***

- Bước 1 :
  + Gán K=U+ (U+ là tập thuộc tính của U)
- Bước 2 : ta có A là thuộc tính của U.
  + Tính bao đóng của (Ki-1 - A)+ nếu bằng U+ ((Ki-1 - A)+ = U+) thì loại bỏ A ra khỏi K tức là Ki =(Ki-1 - A).
  + Nếu (Ki-1 - A)+ != U+ thì Ki = Ki-1.

Bước n: kết quả K=Kn.

Ví dụ

Cho U={A,B,C,D,E} và F={AB->C, AC->B, BC->DE} tìm một khóa của lược đồ quan hệ r xác định trên U và F ?

Bước 1:

+ K=U tức là K=ABCDE

Bước 2:

+ Tính Bao đóng của (K-A)+ nghĩa là tính (BCDE)+ = BCDE, khác U+ nên K=ABCDE

Bước 3:

+ Tính Bao đóng của (K-B)+ : (ACDE)+ = ABCDE, bằng U+ nên loại B ra khỏi K: K=ACDE

Bước 4:

+ Tính Bao đóng của (K-C)+ nghĩa là tính (ADE)+ = ADE, khác U+ nên không bỏ C, K=ACDE

Bước 5:

+ Tính Bao đóng của (K-D)+ nghĩa là tính (ACE)+ = ACEBD = U+ nên bỏ D ra tập K ta có K=ACE

Bước 6:

+ Tính Bao đóng của (K-E)+ nghĩa là tính (AC)+ = ACBDE = U+ nên bỏ E ra tập K ta có K=AC

- ***Thuật toán tìm khóa***

- **Ý tưởng**

  Cho α=<R,F>,

  1. Tìm tất cả tập con khác rỗng của R

2. Loại tập con có bao đóng khác R
3. Loại tập con bao tập con khác
4. Những tập còn lại là khóa của W

- **Thuật toán**

Cho W=<R,F>, R={A, B, C}, F={A → B, A → C, B→A, B →C, AC→B }

	Bao đóng	Siêu khóa	Khóa
A	ABC	A	A
B	ABC	B	B
C	C		
AB	ABC	AB	
AC	ABC	AC	
BC	ABC	BC	
ABC	ABC	ABC	

- **Một số cải tiến**

- Theo tính chất của khóa chúng ta sẽ có một số thuộc tính luôn thuộc khóa. Trong thuật toán tìm khóa sẽ không xét nó và thêm vào khóa

- Một số thuộc tính không thuộc khóa nào cả. Ta loại bỏ nó trong quá trình tìm kiếm khóa

- **Thuật toán**

Cho W=<R,F>, R={A, B, C, D, E, H}, F={A → B, A → C, B→A, B →C, AC→B, E → C, C→H, B→H }

- Chắc chắn D, E tham gia mọi khóa
- H sẽ không tham gia vào khóa nào cả

- Thuật toán tìm khóa: sẽ không cố gắng loại trừ D, E ra khỏi tập. Tập khởi tạo ban đầu có thể là K=R\{K}.
- Thuật toán tìm mọi khóa: Thêm một cột mới luôn chứa D, E. Trong các tập con của thuộc tính còn lại không xem xét đến H.

- **Thuật toán**

	Bao đóng	Siêu khóa	Khóa

DE	A	DEHABC	DEA	DEA
DE	B	DEHABC	DEB	DEB
DE	C	DEHC		
DE	AB	DEHABC	DEAB	
DE	AC	DEHABC	DEAC	
DE	BC	DEHABC	DEBC	
DE	ABC	DEHABC	DEABC	
DE		DEHC		

Bài tập

- Cho lược đồ α=<U+=ABCDEFG, F={AB→C, C→B, ABD→E, F→A}>
  - Tìm khoá của lược đồ α
- Cho lược đồ α=<U+=ABCDEFGH, F={A→C, DC→B, AD→E, F→A, G →H, EF→HG}> .
  - Tìm khoá của lược đồ α
- Cho lược đồ quan hệ: α=<U, F>
  - U={A, B, C, D, E, I} và
  - F={AB→E, AC→I, BC →A, AC→B, CE→D}

  1. Chứng minh F|= BC→E
  2. Tìm tất cả các khoá của lược đồ quan hệ.
- U={A, B, C, D, E, I} và F={AB→C, B→D, CD→E, CI→A}
  - 1. Sử dụng hệ tiên đề Amstrong chứng minh F|= AB→E,
  - 2. Tìm tất cả các khoá của lược đồ quan hệ.

VIII. Các dạng chuẩn

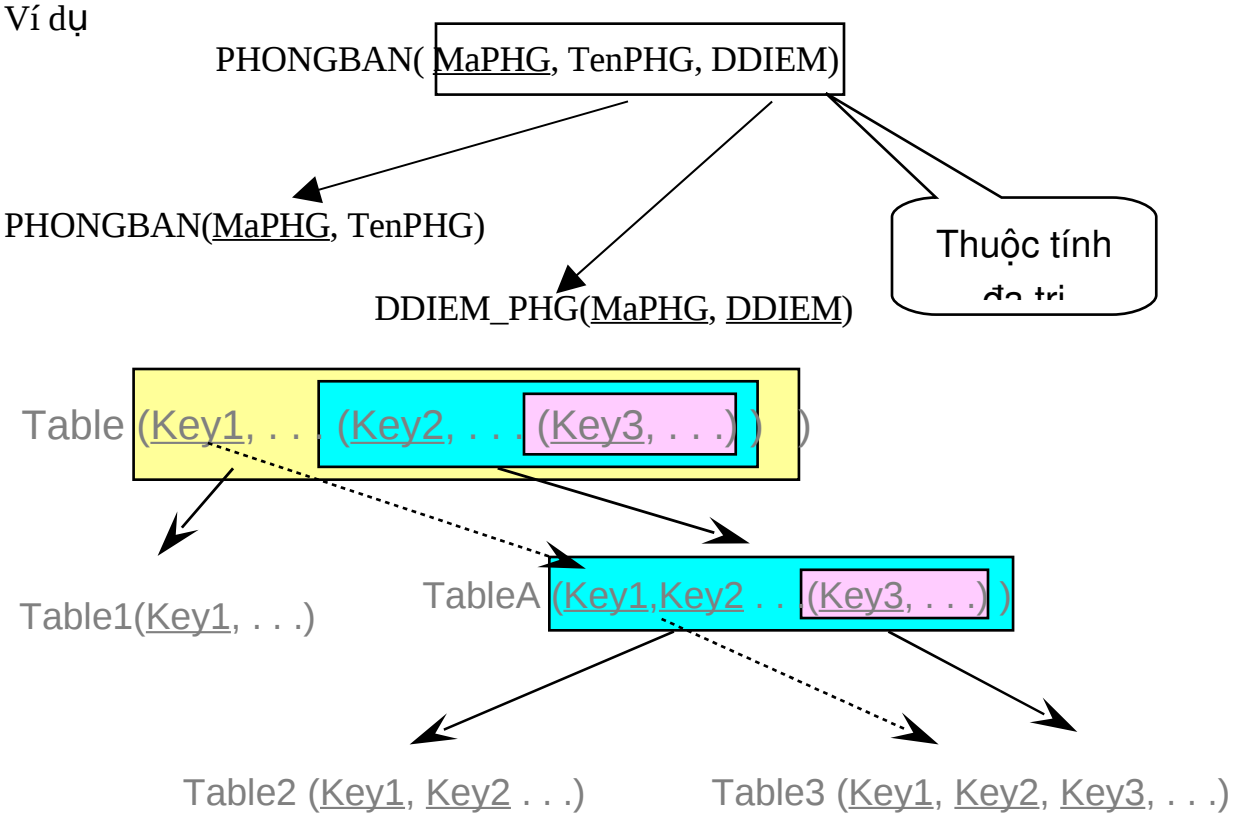
- Mỗi một dạng chuẩn là một tập các điều kiện trên lược đồ nhằm đảm bảo các tính chất của nó (liên quan tới dư thừa và bất thường trong cập nhật)
- Chuẩn hóa dữ liệu: quá trình phân tích lược đồ quan hệ dựa trên các FD và các khóa chính để đạt được
  - Cực tiểu sự dư thừa
  - Cực tiểu các phép cập nhật bất thường
- Thủ tục chuẩn hoá cung cấp
  - Một cơ cấu hình thức để phân tích các lược đồ quan hệ dựa trên các khoá của nó và *các phụ thuộc hàm giữa các thuộc tính* của nó.
  - Một loạt các kiểm tra dạng chuẩn có thể thực hiện trên các lược đồ quan hệ riêng rẽ sao cho cơ sở dữ liệu quan hệ có thể được chuẩn hoá đến một mức cần thiết.
- Tính chất
  - Nối không mất mát (hoặc nối không phụ thêm)
  - Bảo toàn sự phụ thuộc

- nó đảm bảo rằng từng phụ thuộc hàm sẽ được biểu hiện trong các quan hệ riêng rẽ nhận được sau khi tách.

- Phân loại
  - Boyce Codd đề nghị 3 dạng
    - 1NF (first normal form): tương đương với định nghĩa của lược đồ quan hệ (quan hệ và bộ)
    - 2NF: ko có giá trị trong thực tiễn
    - 3NF → BCNF: thường sử dụng nhiều nhất
  - 4NF, 5NF do tính đa trị và phụ thuộc hàm nối

1. Dạng chuẩn 1

- Đn: gọi là 1NF nếu miền giá trị của một thuộc tính chỉ chứa giá trị nguyên tử (đơn, ko phân chia được) và giá trị của mỗi thuộc tính cũng là một giá trị đơn lấy từ miền giá trị của nó
- Ví dụ



- Lược đồ gốc:
  Table (Key1, aaa. . . (Key2, bbb. . . (Key3, ccc. . .)))
- Để thỏa mãn 1NF chúng ta thực hiện
  - Table1(Key1, aaa . . .)
  - Table2(Key1, Key2, bbb . .)
  - Table3(Key1, Key2, Key3, ccc. . .)

2. Dạng chuẩn 2

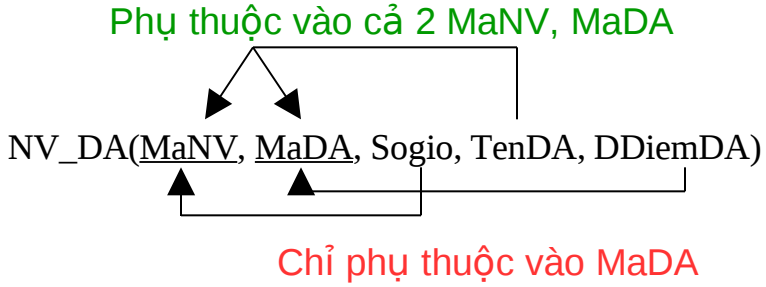
- Phụ thuộc hàm đầy đủ: Một phụ thuộc hàm X $\rightarrow$Y là một phụ thuộc hàm đầy đủ nếu loại bỏ bất kỳ thuộc tính A nào ra khỏi X thì phụ thuộc hàm không còn đúng nữa.

  $\forall$ A, A $\in$ X, (X – {A}) $\rightarrow$Y : là sai.
- Phụ thuộc hàm bộ phận: Một phụ thuộc hàm X $\rightarrow$Y là phụ thuộc bộ phận nếu có thể bỏ một thuộc tính A$\in$ X, ra khỏi X phụ thuộc hàm vẫn đúng, điều đó có nghĩa là với

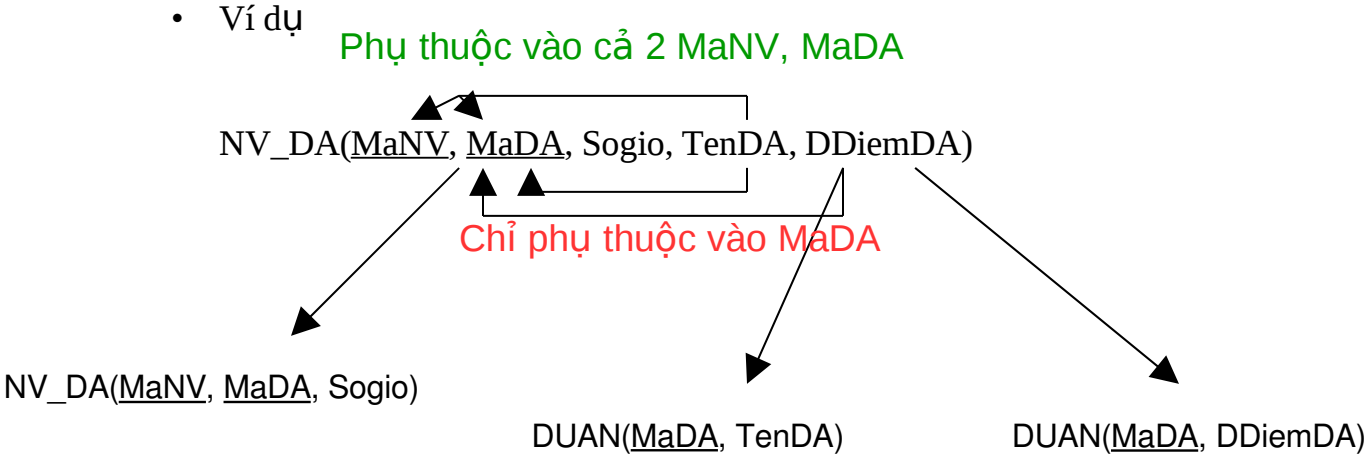
  $\forall$ A$\in$ X, (X – {A}) $\rightarrow$Y
- Tiêu chuẩn Y phụ thuộc đầy đủ vào X

$$\begin{cases} Y \subseteq X_F^+ \\ \forall X' \subset X : Y \not\subset X'^+_F \end{cases}$$

- 2NF:
  - Thỏa mãn 1NF
  - Mọi thuộc tính không khoá đều phụ thuộc đầy đủ vào khoá
- Với các quan hệ có thuộc tính khóa đơn thì ko phải xét
- Chỉ kiểm tra các lược đồ có chứa phụ thuộc hàm bộ phận
- Ví dụ



- Ví dụ

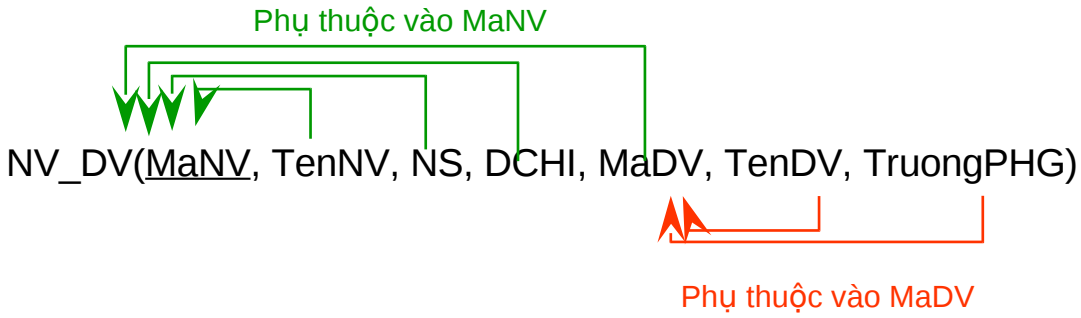


3. Dạng chuẩn 3

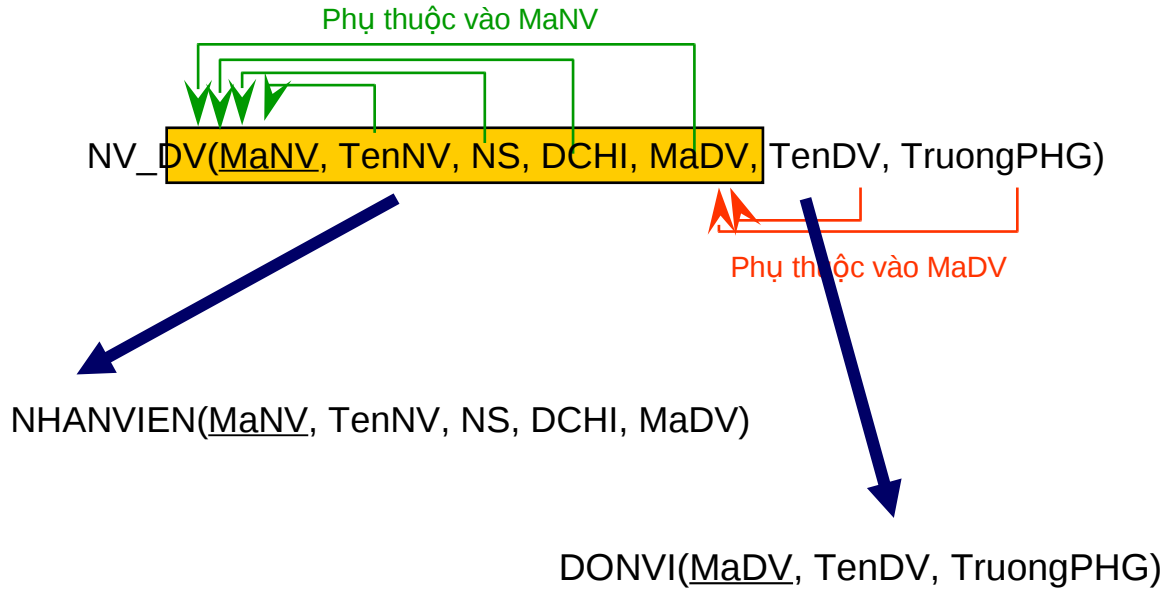
- 3NF dựa trên khái niệm phụ thuộc bắc cầu.
  - Cho lược đồ quan hệ α=<U+, F>, , A là một thuộc tính của U+.
  - Ta nói A phụ thuộc bắc cầu vào X trên α nếu có thể chèn một cầu thực sự Y vào giữa quan hệ A và X. Nghĩa là

$$\begin{cases} X \to Y, Y \to A \ (Y \ không \to X) \\ A \notin XY \end{cases}$$

- ĐN: Một lược đồ quan hệ R là ở 3NF nếu nó thoả mãn (theo Codd)
  - Thỏa mãn 2NF
  - Không có thuộc tính không khoá nào của R là phụ thuộc bắc cầu vào khoá chính.



- Tất cả các thuộc tính phải phụ thuộc vào thuộc tính khóa
  - Một vài thuộc tính phụ thuộc vào thuộc tính ko phải là khóa
  - Chuẩn hóa → Tách nhóm các thuộc tính đó thành quan hệ mới



Ví dụ

- Cho lược đồ quan hệ α như sau:

α=<U+=CTRHSG; F={C →T, HR →C, HT →R, CS →G, HS →R}>

C - giáo trình (Curriculum)T - Giáo viên (Teacher)R - Phòng học (Room)H - Giờ (Hour)S - Sinh viên (Student)G - Lớp (grade)C →T: Mỗi giáo trình có một thầy dạy,

  - HR →C: Chỉ một môn học ở một phòng học tại một thời điểm,
  - HT →R: Tại mỗi thời điểm mỗi giáo viên chỉ có thể dạy ở một phòng học,
  - CS →G: Mỗi sinh viên chỉ ở một lớp học theo mỗi giáo trình.
  - HS →R: Mỗi sinh viên chỉ có thể ở một phòng học tại một thời điểm.
- Yêu cầu: Kiểm tra α có ở dạng chuẩn 3NF hay không? Nếu không α ở dạng chuẩn nào?
  - Xác định tập K các khoá của α và tập N các thuộc tính không khoá.
    - *Nhận xét:* Thuộc tính HS tham gia khoá vì chúng không xuất hiện ở vế phải. Kiểm Tra HS có phải là khoá không.
    - HS+=HSRCTG=U+.
  - Ta có khoá K={HS} là khoá duy nhất, N={RCTG} là các thuộc tính không khoá.
    - *R là thuộc tính không khoá:* HS →HT →R (HT không →HS). Như vậy R phụ thuộc bắc cầu vào khoá HS thông qua cầu HT. Suy ra α không ở dạng chuẩn 3NF.
  - Kiểm tra N=RCTG có phụ thuộc đầy đủ vào HS hay không?
    - C Vậy C phụ thuộc đầy đủ vào HS
    - Tương tự RTG
- KL: α ở dạng chuẩn 2NF.

- ***Thuật toán kiểm tra α có là 3NF***

- Cho lược đồ α=<U+, F> (ở dạng chuẩn 1NF). Kiểm tra ở dạng chuẩn 3NF.
- *Bước 1:* Tính tập K, các khoá của α.
  - Tính tập N các thuộc tính không khoá.
  - Nếu N=∅ thì kết luận α ở dạng chuẩn 3NF. Dừng thuật toán.
- *Bước 2:* (N∅≠)

  - Lấy A∈ N tuỳ ý.
  - Lấy ra Y không chứa khoá nào trong K (Y không là siêu khoá) và A∉ Y.
  - Nếu A ∈ Y+ và A ∉ Y thì kết luận α không ở dạng chuẩn 3NF và dừng.
- *Bước 3:* Nếu không bị dừng ở bước 2 thì kết luận α ở dạng chuẩn 3NF.

Ví dụ

- Cho lược đồ α=<U+={MaSV, TenSV, MaMT, ĐiemThi};
  F={ MaSV→TenSV ; MaSV, MaMT→ĐiemThi}>

α Có ở dạng 3NF không?

- Bước 1.

K = {(MaSV, MaMT)}, N = {TenSV, DiemThi}.

- Bước 2.
  - A = DiemThi,
    - Y = {MaSV}: Y+ = {MaSV, TenSV}=> A ∉ Y+ và A ∉ Y.
    - Y = {MaMT}: Y+ = {MaMT}=> A ∉ Y+ và A ∉ Y.
  - A = TenSV, Y = {MaSV}: Y+ = {MaSV, TenSV}=> A ∈ Y+ và A ∉ Y
    => α không ở dạng 3NF.

Bài tập

- Cho lược đồ quan hệ α=<U+= ABC; F={AB →C; BC →A}>

Hỏi α có ở dạng chuẩn 3NF hay không?

- α=<U+=CTRHSG; F={C →T, HR →C, HT →R, CS →G, HS →R}>

Tóm tắt 3 dạng chuẩn 1-3

NF	Nhận biết	Cách chuẩn hóa
1	Quan hệ ko có thuộc tính đa trị và quan hệ lặp	Chuyển tất cả quan hệ lặp hoặc đa trị thành 1 quan hệ mới
2	Phụ thuộc 1 phần vào thuộc tính khóa	Tách thuộc tính phụ thuộc 1 phần thành lược đồ mới, đảm bảo quan hệ với lược đồ liên quan
3	Phụ thuộc ẩn, tồn tại phụ thuộc hàm giữa các thuộc tính ko phải là khóa	Tách các thuộc tính đó thành lược đồ mới

IX. Dạng chuẩn Boyce-Codd

- Một lược đồ quan hệ R được gọi là ở dạng chuẩn Boyce-Codd (BCNF) nếu nó
  - Thỏa mãn dạng chuẩn 3NF
  - Nếu X →Y là một phụ thuộc hàm không tầm thường thoả mãn trên α thì X phải là siêu khoá của α .
- Hệ quả
  - Nếu R chỉ có hai thuộc tính thì R đạt BCNF

- Nếu một lược đồ quan hệ không thoả mãn điều kiện BCNF, thủ tục chuẩn hóa bao gồm:
  - Loại bỏ các thuộc tính khóa phụ thuộc hàm vào thuộc tính không khóa ra khỏi quan hệ

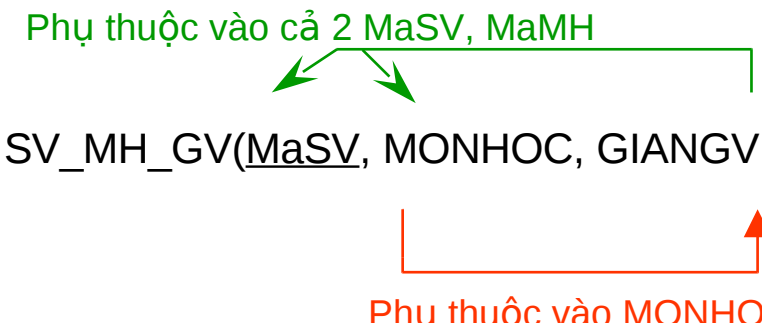
– tách chúng thành một quan hệ riêng có khoá chính là thuộc tính không khóa gây ra phụ thuộc.

- Ví dụ : R (A1,A2,A3,A4,A5)

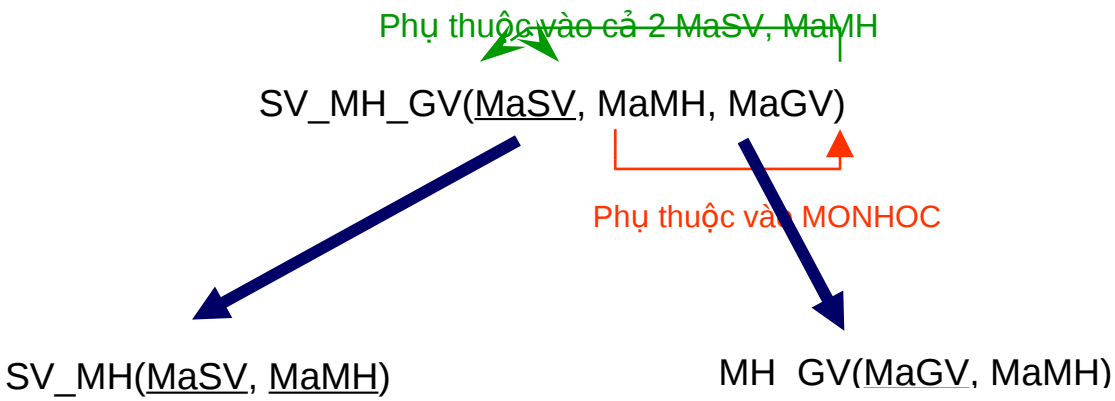
Với các phụ thuộc hàm:

– A1,A2 →A3,A4,A5
– A4 →A2

- lược đồ được tách ra như sau:
  – R1(A4, A2)
  – R2(A1, A4, A3, A5)
- Ví dụ



- Ví dụ



- ***Thuật toán kiểm tra α có ở dạng chuẩn BCNF***

- *Input:* α ở dạng chuẩn 1NF, α=<U+, F>
- *Output:* Kết luận α có ở dạng chuẩn BCNF hay không?
- *Thuật toán:*
  – Test=True
  – For X⊂U+ (X∅≠ and X≠ U+) do
    - If Then Test=False.
  – *If* Test=True *Then* α ở dạng chuẩn BCNF *Else* α chưa ở dạng chuẩn BCNF

Ví dụ

- Cho lược đồ α=<U+=CZS; F={CS →Z, Z →C}>
- α=<U+=ABC; F={AB→C}>

X. Tách kết nối không mất thông tin

- Cho lược đồ quan hệ α=<U+, F>, ta có thể tách lược đồ quan hệ α thành một tập các lược đồ con.

$$\alpha = <U^+, F> = \begin{cases} \alpha_1 = <U_1^+, F_1> \\ \alpha_2 = <U_2^+, F_2> \\ \cdots\cdots\cdots\cdots \\ \alpha_m = <U_m^+, F_m> \end{cases}$$

- Gọi phép tách tập thuộc tính U+ là một bộ D=($U_1^+, U_2^+, ..., U_m^+$) sao cho: $U^+ = U_1^+ \cup U_2^+ \cup ... \cup U_m^+$
- ***Định nghĩa:*** Cho lược đồ quan hệ α=<U+, F>, phép tách D= ($U_1^+, U_2^+, ..., U_m^+$) được gọi là phép tách kết nối không mất thông tin nếu với mỗi quan hệ R thuộc lược đồ α thì $R = R[U_1^+] * R[U_2^+] * ... * R[U_m^+]$

Ví dụ

- Cho lược đồ α=<U+={MaSV, TenSV, MaMT, TenMon, ĐiemThi}; F={MaSV→TenSV; MaMT→TenMon; MaSV, MaMT→ĐiemThi}>
- Tách lược đồ α thành các lược đồ sau:
  - α1=<U1+={MaSV, TenSV}; F1={MaSV→TenSV}>
  - α2=<U2+={MaMT, TenMon}; F2={MaMT→TenMon}>
  - α3=<U3+={MaSV, MaMT, DiemThi}; F3={MaSV, MaMT→ĐiemThi}>
- Lấy quan hệ R là quan hệ KETQUA(MaSV, TenSV, MaMT, TenMon, ĐiemThi) khi đó:
  - R1= KETQUA[MaSV, TenSV] =SINHVIENα∈ 1
  - R2= KETQUA[MaMT, TenMon] =MONTHIα∈ 2
  - R3= KETQUA[MaSV, MaMT, DiemThi] =KQUAα∈ 3
- Khi cần ta có thể khôi phục lại thông tin:
  - KETQUA=SINHVIEN*MONTHI*KQUA

- Kiểm tra tính tách kết nối không mất thông tin

- ***Định lý:*** D=(U_1^+, U_2^+) là phép tách kết nối không mất thông tin của lược đồ α=<U+, F> khi và chỉ khi xảy ra một trong hai điều kiện sau:
  - *i)* $U_1^+ \cap U_2^+ \rightarrow U_1^+ - U_2^+$
  - *ii)* $U_1^+ \cap U_2^+ \rightarrow U_2^+ - U_1^+$

Ví dụ

- Cho lược đồ quan hệ α=<U+={MaSV, TenSV, MaMT, DiemThi}; F={MaSV→TenSV, MaSV MaMT →DiemThi}>
  - U_1^+ = MaSV, TenSV
  - U_2^+ = MaSV, MaMT, DiemThi
  - $U_2^+ \cap U_1^+$ =MaSV; $U_1^+ - U_2^+$=TenSV. Vậy $U_1^+ \cap U_2^+ \rightarrow U_1^+ - U_2^+$ và phép tách trên là phép tách kết nối không mất thông tin.

- Nếu m > 2, sử dụng thuật toán Chase để kiểm tra.

XI. Thuật toán Chase

Input α=<U^+, F> với U^+={A1,A2,…,An} là lược đồ quan hệ. F là tập các PTH; phép tách D=($U_1^+, U_2^+, ..., U_m^+$)

Output: Khẳng định phép tách kết nối không mất hay không?

- Bước chuẩn bị: Xây dựng bảng

  - Xây dựng một bảng gồm n cột và m hàng, cột j tương ứng với thuộc tính A_j , hàng i tương ứng với α_i.
  - Ở vị trí hàng i cột j, ta ký hiệu là a_j nếu A_j thuộc α_i , Ngược lại ta ký hiệu là b_{ij}

Ta có bảng T như sau:

T	A_1	A_1	….	A_n
U_1^+				
U_2^+				
U_n^+				

- Bước lặp: Coi bảng là một quan hệ R trên α và ép cho F thoả bảng.

  - Xét nhiều lần mỗi phụ thuộc hàm $X \rightarrow Y$ thuộc F cho đến khi không xét được nữa. Với mỗi FTH $X \rightarrow Y$ mà trong bảng có các thuộc tính giống nhau trên tập X thì ta cho chúng bằng nhau trên tập Y (chú ý là ưu tiên theo ký tự a).

    Với $\forall t1, t2$ mà $t1[X]=t2[X]$ thì làm bằng nhau tại Y, tức là $t1[Y]=t2[Y]$. Cụ thể:

    + Nếu $\begin{cases} t1[Y] = a_j \\ t2[Y] = b_{ij} \end{cases} \Rightarrow \begin{cases} t1[Y] = a_j \\ t2[Y] = a_j \end{cases}$

    + Nếu toàn bộ là ký tự b thì ta giữ lại 1 bộ và sửa bộ kia theo bộ đó.

  - Thuật toán dừng khi trong bảng có một dòng toàn là ký tự a hoặc cho đến khi không xét được nữa.

- *Bước kết luận*: Nếu có một dòng toàn a thì phép phân rã đó có nối không mất. Ngược lại nối có mất.

Ví dụ

1. Ví dụ 2: Xét phân R=SAIP, R1=SA, R2=SIP, F={S→A, SI→P}

2 Ví dụ 3: R=ABCDE, R1=AD, R2=AB, R3=BE, R4=CDE, R5=AE. Giả sử có tập phụ thuộc hàm F={A→C, B→C, C→D, DE→C, CE→A}

3. Cho lược đồ quan hệ: α=<U, F> ở dạng chuẩn 1NF với
   - U={A, B, C, D, E, I} và
   - F={AB→E, AC→I, BC →A, AC→B, CE→D}
   - Kiểm tra phép tách α thành các lược đồ con {U1=ABE, U2=ABCI, U3=ACD } kết nối có mất thông tin hay không?

4. Cho lược đồ quan hệ: α=<U, F> ở dạng chuẩn 1NF với U={A, B, C, D, E, I} và

F={BC→DE, BE →C, BI→A, CE →I}

- Kiểm tra phép tách α thành {U1=BIA, U2=CEI, U3=BCDE} có mất thông tin?

XII. Đưa về dạng chuẩn BCNF

- ***Bổ đề:***
  - Mỗi lược đồ có hai thuộc tính đều ở dạng chuẩn BCNF.
  - Nếu α=<U+, F> không ở dạng chuẩn BCNF thì chúng ta có thể tìm được các thuộc tính A và B trong α=<U+, F> sao cho (U-AB) →A hoặc (U-AB) →B.

- Chứng minh

- Giả sử ta có lược đồ α=<U={A,B}, F> có các trường hợp xảy ra:

- α=<U={A,B}, F=$\varnothing$>: Ở BCNF.
- α=<U={A,B}, F={A$\twoheadrightarrow$B}>: Khóa A và ở BCNF.
- α=<U={A,B}, F={A$\twoheadrightarrow$B, B$\twoheadrightarrow$A }>: Khóa AB và ở BCNF

- α=<U, F> chưa ở BCNF, giả sử có một vi phạm là X$\rightarrow$A trong lược đồ. Do đó $\exists$B không thuộc XA (Nếu không X là siêu khóa và không vi phạm) để (U-AB) $\rightarrow$A. Đpcm.

Thuật toán

+ *Bước 1:* Kiểm tra α có ở dạng chuẩn BCNF hay không?

Test=True

For X$\subset$U$^+$ (X$\neq\varnothing$ and X$\neq$U$^+$) do

If $X^+ \subset U^+$ and $X^+ \neq X$ Then Test=False.

If Test=True *Then* α ở dạng chuẩn BCNF và dừng thuật toán

Else α chưa ở dạng chuẩn BCNF ta chuyển sang bước 2 để tách

+ *Bước 2:* Tách α thành hai lược đồ con thực sự

$$\alpha = < U^+, F > = \begin{cases} \alpha_1 = < U_1^+, F_1 > : U_1^+ \subset U^+ \\ \alpha_2 = < U_2^+, F_2 > : U_2^+ \subset U^+ \end{cases}$$

Phép tách được thực hiện như sau:

Dựa vào thuật toán kiểm tra ở bước 1, do α chưa ở dạng chuẩn BCNF nên ở bước kiểm tra $\exists X$: $X^+ \neq X$ và $X^+ = U^+$. Ta sử dụng X để tách α như sau:

Xác định $U_1^+ = X \cup A$ với $A \subseteq X^+ - X$ và $U_2^+ = U^+ - A$

Ta có $\begin{cases} U_1^+ \cap U_2^+ = X \\ U_1^+ - U_2^+ = A \end{cases} \Rightarrow U_1^+ \cap U_2^+ \rightarrow U_1^+ - U_2^+ (X \rightarrow A)$.

Vậy phép tách xây dựng như trên là phép tách kết nối không mất thông tin.

Tách thành $\alpha_1 = < U_1^+, F_1 >$ và $\alpha_2 = < U_2^+, F_2 >$

Xác định F_1 như sau:

$F_1 = \emptyset$

For $X \sqsubset U_1^+$, $X \neq U_1^+$, $\emptyset$ Do

If $(X_F^+ \cap U_1^+) - X = Y \neq \emptyset$ Then $F_1 = F_1 \cup \{X \rightarrow Y\}$

Kết quả ta được $\alpha_2 = <U_1^+, F_1>$.

Ta thực hiện bước lặp tương tự đối với $\alpha_2 = <U_2^+, F_2>$, nếu cần ta lại tách đôi.

Thuật toán sẽ dừng vì các lược đồ con có số thuộc tính ít hơn các lược đồ mẹ.

Ví dụ

- Cho lược đồ quan hệ: α=<U, F> ở dạng chuẩn 1NF với
  - U={A, B, C, D, E, I} và
  - F={AB→C, B→D, CD→E, CI→A}
- Tách lược đồ α=<U+=CTRHSG; F={C →T, HR →C, HT →R, CS →G, HS → R}> thành các lược đồ ở dạng chuẩn BCNF
- Cho lược đồ quan hệ: α=<U, F> ở dạng chuẩn 1NF với
  - U={A, B, C, D, E, I} và
  - F={BC→DE, BE →C, BI→A, CE →I}

*** Đưa về dạng chuẩn 3NF**

- *Bước 1:* Loại bỏ tất cả các thuộc tính của U+ nếu các thuộc tính đó không liên quan đến một phụ thuộc hàm nào của F.
- *Bước 2:* Nếu có 1 phụ thuộc hàm nào của F mà liên quan đến tất cả các thuộc tính của U+ thì kết quả chính là lược đồ α với fd đó.
- *Bước 3:* Thực hiện tách: Đưa ra các lược đồ gồm các thuộc tính XA cho phụ thuộc hàm X →A của F. Nếu có X →A1, X →A2,…, X →An, thì thay thế tập thuộc tính XA1A2…An cho XAi (1≤ i≤ n). Quá trình tách cứ tiếp tục.

Ví dụ

- α=<U+=CTRHSG; F={C →T, HR →C, HT →R, CS →G, HS →R}> được tách thành các lược đồ:
- α1=<U1+=CT; F1={C →T}>
- α2=<U2+=HRC; F2={HR →C}>
- α3=<U3+=HRT; F3={HT →R}>
- α4=<U4+=CSG; F4={CS →G}>
- α5=<U5+=HSG; F5={HS →R}>

CHƯƠNG 4. NGÔN NGỮ SQL

I. Định nghĩa dữ liệu

- Là ngôn ngữ mô tả

  - Lược đồ cho mỗi quan hệ
  - Miền giá trị tương ứng của từng thuộc tính
  - Ràng buộc toàn vẹn
  - Chỉ mục trên mỗi quan hệ
- Gồm
  - CREATE TABLE (tạo bảng)
  - DROP TABLE (xóa bảng)
  - ALTER TABLE (sửa bảng)
  - CREATE DOMAIN (tạo miền giá trị)
  - CREATE DATABASE
  - …

2. Kiểu dữ liệu

- Số (numeric)
  - INTEGER
  - SMALLINT
  - NUMERIC, NUMERIC(p), NUMERIC(p,s)
  - DECIMAL, DECIMAL(p), DECIMAL(p,s)
  - REAL
  - DOUBLE PRECISION
  - FLOAT, FLOAT(p)
- Chuỗi ký tự (character string)
  - CHARACTER, CHARACTER(n)
  - CHARACTER VARYING(x)
- Chuỗi bit (bit string)
  - BIT, BIT(x)
  - BIT VARYING(x)
- Ngày giờ (datetime)
  - DATE gồm ngày, tháng và năm
  - TIME gồm giờ, phút và giây
  - TIMESTAMP gồm ngày và giờ

3. Lệnh tạo bảng

- Để định nghĩa một bảng
  - Tên bảng
  - Các thuộc tính
    - Tên thuộc tính
    - Kiểu dữ liệu
    - Các ràng buộc toàn vẹn trên thuộc tính (RBTV)
- Cú pháp

CREATE TABLE <Tên_bảng> **(** <Tên_cột> <Kiểu_dữ_liệu> [<RBTV>], <Tên_cột> <Kiểu_dữ_liệu> [<RBTV>], … [<RBTV>] **)**

Ví dụ - Tạo bảng

```
CREATE TABLE NHANVIEN (MANV CHAR(9), HONV VARCHAR(10), TENDEM VARCHAR(20), TENNV VARCHAR(10), NS DATETIME, DCHI VARCHAR(50), GT CHAR(3), LUONG INT, MA_NQL CHAR(9), PHG INT)
```

- <RBTV>
  - NOT NULL
  - NULL
  - UNIQUE
  - DEFAULT
  - PRIMARY KEY
  - FOREIGN KEY / REFERENCES
  - CHECK
- Đặt tên cho RBTV

CONSTRAINT <Ten_RBTV> <RBTV>

Ví dụ - RBTV

```
CREATE TABLE NHANVIEN (HONV VARCHAR(10) NOT NULL,
        TENDEM VARCHAR(20) NOT NULL,
        TENNV VARCHAR(10) NOT NULL,
         MANV CHAR(9) PRIMARY KEY,
        NS DATETIME, DCHI VARCHAR(50),
        GT CHAR(3) CHECK (GT IN ('Nam', 'Nu'),
        LUONG INT DEFAULT (10000),
        MA_NQL CHAR(9),
        PHG INT
)
CREATE TABLE PHONGBAN (
        TENPB VARCHAR(20) UNIQUE,
        MAPHG INT NOT NULL,
        TRPHG CHAR(9),
        NG_NHANCHUC DATETIME DEFAULT (GETDATE())
)
CREATE TABLE PHANCONG (MA_NVIEN CHAR(9) FOREIGN KEY (MA_NVIEN)
        REFERENCES NHANVIEN(MANV),
        SODA INT REFERENCES DEAN(MADA),
        THOIGIAN DECIMAL(3,1)
)
```

Ví dụ - Đặt tên cho RBTV

```
CREATE TABLE NHANVIEN (
        HONV VARCHAR(10) CONSTRAINT NV_HONV_NN NOT NULL,
        TENDEM VARCHAR(20) NOT NULL,
        TENNV VARCHAR(10) NOT NULL,
        MANV CHAR(9) CONSTRAINT NV_MANV_PK PRIMARY KEY,
        NS      DATETIME,
        DCHI VARCHAR(50),
        GT CHAR(3) CONSTRAINT NV_GT_CHK
                CHECK (GT IN ('Nam', 'Nu')),
        LUONG INT CONSTRAINT NV_LUONG_DF DEFAULT (1000000),
        MA_NQL CHAR(9),
        PHG INT
)
CREATE TABLE PHANCONG (
        MA_NVIEN CHAR(9),
        SODA INT,
        THOIGIAN DECIMAL(3,1),
        CONSTRAINT PC_MANVIEN_SODA_PK PRIMARY KEY (MA_NVIEN, SODA),
        CONSTRAINT PC_MANVIEN_FK FOREIGN KEY (MA_NVIEN)
                REFERENCES NHANVIEN(MANV),
        CONSTRAINT PC_SODA_FK FOREIGN KEY (SODA)
                REFERENCES DEAN(MADA)
)
```

4. Lệnh sửa bảng

- Được dùng để
  - Thay đổi cấu trúc bảng
  - Thay đổi RBTV
- Thêm cột

ALTER TABLE <Tên_bảng> **ADD COLUMN**
<Tên cột> <Kiểu dữ liêu> [<RBTV>]

- Xóa cột

ALTER TABLE <Tên_bảng> **DROP COLUMN** <Tên_cột>

- Mở rộng cột

ALTER TABLE <Tên_bảng> **ALTER COLUMN**<Tên_cột> <Kiểu_dữ_liệu_mới>

- Thêm RBTV

ALTER TABLE <Tên_bảng> **ADD** **CONSTRAINT** <Ten_RBTV> <RBTV>,
CONSTRAINT <Ten_RBTV> <RBTV>,
…

- Xóa RBTV

ALTER TABLE <Tên_bảng> **DROP** <Tên_RBTV>

Ví dụ - Thay đổi cấu trúc bảng

```
ALTER TABLE NHANVIEN ADD
        NGHENGHIEP CHAR(20)
ALTER TABLE NHANVIEN DROP COLUMN NGHENGHIEP
ALTER TABLE NHANVIEN ALTER COLUMN
        NGHENGHIEP CHAR(50)
```

Ví dụ - Thay đổi RBTV

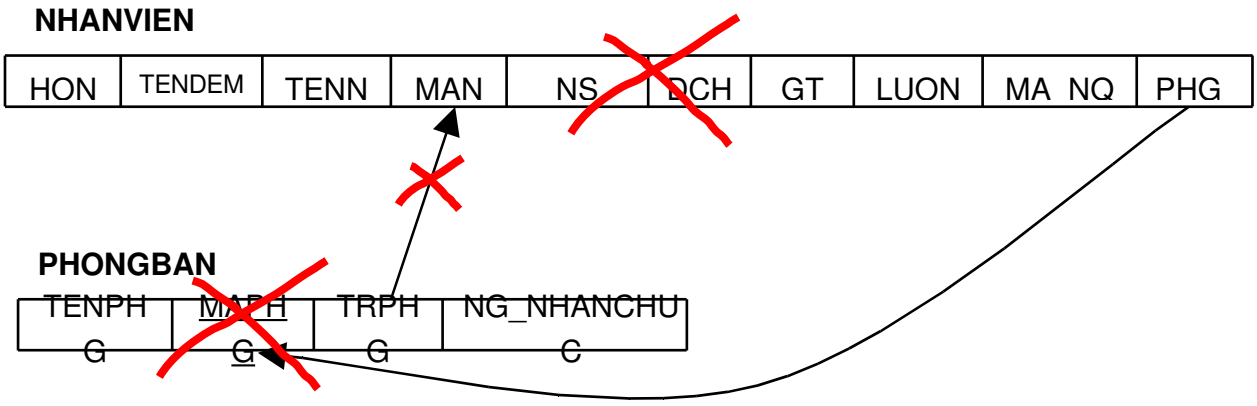
```
CREATE TABLE PHONGBAN (
        TENPB VARCHAR(20),
        MAPHG INT NOT NULL,
        TRPHG CHAR(9),
        NG_NHANCHUC DATETIME
)
ALTER TABLE PHONGBAN ADD
        CONSTRAINT PB_MAPHG_PK PRIMARY KEY (MAPHG),
        CONSTRAINT PB_TRPHG FOREIGN KEY (TRPHG)
            REFERENCES NHANVIEN(MANV),
        CONSTRAINT PB_NGNHANCHUC_DF DEFAULT (GETDATE())
            FOR (NG_NHANCHUC),
        CONSTRAINT PB_TENPB_UNI UNIQUE (TENPB)
```

5. Lệnh xóa bảng

- Được dùng để xóa cấu trúc bảng
  - Tất cả dữ liệu của bảng cũng bị xóa
- Cú pháp

  DROP TABLE <Tên_bảng>
- Ví dụ

```
DROP TABLE NHANVIEN
DROP TABLE PHONGBAN
DROP TABLE PHANCONG
```



6. Lệnh tạo miền giá trị

- Tạo ra một kiểu dữ liệu mới kế thừa những kiểu dữ liệu có sẵn
- Cú pháp

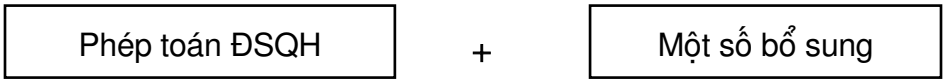
CREATE DOMAIN <Tên_kdl_mới> **AS** <Kiểu_dữ_liệu>

- Ví dụ

CREATE DOMAIN Kieu_Ten AS VARCHAR(30)

II. Truy vấn dữ liệu

- Là ngôn ngữ rút trích dữ liệu thỏa một số điều kiện nào đó
- Dựa trên
-



  - Cho phép 1 bảng có <u>nhiều dòng trùng nhau</u>
  - Bảng là *bag* ≠ quan hệ là *set*

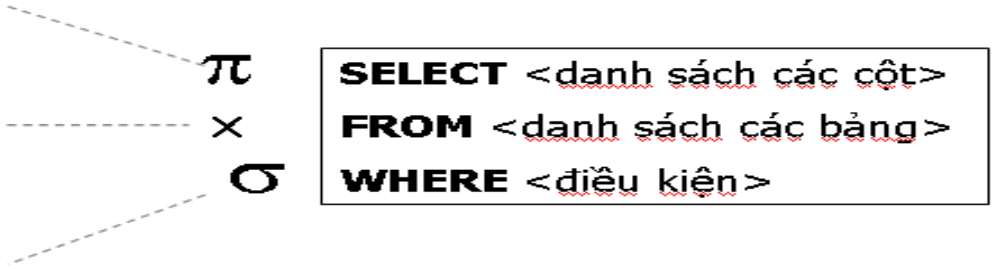
1. Truy vấn cơ bản

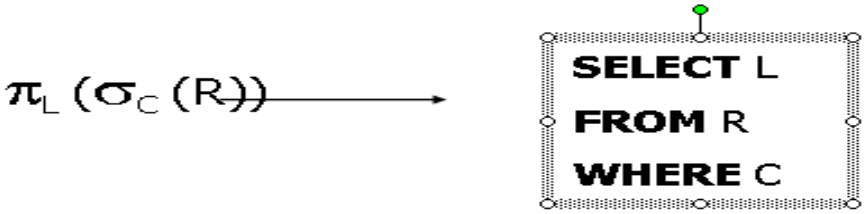
- Gồm 3 mệnh đề

SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <điều kiện>

  - <danh sách các cột>
    - Tên các cột cần được hiển thị trong kết quả truy vấn
  - <danh sách các bảng>
    - Tên các bảng liên quan đến câu truy vấn
  - <điều kiện>
    - Biểu thức boolean xác định dòng nào sẽ được rút trích
    - Nối các biểu thức: AND, OR, và NOT
    - Phép toán: **<** , **>** , **≤** , **≥** , **≠** , **=** , LIKE và BETWEEN

  - SQL và ĐSQH





Ví dụ

Lấy tất cả các cột của quan hệ kết quả

```
SELECT *
FROM NHANVIEN
WHERE PHG=5
```

MANV	HONV	TENDEM	TENNV	NS	DCHI	GT	LUONG	MA_NQL	PHG
33344555	Nguyen	Thanh	Tung	12/08/1955	638 NVC	Nam	40000	88866555	5
98798798	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	33344555	5

1.2. Mệnh đề SELECT

```
SELECT MANV, HONV, TENDEM, TENNV
FROM NHANVIEN
WHERE PHG=5 AND GT='Nam'
```

MANV	HONV	TENDEM	TENNV
333445555	Nguyen	Thanh	Tung
987987987	Nguyen	Manh	Hung

Tên bí danh

```
SELECT MANV, HONV AS HO, TENDEM AS 'TEN DEM', TENNV AS TEN
FROM NHANVIEN
WHERE PHG=5 AND GT='Nam'
```

MANV	HO	TEN DEM	TEN
333445555	Nguyen	Thanh	Tung
987987987	Nguyen	Manh	Hung

Mở rộng

```
SELECT MANV, HONV + ' ' + TENDEM + ' ' + TENNV AS 'HO TEN'
FROM NHANVIEN
WHERE PHG=5 AND GT='Nam'
```

MANV	HO TEN
333445555	Nguyen Thanh Tung
987987987	Nguyen Manh Hung

Mở rộng

```
SELECT MANV, LUONG*1.1 AS 'LUONG10%'
FROM NHANVIEN
WHERE PHG=5 AND GT='Nam'
```

MANV	LUONG10%
333445555	33000
987987987	27500

Loại bỏ các dòng trùng nhau

```
SELECT DISTINCT LUONG
FROM NHANVIEN
WHERE PHG=5 AND GT='Nam'
```

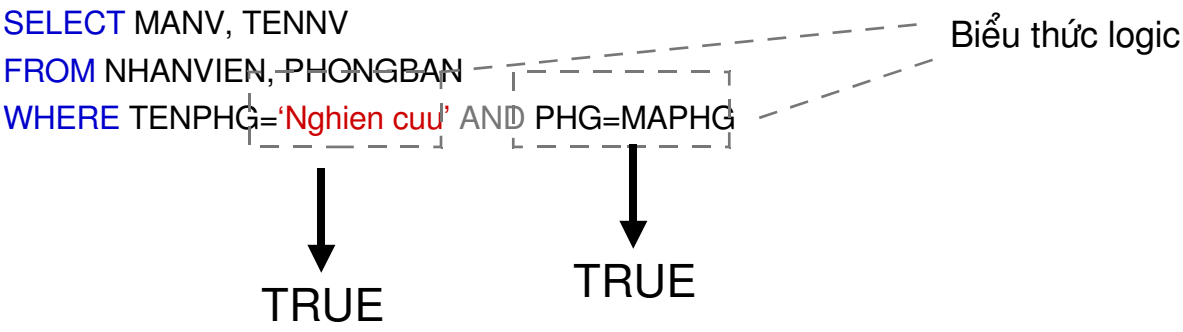
LUONG
30000
25000
38000

- Tốn chi phí
- Người dùng muốn thấy

Ví dụ

- Cho biết MANV và TENNV làm việc ở phòng 'Nghien cuu'

1.3. Mệnh đề WHERE



Độ ưu tiên

```
SELECT MANV, TENNV
FROM NHANVIEN, PHONGBAN
WHERE (TENPHG='Nghien cuu' OR TENPHG='Quan ly') AND    PHG=MAPHG)
```

BETWEE

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE LUONG>20000 AND LUONG<30000
SELECT MANV, TENNV
FROM NHANVIEN
WHERE LUONG BETWEEN 20000 AND 30000
```

NOT BETWEEN

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE LUONG NOT BETWEEN 20000 AND 30000
```

LIKE

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE DCHI LIKE 'Nguyen _ _ _ _'
```

Ký tự bất kỳ

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE DCHI LIKE 'Nguyen %'
```

Chuỗi bất kỳ

NOT LIKE

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE HONV LIKE 'Nguyen'
SELECT MANV, TENNV
FROM NHANVIEN
WHERE HONV NOT NOT LIKE 'Nguyen'
```

ESCAPE

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE DCHI LIKE '% Nguyens_%'  ESCAPE 's'
```

'Nguyen_'

Ngày giờ

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE NGSINH BETWEEN '1955-12-08' AND '1966-07-19'
```

'1955-12-08' YYYY-MM-DD

'12/08/1955' MM/DD/YYYY

'December 8, 1955'

'17:30:00' HH:MI:SS

'05:30 PM'

'1955-12-08 17:30:00'

NULL

- Sử dụng trong trường hợp
  - Không biết (value unknown)
  - Không thể áp dụng (value inapplicable)
  - Không tồn tại (value withheld)
- Những biểu thức tính toán có liên quan đến giá trị NULL sẽ cho ra kết quả là NULL
  - x có giá trị là NULL
  - x + 3 cho ra kết quả là NULL
  - x + 3 là một biểu thức không hợp lệ trong SQL
- Những biểu thức so sánh có liên quan đến giá trị NULL sẽ cho ra kết quả là UNKNOWN
  - x = 3 cho ra kết quả là UNKNOWN

- x = 3 là một so sánh không hợp lệ trong SQL

NULL

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE MA_NQL IS NULL
```

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE MA_NQL IS NOT NULL
```

1.4. Mệnh đề FROM

Không sử dụng mệnh đề WHERE

```
SELECT MANV, MAPHG
FROM NHANVIEN, PHONGBAN
WHERE TRUE
```

MANV	MAPHG
333445555	1
333445555	4
333445555	5
987987987	1
987987987	4
987987987	5

Tên bí danh

```
SELECT TENPHG, DIADIEM
FROM PHONGBAN AS PB, DDIEM_PHG AS DD
WHERE PB.MAPHG=DD.MAPHG
SELECT TENPHG, DIADIEM
FROM PHONGBAN, DDIEM_PHG
WHERE MAPHG=MAPHG
```

```
SELECT TENNV, NV.NGSINH, TENTN, TN.NGSINH
FROM NHANVIEN NV, THANNHAN TN
WHERE MANV=MA_NVIEN
```

```
SELECT TENNV, NGSINH, TENTN, NGSINH
FROM NHANVIEN, THANNHAN
WHERE MANV=MA_NVIEN
```

***Ví dụ 1**: Với những đề án ở 'Ha Noi', cho biết mã đề án, mã phòng ban chủ trì đề án, họ tên trưởng phòng cùng với ngày sinh và địa chỉ của người ấy*
***Ví dụ 2**: Tìm họ tên của nhân viên phòng số 5 có tham gia vào đề án "Sản phẩm X" với số giờ làm việc trên 10 giờ*
***Ví dụ 3**: Tìm họ tên của từng nhân viên và người phụ trách trực tiếp nhân viên đó*
***Ví dụ 4**: Tìm họ tên của những nhân viên được "Nguyen Thanh Tung" phụ trách trực tiếp*

1.5. Mệnh đề ORDER BY

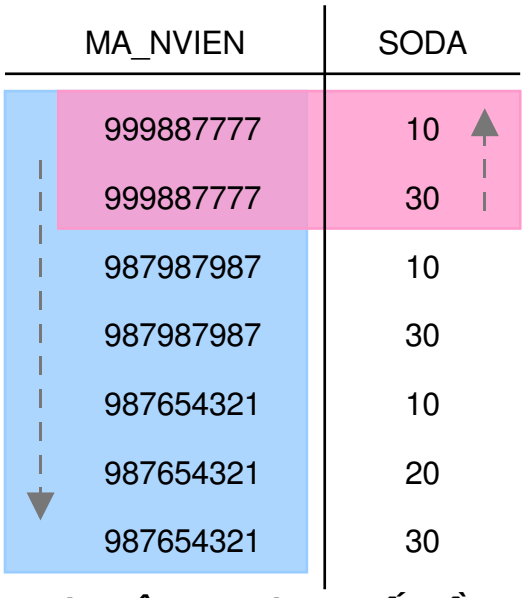
- Dùng để hiển thị kết quả câu truy vấn theo một thứ tự nào đó
- Cú pháp

SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <điều kiện>
ORDER BY <danh sách các cột>

  - ASC: tăng (mặc định)
  - DESC: giảm

– Ví dụ

```
SELECT MA_NVIEN, SODA
FROM PHANCONG
ORDER BY MA_NVIEN DESC, SODA
```



2. Tập hợp, so sánh tập hợp và truy vấn lồng

2.1. Phép toán tập hợp trong SQL

- SQL có cài đặt các phép toán
  - Hợp (UNION)
  - Giao (INTERSECT)
  - Trừ (EXCEPT)
- Kết quả trả về là tập hợp
  - Loại bỏ các bộ trùng nhau
  - Để giữ lại các bộ trùng nhau
    - UNION ALL
    - INTERSECT ALL
    - EXCEPT ALL

Cú pháp

SELECT <ds cột> FROM <ds bảng> WHERE <điều kiện>
UNION [ALL]
SELECT <ds cột> FROM <ds bảng> WHERE <điều kiện>

SELECT <ds cột> FROM <ds bảng> WHERE <điều kiện>
INTERSECT [ALL]
SELECT <ds cột> FROM <ds bảng> WHERE <điều kiện>

SELECT <ds cột> FROM <ds bảng> WHERE <điều kiện>
EXCEPT [ALL]
SELECT <ds cột> FROM <ds bảng> WHERE <điều kiện>

Ví dụ 5*: Cho biết các mã đề án có*

- *Nhân viên với họ là 'Nguyen' tham gia hoặc,*
- *Trưởng phòng chủ trì đề án đó với họ là 'Nguyen'*

Ví dụ 6: *Tìm nhân viên có người thân cùng tên và cùng giới tính*

Ví dụ 7: *Tìm những nhân viên không có thân nhân nào*

2.2 Truy vấn lồng

SELECT MANV, TENNV
FROM NHANVIEN, PHONGBAN
WHERE TENPHG='Nghien cuu' AND PHG=MAPHG

Câu truy vấn ngoài
(Outer query)

SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <so sánh tập hợp> (
    SELECT <danh sách các cột>
    FROM <danh sách các bảng>
    WHERE <điều kiện>)

Câu truy vấn trong
(Subquery)

- Các câu lệnh SELECT có thể lồng nhau ở nhiều mức
- Câu truy vấn con thường trả về một tập các giá trị
- Các câu truy vấn trong trong cùng một mệnh đề WHERE được kết hợp bằng phép nối logic
- Mệnh đề WHERE của câu truy vấn ngoài
  - <biểu thức> <so sánh tập hợp> <truy vấn con>
  - So sánh tập hợp thường đi cùng với một số toán tử
    - IN, NOT IN
    - ALL
    - ANY hoặc SOME
  - Kiểm tra sự tồn tại
    - EXISTS
    - NOT EXISTS
- Có 2 loại truy vấn lồng
  - Lồng phân cấp
    - Mệnh đề WHERE của truy vấn trong không tham chiếu đến thuộc tính của các quan hệ trong mệnh đề FROM ở truy vấn ngoài
    - Khi thực hiện, câu truy vấn trong sẽ được thực hiện trước
  - Lồng tương quan
    - Mệnh đề WHERE của truy vấn trong tham chiếu ít nhất một thuộc tính của các quan hệ trong mệnh đề FROM ở truy vấn ngoài
    - Khi thực hiện, câu truy vấn trong sẽ được thực hiện nhiều lần, mỗi lần tương ứng với một bộ của truy vấn ngoài

Ví dụ - Lồng phân cấp

SELECT MANV, TENNV
FROM NHANVIEN, DIADIEM_PHG

WHERE DIADIEM='TP HCM' AND PHG=MAPHG

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE PHG IN (1, 5)
```

```
(
SELECT MAPHG
FROM DIADIEM_PHG
WHERE DIADIEM='TP HCM' )
```

Ví dụ 7: *Tìm những nhân viên không có thân nhân nào*
Ví dụ 8: *Tìm những nhân viên có lương lớn hơn lương của ít nhất một nhân viên phòng 4*
Ví dụ 9: *Tìm những nhân viên có lương lớn hơn lương của tất cả nhân viên phòng 4*
Ví dụ 10: *Tìm những trưởng phòng có tối thiểu một thân nhân*

Ví dụ - Lồng tương quan

SELECT MANV, TENNV
FROM NHANVIEN, PHONGBAN
WHERE TENPHG='Nghien cuu' AND PHG=MAPHG

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE EXISTS (
    SELECT *
    FROM PHONGBAN
    WHERE TENPHG='Nghien cuu' AND PHG=MAPHG )
```

Ví dụ 6: *Tìm nhân viên có người thân cùng tên và cùng giới tính*
Ví dụ 7: *Tìm những nhân viên không có thân nhân nào*
Ví dụ 8: *Tìm những nhân viên có lương lớn hơn lương của ít nhất một nhân viên phòng 4*
Ví dụ 10: *Tìm những trưởng phòng có tối thiểu một thân nhân*

2.3. Nhận xét IN và EXISTS

- IN
  - <tên cột> IN <câu truy vấn trong>
  - Thuộc tính ở mệnh đề SELECT của truy vấn trong phải có cùng kiểu dữ liệu với thuộc tính ở mệnh đề WHERE của truy vấn ngoài
- EXISTS
  - Không cần có thuộc tính, hằng số hay biểu thức nào khác đứng trước

- Không nhất thiết liệt kê tên thuộc tính ở mệnh đề SELECT của truy vấn trong
- Những câu truy vấn có = ANY hay IN đều có thể chuyển thành câu truy vấn có EXISTS

2.4 Phép chia trong SQL

R	A	B	C	D	E
	α	a	α	a	1
	α	a	γ	a	1
	α	a	γ	b	1
	β	a	γ	a	1
	β	a	γ	b	3
	γ	a	γ	a	1
	γ	a	γ	b	1
	γ	a	β	b	1

S	D	E
b_i	a	1
	b	1

R÷ S	A	B	C
a_i	α	a	γ
	γ	a	γ

- R÷ S là tập các giá trị ai trong R sao cho <u>không có</u> giá trị bi nào trong S làm cho bộ (ai, bi) <u>không tồn tại</u> trong R
- Sử dụng NOT EXISTS để biểu diễn

```
SELECT R1.A, R1.B, R1.C
FROM R, R1
WHERE NOT EXISTS (
        SELECT *
        FROM S
        WHERE NOT EXISTS (
            SELECT *
            FROM R R2
            WHERE R2.D=S.D AND R2.E=S.E
            AND R1.A=R2.A AND R1.B=R2.B AND R1.C=R2.C ))
```

***Ví dụ 11**: Tìm tên các nhân viên được phân công làm <u>tất cả</u> các đồ án*

- *Tìm tên các nhân viên mà <u>không</u> có đề án nào là <u>không</u> được phân công làm*
- *Tập bị chia: PHANCONG(MA_NVIEN, SODA)*
- *Tập chia: DEAN(MADA)*
- *Tập kết quả: KQ(MA_NVIEN)*
- *Kết KQ với NHANVIEN để lấy ra TENNV*
- *Tìm tên các nhân viên được phân công làm <u>tất cả</u> các đồ án*

3. Hàm kết hợp và gom nhóm

3.1. Hàm kết hợp

- COUNT
  - COUNT(*) đếm số dòng
  - COUNT(<tên thuộc tính>) đếm số giá trị khác NULL của thuộc tính

  - COUNT(DISTINCT <tên thuộc tính>) đếm số giá trị khác nhau và khác NULL của thuộc tính
- MIN
- MAX
- SUM
- AVG
- Các hàm kết hợp được đặt ở mệnh đề SELECT

Ví dụ 12: *Tìm tổng lương, lương cao nhất, lương thấp nhất và lương trung bình của các nhân viên*

Ví dụ 13: *Cho biết số lượng nhân viên của phòng 'Nghien cuu'*

Ví dụ 14: *Cho biết số lượng nhân viên của từng phòng ban*

PHG	SL_NV
5	3
4	3
1	1

MANV	HONV	TENLO	TENNV	NGSINH	DCHI	PHAI	LUONG	MA_NQL	PHG
33344555	Nguyen	Thanh	Tung	12/08/1955	638 NVC	Nam	40000	88866555	5
98798798	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	33344555	5
45345345	Tran	Thanh	Tam	07/31/1972	543 MTL	Nu	25000	33344555	5
99988777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	98765432	4
98765432	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	88866555	4
98798798	Tran	Hong	Quang	04/08/1969	980 LHP	Nam	25000	98765432	4
88866555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

3.2. Gom nhóm

- Cú pháp

SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <điều kiện>
GROUP BY <danh sách các cột gom nhóm>

- Sau khi gom nhóm
  - Mỗi nhóm các bộ sẽ có cùng giá trị tại các thuộc tính gom nhóm

Ví dụ 14: *Cho biết số lượng nhân viên của từng phòng ban*

Ví dụ 15: *Với mỗi nhân viên cho biết mã số, họ tên, số lượng đề án và tổng thời gian mà họ tham gia*

MA_NVIEN	SODA	THOIGIAN
123456789	1	32.5
123456789	2	7.5
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
888665555	20	20.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
453453453	1	20.0
453453453	2	20.0

Ví dụ 16: *Cho biết những nhân viên tham gia từ 2 đề án trở lên*

MA_NVIE	SOD	THOIGIAN
12345678	1	32.5
12345678	2	7.5
33344555	2	10.0
33344555	3	10.0
33344555	10	10.0
88866555	20	20.0
98798798	10	35.0
98798798	30	5.0
98765432	30	20.0
98765432	20	15.0
45345345	1	20.0
45345345	2	20.0

bị loại ra

3.3. Điều kiện trên nhóm

- Cú pháp

SELECT <danh sách các cột>
FROM <danh sách các bảng>
WHERE <điều kiện>
GROUP BY <danh sách các cột gom nhóm>
HAVING <điều kiện trên nhóm>

***Ví dụ 16**: Cho biết những nhân viên tham gia từ 2 đề án trở lên*

***Ví dụ 17**: Cho biết những phòng ban (TENPHG) có lương trung bình của các nhân viên lớn hơn 2tr*

**** Nhận xét***

- Mệnh đề GROUP BY
  - Các thuộc tính trong mệnh đề SELECT (trừ những thuộc tính trong các hàm kết hợp) phải xuất hiện trong mệnh đề GROUP BY
- Mệnh đề HAVING
  - Sử dụng các hàm kết hợp trong mệnh đề SELECT để kiểm tra một số điều kiện nào đó
  - Chỉ kiểm tra điều kiện trên nhóm, không là điều kiện lọc trên từng bộ
  - Sau khi gom nhóm điều kiện trên nhóm mới được thực hiện
- Thứ tự thực hiện câu truy vấn có mệnh đề GROUP BY và HAVING
  - (1) Chọn ra những dòng thỏa điều kiện trong mệnh đề WHERE

- (2) Những dòng này sẽ được gom thành nhiều nhóm tương ứng với mệnh đề GROUP BY
- (3) Áp dụng các hàm kết hợp cho mỗi nhóm
- (4) Bỏ qua những nhóm không thỏa điều kiện trong mệnh đề HAVING
- (5) Rút trích các giá trị của các cột và hàm kết hợp trong mệnh đề SELECT

Ví dụ 18: *Tìm những phòng ban có lương trung bình cao nhất*

Ví dụ 19*: Tìm 3 nhân viên có lương cao nhất*

Ví dụ 12: *Tìm tên các nhân viên được phân công làm tất cả các đồ án*

4. Một số dạng truy vấn khác

- Truy vấn con ở mệnh đề FROM
- Điều kiện kết ở mệnh đề FROM
  - Phép kết tự nhiên
  - Phép kết ngoàI
- Cấu trúc CASE

4.1. Truy vấn con ở mệnh đề FROM

- Kết quả trả về của một câu truy vấn phụ là một bảng
  - Bảng trung gian trong quá trình truy vấn
  - Không có lưu trữ thật sự
- Cú pháp

  SELECT <danh sách các cột>
  FROM R1, R2, **(**<truy vấn con>**) AS** tên_bảng
  WHERE <điều kiện>

4.2. Điều kiện kết ở mệnh đề FROM

- Kết bằng

  SELECT <danh sách các cột>
  FROM R1 [**INNER**] **JOIN** R2 **ON** <biểu thức>
  WHERE <điều kiện>
- Kết ngoài

  SELECT <danh sách các cột>
  FROM R1 **LEFT**|**RIGHT** [**OUTER**] **JOIN** R2 **ON** <biểu thức>
  WHERE <điều kiện>

Ví dụ 20: *Tìm mã và tên các nhân viên làm việc tại phòng 'Nghien cuu'*

Ví dụ 21: *Tìm họ tên các nhân viên và tên các đề án nhân viên tham gia nếu có*

4.3 Cấu trúc CASE

- Cho phép kiểm tra điều kiện và xuất thông tin theo từng trường hợp
- Cú pháp

  CASE <tên cột>
  WHEN <giá trị> **THEN** <biểu thức>
  WHEN <giá trị> **THEN** <biểu thức>
  …
  [**ELSE** <biểu thức>]

END

Ví dụ 22: *Cho biết họ tên các nhân viên đã đến tuổi về hưu (nam 60 tuổi, nữ 55 tuổi)*

Ví dụ 23: *Cho biết họ tên các nhân viên và năm về hưu*

***** Kết luận***

SELECT <danh sách các cột>
FROM <danh sách các bảng>
[**WHERE** <điều kiện>]
[**GROUP BY** <các thuộc tính gom nhóm>]
[**HAVING** <điều kiện trên nhóm>]
[**ORDER BY** <các thuộc tính sắp thứ tự>]

5. Cập nhật dữ liệu

5.1. Lệnh INSERT

- Dùng để thêm 1 hay nhiều dòng vào bảng
- Để thêm dữ liệu
  - Tên quan hệ
  - Danh sách các thuộc tính cần thêm dữ liệu
  - Danh sách các giá trị tương ứng

Cú pháp (thêm 1 dòng): **INSERT INTO** <tên bảng>(<danh sách các thuộc tính>)
VALUES (<danh sách các giá trị>)

Ví dụ

```
INSERT INTO NHANVIEN(HONV, TENDEM, TENNV, MANV)
VALUES ('Le', 'Van', 'Tuyen', '635635635')

INSERT INTO NHANVIEN(HONV, TENDEM, TENNV, MANV, DCHI)
VALUES ('Le', 'Van', 'Tuyen', '635635635', NULL)

INSERT INTO NHANVIEN
VALUES ('Le', 'Van', 'Tuyen', '635635635', '12/30/1952', '98 HV', 'Nam', '37000', 4)
```

- **Nhận xét**
  - Thứ tự các giá trị phải trùng với thứ tự các cột
  - Có thể thêm giá trị NULL ở những thuộc tính không là khóa chính và NOT NULL
  - Câu lệnh INSERT sẽ gặp lỗi nếu vi phạm RBTV
    - Khóa chính
    - Tham chiếu
    - NOT NULL - các thuộc tính có ràng buộc NOT NULL bắt buộc phải có giá trị

Cú pháp (thêm nhiều dòng): **INSERT INTO** <tên bảng>(<danh sách các thuộc tính>)
<câu truy vấn con>

Ví dụ

```
CREATE TABLE THONGKE_PB (
        TENPHG VARCHAR(20),
        SL_NV INT,
        LUONG_TC INT
)
```

```
INSERT INTO THONGKE_PB(TENPHG, SL_NV, LUONG_TC)
    SELECT TENPHG, COUNT(MANV), SUM(LUONG)
    FROM NHANVIEN, PHONGBAN
    WHERE PHG=MAPHG
    GROUP BY TENPHG
```

5.2. Lệnh DELETE

- Dùng để xóa các dòng của bảng
- Cú pháp

DELETE FROM <tên bảng>
[**WHERE** <điều kiện>]

Ví dụ

```
DELETE FROM NHANVIEN
WHERE HONV='Tran'

DELETE FROM NHANVIEN
WHERE MANV='345345345'

DELETE FROM NHANVIEN
```

- **Nhận xét**
  - Số lượng số dòng bị xóa phụ thuộc vào điều kiện ở mệnh đề WHERE
  - Nếu không chỉ định điều kiện ở mệnh đề WHERE, tất cả các dòng trong bảng sẽ bị xóa
  - Lệnh DELETE có thể gây ra vi phạm RB tham chiếu
    - Không cho xóa
    - Xóa luôn những dòng có giá trị đang tham chiếu đến
      - CASCADE
    - Đặt NULL cho những giá trị tham chiếu

MANV	HONV	TENLO	TENNV	NGSINH	DCHI	PHAI	LUONG	MA_NQL	PHG
33344555	Nguyen	Thanh	Tung	12/08/1955	638 NVC	Nam	40000	88866555	5
98798798	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	33344555	5
45345345	Tran	Thanh	Tam	07/31/1972	543 MTL	Nu	25000	33344555	5
99988777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	98765432	4
98765432	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	88866555	4
98798798	Tran	Hong	Quang	04/08/1969	980 LHP	Nam	25000	98765432	4
88866555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

MA_NVIE	SOD	THOIGIAN
33344555	10	10.0
88866555	20	20.0
98798798	10	35.0
98798798	30	5.0
98765432	30	20.0
45345345	1	20.0

TENPHG	MAPH	MA_NVIEN	NG_NHANCHU
Nghien cuu	5	333445555	05/22/1988
Dieu hanh	4	987987987	01/01/1995
Quan ly	1	888665555	06/19/1981

MANV	HONV	TENLOT	TENNV	NGSINH	DCHI	PHAI	LUONG	MA_NQL	PHG
33344555	Nguyen	Thanh	Tung	12/08/1955	638 NVC	Nam	40000	88866555	NULL
98798798	Nguyen	Manh	Hung	09/15/1962	Ba Ria VT	Nam	38000	33344555	NULL
45345345	Tran	Thanh	Tam	07/31/1972	543 MTL	Nu	25000	33344555	NULL
99988777	Bui	Ngoc	Hang	07/19/1968	33 NTH Q1	Nu	38000	98765432	4
98765432	Le	Quynh	Nhu	07620/1951	219 TD Q3	Nu	43000	88866555	4
98798798	Tran	Hong	Quang	04/08/1969	980 LHP	Nam	25000	98765432	4
88866555	Pham	Van	Vinh	11/10/1945	450 TV HN	Nam	55000	NULL	1

5.3. Lệnh UPDATE

- Dùng để thay đổi giá trị của thuộc tính cho các dòng của bảng
- Cú pháp

UPDATE <tên bảng>
SET <tên thuộc tính>=<giá trị mới>,
<tên thuộc tính>=<giá trị mới>,
…
[**WHERE** <điều kiện>]

Ví dụ
UPDATE NHANVIEN
SET NGSINH='08/12/1965'
WHERE MANV='333445555'

UPDATE NHANVIEN
SET LUONG=LUONG*1.1
Ví dụ 25: *Với đề án có mã số 10, hãy thay đổi nơi thực hiện đề án thành 'Vung Tau' và phòng ban phụ trách là phòng 5*
UPDATE DEAN
SET DIADIEM_DA='Vung Tau', PHONG=5
WHERE MADA=10

- ***Nhận xét***
  - Những dòng thỏa điều kiện tại mệnh đề WHERE sẽ được cập nhật giá trị mới
  - Nếu không chỉ định điều kiện ở mệnh đề WHERE, tất cả các dòng trong bảng sẽ bị cập nhật
  - Lệnh UPDATE có thể gây ra vi phạm RB tham chiếu
    - Không cho sửa
    - Sửa luôn những dòng có giá trị đang tham chiếu đến
      - CASCADE

6. Khung nhìn (view)

6.1. Khung nhìn

- Bảng là một quan hệ được tổ chức lưu trữ vật lý trong CSDL
- Khung nhìn cũng là một quan hệ
  - Không được lưu trữ vật lý (bảng ảo)
  - Không chứa dữ liệu
  - Được định nghĩa từ những bảng khác
  - Có thể truy vấn hay cập nhật thông qua khung nhìn
- Tại sao phải sử dụng khung nhìn?
  - Che dấu tính phức tạp của dữ liệu
  - Đơn giản hóa các câu truy vấn
  - Hiển thị dữ liệu dưới dạng tiện dụng nhất
  - An toàn dữ liệu

6.2. Định nghĩa khung nhìn

- Cú pháp

  CREATE VIEW <tên khung nhìn> **AS**
  <câu truy vấn>
  DROP VIEW <tên khung nhìn>
- Bảng ảo này có
  - Danh sách thuộc tính trùng với các thuộc tính trong mệnh đề SELECT
  - Số dòng phụ thuộc vào điều kiện ở mệnh đề WHERE

– Dữ liệu được lấy từ các bảng ở mệnh đề FROM

Ví dụ

```
CREATE VIEW NV_P5 AS
        SELECT MANV, HONV, TENDEM, TENVN
        FROM NHANVIEN
        WHERE PHG=5

CREATE VIEW TONGLNG_SLNV_PB AS
        SELECT MAPHG, TENPB, COUNT(*) AS SLNV,
        SUM(LUONG) AS TONGLNG
        FROM NHANVIEN, PHONGBAN
        WHERE PHG=MAPHG
        GROUP BY TENPHG
```

6.3 Truy vấn trên khung nhìn

- Tuy không chứa dữ liệu nhưng có thể thực hiện các câu truy vấn trên khung nhìn

```
SELECT TENNV
FROM NV_P5
WHERE HONV LIKE 'Nguyen'
```

NV_P5 ←πMANV,HONV, TENDEM, TENNV (σPHG=5 (NHANVIEN))

π TENNV (σHONV='Nguyen' (NV_P5))

- Có thể viết câu truy vấn dữ liệu từ khung nhìn và bảng

```
SELECT HONV, TENVN, TENDA, THOIGIAN
FROM NV_P5, PHANCONG, DEAN
WHERE MANV=MA_NVIEN AND SODA=MADA
```

NV_P5 ←πMANV,HONV, TENDEM, TENNV (σPHG=5 (NHANVIEN))

TMP ←NV_P5 MANV=MA_NVIEN PHONGBAN SODA=MADADEAN

πTENNV,TENDA,THOIGIAN(TMP)

6.4. Cập nhật trên khung nhìn

- Có thể dùng các câu lệnh INSERT, DELETE và UPDATE cho các <u>khung nhìn đơn giản</u>
  - Khung nhìn được xây dựng trên 1 bảng và có khóa chính của bảng
- Không thể cập nhật dữ liệu nếu
  - Khung nhìn có dùng từ khóa DISTINCT
  - Khung nhìn có sử dụng các hàm kết hợp
  - Khung nhìn có mệnh đề SELECT mở rộng
  - Khung nhìn được xây dựng từ bảng có RB trên cột
  - Khung nhìn được xây dựng từ nhiều bảng
- Sửa lại họ cho nhân viên mã '123456789' ở phòng 5 là 'Pham'

```
UPDATE NV_P5
SET HONV='Pham'
WHERE MANV= '123456789'
```

7. Chỉ mục (index)

- Chỉ mục trên thuộc tính A là một cấu trúc dữ liệu làm cho việc tìm kiếm mẫu tin có chứa A hiệu quả hơn

```
SELECT *
FROM NHANVIEN
WHERE PHG=5 AND  GT='Nu'
```

Đọc 10.000 bộ

Đọc 200 bộ

Đọc 70 bộ

Bảng NHANVIEN có 10.000 bộ

Có 200 nhân viên làm việc cho phòng 5

- Cú pháp

  CREATE INDEX <tên chỉ mục> **ON** <tên bảng>(<tên cột>)

  DROP INDEX <tên chỉ mục>

- Ví dụ

  CREATE INDEX PHG_IND ON NHANVIEN(PHG)

  CREATE INDEX PHG_GT_IND ON NHANVIEN(PHG, GT)

- ***Nhận xét***
  - Tìm kiếm nhanh trong trường hợp so sánh với hằng số và phép kết
  - Làm chậm đi các thao tác thêm, xóa và sửa
  - Tốn chi phí
    - Lưu trữ chỉ mục
    - Truy xuất đĩa nhiều
- Chọn lựa cài đặt chỉ mục hợp lý???

Ví dụ

- Xét quan hệ
  - PHANCONG(MA_NVIEN, SODA, THOIGIAN)
- Giả sử
  - PHANCONG được lưu trữ trong 10 block
    - Chi phí để đọc toàn bộ dữ liệu của PHANCONG là 10
  - Trung bình một nhân viên tham gia 3 đề án và một đề án có khoảng 3 nhân viên làm
    - Dữ liệu được trải đều trong 10 block
    - Chi phí để tìm một nhân viên hay một đề án là 3
  - Khi sử dụng chỉ mục
    - Chi phí đọc hay cập nhật chỉ mục
  - Thao tác thêm cần 2 lần truy xuất đĩa
- Giả sử có 3 thao tác được thực hiện thường xuyên
  - Q1: SELECT SODA, THOIGIAN
    FROM PHANCONG
    WHERE MA_NVIEN='123456789'

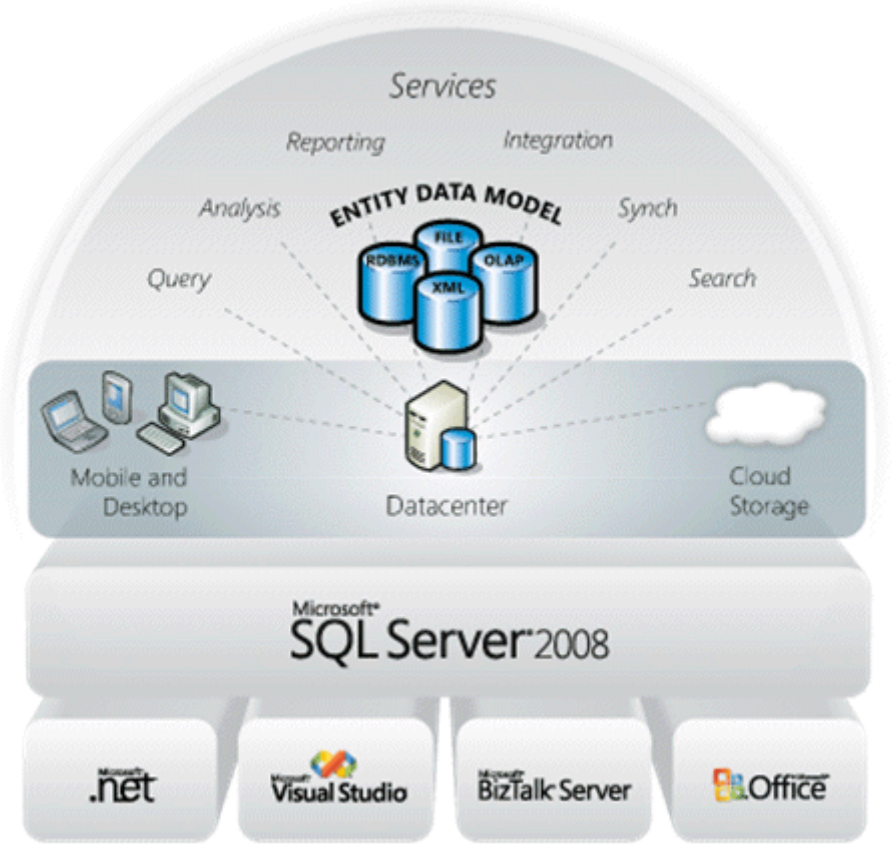
– Q2: SELECT MANV
FROM PHANCONG
WHERE SODA=1 AND THOIGIAN=20.5
– Q3: INSERT INTO PHANCONG
VALUES (123456789', 1, 20.5)

- Bảng so sánh chi phí

Thao tác	Không có chỉ mục	Chỉ mục trên MA_NVIEN	Chỉ mục trên SODA	Chỉ mục trên cả 2 thuộc tính
Q1	10	4	10	4
Q2	10	10	4	4
Q3	2	4	4	6
Chí phí TB	2 + 8p1 + 8p2	4 + 6p2	4 + 6p1	6 - 2p1 – 2p2

Khoảng thời gian thực hiện Q1 là p1
Khoảng thời gian thực hiện Q2 là p2
Khoảng thời gian thực hiện Q3 là 1 - p1 - p2

Giáo trình “Các hệ quản trị cơ sở dữ liệu



Ch□□ng 1 TỔNG QUAN CƠ SỞ DỮ LIỆU ACCESS

1. GIỚI THIỆU MICROSOFT ACCESS

Microsoft Access là một Hệ Quản Trị Cơ Sở Dữ Liệu (QTCSDL) tương tác người sử dụng chạy trong môi trường Windows. Microsoft Access cho chúng ta một công cụ hiệu lực và đầy sức mạnh trong công tác tổ chức, tìm kiếm và biểu diễn thông tin.

Microsoft Access cho ta các khả năng thao tác dữ liệu, khả năng liên kết và công cụ truy vấn mạnh mẽ giúp quá trình tìm kiếm thông tin nhanh. Người sử dụng có thể chỉ dùng một truy vấn để làm việc với các dạng cơ sở dữ liệu khác nhau. Ngoài ra, có thể thay đổi truy vấn bất kỳ lúc nào và xem nhiều cách hiển thị dữ liệu khác nhau chỉ cần động tác nhấp chuột.

Microsoft Access và khả năng kết xuất dữ liệu cho phép người sử dụng thiết kế những biểu mẫu và báo cáo phức tạp đáp ứng đầy đủ các yêu cầu quản lý, có thể vận động dữ liệu và kết hợp các biểu mẫu va báo cáo trong một tài liệu và trình bày kết quả theo dạng thức chuyên nghiệp.

Microsoft Access là một công cụ đầy năng lực để nâng cao hiệu suất công việc. Bằng cách dùng các Wizard của MS Access và các lệnh có sẵn (macro) ta có thể dễ dàng tự động hóa công việc mà không cần lập trình. Đối với những nhu cầu quản lý cao, Access đưa ra ngôn ngữ lập trình Access Basic (Visual Basic For application) một ngôn ngữ lập trình mạnh trên CSDL.

2. KHỞI ĐỘNG VÀ THOÁT KHỎI ACCESS

2.1. Khởi động ACCESS

Chọn nút Start trên thanh Task bar

Chọn Programs

Chọn Microsoft ACCESS

Khung hội thoại Microsoft ACCESS gồm:

Create a New Database Using : Tạo CSDL ứng dụng mới.

Blank Database : Tạo CSDL trống.

Database Wizard : Tạo với sự trợ giúp của Wizard.

Open an Existing Database : Mở một CSDL có sẵn.

2.2. Thoát khỏi ACCESS

Chọn File/Exit hoặc nhấn tổ hợp phím ALT+F4

3. CÁC THAO TÁC TRÊN TẬP TIN CƠ SỞ DỮ LIỆU ACCESS

3.1. Tạo một tập tin CSDL

Thực hiện các thao tác sau:

Chọn File/New hoặc chọn biểu tượng New trên thanh công cụ

Chọn Database, chọn OK

Trong mục *Save in:* Chọn thư mục cần chứa tên tập tin.

File name: Chọn tên tập tin cần tạo (Phần mở rộng mặc định là MDB)

3.2. Mở một CSDL đã tồn tại trên đĩa

Chọn File/Open database (Hoặc click biểu tượng Open)

Trong mục *Look in :* Chọn thư mục cần chứa tên tập tin cần mở.

File name: Chọn tên tập tin cần mở.

Chọn Open

3.3. Đóng một CSDL

Chọn File/Close hoặc ALT+F4

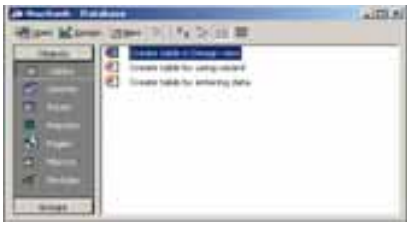
3.4. Các thành phần cơ bản của một tập tin CSDL ACCESS

Một tập tin CSDL ACCESS gồm có 6 thành phần cơ bản sau

Bảng (Tables) : Là nơi chứa dữ liệu

Truy vấn (Queries) : Truy vấn thông tin dựa trên một hoặc nhiều bảng.

Biểu mẫu (Forms) : Các biểu mẫu dùng để nhập dữ liệu hoặc hiển thị dữ liệu.



Báo cáo (Reports) : Dùng để in ấn.

Pages (Trang) : Tạo trang dữ liệu.

Macros (Tập lệnh) : Thực hiện các tập lệnh.

Modules (Đơn thể) : Dùng để lập trình Access Basic

4. CÁCH SỬ DỤNG CỬA SỔ DATABASE

Như đã nói ở trên, một CSDL của Access chứa trong nó 7 đối tượng chứ không đơn thuần là bảng dữ liệu. Sau khi tạo mới một CSDL hoặc mở một CSDL có sẵn Access sẽ hiển thị một cửa sổ Database, trên đó hiển thị tên của CSDL đang mở và liệt kê 7 đối tượng mà nó quản lý, mỗi lớp đối tượng đều được phân lớp rõ ràng để tiện theo dõi.

4.1. Tạo một đối tượng mới

Trong cửa sổ Database, chọn tab chứa đối tượng cần tạo (Bảng, Truy vấn, Biểu mẫu, Báo cáo,...) hoặc thực hiện lệnh

View/D atabase Object - Table/Query/Form/ReportPages/Macros/Modules

Chọn nút New.

4.2. Thực hiện một đối tượng trong CSDL

Trong cửa sổ Database, chọn tab cần thực hiện. Cửa sổ Database liệt kê tên các đối tượng có sẵn, chọn tên đối tượng cần mở.

Chọn nút Open (đối với Bảng, Truy vấn, Biểu mẫu, Trang) hoặc Preview (đối với Báo biểu) hoặc Run (đối với Macro và Module).

4.3. Sửa đổi một đối tượng có sẵn trong CSDL

Trong cửa sổ Database, chọn tab cần thực hiện. Cửa sổ Database liệt kê tên các đối tượng có sẵn, chọn tên đối tượng cần mở, Chọn nút Design.

Ch□□ng 2 BẢNG DỮ LIỆU

Bảng là đối tượng chủ yếu chứa các thông tin cần quản lý, có thể đó chỉ là một vài địa chỉ đơn giản hay cả vài chục nghìn bản ghi chứa đựng thông tin liên quan đến các hoạt động SXKD của một công ty xuất nhập khẩu nào đó. Trước khi ta muốn làm việc với bất kỳ một CSDL nào thì ta phải có thông tin để quản lý, các thông tin đó nằm trong các bảng, nó là cơ sở để cho người sử dụng tạo các đối tượng khác trong CSDL như truy vấn, biểu mẫu, báo biểu...

1. THIẾT KẾ CƠ SỞ DỮ LIỆU

Một CSDL được thiết kế tốt cho phép người sử dụng truy cập nhanh chóng đến những thông tin cần tham khảo, giúp tiết kiệm được thời gian truy xuất thông tin. Một CSDL thiết kế tốt giúp người sử dụng rút ra được những kết quả nhanh chóng và chính xác hơn.

Để thiết kế một CSDL tốt chúng ta phải hiểu cách mà một Hệ QTCSDL quản trị các CSDL như thế nào. MS Access hay bất kỳ một Hệ QTCSDL nào có thể cung cấp các thông tin cho chúng ta một cách chính xác và hiệu quả nếu chúng được cung cấp đầy đủ mọi dữ kiện về nhiều đối tượng khác nhau lưu trữ trong các bảng dữ liệu. Ví dụ ta cần một bảng để chứa thông tin về lý lịch của cán bộ, một bảng khác để chứa các đề tài nguyên cứu khoa học của các cán bộ...

Khi bắt tay thiết kế CSDL, chúng ta phải xác định và phân tích các thông tin muốn lưu trữ thành các đối tượng riêng rẽ, sau đó báo cho Hệ QTCSDL biết các đối tượng đó liên quan với nhau như thế nào. Dựa vào các quan hệ đó mà Hệ QTCSDL có thể liên kết các đối tượng và rút ra các số liệu tổng hợp cần thiết.

CÁC BƯỚC THIẾT KẾ CSDL

Bước 1: Xác định mục tiêu khai thác CSDL của chúng ta. Điều này quyết định các loại sự kiện chúng ta sẽ đưa vào MS Access.

Bước 2: Xác định các bảng dữ liệu cần thiết. Mỗi đối tượng thông tin sẽ hình thành một bảng trong CSDL của chúng ta.

Bước 3: Sau khi đã xác định xong các bảng cần thiết, tiếp đến ta phải chỉ rõ thông tin nào cần quản lý trong mỗi bảng, đó là xác định các trường. Mỗi loại thông tin trong bảng gọi là trường. Mọi mẫu in trong cùng một bảng đều có chung cấu trúc các trường. Ví dụ: Trong lý lịch khoa học cán bộ, những trường (thông tin) cần quản lý là: "HỌ VÀ TÊN", "CHUYÊN MÔN", "HỌC VỊ", "HỌC HÀM",...

Bước 4: Xác định các mối quan hệ giữa các bảng. Nhìn vào mỗi bảng dữ liệu và xem xét dữ liệu trong bảng này liên hệ thế nào với dữ liệu trong bảng khác. Thêm trướng hoặc tạo bảng mới để làm rõ mối quan hệ này. Đây là vấn đề hết sức quan trọng, tạo được quan hệ tốt sẽ giúp chúng ta nhanh chóng truy tìm tìm và kết xuất dữ liệu.

Bước 5: Tinh chế, hiệu chỉnh lại thiết kế. Phân tích lại thiết kế ban đầu để tim lỗi, tạo bảng dữ liệu và nhập vào vài bản ghi, thử xem CSDL đó phản ánh thế nào với những yêu cầu truy xuất của chúng ta, có rút được kết quả đúng từ những bảng dữ liệu đó không. Thực hiện các chỉnh sửa thiết kế nếu thấy cần thiết.

2. KHÁI NIỆM VỀ BẢNG

Bảng là nơi chứa dữ liệu về một đối tượng thông tin nào đó như SINH VIÊN, HÓA ĐƠN,... Mỗi hàng trong bảng gọi là một **bản ghi** (record) chứa các nội dung riêng của đối tượng đó. Mỗi bản ghi của một bảng đều có chung cấu trúc, tức là các **trường** (field). Ví dụ: Cho bảng dưới đây để quản lý lý lịch khoa học cán bộ trong trường đại học, có các trường MACB (Mã cán bộ), TRINHDOVH (Trình độ văn hóa), CHUYENMON (Chuyên môn),...

Trong một CSDL có thể chứa nhiều bảng, thường mỗi bảng lưu trữ nhiều thông tin (dữ liệu) về một đối tượng thông tin nào đó, mỗi một thông tin đều có những kiểu đặc trưng riêng, mà với Access nó sẽ cụ thể thành những kiểu dữ liệu của các trường.

3. TẠO BẢNG MỚI TRONG CƠ SỞ DỮ LIỆU

Trong MS Access có hai cách để tạo bảng, một là cách dùng Table Wizard, nhưng các trường ở đây MS Access tự động đặt tên và không có bàn tay can thiệp của người sử dụng. Ở đây, sẽ đưa ra cách tạo mới bảng hoàn toàn do người sử dụng.

3.1. Tạo bảng không dùng Table Wizard

Trong cửa sổ Database, chọn tab Table (hoặc Lệnh *View/Daatbase object - Table*)

Chọn nút **New,** xuất hiện hộp thoại

Datasheet View: Trên màn hình sẽ xuất hiện một bảng trống với các trường (tiêu đề cột) lần lượt Field1, field2

Design View: Trên màn hình xuất hiện cửa sổ thiết kế bảng, người sử dụng tự thiết kế bảng.

Table Wizard: Thiết kế bảng với sự trợ giúp của MS Access

Import table: Nhập các bảng và các đối tượng từ các tập tin khác vào CSDL hiện thời.

Link table: Tạo bảng bằng cách nối vào CSDL hiện thời các bảng của CSDL khác.

Chọn chức năng Design View, chọn OK.

3.2. Sử dụng Design View

Field Name: Tên trường cần đặt (thông tin cần quản lý)

Data Type: Kiểu dữ liệu của trường

Desciption: Mô tả trường, phần này chỉ mang ý nghĩa làm rõ thông tin quản lý, có thể bỏ qua trong khi thiết kế bảng.

Field properties: Các thuộc tính của trường

Xác định khoá chính của bảng (nếu có)

Xác định thuộc tính của bảng, Lưu bảng dữ liệu

Đặt tên trường

Tên trường ở đây không nhất thiết phải có độ dài hạn chế và phải sát nhau, mà ta có thể đặt tên trường tùy ý nhưng không vượt quá 64 ký tự kể cả ký tự trắng. Lưu ý rằng, tên trường có thể đặt dài nên nó dễ mô tả được thông tin quản lý, nhưng sẽ khó khăn hơn khi ta dùng các phát biểu SQL và lập trình Access Basic. Do đó khi đặt tên trường ta nên đặt ngắn gọn, dễ gợi nhớ và không chứa ký tự trắng.

Kiểu dữ liệu

MS Access cung cấp một số kiểu dữ liệu cơ bản sau:

Kiểu dữ liệu	Dữ liệu vào	Kích thước
Text	Văn bản	Tối đa 255 byte
Memo	Văn bản nhiều dòng, trang	Tối đa 64000 bytes
Number	Số	1,2,4 hoặc 8 byte
Date/Time	Ngày giờ	8 byte
Currency	Tiền tệ (Số)	8 byte
Auto number	ACCESS tự động tăng lên một khi một bản ghi được tạo	4 byte
Yes/No	Lý luận (Boolean)	1 bit
OLE Object	Đối tượng của phần mềm khác	Tối đa 1 giga byte
Lookup Wizard		Trường nhận giá trị do người dùng chọn từ 1 bảng khác hoặc 1 danh sách giá trị định trước
Hyper link	Liên kết các URL	

Quy định thuộc tính, định dạng cho trường

Đặt thuộc tính là một phần không kém quan trọng, nó quyết định đến dữ liệu thực sự lưu giữa trong bảng, kiểm tra độ chính xác dữ liệu khi nhập vào, định dạng

dữ liệu nhập vào ... Mỗi một kiểu dữ liệu sẽ có các thuộc tính và các đặc trưng và khác nhau. Sau đây là các thuộc tính, định dạng của các kiểu dữ liệu.

Để tăng thêm tốc độ xử lý khi nhập dữ liệu cũng như các công việc tìm kiếm sau này thì việc quy định dữ liệu rất quan trọng.

Các trường trong ACCESS có các thuộc tính sau:

3.2.1. Field Size

Quy định kích thước của trường và tùy thuộc vào từng kiểu dữ liệu

Kiểu Text: Chúng ta quy định độ dài tối đa của chuỗi.

Kiểu Number: Có thể chọn một trong các loại sau:

Byte: 0..255

Integer: -32768..32767

Long Integer: -3147483648.. 3147483647

Single:$-3,4x10^{38}..3,4x10^{38}$ (Tối đa 7 số lẻ)

Double: $-1.797x10^{308}$ ..$1.797x10^{308}$ (Tối đa 15 số lẻ)

Decimal Places

Quy định số chữ số thập phân (Chỉ sử dụng trong kiểu Single và Double)

Đối với kiểu Currency mặc định decimal places là 2

3.2.2. Format

Quy định dạng hiển thị dữ liệu, tùy thuộc vào từng kiểu dữ liệu.

Kiểu chuỗi: Gồm 3 phần

<Phần 1>;<Phần 2>;<Phần 3>

Trong đó:

<Phần 1>: Chuỗi định dạng tương ứng trong trường hợp có chứa văn bản.

<Phần 2>: Chuỗi định dạng tương ứng trong trường hợp không chứa văn bản.

<Phần 3>: Chuỗi định dạng tương ứng trong trường hợp null

Các ký tự dùng để định dạng chuỗi

Ký tự	Tác dëch vuû du lëchụng
@	Chuỗi ký tự
>	Đổi tất cả ký tự nhập vào thành in hoa
<	Đổi tất cả ký tự nhập vào thành in thường
“Chuỗi ký tự “	Chuỗi ký tự giữa 2 dấu nháy
\<ký tự>	Ký tự nằm sau dấu \
[black] [White] [red] Hoặc [<số>] Trong đó 0<=số<=56	Màu

Ví dụ

Cách định dạng	Dữ liệu	Hiển thị
@@@-@@@@	123456 abcdef	123-456 abc-def
>	Tinhoc	TINHOC
<	TINHOC	Tinhoc
@;”Không có”;”Không biết”	Chuỗi bất kỳ Chuỗi rỗng Giá trị trống (Null)	Hiển thị chuỗi Không có Không biết

Kiểu Number

Định dạng do ACCESS cung cấp

Dạng	Dữ liệu	Hiển thị
General Number	1234.5	1234.5
Currency	1234.5	$1.234.50
Fixed	1234.5	1234

Standard	1234.5	1,234.50
Pecent	0.825	82.50%
Scientific	1234.5	1.23E+03

Định dạng do người sử dụng

<Phần 1>;<Phần 2>;<Phần 3>;<Phần 4>

<Phần 1>:Chuỗi định dạng tương ứng trong trường hợp số dương.

<Phần 2>: Chuỗi định dạng tương ứng trong trường hợp số âm.

<Phần 3>: Chuỗi định dạng tương ứng trong trường hợp số bằng zero.

<Phần 4>: Chuỗi định dạng tương ứng trong trường hợp null.

Các ký tự định dạng

Ký tự	Tác dụng
.(Period)	Dấu chấm thập phân
,(comma)	Dấu phân cách ngàn
0	Ký tự số (0-9)
#	Ký tự số hoặc khoảng trắng
$	Dấu $
%	Phần trăm

Ví dụ

Định dạng	Hiển thị
0;(0);;"Null"	Số dương hiển thị bình thường Số âm được bao giữa 2 dấu ngoặc Số zero bị bỏ trống Null hiện chữ Null
+0.0;-0.0;0.0	Hiển thị dấu + phía trước nếu số dương Hiển thị dấu - phía trước nếu số âm Hiển thị 0.0 nếu âm hoặc Null

Kiểu Date/Time

Các kiểu định dạng do ACCESS cung cấp

Dạng	Hiển thị
General date	10/30/99 5:10:30PM
Long date	Friday, may 30 , 1999
Medium date	30-jul-1999
Short date	01/08/99
Long time	6:20:00 PM
Medium time	6:20 PM
Short time	18:20

Các ký tự định dạng

Ký tự	Tác dụng
: (colon)	Dấu phân cách giờ
/	Dấu phân cách ngày
d	Ngày trong tháng (1-31)
dd	Ngày trong tháng 01-31)
ddd	Ngày trong tuần (Sun -Sat0
W	Ngày trong tuần (1-7)
WW	Tuần trong năm (1-54)
M	Tháng trong năm (1-12)
MM	Tháng trong năm (01-12)
q	Quý trong năm (1-4)
y	Ngày trong năm (1-366)
yy	Năm (01-99)
h	Giờ (0-23)
n	Phút (0-59)
s	Giây (0-59)

Ví dụ

Định dạng	Hiển thị
Ddd,”mmm d”,yyyy	Mon,jun 2, 1998
Mm/dd/yyyy	01/02/1998

Kiểu Yes/No

Các kiểu định dạng

Định dạng	Tác dụng
Yes/No	Đúng/Sai
True/False	Đúng/Sai
On/Off	Đúng/Sai

Định dạng do người sử dụng: Gồm 3 phần

<Phần 1>;<Phần 2>;<Phần 3>

Trong đó: <Phần 1>: Bỏ trống

<Phần 2>: Trường hợp giá trị trường đúng

<Phần 3>: Trường hợp giá trị trường sai

Ví dụ

Định dạng	Hiển thị	
	Trường hợp True	Trường hợp False
;”Nam”;”Nu”	Nam	Nu
;”co”;”Khong”	Co	Khong

3.2.3. Input mask (Mặt nạ)

Thuộc tính này dùng để quy định mặt nạ nhập dữ liệu cho một trường.

Các ký tự định dạng trong input mask

Ký tự	Tác dụng
0	Bắt buộc nhập ký tự số
9	Không bắt buộc nhập, ký tự số
#	Không bắt buộc nhập, số 0-9, khoảng trắng, dấu + và -
L	Bắt buộc nhập, ký tự chữ
?	Không bắt buộc nhập, ký tự chữ hoặc khoảng trắng
a	Bắt buộc nhập, ký tự chữ hoặc số
A	Không bắt buộc nhập, ký tự chữ hoặc số
&	Bắt buộc nhập, ký tự bất kỳ
C	Không bắt buộc nhập ký tự bất kỳ
<	Các ký tự bên phải được đổi thành chữ thường
>	Các ký tự bên phải được đổi thành chữ hoa
!	Dữ liệu được ghi từ phải sang trái
\<Ký tự>	Ký tự theo sau \ sẽ được đưa thẳng vào

Ví dụ

Input mask	Dữ liệu nhập vào
(000)000-0000	(054)828-8282
(000)AAA-A	(123)124-E

☞ **Chú ý:** Nếu muốn các ký tự gõ vào quy định thuộc tính input mask là password (Khi nhập dữ liệu vào tại các vị trí đó xuất hiện dấu *).

3.2.4. Caption

Quy định nhãn là một chuỗi ký tự sẽ xuất hiện tại dòng tiêu đề của bảng. Chuỗi ký tự này cũng xuất hiện tại nhãn các của các điều khiển trong các biểu mẫu hoặc báo cáo.

3.2.5. Default value

Quy định giá trị mặc định cho trường trừ Auto number và OEL Object

3.2.6. Validation rule và Validation Text

Quy định quy tắc hợp lệ dữ liệu (Validation rule) để giới hạn giá trị nhập vào cho một trường. Khi giới hạn này bị vi phạm sẽ có câu thông báo ở Validation text.

Các phép toán có thểt dùng trong Validation rule

Các phép toán	Phép toán	Tác dụng
Phép so sánh	>, <, >=, <=, =, <>	
Phép toán logic	Or, and , not	Hoặc, và, phủ định
Phép toán về chuỗi	Like	Giống như

☞ ***Chú ý:*** Nếu hằng trong biểu thức là kiểu ngày thì nên đặt giữa 2 dấu #.

Ví dụ

Validation rule	Tác dụng
<>0	Khác số không
Like "*HUE*"	Trong chuỗi phải chứa HUE
<#25/07/76#	Trước ngày 25/07/76
>=#10/10/77# and <=#12/11/77#	Trong khoảng từ 10/10/77 đến 12/12/77

3.2.7. Required

Có thể quy định thuộc tính này để bặt buộc hay không bắt buộc nhập dữ liệu cho trường.

Required	Tác dụng
Yes	Bắt buộc nhập dữ liệu
No	Không bắt buộc nhập dữ liệu

3.2.8. AllowZeroLength

Thuộc tính này cho phép quy định một trường có kiểu Text hay memo có thể hoặc không có thể có chuỗi có độ dài bằng 0.

☞ ***Chú ý:*** Cần phân biệt một trường chứa giá trị null (chưa có dữ liệu) và một trường chứa chuỗi có độ dài bằng 0 (Có dữ liệu nhưng chuỗi rỗng "").

AllowZeroLength	Tác dụng
Yes	Chấp nhận chuỗi rỗng
No	Không chấp nhận chuỗi rỗng

3.2.9. Index

Quy định thuộc tính này để tạo chỉ mục trên một trường. Nếu chúng ta lập chỉ mục thì việc tìm kiếm dữ liệu nhanh hơn và tiện hơn.

Index	Tác dụng
Yes(Dupplicate OK)	Tạo chỉ mục có trùng lặp
Yes(No Dupplicate)	Tạo chỉ mục không trùng lặp
No	Không tạo chỉ mục

3.2.10. New value

Thuộc tính này chỉ đối với dữ liệu kiểu auto number, quy định cách thức mà trường tự động điền số khi thêm bản ghi mới vào.

New value	Tác dụng
Increase	Tăng dần
Random	Lấy số ngẫu nhiên

4. THIẾT LẬP KHOÁ CHÍNH (*primary key*)

4.1. Khái niệm khoá chính

Sức mạnh của một Hệ QTCSDL như Microsoft Access, là khả năng mau chóng truy tìm và rút dữ liệu từ nhiều bảng khác nhau trong CSDL. Để hệ thống có thể làm được điều này một cách hiệu quả, mỗi bảng trong CSDL cần có một trường hoặc một nhóm các trường có thể xác định duy nhất một bản ghi trong số rất nhiều bản ghi đang có trong bảng. Đây thường là một mã nhận diện như Mã nhân viên hay Số Báo Danh của học sinh. Theo thuật ngữ CSDL trường này được gọi là **khóa chính** (primary key) của bảng. MS Access dùng trường khóa chính để kết nối dữ liệu nhanh chóng từ nhiều bảng và xuất ra kết quả yêu cầu.

Nếu trong bảng chúng ta đã có một trường sao cho ứng với mỗi trị thuộc trường đó chúng ta xác định duy nhất một bản ghi của bảng, chúng ta có thể dùng trường đó làm trường khóa của bảng. Từ đó cho ta thấy rằng tất cả các trị trong trường khóa chính phải khác nhau. Chẳng hạn đừng dùng tên người làm trường khóa vì tên trường là không duy nhất.

Nếu không tìm được mã nhận diện cho bảng nào đó, chúng ta có thể dùng một trường kiểu **Autonunter** (ví dụ Số Thứ Tự) để làm trường khóa chính.

Khi chọn trường làm khóa chính chúng ta lưu ý mấy điểm sau:

MS Access không chấp nhận các giá trị trùng nhau hay trống (null) trong trường khóa chính.

Chúng ta sẽ dùng các giá trị trong trường khóa chính để truy xuất các bản ghi trong CSDL, do đó các giá trị trong trường này không nên quá dài vì khó nhớ và khó gõ vào.

Kích thước của khóa chính ảnh hưởng đến tốc độ truy xuất CSDL. Để đạt hiệu quả tối ưu, dùng kích thước nhỏ nhất để xác định mọi giá trị cần đưa vào trường.

4.2. Cách đặt khoá chính

Ta có thể tự chọn trường làm khóa chính cho bảng bằng các bước sau đây:

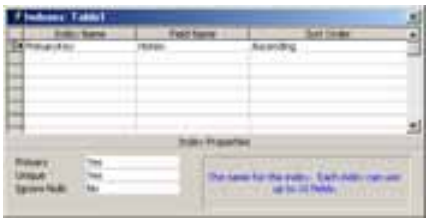
Mở bảng ở chế độ Design View

Nhắp chọn trường cần đặt

Thực hiện lệnh *Edit - Primary Key* hoặc nhắp chọn nút trên thanh công cụ của mục này .

☞***chú ý:*** Không phải mọi trường đều có thể làm khóa chính, mà chỉ có các trường có các kiểu dữ liệu không phải là **Memo** và **OLE Object**., **Hyper Link**.

Để hủy bỏ khóa chính hoặc các đã thiết lập thì thực hiện lệnh *View - Indexes*, trong hộp thoại này chọn và xóa đi những trường khóa đã thiết lập:



5. LƯU BẢNG

Sau khi thiết kế xong, ta tiến hành lưu bảng vào CSDL, có thể thực hiện một trong hai thao tác sau:

Thực hiện lệnh *File - Save*.

Nhắp chọn nút trên thanh công cụ của mục này (Table Design)

6. HIỆU CHỈNH BẢNG

6.1. Di chuyển trường: Các thao tác để di chuyển thứ tự các trường

Đưa con trỏ ra đầu trường đến khi con trỏ chuột chuyển thành ➡ thì nhắp chọn.
Đưa con trỏ ra đầu trường vừa chọn, nhấn và kéo đếnvị trí mới.

6.2. Chen trường : Các thao tác lần lượt như sau

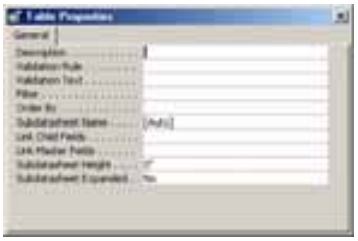
Chọn trường hiện thời là trường sẽ nằm sau trường được chen vào
Thực hiện lệnh *Insert/ Row*

6.3. Xóa trường: Các thao tác lần lượt như sau

Chọn trường cần xóa
Thực hiện lệnh *Edit - Delete Rows*

6.4. Quy định thuộc tính của bảng

Mở bảng ở chế độ Design View, chọn View/Properties



Description: Dòng mô tả bảng

Validation Rule: Quy tắc hợp lệ dữ liệu cho toàn bảng.

Validation Text: Thông báo lỗi khi dữ liệu không hợp lệ.

7. XEM THÔNG TIN VÀ BỔ SUNG BẢN GHI

7.1. Xem thông tin ở chế độ datasheet

Muốn xem thông tin trong một bảng chúng ta phải chuyển bảng sang một chế độ hiển thị khác gọi là *Datasheet*. Trong chế độ hiển thị này, mỗi bản ghi được thể hiện trên một hàng ngang, hàng đầu tiên là các tên trường.

Sau đây là các cách để chuyển sang chế độ hiển thị Datasheet:

Trong của sổ Database của CSDL đang mở, nhấp chọn tab Table. Trong mục này chọn bảng cần hiển thị rồi chọn nút **Open**, bảng sẽ được mở để bổ sung và chỉnh sửa dữ liệu.

Ta có thể chuyển sang chế độ Datasheet ngay khi đang ở trong chế Design, bằng cách nhắp chọn nút thì bảng sẽ chuyển sang chế độ Datasheet, để quay trở về chế độ Design, ta nhắp chọn lại nút . Hoặc chọn lệnh *View - /Design View*.

7.2. Bổ sung bản ghi cho bảng

Sau khi hoàn thành công việc thiết kế cấu trúc bảng, ta tiến hành nhập dữ liệu, tức là bổ sung các bản ghi, cho bảng. Hiển thị bảng ở chế độ hiển thị Datasheet, mỗi hàng đại diện cho một bản ghi. Ở đây có các ký hiệu chúng ta cần biết công dụng của chúng

Bản ghi hiện thời

Bản ghi đang nhập dữ liệu

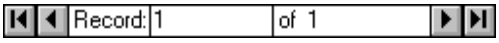
Bản ghi mới

7.2.1. Bổ sung bản ghi mới cho CSDL

Đang đứng tại một bản ghi nào đó (không phải là bản ghi mới), chọn nút trên thanh công cụ. Hoặc thực hiện lệnh *Record - Go To - New*.

7.2.2. Di chuyển giữa các bản ghi

Ta có thể di chuyển qua lại giữ các bản ghi bằng cách dùng công cụ *Chọn lựa bản ghi* (Record Selector).



7.2.3. Nhập dữ liệu cho bản ghi

Khi đang nhập dữ liệu cho một bản ghi nào đó, thì đầu hàng của bản ghi đó xuất hiện biểu tượng .

Tổ hợp phím	Tác dụng
Tab	Sang ô kế tiếp
Shift Tab	Sang ô phía trước
Home	Đến đầu dòng
End	Đến cuối dòng
Ctrl Home	Đến bản ghi đầu tiên
Ctrl End	Đến bản ghi cuối cùng
Shift F2	Zoom

Theo chuẩn, khi nhập dữ liệu thì Access sẽ lấy font mặc định là MS San Serif, điều này có thể giúp cho ta hiển thị được tiếng Việt chỉ khi MS San Serif đó là của VietWare hoặc của ABC.

Để không phụ thuộc vào điều này, ta nên chọn font trước khi tiến hành nhập dữ liệu. Trong chế độ hiển thị Datasheet, thực hiện lệnh *Format - Font...*

Khi nhập dữ liệu là trường cho trường **OLE Object**, ta thực hiện như sau: Lệnh *Edit - Object...*

7.2.4. Chọn các bản ghi

Đánh dấu chọn bản ghi:

Chọn Edit/Select Record: Để chọn bản ghi hiện hành

Chọn Edit/ Select all Record để chọn toàn bộ

7.2.5. Xóa bản ghi

Chọn các bản ghi cần xóa, sau đó thực hiện lệnh *Edit - Delete* (hoặc nhấn phím DELETE) .

8. THIẾT LẬP QUAN HỆ GIỮA CÁC BẢNG

8.1. Các loại quan hệ trong cơ sở dữ liệu ACCESS

8.1.1. Quan hệ một -một (1-1)

Trong quan hệ một -một, mỗi bản ghi trong bảng A có tương ứng với một bản ghi trong bảng B và ngược lại mỗi bản ghi trong bảng B có tương ứng duy nhất một bản ghi trong bảng A.

***Ví dụ*:** Cho 2 bảng dữ liệu

Bảng Danhsach(**Masv**, ten, Ngaysinh, gioitinh) và bảng Diemthi(**Masv**, diem)

Ten	Ngaysinh	Gioitinh	Masv
An	20/10/77	Yes	A001
Bình	21/07/80	No	A002
Thuỷ	02/12/77	Yes	A003
Lan	03/04/80	No	A004
Hồng	12/11/77	No	A005

Masv	diem
A001	9
A002	7
A003	9
A004	4
A005	5

Bảng **Danhsach** và **diemthi** có mối quan hệ **1-1** dựa trên trường Masv.

8.1.2. Quan hệ một nhiều (1-∞)

Là mối quan hệ phổ biến nhất trong CSDL, trong quan hệ một nhiều : Một bản ghi trong bảng A sẽ có thể có nhiều bản ghi tương ứng trong bảng B, nhưng ngược lại một bản ghi trong bảng B có duy nhất một bản ghi tương ứng trong bảng A.

Ví dụ: Trong một khoa của một trường học nào đó có nhiều sinh viên, những một sinh viên thuộc một khoa nhất định. Ta có 2 bảng dữ liệu như sau:

Bảng Danhsachkhoa(**Makhoa**, tenkhoa, sodthoai)

Bảng danhsachsv(Makhoa, Ten, Quequan, lop)

Tenkhoa	Sodthoai	Makhoa
CNTT	826767	01
TOÁN	878787	02
LÝ	868785	03

Makhoa	Ten	Queuqan	Lop
01	Thanh	Huế	K23
01	Tùng	Qbình	K24
02	Thuỷ	Huế	K25
02	Hùng	ĐN	K26
03	Lan	Huế	K25
03	Hương	ĐN	K26

Bảng Danhsachkhoa và bảng danhsachsv có mối quan hệ **1-∞** dựa trên trường Makhoa.

8.3. Quan hệ nhiều nhiều(∞-∞)

Trong quan hệ nhiều nhiều, mỗi bản ghi trong bảng A có thể có không hoặc nhiều bản ghi trong bảng B và ngược lại mỗi bản ghi trong bảng B có thể có không hoặc nhiều bản ghi trong bảng A.

Khi gặp mối quan hệ nhiều- nhiều để không gây nên sự trùng lặp và dư thừa dữ liệu thì người ta tách quan hệ nhiều-nhiều thành 2 quan hệ một-nhiều bằng cách tạo ra một bảng phụ chứa khóa chính của 2 bảng đó.

Ví dụ: Một giáo viên có thể dạy cho nhiều trường và một trường có nhiều giáo viên tham gia giảng dạy. Đây là một mối quan hệ nhiều-nhiều

Bảng Danhsachgv(**Magv**,ten)

Bảng Danhsachtruong(**Matruong**, Tentruong)

Tạo ra bảng Phancongday(**Magv, matruong**)

Ten	Magv
Thanh	G1
Thuý	G2
Hùng	G3

Magv	Matruong
G1	KH
G1	SP
G2	YK
G2	SP
G3	KH
G3	YK

Matruong	tentruong
KH	DHKH
SP	DHSP
YK	DHYK

Bảng **Danhsachgv** và bảng **Phancongday** có mối quan hệ **1-∞** dựa trên trường **Magv**.

Bảng **Danhsachtruong** và bảng **Phancongday** có mối quan hệ **1-∞** dựa trên trường **Matruong**.

8.4. Thiết lập mối quan hệ giữa các bảng dữ liệu (Relationships)

Tại cửa sổ Database, thực hiện lệnh Tools/Relationship

Trong cửa sổ Shov

bảng cần thiết lập quan hệ, sau đó chọn **Add và Close**.

Kéo trường liên kết của bảng quan hệ vào trường của bảng được quan hệ (Table related).

Bật chức năng Enforce Referential Integrity (Nếu muốn quan hệ này bị ràng buộc tham chiếu toàn vẹn), chọn mối quan hệ (one-many) hoặc (one-one).

Chọn nút **Create**.

☞ ***Chú ý***

Quan hệ có tính tham chiếu toàn vẹn sẽ đảm bảo các vấn đề sau:

Khi nhập dữ liệu cho trường tham gia quan hệ ở bên nhiều thì phải tồn tại bên một.

Không thể xoá một bản ghi của bảng bên một nếu trong quan hệ đã tồn tại những bản ghi bên nhiều có quan hệ với bản ghi bên một đó.

Trường hợp vi phạm các quy tắc trên thì sẽ nhận được thông báo lỗi.

8.4.1. Thiết lập thuộc tính tham chiếu toàn vẹn trong quan hệ

Trong khi chọn mối quan hệ giữa các bảng, có 2 thuộc tính tham chiếu toàn vẹn đó là *Cascade update related fields,Cascade Delete related records*, có thiết lập 2 thuộc tính này.

Nếu chọn thuộc tính *Cascade update related fields*, khi dữ liệu trên khoá chính của bảng bên một thay đổi thì Access sẽ tự động cập nhật sự thay đổi đó vào các trường tương ứng (có quan hệ) trên các bảng bên Nhiều, hay nói cách khác, dữ liệu ở bảng bên nhiều cũng thay đổi theo.

Nếu chọn thuộc tính *Cascade Delete related records*, khi dữ liệu trên bảng bên một bị xoá thì dữ liệu trên bảng bên nhiều cũng sẽ bị xoá..

8.4.2. Kiểu kết nối (Join type)

Trong quá trình thiết lập quan hệ giữa các bảng, nếu không chọn nút Create, chọn nút join type để chọn kiểu liên kết

Mục 1: Liên kết nội (Inner join)

Mục 2 và mục 3 là liên kết ngoại (outer join)

8.4.3. Điều chỉnh các mối quan hệ

Mở cửa sổ quan hệ (Tools/Relationship)

Click chuột phải, chọn edit relationship

8.4.4. Xoá các mối quan hệ

Mở cửa sổ quan hệ (Tools/Relationship)

Chọn mối quan hệ giữa các bảng, nhấn delete.

9. SẮP XẾP VÀ LỌC DỮ LIỆU

9.1. Một số phép toán và hàm

9.1.1. Một số phép toán

Ký tự thay thế: ? : Thay thế cho một ký tự bất kỳ

Ký tự * : Thay thế cho một dãy các ký tự.

Phép toán **Like**: Giống như

In: Kiểm tra một giá trị có thuộc một tập các giá trị hay không?

Is Null: Giá trị của một trường là Null.

Is not Null:Giá trị của một trường là không Null.

Between.....and: Kiểm tra xem một giá trị có thuộc một "đoạn" nào đó hay không?

9.1.2. Một số hàm

Hàm **Left$(<Chuỗi>,<n>)**: Trích bên trái chuỗi n ký tự.

Hàm **Right$(<Chuỗi>,<n>)**: Trích bên phải chuỗi n ký tự.

Hàm **Ucase(<Chuỗi>)**: Trả lại một chuỗi in hoa.

Hàm **Lcase(<Chuỗi>)**: Trả lại một chuỗi in thường.

Hàm **IIF(<Điều kiện>,<Giá trị 1>,< Giá trị 2>)**: Nếu <Điều kiện> nhận giá trị true thì hàm trả lại <Giá trị 1>, ngược lại hàm trả lại <Giá trị 2>.

9.2. Sắp xếp dữ liệu

9.2.1. sắp xếp trên một trường

Đặt con trỏ tại trường cần sắp xếp

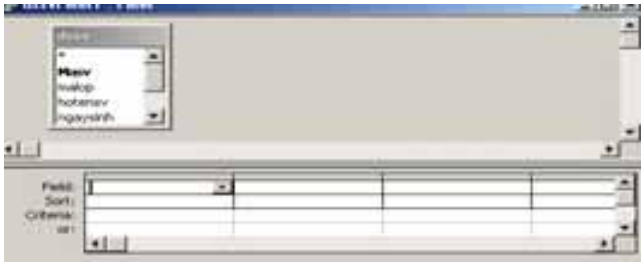
Thực hiện lệnh Records/ Sort/ Sort Ascending (Nếu sắp xếp tăng dần)

/ Sort Descending (Nếu sắp xếp giảm dần)

9.2.2. sắp xếp trên nhiều trường

Mở bảng trong chế độ Datasheet rồi thực hiện lệnh

Records/ Filter/ Advanced Filter sort



Trong hàng Field: Chọn các trường cần sắp xếp (Thứ tự ưu tiên từ trái sang phải)

Trong hàng Sort: Chọn tiêu chuẩn sắp xếp.

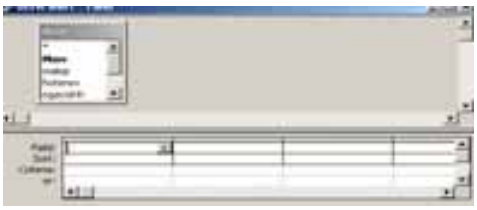
Trong hàng Criteria: Chọn điều kiện sắp xếp (nếu có).

Xem kết quả.: Chọn Filter/Apply filter.

9.3. Lọc dữ liệu

Mở bảng trong chế độ Datasheet rồi thực hiện lệnh

Records/ Filter/ Advanced Filter sort



Trong hàng Field: Chọn các trường làm tiêu chuẩn lọc dữ liệu

Trong hàng Criterie: Chọn tiêu chuẩn lọc dữ liệu.

Thực hiện lệnh Filter/ Apply filter sort để xem kết quả..

Chương 3 TRUY VẤN (QUERY)

Sức mạnh thực sự của CSDL là khả năng tìm đúng và đầy đủ thông tin mà chúng ta cần biết, trình bày dữ liệu sắp xếp theo ý muốn. Để đáp ứng yêu cầu trên, Acces cung cấp một công cụ truy vấn cho phép đặt câu hỏi với dữ liệu đang chứa bên trong các bảng trong CSDL.

1. KHÁI NIỆM TRUY VẤN

Truy vấn là một công cụ cho phép đặt câu hỏi với dữ liệu trong bảng dữ liệu trong CSDL.

Loại truy vấn thông dụng nhất là truy vấn chọn (Select Query). Với kiểu truy vấn này chúng ta có thể xem xét dữ liệu trong các bảng, thực hiện phân tích và chỉnh sửa trên dữ liệu đó, có thể xem thông tin từ 1 bảng hoặc có thể thêm nhiều trường từ nhiều bảng khác nhau.

Ví dụ:

Cho 2 bảng dữ liệu KHOHANG (**MAHANG**, TENHANG, GIA)
BANHANG(MAHANG, TENKHACH, SOLUONG, NGAYMUA). Hãy hiển thị những khách hàng mua hàng trong tháng 7 bao gồm các thông tin: MAHANG, TENHANG, GIA, TENKHACH.

Sau khi thực hiện truy vấn, dữ liệu thỏa mãn yêu cầu được rút ra và tập hợp vào một bảng kết qủa gọi là **Dynaset** (Dynamic set). Dynaset cũng hoạt động như 1 bảng (Table) nhưng nó không phải là bảng vfa kết quả khi hiển thị có thể cho phép sửa đổi.

Một loại bảng thể hiện kết quả truy vấn khác là **Snapshot**, nó tương tự như dynaset tuy nhiên không thể sửa đổi thông tin (Như truy vấn Crosstab....).

1.1. Các loại truy vấn trong Access

Select Query : Truy vấn chọn

Crosstab Query : Truy vấn tham khỏa chéo (Thể hiện dòng và cột)

Action Query : Truy vấn hành động gồm

Truy vấn tạo bảng (make table Query)

Truy vấn nối (append Query)

Truy vấn cập nhật (Update Query)

Truy vấn xóa dữ liệu (Delete Query)

SQL Query : Truy vấn được viết bởi ngôn ngữ SQL.

Pass throught Query : Gởi các lệnh đến một CSDL SQL như Microsoft SQL server.

1.2. Sự cần thiết của truy vấn

Khi đứng trước một vấn đề nào đó trong CSDL, nếu sử dụng công cụ truy vấn thì có thể thực hiện được các yêu cầu sau:

Sự lựa chọn các trường cần thiết.

Lựa chọn những bản ghi.

Sắp xếp thứ tự các bản ghi.

Lấy dữ liệu chứa trên nhiều bảng khác nhau trong CSDL.

Thực hiện các phép tính.

Sử dụng truy vấn làm nguồn dữ liệu cho một biểu mẫu (Form), báo cáo (report) hoặc một truy vấn khác (Query).

Thay đổi dữ liệu trong bảng.

2. CÁC CHẾ ĐỘ HIỂN THỊ TRUY VẤN

2.1. Cửa sổ thiết kế truy vấn (Design view).

Trong chế độ này, người sử dụng có thể tạo, sửa chữa một truy vấn nào đó. Màn hình truy vấn chứa hai phần, phần thứ nhất chứa các bảng (hoặc truy vấn) tham gia truy vấn, phần thứ hai gọi là vùng lưới QBE (Query By Example).

2.2. Cửa sổ hiển thị truy vấn (DataSheet view).

Sử dụng chế độ này để xem kết quả.

2.3. Cửa sổ lệnh SQL (SQL view).

Sử dụng chế độ này để xem mã lệnh của truy vấn đang tạo

3. TẠO TRUY VẤN

3.1.Tạo mới 1 truy vấn

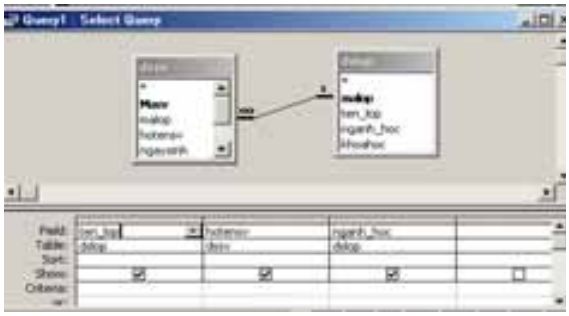
Từ cửa sổ Database, click vào đối tượng Queries.

Chọn nút New.

Chọn Design View, chọn OK

Trong bảng Show Table chọn tables để hiển thị các bảng, chọn các bảng tham gia vào truy vấn và nhấn nút **Add**, sau đó nhấn ***Close***. (Nếu chọn Queries thì hiển thị truy vấn, chọn both thì hiển thị cả truy vấn và bảng dữ liệu).

Đưa các trường từ các bảng vào tham gia truy vần bằng cách kéo các trường và thả vào hàng **Field** trong vùng lưới QBE.



Trong hàng Sort: Sắp xếp dữ liệu (nếu có)

Trong hàng **Criteria** đặt tiêu chuẩn (nếu có)

Lưu truy vấn.

☞ ***Chú ý***

Mỗi truy vấn có:

Tối đa là 32 bảng tham gia.

Tối đa là 255 trường.

Kích thước tối đa của bảng dữ liệu (do truy vấn tạo ra) là 1 gigabyte.

Số trường dùng làm khóa sắp xếp tối đa là 10.

Số truy vấn lồng nhau tối đa là 50 cấp.

Số ký tự tối đa trong ô của vùng lưới là 1024.

Số ký tự tối đa trong dòng lệnh SQL là 64000.

Số ký tự tối đa trong tham số là 255.

3.2. Thay đổi thứ tự, xóa các trường

Các trường trong truy vấn sẽ hiển thị theo thứ tự như xuất hiện trong vùng lưới QBE.

3.2.1.Thay đổi thứ tự của trường

Đưa con trỏ vào thanh chọn sao cho con trỏ biến thành hình mũi tên trỏ xuống

Click để chọn trường

Drag để thay đổi vị trí.

3.2.2.xoá trường

Đưa con trỏ vào thanh chọn sao cho con trỏ biến thành hình mũi tên trỏ xuống

Click để chọn trường

Nhấn phím delete (Nếu muốn xoá tất cả các trường trong vùng lưới QBE: chọn Edit/clear grid)

3.3. Thể hiện hoặc che dấu tên bảng trong vùng lưới QBE

Muốn biết tên trường hiện tại trong vùng lưới QBE là của tên bảng nào, tại chế độ Design View người sử dụng thực hiện View/tables name.

3.4. Xem kết quả của truy vấn.

Tại cửa sổ Database chọn tên truy vấn rồi chọn Open, hoặc trong khi thiết kế truy vấn thực hiện lệnh View/datasheet View.

3.5. Đổi tiêu đề cột trong truy vấn.

Đổi tên tiêu đề cột trong truy vấn mục đích là làm cho bảng kết xuất dễ đọc hơn (Trừ khi đã quy định thuộc tính Caption).

Muốn đổi tên tiêu đề cột thực hiện theo các bước sau:

Mở truy vấn ở chế độ Design View

Chọn vào bên trái ký tự đầu tiên của tên trường trong vùng lưới QBE

Gõ vào tên mới, theo sau là dấu 2 chấm (:).

Ví dụ:

3.6. Định thứ tự sắp xếp

Có thể sử dụng nhanh trong chế độ datasheet View .

Có thể tạo sắp xếp trong khi thiết kế truy vấn bằng cách chọn Ascending (tăng dần) hoặc Descending (giám dần) trong hàng Sort của vùng lưới QBE .

☞ ***Chú ý:*** Nếu có nhiều trường định vị sắp xếp thì theo thứ tự ưu tiên từ trái sang phải.

3.7. Che dấu hay thể hiện các trường trong Dynaset

Tại hàng Show ứng với trường cần che dấu chúng ta không chọn mặt dù nó vẫn tồn tại, vẫn tham gia truy vấn.

3.8. Mối quan hệ giữa thuộc tính của trường trong truy vấn và trong bảng dữ liệu

Theo mặc nhiên, các trường trong truy vấn kế thừa tất cả các thuộc tính của trường trong bảng làm nguồn dữ liệu. Nếu không quy định lại trong truy vấn, các trường trong Dynaset hoặc snapshot luôn kế thừa các thuộc tính của bảng làm nguồn dữ liệu. Nếu thay đổi thiết kế trong bảng làm nguồn dữ liệu và thay đổi thuộc tính của các trường thì thuộc tính này cũng được thay đổi trong truy vấn. Tuy nhiên, nếu quy định lại các thuộc tính cho các trường trong truy vấn thì các thuộc tính của các trường trong bảng làm nguồn dữ liệu không thay đổi.

4. THIẾT KẾ TRUY VẤN CHỌN

4.1. Định nghĩa truy vấn chọn

Truy vấn chọn là loại truy vấn được chọn lựa, rút trích dữ liệu từ các bảng dữ liệu thỏa mãn một hoặc nhiều điều kiện nào đó. Khi thực hiện truy vấn chọn, Access tác động lên dữ liệu và thể hiện các bản ghi thoả mãn các điều kiện đăt ra trong một bảng kết quả gọi là Recordset.

4.2. Lập phép chọn trong truy vấn

4.2.1. Chọn một nhóm các bản ghi thỏa mãn một điều kiện nào đó

Muốn thực hiện các phép chọn trong khi thể hiện truy vấn người ta thường sử dụng các phép toán sau:

Phép toán	Ví dụ	Ý nghĩa
<	<#20/10/99#	Trước ngày 20/10/99
>	>#10/10/98#	Sau ngày 10/10/98
>=	>= #05/05/90#	Sau và trong ngày 05/05/90
<>	<>#01/01/99#	Khác ngày 01/01/99
=	= #10/10/97#	Trong ngày 10/10/97
Between .... and ...	Between #1/2/97# and #1/7/97#	Từ ngày 1/2/97 đến 1/7/97

Ví dụ:

Cho 2 bảng dữ liệu Dslop(**Malop**, Tenlop, Nganh_hoc, khoahoc)

Dssv(Masv, malop, hotensv, ngaysinh, quequan, giotinh, hocbong)

Tạo một truy vấn để hiển thị danh sách những sinh viên có ngaysinh trong khoảng thời gian từ 05/05/75 đến 05/05/79 bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

Trong hàng Criteria của trường Ngaysinh: Between #05/05/75# and #05/05/79#

4.2.2. Ký tự thay thế

Ký tự * : Thay thế một nhóm ký tự bất kỳ.

Ký tự ? : Thay thế 1 ký tự.

Ký tự [ ] : Thay thế các ký tự trong ngoặc vuông.

Ký tự ! : Phủ định.

Ký tự - : Từ ký tự đến ký tự.

Ví dụ

Cho 2 bảng dữ liệu Dslop(**Malop**, Tenlop, Nganh_hoc, khoahoc)

Dssv(Masv, malop, hotensv, ngaysinh, quequan, giotinh, hocbong)

Tạo một truy vấn để hiển thị danh sách những sinh viên có Tenlop bắt đầu là **"T"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

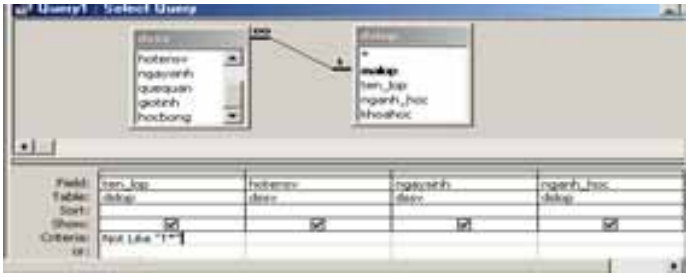
Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

Trong hàng Criteria của trường Tenlop: Like "T*"

4.2.3. Chọn các bản ghi không phù hợp với một giá trị nào đó

Dùng toán tử **Not**

Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Tenlop không bắt đầu là **"T"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, nganh_hoc.



4.2.4. Định nhiều tiêu chuẩn trong lựa chọn

Dùng phép "Và" và phép "Hoặc" trong một trường

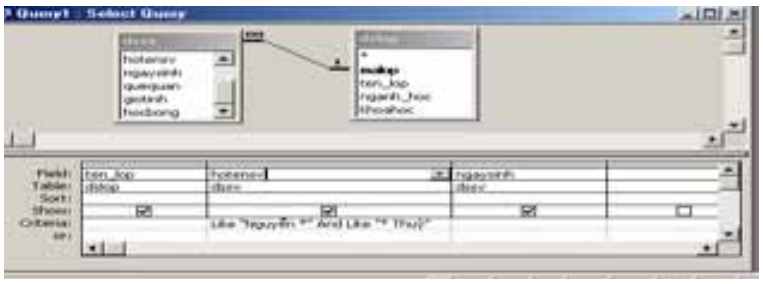
Muốn quy định nhiều tiêu chuẩn trong cùng một trường, chúng ta phải sử dụng toán tử AND (và) cùng toán tử OR (hoặc).

Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và Tên "**Thuỷ"** bao gồm các trường: Tenlop, Hotensv, Ngaysinh.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh vào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Nguyễn *" and "* Thuỷ"



Dùng phép "Và" và phép "Hoặc"trên nhiều trường

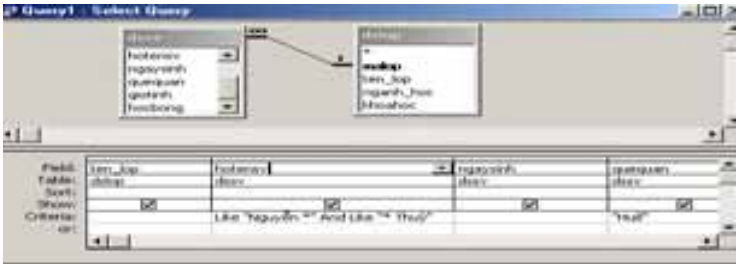
Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và tên "**Thuỷ**" và có quê quán ở "**Huế"** bao gồm các trường: Tenlop, Hotensv, ngaysinh Quequan.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh, quequan vào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Nguyễn *" and "* Thuỷ"

Quequan : Huế



Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Lê"** hoặc có quê quán ở "**Đà Nẵng** bao gồm các trường: Tenlop, Hotensv, Ngaysinh, Quequan.

Tạo truy vấn chọn và đưa 2 bảng dslop và dssv vào tham gia truy vấn

Đưa các trường Tenlop, hotensv, ngaysinh , Quequanvào vùng lưới QBE

Trong hàng Criteria của trường Hotensv : Like "Lê *"

Trong hàng or của trường Quequan : Đà Nẵng

4.2.5. Chọn các bản ghi có chứa có giá trị

Chúng ta có thể chọn các bản ghi có chứa hoặc không chứa giá trị, chẳng hạn như tìm những sinh viên mà không có số điện thoại nhà ở.

Access cung cấp 2 phép toán

Phép toán	Ý nghĩa
IS NULL	Trường không chứa giá trị
IS NOT NULL	Trường có chứa giá trị

Ví dụ:

Tạo một truy vấn để hiển thị danh sách những sinh viên không có học bổng bao gồm các trường: Tenlop, Hotensv, hocbong.

4.2.6. Chọn các bản ghi thuộc danh sách các giá trị nào đó

Chúng ta có thể sử dụng phép toán **IN** (***Danh sách giá trị***)

Ví dụ

Tạo một truy vấn để hiển thị danh sách những sinh viên thuộc lớp "**Tin K23**" hoặc "**Tin K24**" hoặc "**Tin K25**" bao gồm các trường: Tenlop, Hotensv, hocbong.

4.2.7. Tham chiếu đến các trường khác

Nếu trong biểu thức chọn của truy vấn, các tính toán tham chiếu đến các trường phải đặt trong dấu [ ], trường hợp tham chiếu đến trường của bảng khác phải chỉ rõ bảng nguồn của nó. **[Tên bảng]![Tên trường].**

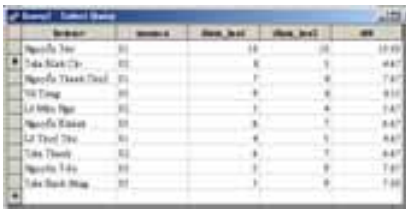
4.2.8. Tạo trường kiểu biểu thức

Ví dụ

Cho 2 bảng dữ liệu Dssv(**Masv**, hotensv, ngaysinh, quequan, gioitinh)

Dsdiem(**Masv**, mamon, diem_lan1, diem_lan2)

Tạo truy vấn để hiển thị các thông tin: Hotensv, mamon, diem_lan1, diem_lan2, dtb, trong đó dtb=(diem_lan1+ diem_lan2*2)/3.



☞ ***Chú ý***

Sau khi thực hiện truy vấn chúng ta không thể thay đổi giá trị trong trường kiểu biểu thức, tuy nhiên nếu thay đổi giá trị trong trường tham gia biểu thức thì kết quả trong trường kiểu biểu thức cũng thay đổi theo.

4.2.9 .Chọn giá trị duy nhất

Theo mặc định, access sẽ chọn tất cả các bản ghi thoã mãn điều kiện, tuy nhiên đôi khi có nhiều giá trị giống nhau được lặp đi lặp lại, do đó để cô đọng dữ liệu thì chúng ta có thể quy định thuộc tính duy nhất trong khi hiển thị

Thuộc tính Unique-values

Chọn Yes: Không thể hiện các giá trị trùng nhau

Chọn No: Thể hiện các giá trị trùng nhau

Thuộc tính Unique-Records

Chọn Yes: Không thể hiện các bản ghi trùng nhau

Chọn No: Thể hiện các bản ghi trùng nhau

4.2.10. Chọn các giá trị đầu

Khi hiển thị truy vấn đôi khi chúng ta muốn hiển thị một số bản ghi đầu tiên nào đó thoả mãn các điều kiện thì sử dụng thuộc tính **Top values**.

5. TRUY VẤN DỰA TRÊN NHIỀU BẢNG DỮ LIỆU

Để tạo truy vấn dựa trên nhiều bảng dữ liệu thì các bảng đó phải được thiết lập mối quan hệ, nếu các bảng không thiết lập mối quan hệ thì khi truy vấn dữ liệu access sẽ cho ra những bộ dữ liệu là tích Đề-Các giữa các bộ dữ liệu trong các bảng.

5.1. Liên kết các bảng trong truy vấn

Khi các bảng dữ liệu được thiết lập mối quan hệ thì trường nối với nhau gọi là trường liên kết, trong access phân biệt 3 loại liên kết sau

5.1.1. Liên kết nội (Inner join)

Đây là loại liên kết rất phổ biến nhất giữa 2 bảng dữ liệu. Trong đó dữ liệu khi thể hiện trên Dynaset sẽ gồm những bản ghi mà dữ liệu chứa trong trường liên kết ở hai bảng phải giống nhau hoàn toàn.

5.1.2. Liên kết ngoại (Outer join)

Đây là loại liên kết cho phép dữ liệu thể hiện trên Dynaset của một trong hai bảng tham gia có nội dung trường liên kết không giống nội dung trong trường tương ứng của bảng còn lại. Liên kết ngoại được chia làm hai loại

Left Outer Join: Trong kiểu liên kết này, dữ liệu ở bảng bên "1" thể hiện toàn bộ trên Dynaset và chỉ những bản ghi bên bảng "nhiều" có nội dung trong trường liên kết giống trường tương ứng bên bảng "1".

Right Outer Join: Trong kiểu liên kết này, dữ liệu ở bảng bên "nhiều" thể hiện toàn bộ trên Dynaset và chỉ những bản ghi bên bảng "1" có nội dung trong trường liên kết giống trường tương ứng bên bảng "nhiều".

5.1.3. Tự liên kết (Seft join)

Là kiểu liên kết của một bảng dữ liệu với chính nó. Trong đó một bản ghi trong bảng dữ liệu sẽ liên kết với những bản ghi khác trong bảng dữ liệu đó. Tự liên kết có thể hiểu như là liên kết nội hay liên kết ngoại từ một bảng vào một bảng sao chính nó. Để thực hiện việc tạo tự liên kết chúng ta phải đưa một bảng vào tham gia truy vấn 2 lần.

5.2.Tạo liên kết ngoại

Muốn tạo liên kết ngoại giữa 2 bảng dữ liệu ta thực hiện

Tools/Relationships

Double click vào đường liên kết giữa 2 bảng dữ liệu, chọn Join Type

Trong hộp thoại Join Properties chọn mục 2 hoặc mục 3.

5.3.Tạo một tự liên kết

Để tạo một tự liên kết chúng ta thực hiện

Tạo truy vấn mới và đưa bảng dữ liệu vào truy vấn 2 lần

Tạo các liên kết

Ví dụ:

Cho bảng Dsdiem(Masv, Hoten, Diem_lan1, Diem_lan2)

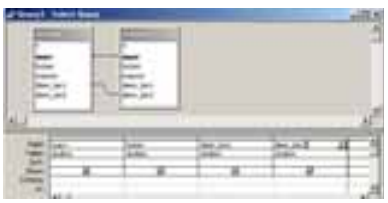
Tạo truy vấn để hiển thị danh sách những sinh viên có điểm thi Lần 1 bằng điểm thi lần 2...

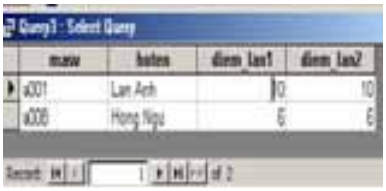
Tạo truy vấn mới và đưa bảng Dsdiem vào tham gia truy vấn 2 lần

Tạo liên kết nội (Inner join) giữa 2 trường Masv

Tạo liên kết nội từ trường Diem_lan1 vào Diem_lan2

Đưa các trường vào vùng lưới QBE và xem kết quả





5.3. Tự động tìm kiếm dữ liệu (Auto lookup)

Khi nhập dữ liệu vào Dynaset, chức năng tự động tìm kiếm dữ liệu cho phép chúng ta chỉ nhập dữ liệu ở các trường của bảng bên "nhiều" (Ở quan hệ 1-∞) còn Acces sẽ tự động tìm kiếm dữ liệu tương ứng trên bảng "1" để hiển thị.

Chức năng Auto Lookup hoạt động trong các truy vấn mà hai bảng tham gia có mối quan hệ 1-∞.

Tạo một truy vấn có sử dụng chức năng Auto Lookup chúng ta thực hiện:

Tạo truy vấn và đưa 2 bảng vào tham gia truy vấn

Đưa trường liên kết cuả bảng bên nhiều vào vùng lưới QBE

Đưa các trường cần hiển thị dữ liệu của bảng bên "1".

☞ ***Chú ý:*** Khi nhập dữ liệu chỉ nhập dữ liệu ở các trường của bảng "nhiều"

6. TÍNH TỔNG TRONG TRUY VẤN CHỌN

Trong thực tế, chúng ta thường có những câu hỏi đặt ra về việc nhóm dữ liệu nào đó, chẳng hạn trong tháng 10 công ty xăng dầu XYZ bán được bao nhiêu lít xăng, tổng thành tiền bao nhiêu?

Trong Access chúng ta có thể thực hiện một số phép tính lên một nhóm bản ghi bằng cách dùng truy vấn tính tổng

Một số phép toán thường sử dụng

Phép toán	Ý nghĩa
Sum	Tính tổng các giá trị của một trường
Avg	Tính giá trị trung bình của một trường
Min	Tính giá trị nhỏ nhất của một trường
Max	Tính giá trị lớn nhất của một trường
Count	Đếm số giá trị khác rỗng có trong một trường
First	Giá trị của trường ở bản ghi đầu tiên trong bảng
Last	Giá trị của trường ở bản ghi cuối cùng trong bảng
Where	Giới hạn điều kiện khi tính tổng
Expression	Trường kiểu biểu thức

☞ ***Chú ý:*** Khi thực hiện truy vấn Total, dữ liệu trong bảng kết quả của nó trình bày không thể chỉnh sửa.

6.1. Tạo truy vấn tính tổng

Tạo truy vấn chọn và đưa các bảng vào tham gia truy vấn

Thực hiện lệnh: View/Totals

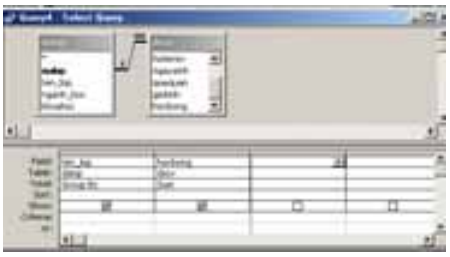
Trong vùng lưới QBE:

Tại hàng Field chọn các trường

Total chọn các phép toán tương ứng.

Criteria: Chọn điều kiện giới hạn tính tổng (Nếu có)

Lưu và thực hiện truy vấn



6.2. Tính tổng của tất cả các bản ghi

Tạo truy vấn chọn.

Đưa các bảng cần thiết vào truy vấn.

Đưa các trường cần thiết vào vùng lưới QBE

Chọn menu View/Totals, dòng Total sẽ xuất hiện trên vùng lưới.

Trong hàng total của mỗi trường chọn phương pháp tính tổng (Sum, Avg, count...). Vì đang tính tổng của tất cả các bản ghi nên không được phép chọn “Group by” ở bất kỳ trường nào.

Chuyển sang DataSheet View để xem kết quả. (View/ Datasheet View)

Ví dụ:

Để quản lý các mặt hàng bán ra trong một cửa hàng người ta sử dụng 2 bảng dữ liệu như sau:

Dshang(**Mahang**, tenhang, dongia)

Dskhach(Mahang, tenkhach, ngaymua, diachi, soluong, thanhtien)

Tạo truy vấn để thống kê xem trong cửa hàng bán bao nhiêu mặt hàng và trung bình đơn giá của mỗi mặt hàng là bao nhiêu?

Tạo truy vấn và đưa bảng Dshang vào tham gia truy vấn

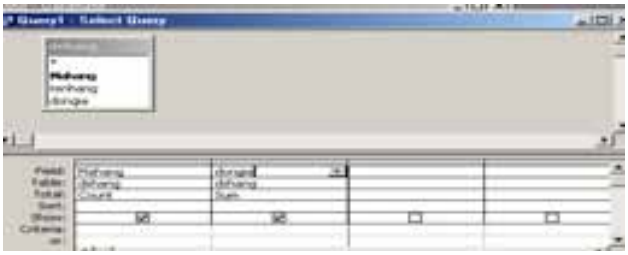
Đưa 2 trường Mahang và dongia vào vùng lưới QBE.

Chọn View/Totals

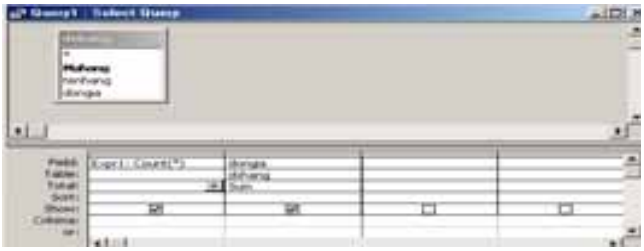
Trong hàng Total của trường Mahang chọn phép toán Count

Trong hàng Total của trường Dongia chọn phép toán Avg

Chọn View/Datasheet View để xem kết quả.



☞ ***Chú ý:*** Có thể đếm số bản ghi bằng cách dùng Count(*)



6.3. Tính tổng trên từng nhóm bản ghi

Trong nhiều trường hợp chúng ta có thể tính toán trên một nhóm bản ghi nào đó. Chúng ta quy định khi thiết kế truy vấn những trường nào sẽ được tính theo nhóm, trường nào sẽ được tính tổng.

Tạo truy vấn

Đưa các bảng tham gia vào truy vấn

Đưa các trường vào vùng lưới

Chọn View/Totals

Tại hàng total

Chọn “Group by” cho trường làm khóa để nhóm

Chọn các phép toán tính tổng (Sum,count..) cho các trường còn lại

Chọn View/Datasheet View để xem kết quả.

Ví dụ:

Tạo một truy vấn để tính tổng soluong, thanhtien của mỗi mặt hàng bán được là bao nhiêu?

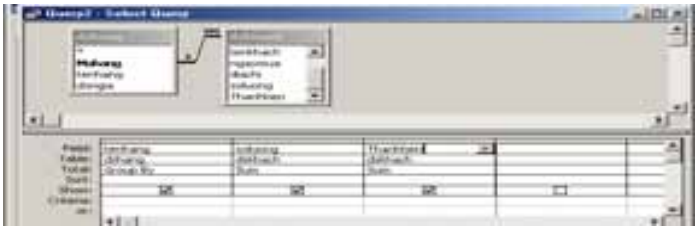
Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

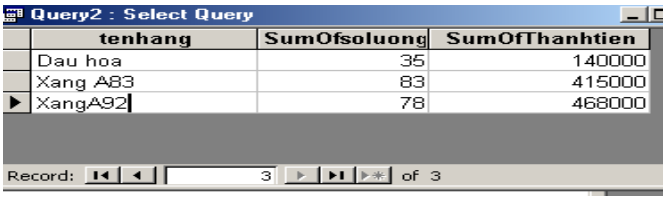
Đưa các trường Tenhang, soluong, thanhtien vào vùng lưới QBE.

Tại hàng Total của trường Tenhang: Chọn phép toán Group by

Tại hàng Total của trường Soluong, thanhtien: Chọn phép toán SUM.



Chọn View/Datasheet View để xem kết quả.

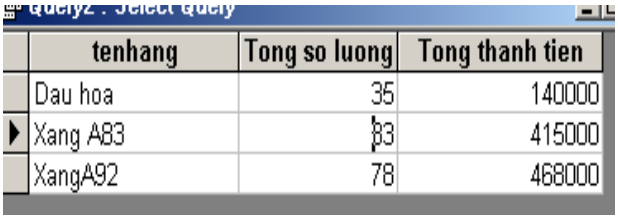


Query2 : Select Query

tenhang	SumOfsoluong	SumOfThanhtien
Dau hoa	35	140000
Xang A83	83	415000
XangA92	78	468000

Record: 3 of 3

Ta có thể thay đổi tiêu đề cột trong khi thực hiện truy vấn tính tổng như sau



tenhang	Tong so luong	Tong thanh tien
Dau hoa	35	140000
Xang A83	83	415000
XangA92	78	468000

6.4. Tính tổng trên nhiều nhóm bản ghi

Access cho phép tính tổng không chỉ trên một mà còn nhiều nhóm bản ghi. Để làm được điều đó chúng ta chọn “Group by” trên nhiều trường và khi thực hiện

Access sẽ theo thứ tự từ trái sang phải trường bên trái là nhóm mức cao hơn, trường kế tiếp theo là nhóm mức thấp hơn.

Ví dụ:

Tạo truy vấn để tính tổng thanhtien của mỗi mặt hàng bán được theo từng năm nào đó?

Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường tenhang, ngaymua, thanhtien vào vùng lưới QBE

Tại hàng Total của trường tenhang chọn phép toán Group by

Tại hàng field của trường ngayban sử dụng hàm year([ngayban]) và tại hàng total chọn phép toán Group by.

Tại hàng Total của trường Thanhtien chọn phép toán Sum.

Thay đổi tiêu đề cột trong truy vấn.

Chọn View/ Datasheet View để xem kết quả

Query4 : Select Query

tenhang	Nam	Tong thanh tien
Dau hoa	1997	24000
Dau hoa	1998	88000
Dau hoa	1999	28000
Xang A83	1997	45000
Xang A83	1998	250000
Xang A83	1999	120000
XangA92	1997	120000
XangA92	1998	168000
XangA92	1999	180000

ecord: 3 of 9

6.5. Lập biểu thức chọn cho các trường dùng để nhóm khi tính tổng

Cũng như với những truy vấn khác, chúng ta có thể lập biểu thức chọn cho truy vấn tính tổng theo từng nhóm.

Để thực hiện công việc này, chúng ta lập biểu thức điều kiện ngay hàng Criteria của trường “group by”.

Ví dụ:

Tạo truy vấn để tính tổng thanhtien của mỗi mặt hàng bán được trong năm 1999.

Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

Đưa các trường tenhang, ngaymua, thanhtien vào vùng lưới QBE

Tại hàng Total của trường tenhang chọn phép toán Group by

Tại hàng field của trường ngayban sử dụng hàm year([ngayban]) và tại hàng total chọn phép toán Group by.

Tại hàng Total của trường Thanhtien chọn phép toán Sum.

Tại hàng Criteria của trường ngaymua: gõ vào giá trị 1999

Thay đổi tiêu đề cột trong truy vấn.

Chọn View/ Datasheet View để xem kết quả

Query4 : Select Query

tenhang	Nam	Tong thanh tien
Dau hoa	1999	28000
Xang A83	1999	120000
XangA92	1999	180000

Record: 1 of 3

6.6. Lập biểu thức chọn để giới hạn những bản ghi.

Trong các phần trước, chúng ta đã biết cách giới hạn các bản ghi trong truy vấn theo một điều kiện nào đó, sự giới hạn này gọi là **giới hạn sau khi tính tổng.**

Bây giờ chúng ta lập biểu thức chọn giới hạn số bản ghi trước khi đưa vào tính tổng trong truy vấn gọi là **giới hạn trước khi tính tổng** .

Cách tạo

Tạo truy vấn mới và đưa các bảng tham gia vào truy vấn

Đưa các trường vào vùng lưới QBE.

Chọn menu View/Total

Thiết lập hàng Total thành Where đối với trường chúng ta muốn dùng để đặt biểu thức điều kiện giới hạn số bản ghi trước khi tính tổng.

Gõ biểu thức điều kiện tại hàng Criteria tương ứng.

Chuyển sang chế độ datasheet view để xem kết quả.

Ví dụ:

Tạo truy vấn để tính tổng soluong, thanhtien của mỗi mặt hàng bán được đối với khách mua hàng có Queuqan ở "**Huế**"

☞ ***Chú ý:*** Trong đa số trường hợp, đặt điều kiện lọc trước và sau khi tính tổng có giá trị khác nhau.

6.7. Dùng truy vấn để cập nhật bản ghi

Khi truy vấn chỉ dựa trên một bảng, hoậchi bảng có quan hệ 1-1 thì tất cả các trường đều có thể thay đổi, cập nhật. Trong trường hợp có nhiều hơn hai bảng tham gia truy vấn mà có quan hệ 1-∞ thì sẽ phức tạp hơn.

6.7.1. Khi nào dữ liệu trong trường có thể sửa đổi được

Bảng sau đây liệt kê các trường hợp khi nào một trường trong kết quả truy vấn hay trong biểu mẫu có thể sửa đổi được.

Loại truy vấn hay truờng	Dữ liệu trong trường có cho phép sửa đổi hay không?
Truy vấn dựa trên 1 bảng	Có
Truy vấn dựa trên nhiều bảng có quan hệ 1-1	Có

Truy vấn dựa trên nhiều bảng có quan hệ 1-∞	Thông thường
Truy vấn Tham khảo chéo	Không
Truy vấn tính tổng	Không
Truy vấn với thuộc tính Unique values được thiết lập thành Yes	Không
Truy vấn hội	Không
Truy vấn chuyển nhượng	Không
Trường kiểu biểu thức	Không
Trường trong bản ghi đã bị xoá hoặc bị khoá bởi một người khác trong môi trường nhiều người sử dụng	Không

6.7.1. Chỉnh sửa bản ghi trong truy vấn dựa trên hai bảng có quan hệ 1-∞

Trong truy vấn dựa trên dữ liệu là hai bảng có quan hệ 1-∞, chúng ta có thể sửa đổi tất cả các trường trừ trường liên kết của bảng bên "1". Tuy nhiên có hai trường hợp mà vẫn có thể sửa đổi dữ liệu trong trường liên kết bên "1" là:

Có thể sửa đổi dữ liệu trường liên kết bên bảng "1" trong trường hợp liên kết ngoại và trương tương ứng của liên kết bên bảng "nhiều" không chứa giá trị.

Có thể sửa đổi dữ liệu trường liên kết bên bảng "1" trong trường hợp đã khai báo thuộc tính tham chiếu toàn vẹn.

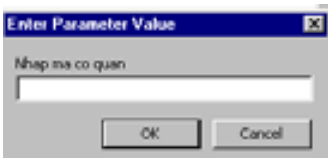
7. TRUY VẤN THAM SỐ (Parameter Query)

7.1. Khái niệm

Nếu thường xuyên chạy cùng một truy vấn, nhưng mỗi lần một tiêu chuẩn khác nhau, thay vì phải thiết kế lại truy vấn sau mỗi lần thực hiện, có thể tiết kiệm thời gian bằng cách tạo truy vấn tham số. Khi thực hiện loại này Access sẽ nhắc nhập điều kiện chọn trong hộp thoại enter parameter Value.

Ví dụ:

Giả sử thường xuyên chạy một truy vấn để liệt kê danh sách nhân viên của một cơ quan nào đó có mã cơ quan nhập vào bất kỳ.



☞ ***Chú ý:*** Nội dung các tham số mà chúng ta nhập vào có thể là hằng (số,chuỗi,ngày..) nhưng không được biểu thức.

7.2. Tạo truy vấn tham số

Tạo truy vấn chọn và đưa các bảng cần thiết vào tham gia truy vấn.

Kéo các trường cần thiết vào vùng lưới QBE.

Tại hàng Criteria gõ vào biểu thức có chứa tham số với chú ý tên tham số phải nằm giữa 2 dấu ngoặc vuông ([ ])

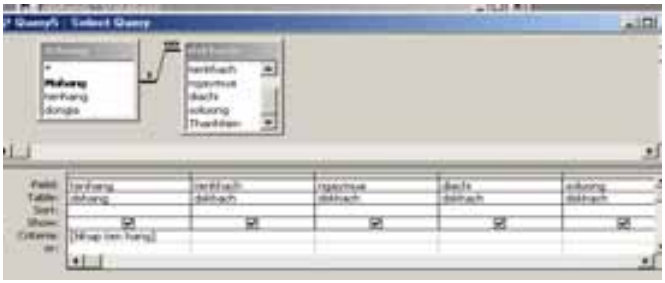
Tên tham số cũng là chuỗi nhắc nhở.Access cho phép có khoảng trắng và độ dài tối đa 255 ký tự.

Quy định kiểu dữ liệu cho tham số: Chọn queries/ parameter query.

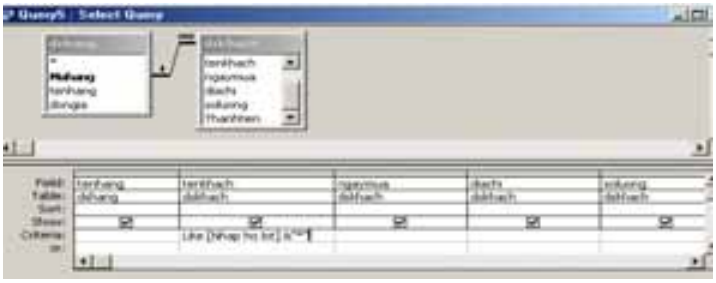
Trong hộp thoại query parameters: Trong mục Parameter chọn tham số, trong mục Data type chọn kiểu dữ liệu tương ứng.

Ví dụ:

Tạo truy vấn để hiển thị danh sách các khách hàng mua một mặt hàng nào đó (mặt hàng được nhập bất kỳ từ bàn phím).



Tạo truy vấn để hiển thị danh sách các khách hàng mua hàng có họ lót được nhập từ bàn phím.



7.3. Truy vấn nhiều tham số

Có thể tạo truy vấn , khi chạy truy vấn nhập nhiều dữ liệu cho điều kiện chọn lựa. Muốn vậy tạo truy vấn nhiều tham số.

Ví dụ:

Tạo một truy vấn hiển thị danh sách các khách hàng mua hàng trong khoảng thời gian nào đó (Thời gian được nhập từ bàn phím).

Tạo truy vấn chọn và đưa các 2 bảng dshang và dskhach vào tham gia truy vấn.

Kéo các trường tenhang, tenkhach, ngaymua vào vùng lưới QBE.

Tại hàng Criteria của trường NGAYSINH chọn:

Between [Từ ngày] and [Đến ngày]

Chọn query/Parameter khai báo kiểu dữ liệu cho 2 tham số là date/time.

Khi chạy truy vấn sẽ cho kết quả sau

7.4. Kết hợp giữa truy vấn tham số và truy vấn tính tổng

Trong nhiều bài toán quản lý người ta thường gặp những yêu cầu như: Hãy thống kê xem mỗi loại hàng trong một tháng nào đó bán được với tổng số lượng là bao nhiêu? Tổng thành tiền là bao nhiêu? (Tháng được nhập từ bàn phím). Vì vậy trước hết chúng ta phải thực hiện truy vấn tính tổng xong mới kết hợp truy vấn tham số.

Ví dụ:

Hãy tạo một truy vấn để hiển thị tổng thanhtien của mỗi mặt hàng bán được trong một năm nào đó (Năm được nhập từ bàn phím).

Tạo một truy vấn chọn, đưa bảng Dshang và dskhach vào tham gia truy vấn.

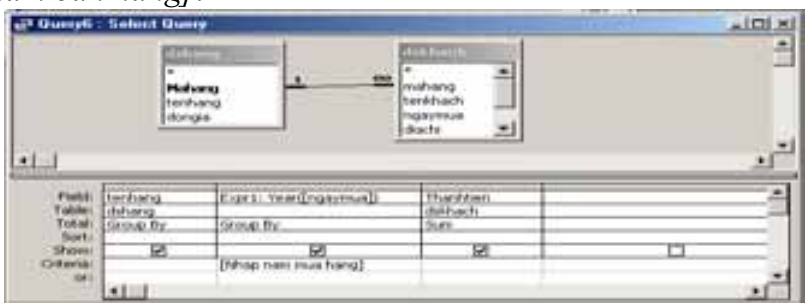
Đưa các trường tenhang, ngayban, thanhtien vào vùng lưới QBE.

Chọn View/Totals

Trong hàng Total: Chọn Group by đối với trường tenhang, và year([ngayban])

Trong hàng Criteria đối với trường Ngaysinh, ta chọn tham số sau:

[Nhap vao nam ban hang].



8. TRUY VẤN THAM KHẢO CHÉO (Crosstab query)

8.1. Khái niệm

Truy vấn tham khảo chéo là loại truy vấn dùng để tóm lược dữ liệu và trình bày kết quả theo dạng như một bảng tính. Truy vấn tham khảo chéo cũng có thể thống kê một khối lượng dữ liệu lớn và trình bày đơn giản hơn do đó thường sử dụng để so sánh dữ liệu.

8.2. Tạo truy vấn tham khảo chéo

Muốn tạo một truy vấn tham khảo chéo chúng ta phải xác định được 3 yếu tố chính: Trường làm tiêu đề cột (Duy nhất 1 trường), trường làm tiêu đề hàng (Có thể nhiều trường), trường tính giá trị (Duy nhất 1 trường).

Cách tạo

Tạo truy vấn chọn và đưa các bảng vào tham gia truy vấn

Đưa các trường vào vùng lưới QBE

Chọn Query/Crosstab

Quy định trường làm tiêu đề cột

Tại hàng Total: Bắt buộc chọn phép toán Group by

Tại hàng Crosstab: Chọn Column heading

Quy định trường làm tiêu đề hàng

Tại hàng Total: Ít nhất một trong các trường phải chọn phép toán Group by

Tại hàng Crosstab: Chọn Row heading

Quy định trường tính giá trị

Tại hàng Total: Chọn phép toán thích hợp

Tại hàng Crosstab: Chọn Value

Ví dụ:

Cho 2 bảng dữ liệu Dstruong(**matruong**, tentruong, sodt)

Danhsach(matruong, hoten, ngaysinh, gioitinh, xeploai)

Tạo một truy vấn Crosstab để phản ánh tổng số lượng sinh viên xếp mỗi loại của trong từng trường bao nhiêu.?

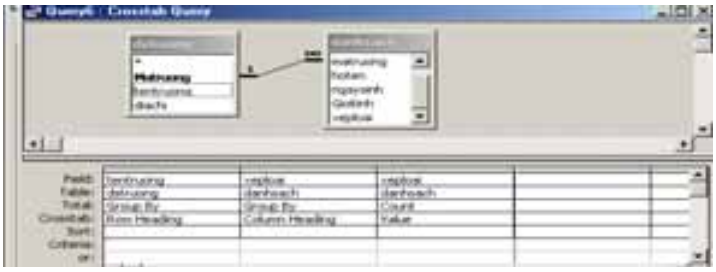
Tạo truy vấn và đưa 2 bảng dữ liệu vào tham gia truy vấn, đưa các trường tentruong và xeploai vào vùng lưới QBE. (Trường Xeploai đưa vào 2 lần)

Chọn Query/ crosstab query

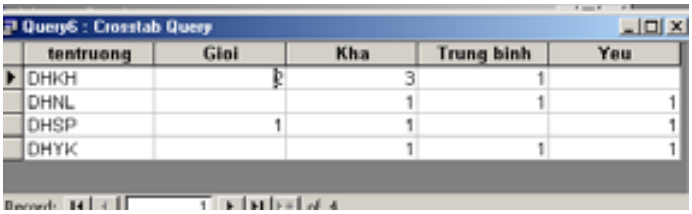
Tại hàng Total của trường tentruong: Chọn phép toán Group by, hàng crosstab: chọn Row heading

Tại hàng Total của trường Xeploai: Chọn phép toán Group by, hàng Crosstab chọn Column heading.

Tại hàng Total của trường Xeploai: Chọn phép toán Count, hàng Crosstab chọn Value.



Chọn View/ Datasheet View để xem kết quả



Query6 : Crosstab Query

tentruong	Gioi	Kha	Trung binh	Yeu
DHKH	2	3	1	
DHNL		1	1	1
DHSP	1	1		1
DHYK		1	1	1

8.3. Định dạng cho tiêu đề cột

Với truy vấn Crosstab, chúng ta có thể can thiệp nhiều hơn về cách trình bày tiêu đề cột trong bảng. Chúng ta có thể thay đổi bằng cách đặt lại thuộc tính Column Heading của truy vấn. Thuộc tính này cho phép chúng ta: Chỉ định sắp xếp các tiêu đề cột.

Muốn định dạng tiêu đề cột thực hiện các bước sau:

Tạo truy vấn Crosstab.

Chuyển sang chế độ Design View.

Mở bảng thuộc tính truy vấn.

Tại hàng Column Heading : Gõ các tiêu đề cột theo thứ tự mà chúng ta muốn

Các giá trị này phải cách nhau bởi dấu chấm phẩy (;)

9. TRUY VẤN HÀNH ĐỘNG

9.1. Các loại truy vấn hành động

Truy vấn hành động giúp người sử dụng tạo bảng mới hay sửa đổi dữ liệu trong các bảng. Có 4 loại truy vấn hành động:

Truy vấn tạo bảng (Make table query): Tạo bảng mới từ một bảng hay nhiều bảng đã tồn tại dữ liệu.

Truy vấn cập nhật (Update query): Dùng để cập nhật dữ liệu cho một hoặc nhiều trường trong bảng dữ liệu.

*Truy vấn xoá (Delete query):*Xoá các bản ghi thoả mãn các điều kiện từ một hay nhiều bảng dữ liệu.

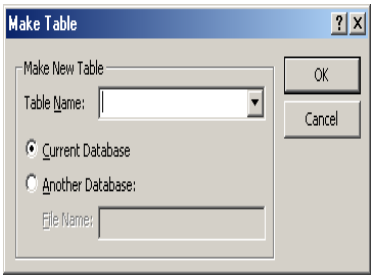
Truy vấn nối (Append query): Nối một số bản ghi từ một hoặc nhiều bảng dữ liệu vào sau một hoặc nhiều bảng dữ liệu khác.

9.2. Truy vấn tạo bảng

Truy vấn tạo bảng sẽ tạo ra một bảng mới bằng cách rút các bản ghi thoã mãn các điều kiện nào đó.

Cách tạo truy vấn

Để tạo truy vấn tạo bảng chúng ta tạo truy vấn chọn và đưa bảng vào tham gia truy vấn. Đưa các trường vào vùng lưới QBE



Chọn query/ make table query

Trong mục Table name: Đặt tên mới cho bảng muốn tạo.

Chọn Current Database: CSDL hiện thời

Another Database: Tạo bảng trong CSDL khác.

Chọn các điều kiện (Nếu có).

9.3. Truy vấn xoá

Truy vấn xoá giúp chúng ta loại bỏ các bản ghi thoả mãn một số điều kiện nào đó

Cách tạo truy vấn

Để tạo truy vấn xoá chúng ta tạo truy vấn chọn và đưa bảng vào tham gia truy vấn.

Chọn query/ Delete query

Trong vùng lưới QBE tại hàng Field chọn các trường cần so sánh với điều kiện xoá

Tại hàng Delete: Chọn phép toán Where

Tại hàng Criteria: Chọn điều kiện xoá

Ví dụ:

Tạo truy vấn để xoá những sinh viên có matruong là "SP"

9.3. Truy vấn cập nhật

Truy vấn này dùng để cập nhật giá trị hoặc sửa đổi giá trị của các trường trong bảng dữ liệu.

Cách tạo truy vấn

Tạo một truy vấn chọn và đưa bảng vào tham gia truy vấn

Chọn Query/Update query

Tại hàng Field: Chọn trường cần cập nhật dữ liệu

Tại hàng Update to: Chọn Biểu thức cần tính giá trị

Tại hàng Criteria: Chọn điều kiện (nếu có).

Ví dụ:

Cho 2 bảng dữ liệu Dsphong(**tenphong**, dongia)

Dskhachtro (tenphong, ngayden, ngaydi, thanhtien)

Tạo truy vấn để cập nhật giá trị cho trường Thanhtien=(ngaydi-ngayden)*dongia

9.4. Truy vấn nối dữ liệu

Truy vấn nối dữ liệu dùng để nối dữ liệu từ một bảng này vào sau một bảng khác.

Cách tạo truy vấn

Tạo truy vấn chọn và đưa bảng dữ liệu vào để nối với bảng khác tham gia truy vấn.

Chọn Queries/Append query

Trong mục Table name: Chọn bảng cần nối vào và chọn OK.

Chọn Current Database: CSDL hiện thời

Another Database: Tạo bảng trong CSDL khác.

Trong vùng lưới QBE của truy vấn tại hàng Field:

Đưa các trường của bảng gốc vào.

Trong hàng Append to: Đưa các trường tương ứng của bảng cần nôi và đặt điều kiện nếu cần thiết.

☞ ***Chú ý***

Trong truy vấn nối dữ liệu thì các trường được nối với nhau tương ứng phải cùng kiểu dữ liệu. Nếu các trường tương ứng không có cùng kiểu dữ liệu thì sẽ không được nối. Nếu các trường có Field size không phù hợp thì tuỳ theo việc nối dữ liệu mà sẽ cắt bớt hoặc thêm vào ký tự trắng.

Ch□□ng 4 NGÔN NGỮ SQL

1. GIỚI THIỆU VỀ NGÔN NGỮ SQL

SQL là một ngôn ngữ dùng để truy xuất dữ liệu, cập nhật, thay đổi và quản lý các CSDL quan hệ.

Khi tạo một truy vấn thì ACCESS tự động xây dựng các câu lệnh SQL tương ứng.

Một số truy vấn của ngôn ngữ SQL như sau:

Truy vấn hội (Union Query)

Truy vấn chuyển nhượng (pass through Query)

Truy vấn định nghĩa dữ liệu (Data Definition Query)

Truy vấn con (Sub Query)

☞ ***Chú ý***

Trong SQL mỗi câu lệnh có thể được viết trên nhiều hàng và kết thúc mỗi câu lệnh là dấu chấm phẩy (;)

2. SQL XỬ LÝ TRÊN BẢNG DỮ LIỆU

2.1. Tạo bảng mới

Cú pháp

Create table <Table name>(<Field1> <Type>[(Size)][, <Field2> <Type>[(Size)].....

[Constraint < Primary key name> primary key <Fieldname key>]

[Constraint <Index name> Unique <Field name Index>]

Chức năng: Tạo cấu trúc của một bảng mới.

Giải thích chức năng các tham số

Table name: Tên bảng cần tạo

Fieldname1, Fieldname2....: Các trường trong bảng cần tạo, tối thiểu 1 trường.

Type: Các kiểu dữ liệu tương ứng.

Size : Kích thước của trường

Primary key name : Tên khoá chính

Fieldname key : Trường làm khoá chính

Index name : Tên chỉ mục

Fieldname Index : Trường làm chỉ mục

Constraint.....Primary key : Dùng để thiết lập khoá chính

Constraint..... Unique: Thiết lập chỉ mục (Không trùng nhau)

☞ ***Chú ý:*** Đối với các kiểu dữ liệu khi sử dụng trong SQL

Kiểu dữ liệu chuẩn	**Khai báo tương ứng trong SQL**
Text	Text(Size)
Byte	Byte
Integer	Short
Long Integer	Long
Single	Single
Double	Double
Date/Time	Datetime
Yes/No	Yesno
OLE Object	OLE Object
Currency, Memo, Counter	Currency, Memo, Counter

Ví dụ : Tạo bảng DSTRUONG có cấu trúc sau

Fieldname	Data Type	size
Matruong	Text	2
Tenruong	Text	20
SODT	Text	9

Create table dstruong(matruong text(2), tentruong text(20), Sodt text(9));

Ví dụ Tạo bảng DSHOCVIEN có cấu trúc sau

Fieldname	Data Type	size	
Mahv	Text	4	(Khoá chính)
Tenhv	Text	30	
Ngaysinh	Date/time		

SODT Text 9 (Lập chỉ mục)

Create table dshocvien(mahv text(4), tenhv text(30), ngaysinh Datetime, sodt text(9), Constraint khoa primary key (mahv), Constraint chimuc unique (sodt));

☞ ***Chú ý:*** Nếu tên trường có ký tự trắng hoặc tên bảng, tên khoá chính, tên chỉ mục thì phải đặt trong cặp dấu [...]

Ví dụ

Create table [Bang NV] ([Ma nv] text(2), [ho va ten] text(30));

2.2. Thay đổi cấu trúc của bảng

2.2.1.Thêm hoặc loại bỏ trường

Cú pháp

Alter table <Table name>[add column <fieldname><type>]

[Drop column <Field name>]

[Add Constraint <Index name> unique <fieldname>]

[Drop Constraint <Index name>]

Chức năng: Thay đổi cấu trúc của bảng

Giải thích:

ADD Column..: Thêm trường vào bảng

Drop column..: Loại bỏ trường ra khỏi bảng

Add Constraint.....: Thêm tên chỉ mục

Drop Constraint..... Loại bỏ tên chỉ mục

Ví dụ: Giả sử đã tồn tại bảng MATHANG cấu trúc sau

Fieldname	*Data Type*	*size*
MAHANG	Text	4
TENHANG	Text	20
GIA	Integer	
MAXN	Text	2
Ngaynhap	Date/time	

Thêm trường SOLUONG có kiểu byte vào bảng MATHANG

Alter Table mathang soluong byte;

Ví dụ : Thêm chỉ mục có ten cmx cho trường MAXN

Alter table mathang add constraint cmx unique Maxn

Ví dụ : Loại bỏ chỉ mục cmx

Alter table mathang drop constraint cmx

Ví dụ : Loại bỏ trường ngaynhap ra khỏi bảng MATHANG

Alter table mathang drop column ngaynhap

2.2.2. Loại bỏ chỉ mục

Cú pháp

Drop Index <Index name> on <Table name>

Chức năng: Loại bỏ 1 chỉ mục nào đó.

2.3. Xoá bảng

Cú pháp

Drop table <Table name>

Chức năng: Xoá bảng dữ liệu nào đó.

Ví dụ: Xoá bảng MATHANG

Drop table MATHANG

3. SQL XỬ LÝ TRÊN TRUY VẤN

3.1.Truy vấn chọn (Select query)

Cú pháp

Select <Scope> <Fieldname1> [AS <New name>].....

From <Table name>

[Where <Condition>]

Chức năng: Tạo truy vấn chọn

Trong đó:

Scope: Phạm vi (Mặc định là ALL, Top n: Lấy n bản ghi đầu tiên)

Nếu có AS <New name> thì sẽ thay thế tên cho Fieldname tương ứng.

Table name: Tên bảng cần lấy dữ liệu.

Condition: Điều kiện để hạn chế dữ liệu.

Ví dụ: Cho bảng dữ liệu DOCGIA sau

Fieldname	Data Type	Size	Description
MADG	Text	2	Mã độc giả (Khoá chính)
MASACH	Text	4	
HOTEN	Text	30	
QUEQUAN	Text	30	
NGAYSINH	Date/time	8	
NGAYMUON	Date/time	8	

Chọn 2 trường MADG và HOTEN

Select MADG, hoten

From docgia;

Chọn 2 trường Masach và hoten mà chỉ những masach bắt đầu là T

Select MADG, hoten

From docgia where masach like "T*";

Chọn Hoten, quequan, madg cho những độc giả có quê quán ở Huế và đổi tên trường hoten thành Họ và tên

Select hoten AS [Họ và tên], queuqan, madg

From docgia where quequan="Huế";

Chọn những độc giả mượn sách trong tháng 8 hoặc năm 1999.

Select * from docgia where month([ngaymuon])=8 or year([ngaymuon])=1999;

3.2. Truy vấn tính tổng (Total query)

Cú pháp

Select .....from.....[where < Condition>]

Group by [Group fieldname]

[Having <Group Condition>];

Chức năng: Tạo một truy vấn tính tổng.

Ví dụ: Cho bảng dữ liệu BANHANG có cấu trúc

Fieldname	Data type	size
Mahang	text	2
Soluong	integer	

Ngayban Date/time 8

Tạo một truy vấn để thống kê xem mỗi loại hàng bán được với số lượng là bao nhiêu?

```
Select mahang, sum([soluong])
From banhang
Group by mahang;
```

Tạo truy vấn để thống kê xem mỗi loại hàng trong thàng 7 bán được với số lượng bao nhiêu? Chỉ hiển thị những loại hàng mà số lượng bán trên 20.

```
Select mahang, sum([soluong])
From banhang
where month([ngayban])=7
Group by mahang
 having sum([soluong])>20;
```

3. 3.Truy vấn tham khảo chéo

Cú pháp

Transform <Value Express>

Select.....From.....Where.....

Group by <Row Heading Field>

Pivot <Column heading Field>

Chức năng: Tạo truy vấn tham khảo chéo

Ví dụ

```
Transform sum([soluong])
select   Mahang, tenhang, sum([soluong]) from dskhang
Group by  tenhang
Pivot Mahang;
```

3.4. Truy vấn tạo bảng

Cú pháp

Select <Field select> into <New Table name)

From <Old Table name>

[Where <Condition>]

Chức năng: Tạo một truy vấn tạo bảng

Ví dụ

Select Mahang, tenhang into Luu

From Dskhang

Where Mahang Like "A*";

3.5. Truy vấn nối dữ liệu

Cú pháp

Insert into <append Table name>

Select <field select>

From <Table name>

[Where <Condition>]

Chức năng: Tạo truy vấn nối dữ liệu

Nếu chỉ thêm 1 bản ghi với các giá trị cụ thể thì ta thực hiện câu lệnh

Insert into <Table name and Field list>

values <append values>

Ví dụ:

Insert into luu1(Hoten, quequan)

Values ("Nguyen an", "Hue")

3.6. Truy vấn cập nhật dữ liệu

Cú pháp

Update <Update Table name>

Set <Field name>=<Express>

[Where <Condition>]

Chức năng: Tạo một truy vấn dùng để cập nhật dữ liệu

Ví dụ:

Update dssv

set [hocbong]=[hocbong]+200000

Where Uutien="1";

3.7. Truy vấn xoá

Cú pháp

Delete <Table.>*

From <Delete Table name>

[Where <Condition>]

Chức năng: Dùng để tạo một truy vấn xoá các bản ghi trong bảng theo một hoặc nhiều điều kiện nào đó.

☞ ***Chú ý:*** nếu mệnh đề From chỏ có 1 bảng duy nhất thì không cần liệt kê các bảng trong mệnh đề DELETE.

Ví dụ:

Có 2 bảng dữ liệu DSHS và DSDTHI có quan hệ 1-1 trên trường MAHS. Hãy xoá những học sinh có điểm thi <5 trong bảng DSHS và DSDTHI.

Delete DSHS.*

From DSHS INNER JOIN DSDTHI ON DSHS.MAHS=DSDTHI.MAHS

Where diem<5.;

3.8. Tạo mối quan hệ giữa các bảng

Muốn tạo một truy vấn để truy xuất dữ liệu từ 2 hay nhiều bảng thì phải tạo các mối quan hệ giữa các bảng đó.

Cú pháp

..... From <Table name 1> inner join <Table name 2> ON <Table name 1>.<Field name1>=<Table name 2>.<Field name 2>.....

Hoặc

.....From <Table name 1> Left join/ Right join <Table name 2> ON <Table name 1>.<Field name1>=<Table name 2>.<Field name 2>.....

Ví dụ:

Tạo truy vấn gồm : Matruong, tentruong, hoten từ 2 bảng Dstruong và DSHS dựa vào trường liên kết Matruong.

Select Matruong, tentruong, hoten From Dstruong inner join dshs on Dstruong.matruong=dshs.matruong;

Ví dụ:

Cho 3 bảng dữ liệu Dstruong(**Matruong**, tentruong)

Dskhoa(Matruong, tenkhoa, **Makhoa**)

DSSV(Makhoa, Hoten, Ngaysinh, quequan)

Tạo một truy vấn để hiển thị danh sách sinh viên thuộc mỗi khoa của mỗi trường

Select Distinctrow

Matruong, tentruong, tenkhoa, hoten

From Dstruong inner join (Dskhoa inner join DSSV ON Dskhoa.Makhoa=DSSV.Makhoa)

ON Dstruong.Matruong=Dskhoa.Matruong;

☞ ***Chú ý***

Có thể sử dụng liên kết ngoại trái Left join hoặc phải Right join

3.9. Truy vấn con (Sub query)

Truy vấn con là một mệnh đề Select.....From.....Wheres được lồng ghép vào một trong các mệnh đề sau:

Select.....From.....Where

Select.....Into.....

Insert.....Into.....

Delete.....

Update.....

Cú pháp

Select.....From.....Where.....

<Biểu thức so sánh> ANY|SOME|ALL <Mệnh đề truy vấn con>

<Biểu thức tìm kiếm> IN | NOT IN <Mệnh đề truy vấn con>

EXIST | NOT EXISTS <Mệnh đề truy vấn con>;

Chức năng: Tạo một truy vấn con

Giải thích các tham số

<Biểu thức so sánh>: Là một biểu thức và một phép toán so sánh.

<Biểu thức tìm kiếm>: Là một biểu thức mà tập hợp kết quả của truy vấn con sẽ được tìm kiếm.

<Mệnh đề truy vấn con>: Là dạng mệnh đề ở trong cú pháp và đặt giữa hai dấu ( ).

ANY, SOME: Các bản ghi trong truy vấn chính thoả mãn điều kiện so sánh với bất kỳ hoặc một vài các bản ghi nào truy xuất được từ truy vấn con.

ALL: Các bản ghi trong truy vấn chính thoả mãn với điều kiện so sánh với tất cả bản ghi nào truy xuất được từ truy vấn con.

IN: Các bản ghi trong truy vấn chính mà có tồn tại một vài bản ghi trong truy vấn con có giá trị bằng nó.

NOT IN: Các bản ghi trong truy vấn chính mà không tồn tại một vài bản ghi trong truy vấn con có giá trị bằng nó.

EXISTS (NOT EXISTS): Phép so sánh True/ False để xác định nhận truy vấn con có kết quả là bản ghi nào không.

Ví dụ:

Cho 2 bảng dữ liệu KHO(Mahang, Tenhang, Dongia)

NKBAN (Mahang, Hoten, Ngaymua, Giamgia, Dongia)

Tìm tất cả những mặt hàng mà đơn giá lớn hơn vài mặt hàng được bán với Giamgia là 20%.

*Select * From Kho*

Where dongia> ANY (select dongia From NKBAN Where giamgia=20%);

Tìm những mặt hàng bán ra với giảm giá >=10%.

*Select * from Kho*

Where Mahang IN (Select mahang From NKBAN Where giamgia>=0.1);

Ví dụ:

Cho 2 bảng danh sách DSKH(MAKHACH, HOTEN, QUEQUAN, SDT)

DATHANG(MAKHACH, SOLUONG, NGAYDAT)

Tìm những người khách đặt hàng trước 10/10/99 bao gồm Hoten, Quequan.

Select Hoten, quequan From DSKH

Where Makhach IN (Select makhach from DATHANG Where NGAYDAT<=#10/10/99#;.

3.10. Truy vấn hội (Union Query)

Dùng để nối (Kết hợp) dữ liệu các trường tương ứng từ 2 hay nhiều bảng hoặc truy vấn vào trường.

Cú pháp: *Select .....From.....Where.....*

UNION | UNION ALL

Select.....

Chức năng: Tạo truy vấn hội

Giải thích:

UNION: Không muốn các bản ghi trùng nhau hiển thị

UNION ALL: Hiển thị các bản ghi trùng nhau

Ví dụ:

Cho 2 bảng dữ liệu DHSVIEN(Hoten, Lop, Matruong, Diachi)

CDHSVIEN(Hoten, Lop, Matruong, Diachi)

Sử dụng truy vấn hội để liệt kê Hoten, Lop, Matruong của sinh viên 2 hệ (Địa học, Cao đẳng).

Select Hoten, Lop, Matruong from DHSVIEN

UNION Select Hoten, Lop, Matruong from DHSVIEN

Liệt kê Hoten, Lop của những sinh viên 2 hệ và có quê quán ở Huế

Select Hoten, Lop from DHSVIEN

UNION Select Hoten, Lopfrom DHSVIEN Where Diachi=”Huế”;

Ch□□ng 5 BIỂU MẪU

Từ trước đến nay chúng ta vẫn làm việc một cách đơn điệu với các bảng, truy vấn với cách trình bày dữ liệu hiệu quả nhưng không đẹp mắt. Với biểu mẫu (form) trong Access sẽ giúp chúng ta khắc phục điều này. Biểu mẫu trong Access rất linh động, chúng ta có thể dùng biểu mẫu để nhập, xem, hiệu chỉnh dữ liệu. Hoặc là dùng biểu mẫu để tạo ra các bảng chọn công việc làm cho công việc của chúng ta thuận lợi và khoa học hơn. Hoặc dùng biểu mẫu để tạo ra các hộp thoại nhằm thiết lập các tùy chọn cho công việc quản lý của mình.

1. KHÁI NIỆM VỀ BIỂU MẪU

Nếu chúng ta đã quen điền các tờ biểu, mẫu trong cuộc sống hàng ngày thì chúng ta có thể hình dung một biểu mẫu trong Access cũng vậy. Một biểu mẫu trong Access định nghĩa một tập dữ liệu chúng ta muốn lấy và từ đó đưa vào CSDL. Cũng vậy biểu mẫu cùng có thể dùng để xem xét dữ liệu hay in ra máy in.

Trong môi trường của Hệ QTCSDL Access chúng ta có thể thiết kế các biểu mẫu có hình thức trình bày đẹp, dễ sử dụng và thể hiện đúng các thông tin cần thiết. Chúng ta có thể đưa vào biểu mẫu các đối tượng như văn bản, hình ảnh, đường vẽ két hợp với các màu sắc sao cho biểu mẫu của chúng ta đạt được nội dung và hình thức trình bày ưng ý nhất. Hình thức và cách bố trí các đối tượng ra sao trên biểu mẫu hoàn toàn tùy thuộc vào khả năng thẩm mỹ và ng khiếu trình bày của chúng ta.

2. TÁC DỤNG VÀ KẾT CẤU CỦA BIỂU MẪU

2.1.Tác dụng của biểu mẫu

Biểu mẫu cung cấp một khả năng thuận lợi để hiển thị dữ liệu. Chúng ta có thể xem mọi thông tin của một bản ghi thay vì ở chế độ Datasheet nghèo nàn trước đây bằng chế độ Form View, một phương cách tiên tiến hơn.

Sử dụng biểu mẫu tăng khả năng nhập dữ liệu, tiết kiệm thời gian và ngăn ngừa các lỗi do đánh sai. Chẳng hạn thay vì gõ vào các giá trị của tất cả các trường chúng ta có thể tạo những danh sách (gọi là combo box) để chọn trên biểu mẫu (đây là phương cách áp dụng rất hiệu quả để tránh đánh sai dữ liệu).

Biểu mẫu cung cấp một hình thức trình bày hết sức tiện nghi để xem, nhập và hiệu chỉ các bản ghi trong CSDL. Access cung cấp các công cụ thiết kế biểu mẫu hỗ trợ rất đắc lực cho chúng ta trong việc thiết kê những biểu mẫu dễ sử dụng mà lại có thể tận dụng được các khả năng:

Hình thức thể hiện dữ liệu đẹp, trình bày lôi cuốn với các kiểu font và hiệu ứng đồ họa đặc biệt khác ...

Quen thuộc với người sử dụng vì nó giống các biểu mẫu trên giấy thông thường.

Có thể tính toán được.

Có thể chứa cả biểu đồ.

Có thể hiển thị dữ liệu từ nhiều bảng (hoặc truy vấn)

Tự động hóa một số thao tác phải làm thường xuyên.

2.2. Kết cấu của biểu mẫu

Các thông tin trên biểu mẫu có thể lấy dữ liệu từ một bảng hay truy vấn nào đó, nhưng cũng có thể độc lập đối với cả bảng lẫn truy vấn, chẳng hạn như các đối tượng đồ họa. Dáng vẻ trình bày của biểu mẫu được thực hiện trong quá trình thiết kế.

Tất cả các thông tin thể hiện trên biểu mẫu được chứa trong những đối tượng gọi là *điều khiển* (control). Điều khiển có thể dùng để thể hiện dữ liệu hoặc thực hiện các hàng động hoặc trang trí cho biểu mẫu.

Một số điều khiển được buộc vào với các trường của bảng hay truy vấn, gọi là bảng cơ sở hay truy vấn cơ sở. Do đó chúng ta có thể dùng biểu mẫu để nhập dữ liệu vào

các trường hay lấy dữ liệu từ các trường đó ra để xem. Ví dụ dùng Text box để nhập hay hiển thị chuỗi và số, dùng Object frame để thể hiện hình ảnh.

Một số điều khiển khác trình bày thông tin được lưu trữ trong thiết kế bảng. Ví dụ dùng Label (nhãn) để thể hiện thông tin có tính chất mô tả; đường và các hình khối để tổ chức dữ liệu và làm biểu mẫu có hình thức hấp dẫn hơn.

3. TẠO BIỂU MẪU

3.1. Tạo biểu mẫu tự động với Autoform

Access cung cấp chức năng Autoform cho phép chúng ta tạo biểu mẫu dựa trên các bảng hoặc truy vấn đã được xây dựng trước đó.

Cách tạo

Trong cửa sổ Database, chọn form, chọn New

Trong mục *Choose the table or query Where the object's data comes from:*

Chọn bảng hoặc truy vấn làm nguồn dữ liệu cho form.

Chọn **Autoform Columnar**: Nếu muốn tạo lập biểu mẫu dạng cột, trong đó mỗi trường trong bảng hay truy vấn là một dòng.

Chọn **Autoform Tabular**: Nếu muốn tạo lập biểu mẫu dạng hàng, trong đó mỗi trường trong bảng hay truy vấn là một cột và một bản ghi trong một dòng.

Chọn **Autoform Datasheet**: Nếu muốn tạo lập biểu mẫu theo dạng bảng, trong đó mỗi cột tương ứng một trường và mỗi dòng là một bản ghi.

Chọn OK.

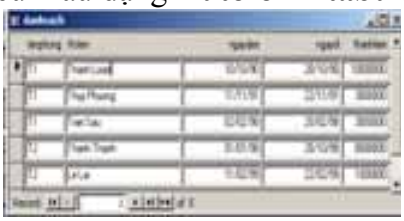
Ví dụ

Cho bảng Danhsach(Tenphong, hoten, ngayden, ngaydi, thanhtien)

Hãy thiết kế biểu mẫu dựa trên chức năng Autoform sử dụng bảng Danhsach làm nguồn dữ liệu.

Biểu mẫu dạng **Autoform Columnar**

Biểu mẫu dạng **Autoform tabular**



Biểu mẫu dạng **Autoform Datasheet**

3.2. Tạo biểu mẫu sử dụng Wizard

Tạo biểu mẫu sử dụng công cụ Autoform thì Access không cho phép người sử dụng can thiệp vào quá trình tạo biểu mẫu, chẳng hạn như hạn chế số trường..... thì Form Wizard cho phép người sử dụng can thiệp vào quá trình tạo biểu mẫu.

Cách tạo

Trong cửa sổ Database chọn Form, chọn New

Trong mục *Choose the table or query Where the object's data comes from:*

Chọn bảng hoặc truy vấn làm nguồn dữ liệu cho form.

Chọn Form Wizard

Chọn OK

Trong mục Avaiable Field: Chọn các trường đưa vào biểu mẫu, nhấn nút >>

Chọn nút **Next**.

Chọn *Columnar* : Biểu mẫu hiển thị theo dạng cột

Tabular : Biểu mẫu hiển thị theo dạng hàng

Datasheet : Biểu mẫu hiển thị theo dạng bảng

Justified : Biểu mẫu hiển bình thường (đều).

Chọn **Next**

Chọn loại biểu mẫu

Chọn **Next**

Đặt tiêu đề cho Form

Chọn *Open the form to view or enter information*: Nếu muốn mở Form sau khi chọn Finish.

Chọn *Modify the form's design:* Nếu muốn form ở dạng thiết kế.

Chọn **Finish** .

Lưu form.

3.3. Tạo biểu mẫu không sử dụng Wizard (Do người sử dụng tự thiết kế)

Tạo biểu mẫu sử dụng công cụ Autoform và Form wizard người sử dụng có thể nhanh chóng thiết kế các biểu mẫu nhờ vào các đặc tính hỗ trợ của Access. Nhưng đối với hai cách trên chỉ cung cấp một số hạn chế các phương án xây dựng biểu mẫu mà không thoã mãn yêu cầu của người sử dụng khi muốn thiết kế biểu mẫu teho ý của riêng mình. Do đó người sử dụng phải tự thiết kế một biểu mẫu không cần sự hỗ trợ của Access.

Cách tạo

Trong cửa sổ Database chọn Form, chọn New

Chọn bảng dữ liệu hoặc truy vấn làm nguồn dữ liệu cho form, chọn **OK**

Xây dựng các điều khiển cho biểu mẫu (Đưa các trường trong bảng dữ liệu vào biểu mẫu).

Thiết lập các thuộc tính cho các điều khiển.

Lưu biểu mẫu.

4. CÁC CHẾ ĐỘ HIỂN THỊ VÀ CÁC THÀNH PHẦN CỦA BIỂU MẪU

4.1. Các chế độ hiển thị

Có 4 chế độ hiển thị của biểu mẫu

4.1.1.Chế độ Design View

Dùng để tạo biểu mẫu mới hay thay đổi cấu trúc của một biểu mẫu đã tồn tại.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ chọn Design (Có

thể click chuột phải rồi chọn Design).

Khi đang ở chế độ Form view: Chọn View/ Form Design

4.1.2. Chế độ Form View

Dùng để nhập, thay đổi và xem dữ liệu. Trong chế độ Form View người sử dụng có thể xem tất cả các trường của một bản ghi tại một thời điểm.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ chọn Open (Có thể click chuột phải rồi chọn Open).

Khi đang ở chế độ Design view: Chọn View/ Form View.

4.1.3. Chế độ hiển thị biểu mẫu dưới dạng bảng (Datasheet View)

Dùng để nhập, thay đổi và xem dữ liệu trong biểu mẫu theo dạng bảng biểu.

Để mở chế độ hiển thị dạng Datasheet View chúng ta thực hiện như sau:

Khi đang ở chế độ Design View: Chọn View/Datasheet.

Khi đang ở chế độ Form View: Chọn View/ Datasheet View.

4.1.4. Chế độ hiển thị Print Preview

Dùng để xem biểu mẫu trước khi quyết định in ấn. Trong chế độ Print Preview sẽ duy trì hình dạng trình bày dữ liệu đã được thiết kế trước đó.

Khi đang ở trong cửa sổ Database: Chọn form /chọn tên form/ Chọn File/Print Preview.

4.2. Các thành phần biểu mẫu trong chế độ Design View

Khi muốn thiết kế biểu mẫu thì người sử dụng phải làm việc trong chế dộ Design View khi đó biểu mẫu có các thành phần chính sau:

Thước(Ruler): Điều chỉnh kích thước của các điều khiển.

Tiêu đề form (form header):Sử dụng để trình bày tiêu đề của form, tiêu đề form luôn được trình bày phần trên cùng, đầu tiên của biểu mẫu và trang in biểu mẫu.

Chân form (Form Footer): Sử dụng để trình bày chân của form, chân form luôn được trình bày phần dưới cùng, xuất hiện cuối biểu mẫu và trang in biểu mẫu.

Tiêu đề trang (Page header): Sử dụng để chứa tiêu đề trang

Chân trang (Page footer): Sử dụng để chứa chân trang nhưng xuất hiện phần trước của Form footer trong trang biểu mẫu in.

☞ ***Chú ý***

Page header và Page footer chỉ xuất hiện trong trang biểu mẫu in nên chúng không có những tính chất thông thường như Form header và Form footer.

Chọn View/ Page header/ footer (Nếu 2 thành phần này chưa xuất hiện trên biểu mẫu).

Chi tiết form (Detail): Đây là phần rất quan trọng chứa các điều khiển nhằm trình bày các dạng dữ liệu từ các bảng dữ liệu hoặc các truy vấn. Các loại điều khiển có thể là điều khiển buộc, không buộc hoặc tính toán.

5. CÁC LOẠI ĐIỀU KHIỂN

Tất cả thông tin trên biểu mẫu được chứa trong những đối tượng gọi là điều khiển (Control). Điều khiển có thể dùng để thể hiện dữ liệu, thực hiện các hành động hoặc thiết kế biểu mẫu đẹp mắt. Trong ACCESS hệ thống định nghĩa một số loại điều khiển như sau:

Điều khiển nhãn (Label).

Điều khiển hộp văn bản (Text box).

Điều khiển nhóm lựa chọn (Option group).

Điều khiển loại hộp Combo (Combo box) và hộp danh sách (List Box).

Ngoài ra còn có một số điều khiển khác như command button.....

Khi tạo lập điều khiển, chúng ta thường xác định hình thức dữ liệu trình bày trong chúng. Có những điều khiển lấy dữ liệu từu các trường trong bảng hay truy vấn, có điều khiển chỉ dùng vào mục đích trang trí, làm tiêu đề, có những điều khiển lấy dữ liệu từ một biểu thức nào đó. Vì vậy người ta phân ra thành ba nhóm điều khiển chính:

Điều khiển buộc (Bound control)

Điều khiển không buộc (Unbound control)

Điều khiển tính toán (Caculated control)

5.1. Điều khiển bị buộc (bound), không buộc (unbound) và tính toán được (calculated).

Khi tạo một điều khiển trong biểu mẫu thì phải xác định nó lấy dữ liệu từ nguồn nào để thể hiện.

Ví dụ

Tạo một điều khiển loại hộp văn bản (Text box) để hiển thị tên các mặt hàng, chúng ta phải chỉ định cho điều khiển lấy dữ liệu trong trường TEN_HANG của bảng MAT_HANG. Hộp văn bản này gọi là bị buộc.

Điều khiển cũng có thể thể hiện những thông tin không có trong CSDL (Không bị buộc).

Ví dụ: Tạo tiêu đề cho biểu mẫu...

Tóm lại

Điều khiển bị buộc (Bound Control) là điều khiển mà nguồn dữ liệu của nó lấy từ một trường trong bảng hoặc truy vấn Trong biểu mẫu dùng điều khiển buộc vào các trường để hiển thị nội dung hoặc cập nhật các trường của CSDL, các giá trị cập nhật có thể là: Văn bản, Date,Num ber, yes/No, Picture, chart trong đó dạng văn bản là phổ biến nhất.

Điều khiển không bị buộc (Unbound Control) là điều khiển không lấy dữ liệu từ một nguồn nào cả là điều khiển không bị buộc. Dùng điều khiển không buộc để trình bày thông tin không có trong các bảng hay rút được từ truy vấn.

Điều khiển tính toán (Calculated Control) là điều khiển mà nguồn dữ liệu của nó không phải là một trường mà là một biểu thức gọi là điều khiển tính toán (Calculated Control). Chúng ta qui định giá trị xuất hiện trong điều khiển bằng cách lập biểu thức cho nó. Biểu thức này là nguồn dữ liệu của điều khiển .Trong biểu thức có thể dùng các toán tử (+, -,=...) với tên điều khiển.

Ví dụ: Tạo một điều khiển THANHTIEN=SOLUONG*DONGIA

5.2. Tạo điều khiển loại hộp văn bản (text box)

Text box có thể là một điều khiển bị buộc, không buộc hoặc tính toán.

Tạo hộp văn bản bị buộc (Bound Text box).

Chúng ta buộc điều khiển Text box vào một trường bằng cách chỉ định điều khiển đó lấy dữ liệu trên trường nào. Chọn trường để buộc vào điều khiển bằng cách Click biểu

tượng Field List để mở danh sách các trường của bảng hay truy vấn làm nền tảng cho biểu mẫu. Theo mặc định như vậy thì hệ thống sẽ tạo một điều khiển loại Text box.

Một cách khác dùng hộp dụng cụ Toolbox để tạo điều khiển và sau đó gõ tên trường muốn buộc vào hộp văn bản.

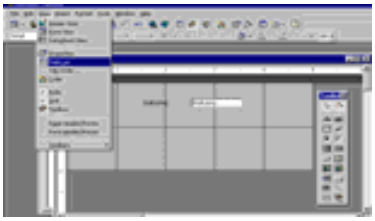
Mở hộp danh sách trường

Mở biểu mẫu trong chế độ Design (Tạo biểu mẫu trước).

Trong cửa sổ Database Chọn Form/New (Cọn bảng hoặc truy vấn làm nền cho biểu mẫu).

Chọn View/Field List hoặc click vào biểu tượng Field List trên thanh công cụ).

Tạo một Text Box bị buộc



Từ danh sách trường chọn một hoặc nhiều trường kéo và đưa vào biểu mẫu.

5.3. Tạo một điều khiển khác dùng hộp công cụ

Dùng hộp công cụ (Toolbox) để tạo những điều khiển không buộc (Unboud control) hoặc để tính toán. Đặc biệt dùng các tính năng của hộp này để tạo những điều khiển bị buộc khác ngoài buộc khác ngoài (Text box). Để bật hộp công cụ ta chọn View/Toolbars.

Tạo điều khiển dùng hộp công cụ

Click vào biểu tượng công cụ tương ứng với điều khiển muốn tạo.

Tạo điều khiển bị buộc bằng cách chọn một trường trong danh sách trường (Field List) và kéo vào biểu mẫu.

Hoặc tạo điều khiển không buộc hay dùng để tính toán bằng cách click vào một vị trí trên biểu mẫu.

Tạo điều khiển dùng để tính toán

Nếu muốn trình bày kết quả của một phép toán trong biểu mẫu, khai báo nguồn dữ liệu của điều khiển là một biểu thức. Sau này mỗi lần mở biểu mẫu, ACCESS tính toán lại kết quả của vùng dữ liệu được cập nhật mới nhất từ các bảng.

Tạo điều khiển nhãn (Label Control)

Muốn trình bày một chuỗi ký tự trên biểu mẫu như để làm tiêu đề... chúng ta dùng loại điều khiển gọi là điều khiển nhãn (label control). Nhãn không thể hiện dữ liệu của một trường hay biểu thức nào , chúng luôn luôn là không bị buộc.

Click vào biểu tượng Label trong Toolbox

Click vị trí muốn đặt nhãn trên biểu mẫu.

Thay đổi các thuộc tính của điều khiển.

Sau khi tạo biểu mẫu, biến đổi một số thuộc tính của điều khiển có thể hoàn thiện thêm về thiết kế và hình thức trình bày của số liệu.

Nhắp kép vào điều khiển.

Trong hộp lựa chọn trên đầu bảng thuộc tính, chọn một lớp thích hợp để làm việc.

All Properties: Trình bày tất cả các thuộc tính của điều khiển

Data Properties: Ấn định các đặc tính thể hiện dữ liệu trong điều khiển như giá trị mặc nhiên, định dạng số.

Even Properties: Qui định một tập lệnh (Macro) hay thủ tục (Procedure)..

Layout Properties: Định nghĩa các hình thức của điều khiển như cao, rộng.

Other Properties: Một số thuộc tính khác như tên điều khiển, thông tin mô tả ở dòng trạng thái.

Click chọn một trong các thuộc tính của bảng để thực hiện.

5.4. Thiết lập một số thuộc tính bổ sung

5.4.1. Các thuộc tính hỗ trợ nhập liệu

Thuộc tính **Default value:** Gán giá trị mặc định vào nội dung trình bày trong điều khiển.

Thuộc tính **ValidationRule và ValidationRule Text**: Kiểm tra tính hợp lệ khi nhập dữ liệu cho một điều khiển và thông báo lỗi nếu dữ liệu không hợp lệ

5.4.2. Các thuộc tính gióng lề

Thuộc tính **General:** Gióng hàng văn bản theo lề trái, dữ liệu số và ngày tháng theo lề phải.

Thuộc tính **Left:** Gióng hàng văn bản theo lề trái.

Thuộc tính **Center:** Xác lập hàng văn bản ở chính giữa.

Thuộc tính **Center:** Gióng hàng văn bản theo lề phải.

5.4.3. Các thuộc tính màu sắc

Thuộc tính **Backcolor**: Thiết lập màu nền cho điều khiển hay cho biểu mẫu

Thuộc tính **Forecolor**: Thiết lập màu cho hàng chữ trong điều khiển.

Thuộc tính **Bodercolor**: Thiết lập màu cho khung bao quanh trong điều khiển.

5.4.3. Các thuộc tính khung bao

Thuộc tính **BoderStyle**:Thiết lập loại khung cho điều khiển

Thuộc tính **BoderWidth**: Thiết lập độ dày hay đậm của khung bao.

Thuộc tính **BoderColor**: Thiết lập màu của khung bao.

6. NÂNG CẤP BIỂU MẪU

Access cung cấp nhiều loại điều khiển để làm cho biểu mẫu dễ sử dụng và có nhiều hiệu ứng tốt. Có thể thay thế một Textbox với một List box hay Combo box để chọn từ các giá trị có sẵn thay vì buộc người sử dụng phải nhớ để nhập giá trị vào.

6.1. Dùng điều khiển List box và Combo box để tạo danh sách chọn lựa.

Trong nhiều trường hợp, chọn một danh sách có sẵn thường tiện lợi hơn phải nhớ để gõ vào từ bàn phím. Access cung cấp hai khả năng điều khiển tạo danh sách chọn lựa:

List box và Combo box.

List box đơn giản là một danh sách để chọn, combo box tương tự như một text box và một combo box kết hợp vào một điều khiển, có nghĩa là có thể gõ thẳng giá trị vào text box hay chọn từ một danh sách có sẵn.

Ưu điểm của List box: Danh sách luôn được thể hiện và người dùng chỉ được phép chọn trong danh sách, do đó dữ liệu nhập luôn luôn là hợp lệ.

Ưu điểm của Combo box: Danh sách không được thể hiện cho đến khi người dùng mở hộp điều khiển, do đó ít tốn chỗ trên biểu mẫu hơn.

6.2. Tạo List box và Combo box không sử dụng Wizard

Tắt chức năng Control Wizard trong menu View hoặc trong thanh công cụ.

Click biểu tượng List box hoặc Combo box trong hộp công cụ.

Tạo điều khiển buộc bằng cách click biểu tượng Field list trên thanh công cụ để mở danh sách trường làm nền cho biểu mẫu. Chọn một trường trong danh sách kéo vào biểu mẫu, hoặc tạo điều khiển không buộc bằng cách click biểu mẫu nơi muốn đặt điều khiển.

Lập các thuộc tính của điều khiển để định nghĩa các hàng dùng làm chọn lựa trong danh sách.

Muốn danh sách thể hiện	**Lập thuộc tính Row Source Type thành**	**Lập Row Source thành**
Các hàng từ một bảng hay truy vấn	Table/Query (Default)	Tên của bảng hay truy vấn đó
Các hàng lấy từ lệnh Select của SQL	Table/Query (Default)	Một câu lệnh SQL
Một danh sách với các giá trị do người dùng tự đặt	Value List	Danh sách các giá trị đó phân cách nhau bởi dấu chấm phẩy
Tên các trường trong một bảng hoặc truy vấn	Field List	Tên của bảng hoặc truy vấn đó
Các giá trị định nghĩa bởi một hàm trong Access Basic	Tên hàm xây dựng	Để trống

6.2.1. Định nghĩa các hàng trong danh sách

Định nghĩa nguồn dữ liệu để hình thành các hàng dùng làm trong danh sách lựa chọn trong List box và Combo box bằng cách lập thuộc tính thích hợp cho RowSource Type và RowSource của điều khiển, hai thuộc tính này phối hợp với nhau để tạo nên các hàng trong danh sách.

Thuộc tính RowSource Type cho biết danh sách sẽ hình thành từ nguồn dữ liệu loại nào, thuộc tính RowSource là thuộc tính xác định nguồn dữ liệu của danh sách.

6.2.2. Danh sách lấy dữ liệu từ bảng hay truy vấn

Có thể tạo List box hay Combo box lấy dữ liệu từ một hoặc nhiều trường trong bảng (hoặc truy vấn) để lập thành danh sách chọn lựa cho điều khiển. Thực hiện điều này bằng cách đặt thuộc tính RowSource Type thành Table/Query và RowSource thành tên bảng hay tên truy vấn có chứa dữ liệu để lập thành danh sách đó.

6.2.3. Tạo danh sách nhiều cột

Có thể tạo danh sách gồm hai hay nhiều cột để có thêm thông tin lựa chọn. Xác định số cột bằng cách lập thuộc tính Columcount và ColumnWidth cho điều khiển. Thuộc tính columcount quy định số cột và nếu trong RowSource là tên một bảng hay truy vấn thì các trường đầu tiên tương ứng số cột của bảng hay truy vấn đó (tính từ trái sang phải) sẽ được đưa vào danh sách. Thuộc tính columnwidth quy định bề rộng mỗi cột tính theo inch hay cm.

6.2.4. Sử dụng danh sách giá trị

Trường hợp điều khiển chỉ có một số nhỏ chọn lựa và các giá trị này không thay đổi, có thể dùng phương pháp đơn giản là danh sách giá trị. Danh sách này chúng ta tự lập bằng cách đưa các giá trị dùng làm chọn lựa vào thuộc tính RowSource của điều khiển. Sử dụng dấu chấm phẩy để phân cách các mục chọn lựa của danh sách.

6.3. Sử dụng hộp kiểm tra (check box), nút chọn lựa (Option button), nút bật tắt (Toggle button).

Hộp kiểm tra, nút chọn lựa, nút bật tắt thường được sử dụng như các điều khiển độc lập để nhận các chọn lựa Yes/No. Các điều khiển này thực chất chỉ khác nhau về hình thức, do đó chúng ta có thể sử dụng bất cứ nút nào.

Khi được chọn điều khiển này biểu diễn trị Yes (hay True), ở trạng thái không chọn thì nhận giá nị No (hay False).

Check box **:** Được chọn khi có dấu **x** trong hộp.

Option button **:** Được chọn khi có hạt đậu trong nút.

Toggle button **:** Được chọn khi nó trông như bị nhấn xuống.

6.4. Sử dụng nhóm chọn lựa (Option group)

Chúng ta có thể dùng nhóm chọn lựa để trình bày một số nhỏ các chọn lựa trên biểu mẫu. Nhóm chọn lựa gồm một khung nhóm (Group frame) có gắn nhãn và một số check box, option button hay toggle button.

Nếu nhóm lựa chọn được buộc vào một trường, chỉ có khung nhóm là bị buộc vào trường đó, không phải các thành phần để chọn lựa trong khung. Khung nhóm có thể không bị buộc vào trường nào hoặc chứa một biểu thức, mỗi lần chỉ có thể chọn một thành phần trong nhóm.

Khi sử dụng hộp kiểm tra, nút chọn lựa và nút bật tắt trong nhóm chọn lựa, thuộc tính của chúng khác với như khi được dùng với các điều khiển độc lập. Mỗi phần tử trong nhóm chọn lựa không có thuộc tính Control Source mà thay bằng thuộc tính Option Value. Lập thuộc tính Option Value của mỗi phần tử dùng làm chọn lựa trong nhóm thành một trị số có nghĩa đối với trường mà khung nhóm được buộc vào. Khi một phần tử của nhóm được chọn, Access đưa giá trị lập trong Option value của phần tử đó cho điều khiển Option Group. Đến lượt Option Group trả lại giá trị đó cho trường mà nó bị buộc vào.

7. BIỂU MẪU DỰA TRÊN NHIỀU BẢNG

7.1. Biểu mẫu phụ (Subform)

Biểu mẫu phụ là phương pháp để đưa thông tin từ nhiều bảng vào một biểu mẫu. Biểu mẫu phụ có nghĩa là một biểu mẫu được lồng trong biểu mẫu khác. Trong Access biểu mẫu chính gọi là Main form, biểu mẫu nằm trong Main form gọi là biểu mẫu phụ (Sub form). Khi dùng biểu mẫu phụ chúng ta dễ nhận thấy mối quan hệ giữa các bản ghi của hai hay nhiều bảng.

Biểu mẫu phụ đặc biệt hữu hiệu khi dùng để hiển thị dữ liệu từ nhiều bảng hay truy vấn có quan hệ một-nhiều với nhau. Biểu mẫu chính đại diễn cho bên một, biểu mẫu phụ đại diện cho bên nhiều.

7.2. Các loại biểu mẫu phụ

Khi tạo biểu mẫu phụ chúng ta có thể thiết kế nó thành dạng bảng, hoặc dạng biểu mẫu, hoặc cả hai dạng trên.

Biểu mẫu dạng bảng: Là loại dễ tạo nhất và có thể sử dụng như bất kỳ bảng nào khác như sắp xếp....

Biểu mẫu phụ dạng biểu mẫu: Cho chúng ta thực sự linh hoạt và mềm dẽo khi thiết kế hơn.

7.3. Thiết kế biểu mẫu phụ

Thông thường chúng ta dùng bảng hay truy vấn làm nguồn dữ liệu cho biểu mẫu chính, một bảng hay truy vấn khác làm nguồn dữ liệu cho biểu mẫu phụ. Nếu dữ liệu trong biểu mẫu chính và biểu mẫu phụ có liên quan với nhau, chúng ta cần đánh giá một số vấn đề sau:

Các bảng hoặc truy vấn có quan hệ một-nhiều với nhau không? Nếu dùng biểu mẫu phụ để thể hiện quan hệ một-nhiều, chúng ta nên dùng bảng bên một đối với bảng chính, bảng bên nhiều đối với bảng phụ.

Các bảng hoặc truy vấn làm nguồn dữ liệu cho biểu mẫu chính/phụ có các trường liên kết không? Access dùng trường kết nối để giới hạn số lượng bản ghi thể hiện trong biểu mẫu phụ.

Cách tạo biểu mẫu chính/phụ

Thiết kế hai biểu mẫu riêng biệt, sau đó kéo biểu mẫu phụ vào biểu mẫu chính.

Thiết kế biểu mẫu chính

Tạo biểu mẫu chính, dành chỗ trên biểu mẫu này để chứa biểu mẫu phụ.

Lưu và đóng biểu mẫu chính.

Thiết kế biểu mẫu phụ

Có thể thiết kế biểu mẫu phụ để chỉ thể hiện dữ liệu dưới dạng bảng, biểu mẫu này có cả hai khả năng trên.

Tạo biểu mẫu mới, lập hai thuộc tính ViewAllowed và Default View của biểu mẫu tuỳ theo yêu cầu sử dụng như sau:

Biểu mẫu phụ chỉ trình bày dưới dạng bảng: Đặt các trường trên biểu mẫu theo thứ tự chúng ta muốn chúng xuất hiện trong bảng. Lập cả hai thuộc tính ViewAllowed và Default View thành Datasheet.

Biểu mẫu phụ chỉ trình bày dữ liệu dưới dạng biểu mẫu: Sắp đặt các điều khiển như trên. Lập thuộc tính ViewAllowed thành Form và Default View thành Single form hay Continuous form.

Đưa biểu mẫu phụ vào biểu mẫu chính

Mở biểu mẫu chính ở chế độ Design View

Chuyển sang cửa sổ Database, nhấn F11

Kéo biểu mẫu phụ tử cửa sổ Database và đặt vào một vị trí trên biểu mẫu chính.

Di chuyển biểu mẫu phụ đến vị trí khác, thay đổi nội dung nhãn hoặc kích thước nếu cần. Chuyển sang chế độ Form View để xem kết quả.

7.4. Liên kết biểu mẫu chính và biểu mẫu phụ

Trong chế độ Design View của biểu mẫu chính, mở bảng thuộc tính của điều khiển biểu mẫu phụ.

Lập thuộc tính LinkChildFields thành tên trường nối kết trong biểu mẫu phụ, nếu có nhiều trường nối kết, phân cách nhau bởi dấu phẩy.

Lập thuộc tính LinkMasterFields thành tên trường nối kết hoặc tên điều khiển trong biểu mẫu chính, nếu có nhiều trường nối kết, phân cách nhau bởi dấu phẩy.

Chương 6 BÁO CÁO (REPORT)

1. TỔNG QUAN VỀ BÁO CÁO

Báo cáo là phương thức hữu hiệu giúp người sử dụng trình bày dữ liệu dưới dạng đầy đủ và dễ hiểu, nhanh chóng, đẹp mắt để khi in ấn. Người sử dụng có thể tích hợp trong báo cáo các dạng thức trình bày dữ liệu khác nhau như: Hình ảnh, biểu đồ, văn bản.....

Báo cáo được xây dựng trên một nguồn dữ liệu đó là bảng hoặc truy vấn, một câu lệnh SQL hoặc một dạng biểu mẫu nào đó

1.1. Các dạng mẫu của báo cáo

Báo cáo dạng cột (columnar): báo cáo dạng này sẽ được trình bày theo dạng một cột và kèm theo phần nhãn của mỗi cột dữ liệu bên trái, mỗi dòng tương ứng với một trường dữ liệu.

Báo cáo dạng hàng (Tabular): Báo cáo sẽ trình bày dữ liệu theo dạng bảng bao gồm nhiều hàng và nhiêu cột.

Báo cáo dạng nhóm/ Tổng (Group/Total): Báo cáo dạng này sẽ tổ chức dữ liệu thành các nhóm, mỗi nhóm sẽ trình bày dữ liệu theo dạng Tabular. Người sử dụng có thể nhóm dữ liệu theo cấp và có thể tính toán giá trị tổng cho mỗi nhóm và một giá trị tính tổng cho toàn bộ các nhóm

Báo cáo dạng biểu đồ (Chart)

Báo cáo dạng nhãn (Label Report)

Báo cáo với báo cáo con

1.2. Các chế độ hiển thị của báo cáo

Báo cáo có thể được trình bày theo 3 chế độ sau

Report design: Chế độ thiết kế báo cáo.

Layout PreView: Chế độ trình bày dữ liệu trong báo cáo.

Print PreView: Chế độ xem hình thức báo cáo trước khi in ấn.

2. TẠO BÁO CÁO SỬ DỤNG CÔNG CỤ AUTO REPORT VÀ REPORT WIZARD

2.1.Tạo báo cáo sử dụng Auto report

Click biểu tượng Report trong cửa sổ database (hoặc chọn View/Report)

Chọn New

Chọn bảng hoặc Truy vấn làm nguồn dữ liệu báo cáo .

Chọn AutoReport Columnar: Nếu muốn báo cáo hiển thị dạng cột.

AutoReport Tabular: Nếu muốn báo cáo hiển thị dạng hàng

Chọn OK

Lưu Báo cáo.

2.2.Tạo báo cáo sử dụng Report Wizard

Click biểu tượng Report trong cửa sổ database (hoặc chọn View/Report)

Chọn New

Chọn bảng hoặc Truy vấn làm nguồn dữ liệu báo cáo .

Chọn Report Wizard

Chọn OK

Chọn các trường cần thiết cho báo cáo.

Chọn **Next**

Chọn các trường cần nhóm, chọn **Next**

Chọn các trường cần sắp xếp, chọn **Next**

Chọn dạng thể hiện của Report, chọn **Next**

Chọn nền thể hiện của Report, chọn **Next**

Đặt tiêu đề cho Report, chọn **Finish**

3. TẠO ĐIỀU KHIỂN TRONG BÁO CÁO

3.1. Tạođiều khiển Text Box (Hộp văn bản)

Text box có nhiều tính năng, có thể là điều khiển bị buộc, không buộc hay dùng tính toán. Nếu trong báo cáo có cả 3 loại điều khiển này, chúng ta nên ưu tiên điều khiển bị buộc trước.

Tạo một text box bị buộc và không buộc

Chúng ta buộc điều khiển Text box vào một trường bằng cách chỉ định trường cho text box đó lấy dữ liệu. Text box đó lấy dữ liệu có thể thực hiện bằng cách kéo trường muốn buộc vào điều khiển từ danh sách trường (Field List) vào biểu mẫu đang thiết kế.

Cách khác để tạo Text box là dùng hộp công cụ (Toolbox), sau đó gõ tên trường muốn buộc vào hộp văn bản hoặc bảng thuộc tính của điều khiển.

Dùng danh sách trường là phương pháp tốt nhất để tạo một điều khiển Text box bị buộc vì hai lý do sau:

Điều khiển đó được hệ thống tự động gắn nhãn và nhãn lấy tên trường được kéo làm tiêu đề.

Text box bị buộc đó thừa kế các thiết lập thuộc tính của trường từ bảng hay truy vấn.

Muốn chuyển một điều khiển không buộc thành bị buộc, lập thuộc tính Control Source của điều khiển thành một trường.

Một số thao tác khi thực hiện thiết kế báo cáo.

Mở báo cáo trong chế độ Design View

Chọn Field List từ menu View (Hoặc click biểu tượng Field List trên thanh công cụ)

Tạo điều khiển Text box bị buộc.

Mở bảng danh sách trường, chọn trường hoặc các trường muốn đặt vào báo cáo

Chọn một trường, click vào trường đó.

Chọn nhiều trường liền nhau, clik trường đầu, giữ phím shift, click trường cuối.

Chọn nhiều trường không liền nhau, giữ phím Ctrl và lần lượt click từng trường.

Chọn tất cả các trường trong danh sách, nhắp kép vào thanh tiêu đề của danh sách trường.

Kéo trường (hoặc các trường) được chọn và đặt vào một vị trí trên mẫu báo cáo.

Click biểu tượng Simple preview trên thanh công cụ để xem kết quả.

3.2. Tạo các điều khiển khác dùng hộp công cụ

Muốn tạo các điều khiển không buộc hay dùng để tính toán, phải dùng công cụ trong Toolbox.

Chọn View/Toolbars để hiển thị thanh công cụ

Tạo điều khiển dùng Toolbox

Click công cụ tương ứng loại điều khiển muốn tạo trong báo cáo.

Tạo điều khiển bị buộc bằng cách chọn một trường trong Field list và kéo nó vào trong báo cáo.

3.3. Tạo điều khiển dùng tính toán

Click vào biểu tượng Text box trong hộp công cụ

Click vào một vị trí trên báo cáo. Access tự động gắn nhãn cho điều khiển vừa tạo, tiêu đề mặc nhiên thường có dạng "Field0", có thể thay đổi tiêu đề này theo ý thích.

Đưa con trỏ vào bên trong Text box.

Gõ dấu = , theo sau là biểu thức muốn lập

Ví dụ: =[SOLUONG]*[DONGIA]

Click vào Sample Preview để xem kết quả.

3.4. Tạo điều khiển nhãn

Nhãn là một điều khiển không buộc, nội dung nhãn không thay đổi từ trang này qua trang khác hay từ bản ghi này qua bản ghi khác.

Click biểu tượng Label trong Toolbox.

Click vào một vị trí trên báo cáo để tạo nhãn. Nhãn được tạo như vậy sẽ có kích thước tự mở rộng khi gõ nội dung vào.

Nếu muốn trình bày văn bản thành nhiều dòng, bấm Ctrl+Enter cuối dòng thứ nhất.

4. CÁC THUỘC TÍNH CỦA ĐIỀU KHIỂN TRONG BÁO CÁO

Thuộc tính xác định các đặc trưng của đối tượng, mỗi điều khiển trong báo cáo cũng có những thuộc tính riêng. Muốn mở bảng thuộc tính của điều khiển, chọn điều khiển đó và click biểu tượng Properties trên thanh công cụ.

Ta xét một số thuộc tính sau:

Cangrow: Dùng thuộc tính này để làm cho Text box có thể tự điều chỉnh kích thước theo phương dọc đối với khối dữ liệu chứa trong trường nó bị buộc.(chọn Yes).

CanShrink: Khi Text box khôngcó dữ liệu hoặc dữ liệu là chuỗi rỗng Access sẽ chừa trống chỗ đó trên giấy. Điều này có thể làm cho báo cáo quá trống trải nếu có nhiều chỗ như vây. Chúng ta lập thuộc tính của Text box này thành Yes.

HideDuplicate: Dùng thuộc tính này để che Text box khi giá trị trong đó trùng bản ghi trước

5. SẮP XẾP VÀ TẬP HỢP DỮ LIỆU THEO NHÓM

Sắp xếp là phương pháp phổ biến nhằm tổ chức dữ liệu theo một trật tự nào đó để tìm kiếm và phân loại thông tin.

5.1. Sắp xếp dữ liệu

Khi in báo cáo người dùng thường muốn tổ chức các bản ghi theo một trật tự nào đó. Ví dụ in danh sách cán bộ theo thứ tự giảm dần của lương.

Các bước thực hiện sắp xếp trên báo cáo.

Mở báo cáo ở chế độ Design View.

Chọn Sorting And Grouping trong menu View.

Trong hộp thoại.

Field/Expression: Chỉ định sắp xếp theo trường hoặc biểu thức nào đó.

Sort Order: Chọn Tăng dần hoặc giảm dần.

5.2. Nhóm dữ liệu

Trong nhiều báo cáo, sắp xếp các bản ghi không cũng chưa đủ mà cần phân thành các nhóm. Nhóm là tập hợp các bản ghi cùng với thông tin tóm lược tiêu biểu cho một thể loại thông tin. Một nhóm thường được cấu tạo như sau:

Tiêu đề nhóm (group header), nhóm con (nếu có), các bản ghi chi tiết và chân nhóm (Group footer)

Tiêu đề nhóm 1

Tiêu đề nhóm 2

Tiêu đề nhóm 3

.....

.....

Tiêu đề nhóm 10

Các bản ghi chi tiết

Chân nhóm 10

.....

.....

Chân nhóm 3

Chân nhóm 2

Chân nhóm 1

Ch□□ng 7 MACRRO

1. KHÁI NIỆM

Macro trng MS Access là tập hợp các lệnh (Hành động, hành động.....) được định sẵn nhằm tự động thực hiện chuỗi các tác vụ nào đó mà không cần sự can thiệp từng bước của người sử dụng. Macro có thể liên kết các đối tương trong tập tin cơ ở dữ liệu (CSDL) như: Table, Query, form, report..... nhằm tạo ra các ứng dụng để khai thác có hiệu quả..

Macro được dùng khi có các hành động nào thường xuyên lặp lại trong MS Access hoặc được dùng khi cần kết hợp các hành động đơn giản nhằm giải quyết một vấn đề nào đó khi xây dựng các ứng dụng. Việc tự động hoá các hành động này bởi macro sẽ được thực hiện một cách nhanh chóng và chính xác.

2.TẠO VÀ THI HÀNH MỘT MACRO

2.1.Tạo một macro

Tại cửa sổ database, chọn đối tượng Macro/ New

Trong mục Action : chọn các hành động cho Macro.

Tong mục Action arguments: Chọn các tham số tương ứng cho hành động.

Lưu trữ Macro: File/save.

2.2.Thi hành macro

Tại cửa sổ database: Chọn đối tượng Macro/Run.

3. CÁC HÀNH ĐỘNG VÀ CÁC THAM SỐ

Ms Acces cung cấp một số hành động để tạo macro, mỗi hành động thực hiện một tác vụ nào đó trên CSDL và tuỳ thuộc vào tham số của mỗi hành động.

Open table: Mở 1 bảng dữ liệu

Table name: Tên bảng cần mở

View: Chọn dạng thể hiện của bảng (Datasheet/ Design/ Print Preview

Data mode:

Add: Dùng để nhập dữ liệu

Edit: Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only: Mở bảng để chỉ xem dữ liệu

Open Query: Mở 1 truy vấn

Query name: Tên truy vấn cần mở

View: Chọn dạng thể hiện của bảng (Datasheet/ Design/ Print Preview

Data mode:

Add: Dùng để nhập dữ liệu

Edit: Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only: Mở bảng để chỉ xem dữ liệu

Open Form: Mở 1 biểu mẫu

Form name: Tên biểu mẫu

View: Chọn dạng thể hiện (Form/ Design/ Print preview/ Datasheet)

Filter name: Tên Query lọc các dữ liệu để hiển thị trong form.

Where condition: Điều kiện lọc dữ liệu hiển thị trong form.

Data mode:

Add : Dùng để nhập dữ liệu

Edit : Dùng để thêm, xoá, sửa dữ liệu trong bảng

Read Only : Mở bảng để chỉ xem dữ liệu

Window mode:

Normal : Dạng cửa sổ form bình thường.

Hidden : Dạng cửa sổ form được ẩn đi.

Icon : Cửa sổ form thu nhỏ thành 1 biểu tượng.

Dialog : Dạng hộp thoại.

Open Report: Mở 1 báo cáo

Report name: Tên báo cáo

View: Chọn kiểu in

Print preview:In ra màn hình.

Design: Dạng thiết kế báo cáo

Print: In ra máy in

Filter name: Tên Query lọc các dữ liệu để hiển thị trong Report.

Where condition: Điều kiện lọc dữ liệu hiển thị trong Report.

Run macro: Thực hiện một tập lệnh

Macro name: Tên macro cần thực hiện

Repeat count: Số lần thực hiện macro sẽ lặp lại.

Repeat Expression: Biểu thức điều kiện để lặp lại khi thực hiện macro. Macro chỉ dừng khi khi biểu thức điều kiện nhận giá trị False.

Open module: Mở cửa sổ soạn thảo thủ tục trong 1 module.

Module name: Tên module chứa thủ tục cần mở.

Procedure name: Tên thủ tục sẽ mở.

Run code: Gọi thực hiện một hàm của Access Basic

Function name: Tên hàm cần thực hiện và các đối số của hàm.

Run App: Cho thực hiện một ứng dụng nào đó trong môi trường Windows

Command line: đường dẫn đến tập tin của một ứng dụng.

Run SQL: Cho thực hiện câu lệnh SQL

SQL Statement: Nội dung câu lệnh SQL

Maximize: Cực đại cửa sổ hiện thời

Minimize:Cực tiểu cửa sổ hiện thời thành một biểu tượng.

Restore: Phục hồi cửa sổ trở về kích thước cũ.

Move size: Di chuyển hoặc thay đổi kích thước cửa sổ hiện thời.

Right: Khoảng cách từ góc trên trái của cửa sổ này đến cạnh trái của cửa sổ chứa nó.

Down: Khoảng cách từ góc trên trái của cửa sổ này đến đến cạnh trên của cửa sổ chứa nó.

Width: Chiều rộng của cửa sổ này.

Height: Chiều cao của cửa sổ này.

Stop Macro: Dừng macro đang thực hiện

Beep: Phát tiếng kêu bíp

Hourglass: Đổi dạng con trỏ thành đồng hồ cát trong khi macro đang chạy

Hourglass On: Yes/No (Đổi/ Không đổi)

Close: Đóng một cửa sổ đang hoạt động

Object Type: Loại cửa sổ của đối tượng cần đóng như Table, Query, form, Report, Macro hoặc Module.

Object name: Tên của đối tượng cần đóng.

Quit: Thoát khỏi MS Access và trở về Windows

Option

Prompt: Hiển thị hộp thoại có lưu trữ không? Nếu đối tượng có thay đổi.

Save all: Lưu trữ tất cả mọi đối tượng.

Exit: Thoát mà không cần lưu trữ.

Print: In đối tượng hiện thời

Print Range: Phạm vi cần in ấn.

All: In tất cả các đối tượng

Selection: In phần trang được chọn

Pages: In các trang được chọn

Page from: Trang bắt đầu in

Page to: Trang kết thúc in

Print Quality: Chất lượng in

Copies: Số bản cần in

Collate Copies: Có sắp xếp thứ tự các bản in teo trang.

Msg Box: Hiển thị hộp thông báo

Message: Câu thông báo cần hiển thị

Beep: Yes/ No: Có/ Không phát ra tiếng Bíp khi hiển thị hộp thông báo.

Type: Loại hộp thông báo.

Title: Tiêu đề của hộp thông báo.

CancelEvent: Huỷ bỏ một sự kiện đang thực hiện

Requery: Cập nhật dữ liệu cho một đối tượng đang hoạt động bằng cách cập nhật lại dữ liệu nguồn của đối tượng đó.

Control name: Tên của đối tượng cần cập nhật dữ liệu (Nếu không chỉ ra thì sẽ cập nhật lại dữ liệu nguồn của chính đối tượng đang hoạt động).

Select Object: Chọn đối tượng trong CSDL

Object Type: loại đối tượng cần chọn.

Object name: Tên đối tượng cần chọn

In Database Window: (Yes/No) Xác định MS access có chọn đối tượng trong cửa sổ CSDL không, mặc định là No.

Set value: Gán một giá trị cho 1 trường, 1 điều khiển, hoặc một thuôc tính trên một Form hoặc 1 Report.

*Item:*Tên trường, đối tượng hay thuộc tính muốn gán giá trị.

Expression: Biểu thức cần gán giá trị cho Item.

☛ ***Chú ý:*** Nếu tên trường, tên đối tượng, tên thuộc tính ở 1 Form hoặc 1 Report khác thì phải mô tả đầy đủ.

Trong Form khác: [Forms]![Tên Form]![Tên trường/Tên đối tượng]

Trong Report khác: [Reports]![Tên Report]![Tên trường/Tên đối tượng]

Đối với các thuộc tính

[Forms/Reports]![Tên Form/Tên Report]![Tên trường].[Tên thuộc tính]

Add menu: Tạo thêm một Drop Down Menu vào một menu bar cho một form hoặc Report.

Menu name: Tên của Drop Down Menu muốn thêm vào menu bar.

Menu macro name: Tên macro chứa các lệnh về việc tạo menu.

Status bar: Thông báo ở thanh trạng thái khi chọn menu này.

Apply Filter: Lọc (Truy vấn) các dữ liệu khi xử lý Table, Form, Report.

Filter name: Tên của truy vấn lọc dữ liệu.

Where condition: Điều kiện lọc dữ liệu.

FindRecord: Tìm bản ghi đầu tiên nằm trong phạm vi và thoả mãn điêu kiện.

Find What: Nội dung dữ liệu cần tìm là một giá trị hoặc một biểu thức, nếu là biểu thức sẽ bắt đầu dấu "=".

Where: Qui định cách so sánh giá trị cần tìm với giá trị của trường.

Any part of field: Một phần bất kỳ của trường.

Match Whole field: Giá trị cần tìm bằng giá trị của trường.

Start of field: Giá trị cần tìm là phần đầu của trường.

Match Case: Yes/No: Có/Không phân biệt chữ in hoa và chữ in thường.

Direction: Quy định hướng tìm

All: Tìm toàn bộ

Up: Tìm từ bản ghi hiện thời lên phía trên.

Down: Tìm từ bản ghi hiện thời lên phía dưới.

Search As Formatted: Qui định việc tìm có dựa trên dữ liệu sau khi đã định dạng trong các trường hay không?

Search in: Qui định việc tìm trên trường hiện thời hoặc trên tất cả các trường.

Current Field: Tìm trên trường hiện thời.

All Fields: Tìm tất cả các trường.

Find First: Yes/No: Qui định tìm từ bản ghi đầu tiên hay tìm từ bản ghi hiện thời.

Findnext:Tìm bản ghi kế tiếp thoả mãn điều kiện tìm kiếm của lệnh FindRecord.

CopyObject: Sao chép một đối tượng trong tập tin CSDL hiện thời thành một đối tượng khác của tập tin CSDL khác trong MS Access.

Destionation Database: Tên tập tin CSDL đích

New name: Tên mới của đối tượng sau khi sao chép.

Source Object Type: Kiểu của đối tượng nguồn.

Source Object Name: Tên của đối tượng nguồn.

DeleteObject: Xoá một đối tượng trong tập tin CSDL hiện thời.

Object Type: Kiểu của đối tượng.

Object Name: Tên của đối tượng

4. NHÓM TẬP LỆNH VÀ TẬP LỆNH CÓ ĐIỀU KIỆN

4.1. Nhóm tập lệnh

Là Macro chứa các macro con, thay vì tạo ra nhiều macro với nhiều tên khác nhau thì các macro này được gom lại thành một tên chung nhằm giảm bớt số lượng và thuận lợi trong quá trình sử dụng. Tuy nhiên các Macro được nhóm khi chúng có liên quan với nhau.

Có thể có nhiều Macro trong nhóm có cùng hành động, tuy nhiên chúng được phân biệt bởi tên Macro.

Đặt tên cho Macro ta thực hiện:

Tại chế độ thiết kế Macro:

View/Macro name

Đặt tên cho Macro tại cột Macro name.

Cách thực hiện 1 macro trong macro name

<Tên Macro Group>.<Tên Macso cần thực hiện>

4.2. Macro có điều kiện

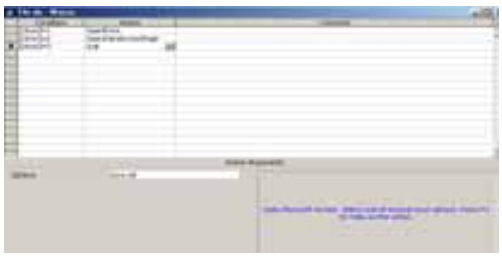
Là macro có chứa các điều kiện thi hành cho mỗi hành động.

Cách tạo Macro có điều kiện

Tại chế độ thiết kế Macro: Chọn View/Conditions

Tại cột Condition : Đặt điều kiện thi hành cho mỗi hành động.

Ví dụ



4.3. Áp dụng Macro cho form và Report

4.3.1. Quy tắc chung khi gọi một đối tượng

Đối với form : *Forms![Tên form]![Tên đối tượng]*

Đối với Report : *Reports![Tên Report]![Tên đối tượng]*

4.3.2. Các thuộc tính của một đối tượng

Muốn gắn một nút lệnh trên một biểu mẫu hoặc báo cáo với một Macro nào đó vào nút lệnh này ta thực hiện: Click chuột phải vào nút lệnh, chọn Properties và gắn Macro vào các hành động tương ứng.

On Enter: Macro thi hành khi nhấn Enter vào bên trong đối tượng

On Exit: Macro thi hành khi thoát khỏis đối tượng

On Got Focus: Thiết lập nhận biết khi có di chuyển con trỏ đến một form hoặc 1 trường trên form đang mở.

On Click: Macro thi hành khi click vào đối tượng

On Dbl Click: Macro thi hành khi Double click vào đối tượng.

On Mouse Down: Macro thi hành khi ấn và giữ chuột tại đối tượng

On Mouse Move: Macro thi hành khi di chuyển chuột ra khỏi đối tượng

On Mouse Up: Macro thi hành khi nhã chuột ra khỏi đối tượng.

On Key Down: Macro thi hành khi ấn và giữ một phím đối tượng.

On Key Press: Macro thi hành khi ấn một phím đối tượng.

On Key Up: Macro thi hành khi nhã một phím đối tượng.

4.3.3. Macro tự động thực hiện sau khi mở tập tin CSDL

Chúng ta có thể tạo ra một Macro mà mỗi khi mở một tập tin CSDL thì Macro này tự động thực hiện.

Để tạo Macro tự động thực hiện ta tiến hành các thao tác sau:

Tạo macro

Lưu trữ Macro với tên AutoExec.

5. THIẾT KẾ MENU TRONG ACCESS

Trong các ứng dụng, thường chúng ta phải tổ chức Menu để cho phép người sử dụng thực hiện các hành động thông qua các chức năng trên Menu này.

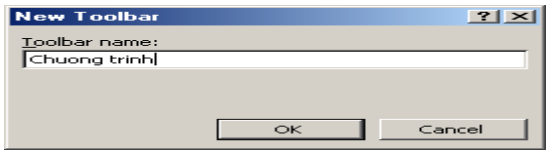
Có 2 cách hiển thị các Menu của người sử dụng (Custom Menu) trong các ứng dụng đó là:

Menu ứng với 1 form xác định nào đó: Menu này chỉ xuất hiện khi truy xuất đến Form này.

Menu toàn cục (Globally): Menu toàn cục này sẽ xuất hiện trong các ứng dụng và nó chỉ được thay thế khi có 1 menu ứng với form nào đó được mở.

5.1. Tạo menu của người sử dụng

Chọn View/Toolbars/Customize/New/ Đặt tên cho thanh menu

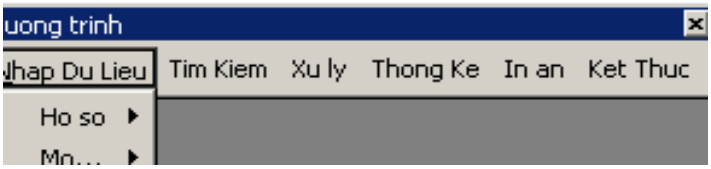


Chọn Properties và chọn Type là Menu bar, close

Chọn command, trọng mục Categories chọn New menu và kéo sang thanh menu vừa tạo.

Click chuột phải để thay đổi các tiêu đề cho phù hợp trên thanh menu.

Tương tự cho các nhóm khác.



Có thể tạo các chức năng là các hành động như mở bảng, truy vấn, biểu mẫu.... bằng cách kéo các biểu tượng này trong mục command vào menu đang tạo.

5.2. Tạo menu toàn cục

Nếu muốn tạo một menu toàn cục sẽ thay thế menu có sẵn trong Access khi CSDL này được mở ta thực hiện các bước sau:

Mở hoặc tạo ra một Macro có tên là Autoexec.

Thêm vào hành động: SetValue

Trong mục các tham số (arguments):

Item: Application.menubar

Expression: Tên của Menubar người sử dụng tạo ra và đặt giữa cặp dấu ngoặc kép "".

BÀI SỐ 1

Khởi động MSACCESS, tạo một CSDL có tên NHANSU.MDB rồi lần lượt tạo các bảng dữ liệu sau

Bảng 1: Nhanvien

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manv**	Text	4	Khoá chính
2	Holot	Text	20	
3	Ten	Text	10	
4	Ngaysinh	Date/time	8	
5	Gioitinh	Yes/No	1	**Yes:Nam, No:Nu**
6	Maphong	Text (Lookup)	2	Lấy từ bảng Dsphong
7	MaPXuong	Text (Lookup)	2	Lấy từ bảng dspxuong
8	Diachi	Text	30	
9	Ghichu	Memo		

Bảng 2: Dsphong

STT	Fieldname	Data Type	Fieldsize	Note
1	**Maphong**	Text	2	Khoá chính
2	Tenphong	Text	30	
3	Sodthoai	Text	11	

Bảng 3: Thunhap_NV

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manv**	Text (Lookup)	4	Khoá chính
2	Luongchinh	Number	Double	
3	Heso	Number	Integer	
4	Phucap	Number	Double	
5	Thue	Number	Double	Thuế
6	Thamnien	Number	Byte	Số năm thâm niên
7	Tongluong	Number	Double	

Bảng 4: DSpxuong

STT	Fieldname	Data Type	Fieldsize	Note
1	**Mapxuong**	Text	2	Khoá chính
2	Tenpxuong	Text	20	
3	Sodoanvien	Number	Byte	Số đoàn viên

1. Hãy nhập dữ liệu thích hợp cho các bảng trên.

2. Mở bảng Nhanvien sắp xếp tăng dần theo **manv**, sắp xếp giảm dần theo **ten**.

3. Sắp xếp tăng dần theo **ten**, nếu **ten** trùng nhau thì sắp xếp giảm dần theo **holot** nếu **holot** trùng nhau thì sắp xếp giảm dần theo **maphong**.

4. Lọc và hiển thị danh sách những nhân viên có tên **Thanh**.

5. Lọc và hiển thị những nhân viên có họ **Nguyen** hoặc **Cao** và có tên **anh**.

6. Lọc và hiển thị nhữnh nhân viên có **Manv** bắt đầu là **B** và **maphong** là **A1** hoặc **Manv** bắt đầu là **C** và **maphong** là **A2**.

7. Lọc và hiển thị những nhân viên **nam** và có năm sinh **1975**.

8. Lọc và hiển thị những nhân viên **nam** sinh trong thời gian từ **12/12/76** đến **12/12/79**.

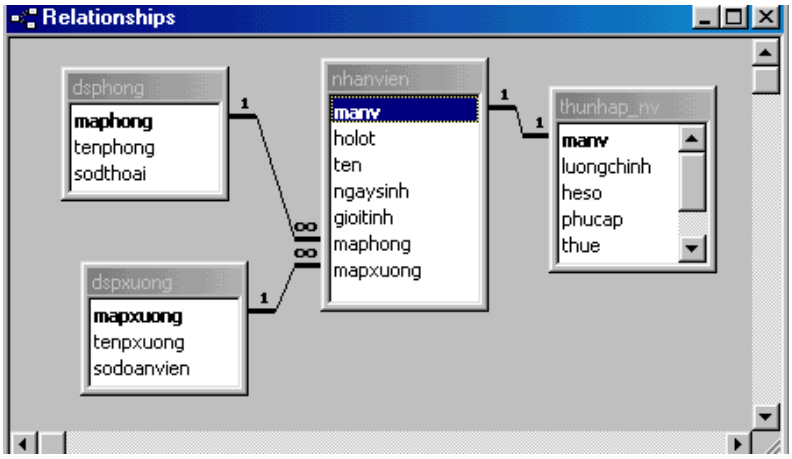
9. Lọc và hiển thị những nhân viên **nữ** có **Mapxuong** là **P1** và sinh trong tháng **10** năm **1975** hoặc nhân viên **nam** có năm sinh **1972** đến **1976**.

10. Mở bảng **Thunhap_nv** để hiển thị những nhân viên có **heso>100 và** có **luongchinh<500.000** hoặc có **phucap** từ **100.000** đến **500.000**.

BÀI SỐ 2

1. Mở CSDL NHANSU.MDB trong bài thực hành số 1 rồi thực hiện các nhiệm vụ sau:

- ❖ Đặt khoá chính cho trường Manv (nhanvien), Maphong(dsphong), manv(thunhap_nv) và mapxuong (dspxuong).
- ❖ Thiết lập các mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau



❖ Hãy thiết lập các thuộc tính tham chiếu toàn vẹn và các thuộc tính như Format, caption, inputmask..... cho các trường một cách hợp lệ.

Đặt thuộc tính Format của trường **ngaysinh :** dd-mm-yy , Inputmask: 00/00/00

Đặt thuộc tính Inputmask của trường **sodthoai** : (000)000000

2. Tạo một truy vấn **query1** để hiển thị những thông tin sau:

Manv, holot, ten (trong bảng **nhanvien**), Luongchinh, phucap (Trong bảng **Thunhap_nv**), tenphong, maphong (Trong **dsphong**).

3. Tạo một truy vấn **query 2** để hiển thị những thông tin sau:

Manv, hoten (Nối holot và ten), maphong, tenphong, mapxuong, tenpxuong, tongthunhap trong đó tongthunhap được tính theo công thức:

Tongthunhap=luongchinh*heso+phucap-thue nếu là nhân viên **nam**.

Tongthunhap=luongchinh*heso+phucap nếu là nhân viên **nữ** .

4. Tạo một truy vấn **query3** để hiển thị những nhân viên có **maphong** là **p1** và **mapxuong** là **x1** sinh trong tháng **7** năm **1976** bao gồm những tin sau:

Hoten, gioitinh, maphong, tenphong, mapxuong, tenpxuong.

5. Tạo một truy vấn **query4** để hiển thị những nhân viên **nữ** và có **maphong** là **p2** hoặc **p3** sinh trong ngày **20** tháng **12** bao gồm những thông tin:

Hoten, maphong, ngaysinh, gioitinh (chú ý giá trị trường gioitinh phải hiển thị nam hoặc nữ).

6. Tạo một truy vấn **query5** để hiển thị những nhân viên **nam** có **tongthunhap>=60000** hoặc thuộc **maphong** là **p2** và không phải họ **Nguyãùn** bao gồm những thông tin: Ten, maphong, tenphong, tongthunhap.

7. Tạo một truy vấn **query6** để hiển thị những nhân viên **nữ** có **maphong** không bắt đầu là **p** hoặc những nhân viên thuộc phân xưởng không có đoàn viên nào bao gồm những thông tin sau: Holot, ten, gioitinh, tenpxuong.

8. Tạo một truy vấn **query7** để hiển thị những nhân viên **nam** tên **Thanh** hoặc **Long** hoặc những nhân viên **nữ** không phải họ **Lê** hoặc **trần** bao gồm những thông tin: Hoten, gioitinh, maphong, tenphong.

9. Tạo một truy vấn **query8** để hiển thị những nhân viên **nam** sinh trong khoảng thời gian từ năm **1973** đến **1980** thuộc phân xưởng có **mapxuong** là **x2** hoặc những nhân viên không có **thuế** sinh trong tháng **4** đến tháng **8** năm **1975** bao gồm những thông tin: Holot, ten, thangsinh, namsinh, mapxuong, tenpxuong.

10. Tạo một truy vấn **query9** để hiển thị những thông tin: Hoten, namsinh, thamnien, luongchinh, trong đó nếu

Luongchinh>=3000 và nhân viên **nữ** thì thamnien là **35**.

Luongchinh>=4000 và nhân viên **nam** thì thamnien là **30**.

BÀI SỐ 3

Tạo một CSDL có tên QLTV.MDB, rồi lần lượt tạo các bảng dữ liệu sau:

Bảng 1: **Loaisach**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masach**	Text	4	Khoá chính
2	Tensach	Text	20	
3	Tentacgia	Text	20	
4	Namxb	Date/time	8	
5	Soluongco	Number	3	Số lượng có
6	Sotrang	Number	5	Số trang sách
7	Manxb	Text	4	Mã nhà xuất bản

Bảng 1: **Docgiamuon**

STT	Fieldname	Data Type	Fieldsize	Note
1	Hoten	Text	30	
2	Quequan	Text	30	
3	**Madocgia**	Text	4	Khoá chính
4	Masach	Text (Lookup)	4	Lấy dữ liệu từ Loaisach
5	Sluong	Number	1	Số lượng mượn
6	Ngaymuon	Date/time	8	Ngày mượn
7	Ngayhen	Date/time	8	Ngày hẹn trả

Bảng 3: **Docgiatra**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Madocgia**	Text	4	Khoá chính
2	Masach	Text	4	
3	Ngaytra	Date/time	8	Ngày trả

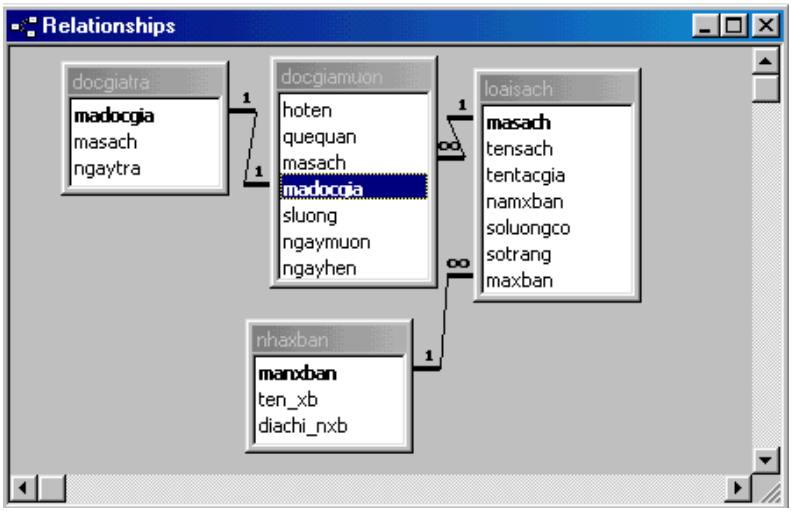
Bảng 4: : **Nhaxban**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Manxban**	Text	4	Khoá chính
2	Ten_xb	Text	30	Tên nhà xuất bản
3	Diachi_nxb	Text	30	Địa chỉ Nhà XB

1. Nhập dữ liệu thích hợp cho các bảng dữ liệu.

2. Đặt khoá chính cho các trường: masach(Loaisach), Madocgia(Docgiamuon), Madocgia(Docgiatra), manxban(Nhaxban).

3. Thiết lập các mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau



4. Tạo một truy vấn **TV1** để hiển thị danh sách độc giả mượn sách trong tháng 10 năm 98 bao gồm những thông tin: Hoten, ngaymuon, tensach, tentacgia.

5. Tạo một truy vấn **TV2** để hiển thị số lượng còn của mỗi loại sách của nhà xuất bản **Kim đồng** bao gồm những thông tin: tensach, nxban, soluongcon.

6. Tạo một truy vấn **TV3** để hiển thị danh sách những độc giả mượn sách quá hạn bao gồm những thông tin: Hoten, Ngaymuon, Ngayhen, songayqua (Số ngày quá hạn).

7. Tạo một truy vấn **TV4** để hiển thị danh sách những độc giả mượn sách quá hạn bao gồm những thông tin Hoten, sluong, mucquahan

Trong đó: Mucquahan là Mức 1 nếu songayqua<5

Mucquahan là Mức 2 nếu songayqua<10

Mucquahan là Mức 3 nếu songayqua>=10

8. Tạo một truy vấn **TV5** để hiển thị danh sách những độc giả trả sách đúng hạn hoặc sớm hạn bao gồm những thông tin: Hoten, masach, tensach.

9. Tạo một truy vấn **TV6** để hiển thị những loại sách có số trang >100 và số lượng còn là 20 của **nhà xuất bản giáo dục** hoặc tên sách có chữ **Tin học** bao gồm Tensach, sotrang, ten_xb, tentacgia.

BÀI SỐ 4

1. Sử dụng CSDL QLTV.MDB, tạo một truy vấn **Truyvan1** để hiển thị tổng số lượng mượn của từng loại sách trong thư viện.

2. Tạo một truy vấn **Truyvan2** để hiển thị tổng số lượng mượn của từng loại sách trong tháng **12** năm **1998**.

3. Tạo một truy vấn **Truyvan3** để hiển thị tổng số lượng mượn của từng loại sách theo từng tháng trong năm **1999**.

4. Tạo một truy vấn **Truyvan4** để hiển thị tổng số lượng mượn của từng loại sách theo từng tháng của một năm nào đó(Tháng và năm được nhập từ bàn phím).

5. Tạo một truy vấn **Truyvan8** để hiển thị số lượng mượn của từng loại sách trong năm **1999** và có số lượng mượn của mỗi độc giả **>2**.

6. Tạo một truy vấn **Truyvan9** để hiển thị tổng số loại sách có trong thư viện.

7. Tạo một truy vấn **Truyvan10** để hiển thị tổng số loại sách trong thư viện cho mượn trong năm **1998**.

8. Tạo một truy vấn **Truyvan11** để hiển thị tên của độc giả nào đó mượn sách trong **tháng 1 năm 2001** (Tên được nhập từ bàn phím).

9. Tạo một truy vấn **Truyvan12** để hiển thị tên sách và tên tác giả xuất bản sách trong năm **2002** (Tên sách nhập từ bàn phím, Họ của tác giả nhập từ bàn phím).

10. Tạo một truy vấn tham khảo chéo **Truyvan13** để phản ánh tổng số mỗi loại sách mượn cho mượn trong **tháng 11 năm 2000**.

11. Tạo một truy vấn tham khảo chéo **Truyvan14** để hiển thị tổng số laọi sách xuất bản trong năm **1995**.

12. Tạo một truy vấn **Truyvan14** để xoá những sinh viên đã mượn sách quá hạn **5** ngày.

BÀI SỐ 5

Tạo một CSDL có tên QLSV.MDB, rồi tạo các bảng dữ liệu sau:

Bảng 1: **DSSV**

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masv**	Text	4	Khoá chính
2	Malop	Text	4	
3	Hotensv	Text	30	
4	Ngaysinh	Date/time	8	
5	Quequan	Text	30	
6	Gioitinh	Yes/No	1	
7	Hocbong	Number	Double	

Bảng 2: DSDIEM

STT	Fieldname	Data Type	Fieldsize	Note
1	**Masv**	Text	4	Khoá chính
2	Mamon	Text	2	Mã môn học
3	Diem_lan1	Number	Double	Điểm thi lần 1
4	Diem_lan2	Number	Double	Điểm thi lần 2

Bảng 3: DSLOP

STT	Fieldname	Data Type	Fieldsize	Note
1	**Malop**	Text	4	Khoá chính
2	Tenlop	Text	20	
3	Nganh_hoc	Text	30	
4	Khoahoc	Text	2	

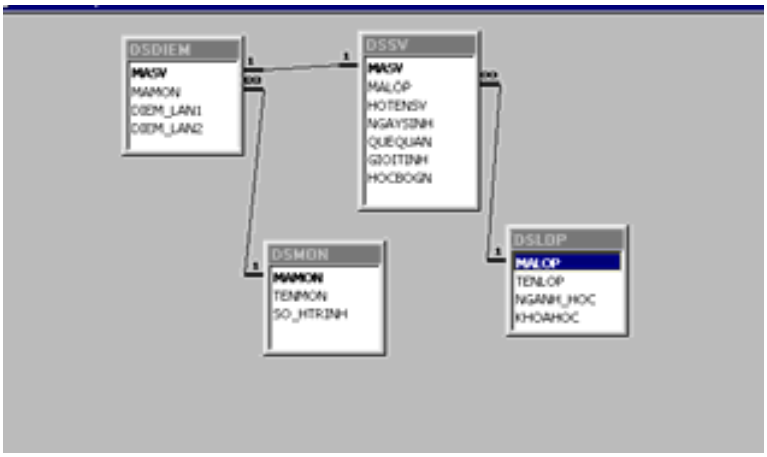
Bảng 3: DSMON

STT	Fieldname	Data Type	Fieldsize	Note
1	**Mamon**	Text	2	Khoá chính
2	Tenmon	Text	20	
3	So_htrinh	Number	Byte	Số học trình

1. Nhập dữ liệu thích hợp cho các bảng dữ liệu.

2. Đặt khoá chính cho các trường: MASV(DSSV), MASV(DSDIEM), MALOP(DSLOP), Mamon(DSMON).

3. Thiết lập các mối quan hệ giữa các bảng theo sơ đồ sau:



4. Tạo một truy vấn **BT1** để hiển thị tổng số sinh viên của mỗi lớp là bao nhiêu?

5. Tạo một truy vấn tham khảo chéo **BT2** để phản ánh tổng số sinh viên xếp loại ***Xsắc, Giỏi, khá, Tb, Yếu*** của mỗi lớp.

6. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT3** để tạo ra bảng dữ liệu DSLUU lưu trữ những sinh viên có Mã lớp bắt đầu là **T** hoặc **H.**

7. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT4** để tạo ra một bảng dữ liệu DSNAM để lưu trữ những sinh viên **nam** sinh trong năm **1976** bao gồm các thông tin: HOTENV, QUEQUAN, GIOTINH.

8. Từ bảng dữ liệu DSSV, tạo một truy vấn **BT5** để tạo ra một bảng dữ liệu DSNU để lưu trữ những sinh viên **nữ** sinh trong **quý 3** năm **1979** bao gồm các thông tin: HOTENV, QUEQUAN, GIOTINH.

9. Tạo một truy vấn **BT6** để tăng học bổng cho những sinh viên **nữ** thêm **30%**.

10. Tạo một truy vấn **BT7** để nối dữ liệu từ bảng DSNAM vào bảng DSNU.

11. Tạo truy vấn **BT8** để xoá những sinh viên có quê quán ở **Đà Nẵng** và có mã lớp bắt đầu là **B**.

BÀI SỐ 6

1. Sử dụng ngôn ngữ SQL để tạo cấu trúc của các bảng dữ liệu sau:

DSTRUONG (**MATRUONG**, TENTRUONG, DIACHI)
DSKHOA (MATRUONG, **MAKHOA**, TENKHOA, SODT)
DANHSACH (**MASV**, MAKHOA, HOTEN, NGAYSINH, LOP, HOCBONG)
BANGDIEM (**MASV**, DTBK1, DTBK2, DTBK3, DTBK4)

Trong đó: Các trường in đậm và gạch chân là khoá chính, kiểu dữ liệu và kích thước của các được mô tả như sau:

Bảng DSTRUONG

MATRUONG TEXT(2), TENTRUONG TEXT(20), DIACHI TEXT(30)

Bảng DSKHOA

MATRUONG TEXT(2), MAKHOA TEXT(4), TENKHOA TEXT(10), SODT TEXT(6). Trường SODT được lập chỉ mục.

Bảng DANHSACH

MASV TEXT(4), MAKHOA TEXT(4), HOTEN TEXT(30), NGAYSINH(DATE/TIME), LOP TEXT(10), HOCBONG (DOUBLE). Trường MASV được lập chỉ mục.

Bảng BANGDIEM

MASV TEXT(4), DTBK1 (DOUBLE), DTBK2 (DOUBLE), DTBK3 (DOUBLE), DTBK4 (DOUBLE).

☞ **Chú ý**

Sau khi tạo cấu trúc các bảng dữ liệuxong, hãy nhập dữ liệu và thiết lập mối quan hệ giữa các bảng phù hợp. Trường HOCBONG trong bảngg DANHSACH chỉ nhập một trong ba giá trị 120000, 180000 hoặc 240000.

2. Sử dụng ngôn ngữ SQL để thay đổi cấu trúc của bảng dữ liệu

a. Thêm trường GHICHU có kiểu MEMO vào trong bảng DANHSACH

b. Thêm trường TBCONG có kiểu DOUBLE vào trong bảng BANGDIEM

c. Thêm trường QUEQUAN có kiểu TEXT và GIOITINH có kiểu YES/NO vào bảng DANHSACH và lập chỉ mục trường QUEQUAN.

3. Sử dụng ngôn ngữ SQL để tạo các truy vấn chọn sau

a. Chọn MATRUONG, MAKHOA, TENKHOA trong bảng DSKHOA.

b. Chọn MATRUONG, MAKHOA, SODT trong bảng DSKHOA của những trường có MATRUONG bắt đầu là Q.

c. Chọn MASV, MAKHOA, HOTEN của những sinh viên sinh trong khoảng thời gian từ 20/10/74 đến 20/10/76 trong bảng DANHSACH.

d. Chọn MASV, HOTEN, LOP, HOCBONG của những sinh viên có MASV bắt đầu là T và thuộc lớp Tin học hoặc Hoá học hoặc kinh tế trong bảng DANHSACH (HOTEN đổi thành Họ và tên).

e. Chọn những sinh viên có tên THANH sinh trong tháng 10/76 hoặc có HOCBONG trong khoảng từ 150000 đến 200000.

4. Sử dụng ngông ngữ SQL để tạo các truy vấn tính tổng sau:

a. Tạo một truy vấn để tính tổng HOCBONG của mỗi khoa.

b. Tạo một truy vấn thống kê xem mỗi khoa số lượng sinh viên là bao nhiêu?

c. Tạo một truy vấn để tính tổng HOCBONG của mỗi lớp trong mỗi khoa.

d. Tạo một truy vấn thống kê xem mỗi trường có bao nhiêu khoa?

5. Sử dụng ngôn ngữ SQL để tạo các truy vấn tham số sau:

a. Tạo một truy vấn tham số thống kê xem mỗi mức HOCBONG mỗi khoa có bao nhiêu sinh viên.

b. Tạo một truy vấn tham số để thống kê xem số lượng sinh viên sinh trong mỗi tháng của năm 1979 của mỗi khoa là bao nhiêu?

6. Sử dụng ngông ngữ SQL để tạo các truy vấn tạo bảng sau:

a. Tạo một bảng DIEMLUU gồm tất cả các trường trong bảng DIEMTHI.

b. Tạo một bảng DSLUU gồm các trường MASV, MAKHOA, HOTEN từ bảng DANHSACH của những sinh viên sinh trước ngày 20/11/76.

c. Tạo một bảng DSLUU1 gồm các trường HOTEN, LOP của những sinh viên thuộc lớp **Tin K25A** và sinh năm 1985 hoặc trước năm 1978.

7. Sử dụng ngôn ngữ SQL để tạo các truy vấn nối dữ liệu sau:

a. Tạo một truy vấn nối dữ liệu từ bảng DIEMLUU vào DIEMTHI nhưng chỉ gồm các trường DTBK1, DTBK2.

b. Tạo một truy vấn nối dữ liệu từ bảng DIEMLUU vào DIEMTHI nhưng chỉ gồm các trường DTBK1, DTBK2, DTBK3 đối với những bản ghi ghi có DTBK3>=8.

8. Sử dụng ngôn ngữ SQL để tạo các truy vấn cập nhật dữ liệu sau:

a. Tính giá trị trường DTBCONG của bảng DIEMTHI theo công thức (DTBK1+ DTBK2+ DTBK3+ DTBK4)/4.

b. Tăng HOCBONG thêm 100.000 cho những sinh viên có MASV bắt đầu là A trong bảng DANHSACH.

c. Giảm HOCBONG đi 50.000 cho những sinh viên có năm sinh từ 1975 đến 1978 hoặc những sinh viên có tên NHAN.

9. Sử dụng ngôn ngữ SQL để tạo các truy vấn xoá sau:

a. Tạo một truy vấn xoá những sinh viên thuộc khoa toán.

b. Tạo một truy vấn xoá những sinh viên sinh trong khoảng thời gian từ 20/10/74 đến 20/10/76 hoặc có HOCBONG=120.000.

c. Tạo một truy vấn xoá những sinh viên có họ NGUYEN hoặc tên THANH sinh trong tháng 7 năm 1978.

10. Sử dụng ngôn ngữ SQL để tạo các truy vấn dựa trên nhiều bảng sau:

a. Tạo truy vấn để hiển thị HOTEN, TENKHOA, LOP, HOCBONG từ 2 bảng dữ liệu DSKHOA và DANHSACH.

b. Tạo một truy vấn để hiển thị TENTRUONG, TENKHOA, HOTEN của những sinh viên sinh trong năm 1980 từ 3 bảng DSTRUONG, DSKHOA, DANHSACH.

c. Tạo một truy vấn để hiển thị TENTRUONG, TENKHOA, HOTEN, DTBK1, DTBK2 tử 4 bảng dữ liệu DSTRUONG, DSKHOA, DANHSACH, DIEMTHI.

11. Sử dụng ngôn ngữ SQL để tạo các truy vấn con sau:

a. Tạo một truy vấn để hiển thị HOTEN, NGAYSINH, LOP của những sinh viên có DTBK4>=5.

b. Tạo một truy vấn để hiển thị it nhất một sinh viên có DTBK2<=4.

12. Sử dụng ngôn ngữ SQL để tạo các truy vấn hội sau:

a. Tạo truy vấn hội để hiển thị HOTEN, DTBK1, DTBK2 từ 2 bảng dữ liệu DIEMTHI và DIEMLUU.

b. Tạo truy vấn hội để hiển thị HOTEN, DTBK1, DTBK2, DTBK3 từ 2 bảng dữ liệu DIEMTHI và DIEMLUU nhưng chỉ hiển thị những sinh viên có DTBK3>=7.

BÀI SỐ 7

Sử dụng CSDL **QLSV.MDB** trong **bài tập số 5** để thực hiện các yêu cầu sau:

1. Tạo một form có tên **Nhaplop** để nhập dữ liệu cho bảng **dslop** như sau:

2. Tạo một form chính phụ như sau:

3. Tạo một form có tên **Hienthi** để hiển thị danh sách sinh viên như sau :

4. Hãy thiết kế form theo yêu cầu sau:

Hãy gắn các chức năng phù hợp với các mục chọn.

5. Hãy thiết kế một **Report** theo yêu cầu sau:

BÀI SỐ 8

Cho CSDL **QLSVIEN.MBD** gồm các bảng dữ liệu sau:

DSSVIEN(**MASV**, MALOP, HOTENSVIEN, NGAYSINH, GTINH, QQUAN)

DSMONHOC(**MAMON**, TENMON, SOH_TRINH)

DSDIEMTHI(**MASV**, MAMON, DIEM_LAN1, DIEM_LAN2)

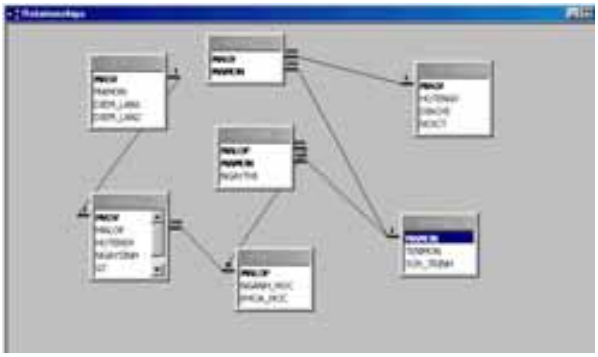
DSGIAOVIEN(**MAGV**, HOTENGVIEN, DIACHI, NOI_CT)

DSLOPHOC(**MALOP**, NGANH_HOC, KHOA_HOC)

GVMONHOC(**MAGV**, **MAMON**)

DSLOPMON(**MALOP**, **MAMON**, NGAYTHI)

Mối quan hệ giữa các bảng dữ liệu theo sơ đồ sau:



1. Tạo các biểu mẫu để cập nhật dữ liệu cho các bảng trên, chú ý sử dụng combo box hoặc List box đối với các trường cần thiết.

2. Tạo báo cáo để hiển thị danh sách sinh viên trong một lớp học nào đó, bao gồm cả ngành học tương ứng.

3. Tạo báo cáo để hiển thị danh sách sinh viên trong từng lớp học, bao gồm cả ngành học, trong đó các sinh viên cùng một lớp thì tên lớp được nhóm lại với nhau.

4. Tạo báo cáo để hiển thị danh sách giáo viên đã giảng dạy các một học cho một lớp nào đó.

5. Tạo báo cáo để hiển thị danh sách giáo viên đã tham gia giảng dạy các môn học cho nhiều lớp học, trong đó các giáo viên giảng dạy các môn học cho một lớp học được nhóm lại với nhau.

6. Tạo báo cáo để hiển thị điểm thi các môn học của một sinh viên.

7. Tạo báo cáo để hiển thị điểm thi một môn học của một lớp nào đó.

CHƯƠNG TRÌNH TỔNG THỂ CẢI CÁCH HÀNH CHÍNH

CỦA CHÍNH PHỦ GIAI ĐOẠN (2001-2010)

BAN ĐIỀU HÀNH ĐỀ ÁN 112

GIÁO TRÌNH
PHÂN TÍCH, THIẾT KẾ, XÂY DỰNG VÀ QUẢN TRỊ CÁC HỆ THỐNG CƠ SỞ DỮ LIỆU

(Giáo trình nộp lần 2)

Tháng 8 - 2004

MỤC LỤC

Lời giới thiệu 10

Chủ đề I: Khái niệm cơ sở dữ liệu 16

1. Giới thiệu chung 16
2. Các khái niệm cơ bản 17
   2.1. Cơ sở dữ liệu 17
      a. Các hệ thống xử lý tệp truyền thống 18
      b. Cách tiếp cận cơ sở dữ liệu 22
   2.2. Dữ liệu và thông tin 26
      a. Dữ liệu 26
      b. Dữ liệu và thông tin 28
      c. Siêu dữ liệu 30
   2.3. Kiểu thực thể và bảng dữ liệu 32
      a. Thực thể 32
      b. Kiểu thực thể 32
      c. Bảng dữ liệu hay quan hệ 35
   2.4. Thuộc tính và trường dữ liệu 39
      a. Thuộc tính 39
      b. Trường dữ liệu 41
   2.5. Liên kết giữa các thực thể 42
   2.6. Khái niệm toàn vẹn dữ liệu 44
3. Ví dụ về một cơ sở dữ liệu 45
4. Các mô hình dữ liệu 48
   4.1. Khái quát chung 48
   4.2. Mô hình dữ liệu quan hệ 50
   4.3. Một số mô hình dữ liệu khác 51
      a. Mô hình dữ liệu mạng và mô hình dữ liệu phân cấp 51
      b. Mô hình dữ liệu hướng đối tượng 52
5. Kho dữ liệu 53

6. Kết luận 54
7. Bài tập 55
Chủ đề II: Phương pháp phân tích, thiết kế và xây dựng các hệ thống CSDL 56
1. Khái quát chung 56
2. Phân tích yêu cầu 57
2.1. Xác định dữ liệu cần lưu trữ 60
2.2. Xác định ứng dụng sẽ được cài đặt trên cơ sở dữ liệu 61
2.3. Xác định các thao tác thường xuyên được thực hiện 62
2.4. Một số kỹ thuật phân tích yêu cầu 63
a. Kỹ thuật phỏng vấn 63
b. Kỹ thuật sử dụng phiếu khảo sát 65
c. Kỹ thuật phân tích tài liệu 65
3. Thiết kế cơ sở dữ liệu mức khái niệm 66
3.1. Lựa chọn giữa kiểu thực thể mạnh và kiểu thực thể yếu 67
a. Kiểu thực thể mạnh 67
b. Kiểu thực thể yếu 68
3.2. Lựa chọn giữa các loại thuộc tính 70
a. Thuộc tính đơn và gộp 70
b. Thuộc tính đơn trị và thuộc tính đa trị 71
c. Thuộc tính lưu trữ và thuộc tính dẫn xuất 72
d. Thuộc tính khoá hay thuộc tính định danh 73
3.3. Lựa chọn giữa các loại liên kết 77
a. Ba khái niệm cơ bản về liên kết 77
b. Các liên kết một-một (1:1) 81
c. Các liên kết một-nhiều (1:N) 82
d. Các liên kết nhiều-nhiều (M:N) 83
e. Thuộc tính cho liên kết 85

3.4. Lựa chọn giữa kiểu thực thể và thuộc tính.....87
3.5. Lựa chọn giữa kiểu thực thể và liên kết.....89
3.6. Các phân cấp “là-một”.....92
3.7. Một sơ đồ thực thể liên kết.....95
3.8. Mô tả các ràng buộc trên dữ liệu.....105
4. Thiết kế cơ sở dữ liệu mức logic.....109
4.1. Giới thiệu chung về ngôn ngữ SQL.....109
4.2. Ràng buộc toàn vẹn.....110
a. Ràng buộc miền.....111
b. Ràng buộc khoá.....113
c. Các kích hoạt.....114
4.3. Lựa chọn hệ quản trị CSDL.....116
4.4. Bước một: Chuyển đổi các kiểu thực thể thông thường.....117
4.5. Bước hai: Chuyển đổi các kiểu thực thể yếu.....121
4.6. Bước ba: Chuyển đổi các liên kết giữa hai kiểu thực thể.....124
a. Chuyển đổi các liên kết một-nhiều.....124
b. Chuyển đổi các liên kết nhiều-nhiều.....126
c. Chuyển đổi các liên kết một-một.....129
4.7. Bước bốn: Chuyển đổi các kiểu thực thể kết nối 132
a. Trường hợp thuộc tính định danh không xác định 133
b. Trường hợp thuộc tính định danh đã xác định...135
4.8. Bước năm: Chuyển đổi các liên kết đơn.....138
a. Chuyển đổi các liên kết đơn một-nhiều.....139
b. Chuyển đổi các liên kết đơn nhiều-nhiều.....141
4.9. Bước sáu: Chuyển đổi các liên kết giữa nhiều kiểu thực thể.....143
4.10. Bước bảy: Chuyển đổi các phân cấp “là một”..146
4.11. Một sơ đồ cơ sở dữ liệu mức logic.....150

4.12. Kết luận 155
5. Tinh chỉnh sơ đồ dữ liệu và các dạng chuẩn 157
5.1. Tinh chỉnh các lược đồ quan hệ 157
5.2. Phụ thuộc hàm 159
5.3. Các dạng chuẩn 161
a. Dạng chuẩn 1NF 161
b. Dạng chuẩn 2NF 163
c. Dạng chuẩn 3NF 166
6. Thiết kế cơ sở dữ liệu vật lý 171
6.1. Phân tích kích thước và nhu cầu sử dụng dữ liệu 172
6.2. Thiết kế các trường 173
a. Lựa chọn kiểu dữ liệu 173
b. Kiểm soát các toàn vẹn dữ liệu 174
6.3. Tổ chức các bản ghi vật lý và phi chuẩn 177
a. Tổ chức các bản ghi vật lý 177
b. Phi chuẩn 179
6.4. Tổ chức các tệp vật lý 188
a. Tổ chức tệp tuần tự 189
b. Tổ chức tệp chỉ mục 190
c. Tổ chức các tệp băm 196
d. Tổ chức các tệp được phân cụm (clustering) 196
6.5. Lựa chọn và sử dụng các loại chỉ mục 198
6.6. RAID 207
6.7. Tối ưu hoá hiệu năng truy vấn 217
7. Thiết kế an toàn bảo mật 219
7.1. Xác định các nhóm sử dụng và các quyền hạn ... 221
7.2. Xác định quyền hạn thao tác trên dữ liệu 222
7.3. Mã hoá dữ liệu 240
7.4. Vai trò của người quản trị 245
8. Kết luận 247
9. Bài tập 247

Chủ đề III: Quản trị hệ thống cơ sở dữ liệu 250
1. Khái quát chung 250
2. Các mô hình tổ chức cơ sở dữ liệu 251
2.1. Các hệ cơ sở dữ liệu tập trung 252
a. Hệ cơ sở dữ liệu cá nhân 252
b. Hệ cơ sở dữ liệu trung tâm 253
c. Hệ cơ sở dữ liệu khách/chủ 254
2.2. Các hệ cơ sở dữ liệu phân tán 254
a. Hệ cơ sở dữ liệu phân tán thuần nhất 255
b. Hệ cơ sở dữ liệu phân tán không thuần nhất 256
3. Quản trị hệ thống cơ sở dữ liệu 257
3.1. Quản trị nội dung cơ sở dữ liệu 263
a. Theo dõi cơ sở dữ liệu 263
b. Hiệu chỉnh cơ sở dữ liệu 265
c. Sao lưu và phục hồi dữ liệu 266
3.2. Quản trị kỹ thuật 267
a. Cài đặt, cấu hình hệ thống 268
b. Đảm bảo về đường truyền, mạng 269
c. Quản trị tài nguyên cơ sở dữ liệu 269
3.3. Đảm bảo an ninh, phân quyền truy nhập 270
3.4. Một số lỗi thường gặp và cách khắc phục 271
a. Không thể truy cập được cơ sở dữ liệu 271
b. Hệ thống báo lỗi timeout (chậm quá thời gian cho phép) 272
c. Hệ thống báo lỗi Out of Memory (tràn bộ nhớ). 273
d. Không thể cung cấp được dữ liệu 274
e. Virus 274
4. Kết luận 275
5. Bài tập 276
Chủ đề IV: Một số hệ quản trị cơ sở dữ liệu phổ biến 277

1. Giới thiệu chung 277
2. Hệ quản trị cơ sở dữ liệu 280
2.1. Kiến trúc của một hệ quản trị cơ sở dữ liệu 280
2.2. Lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu 282
2.3. Truy vấn trong một hệ quản trị cơ sở dữ liệu 283
3. Một số hệ thống quản trị cơ sở dữ liệu thông dụng . 284
3.1. Các hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ 284
a. Hệ quản trị cơ sở dữ liệu Access 284
b. Hệ quản trị cơ sở dữ liệu Visual Foxpro 295
3.2. Các hệ quản trị cơ sở dữ liệu cỡ lớn 311
a. Hệ quản trị cơ sở dữ liệu SQL Server 311
b. Hệ quản trị cơ sở dữ liệu Oracle 340
3.3. Hệ quản trị cơ sở dữ liệu mã nguồn mở 368
3.4. Nhận xét chung 370
4. Minh hoạ đối với một hệ quản trị cơ sở dữ liệu cụ thể 373
5. Kết luận 373
6. Bài tập 374

Phụ lục A: Các hệ thống CSDL sẽ được xây dựng trong khuôn khổ dự án 112 375

1. Mô hình tổ chức hệ thống CSDL các cấp 375
2. Các cơ sở dữ liệu phục vụ điều hành, quản lý 377
a. Cơ sở dữ liệu phục vụ hệ thống thông tin quản lý đơn thư khiếu nại tố cáo 377
b. Cơ sở dữ liệu phục vụ HTTT luật và văn bản qui phạm pháp luật 378
c. Các cơ sở dữ liệu ứng dụng chuyên ngành 379
3. Các cơ sở dữ liệu phục vụ dịch vụ công 380
4. Các cơ sở dữ liệu quốc gia 382

Phụ lục B: Phân tích, thiết kế cơ sở dữ liệu cho một cửa hàng sách 383
1. Phân tích yêu cầu 383
2. Thiết kế cơ sở dữ liệu mức khái niệm 384
3. Thiết kế cơ sở dữ liệu mức logic 386
4. Tinh chỉnh lược đồ 390
5. Thiết kế cơ sở dữ liệu vật lý 392
6. Vấn đề an toàn bảo mật 395

Tài liệu tham khảo 399

Lời giới thiệu

Tài liệu này được biên soạn trong khuôn khổ đào tạo thuộc Đề án 112 với thời gian tập huấn chính thức là 45 tiết. Mục tiêu chính của tài liệu này là:

- Nhằm trang bị cho học viên các kiến thức và kỹ năng cơ bản trong việc phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu.
- Giúp học viên có đủ kiến thức cơ sở và những kỹ năng cần thiết để có thể quản trị các hệ thống cơ sở dữ liệu
- Cung cấp cho học viên một số kinh nghiệm cần thiết để có thể vận hành, quản trị hệ thống và khắc phục một số sự cố và sửa lỗi thông thường đối với các hệ quản trị CSDL.

Trong khi trình bày, chúng tôi đã cố gắng Việt hoá hầu hết các thuật ngữ tiếng Anh. Với các ví dụ chúng tôi cố gắng sử dụng tiếng Việt để diễn giải và đặt tên các đối tượng bằng tiếng Anh (ví dụ: kiểu thực thể nhân viên EMPLOYEE, thuộc tính mã hiệu nhân viên Emp_no,...).

Đối với phần thực hành của giáo trình, tuỳ theo từng đối tượng học viên khác nhau, giảng viên có thể lựa chọn một trong số các hệ quản trị cơ sở dữ liệu quen thuộc như: Access, Visual Foxpro, Sql Server, Oracle,....

Khi chuẩn bị môi trường thực hành, giảng viên nên lưu ý những điểm sau:

- Đối với hai hệ quản trị cơ sở dữ liệu Access và Visual Foxpro, nên cài đặt trên mỗi máy của học viên.
- Đối với hệ quản trị cơ sở dữ liệu Microsoft SQL Server nên cài đặt một máy làm Server chứa cơ sở dữ liệu, các máy thực hành của học viên chỉ cài ở chế độ Client và kết nối đến máy Server trên. Tạo cho mỗi học viên một tài khoản và một cơ sở dữ liệu khác nhau.
- Đối với hệ quản trị cơ sở dữ liệu Oracle, nên sử dụng một máy có cấu hình tương đối mạnh để cài đặt làm máy Server. Trong trường hợp có thể cho học viên thực hành làm quen với công cụ thiết kế Oracle Designer, cần thiết lập thêm kho thiết kế repository. Sau đó tạo cho mỗi học viên một tài khoản người dùng khác nhau và cấp quyền thiết kế tương ứng. Tại mỗi máy thực hành, cài đặt thêm công cụ thiết kế Oracle Designer và cấu hình kết nối đến Server trên. Cố gắng hướng dẫn học viên từ cách tạo các kiểu thực thể, thuộc tính, liên kết,... bằng tiện ích Entity Relationship Diagrammer, sau đó chuyển đổi sang sơ đồ dữ liệu Data Diagram, rồi cuối cùng sinh ra mã để tạo cơ sở dữ liệu.
- Nếu lựa chọn một trong hai hệ quản trị cơ sở dữ liệu là Microsoft SQL Server hoặc Oracle làm bài giảng thực hành, giảng viên nên bố trí thời gian cài đặt và cấu hình trước để có thêm thời gian trên lớp.

Đối với bài toán lựa chọn để thực hành, giảng viên có thể tự lựa chọn một bài toán phù hợp với mình và học viên, hoặc có thể lựa chọn bài toán đã trình bày trong giáo trình

nêu ở mục 3 chủ đề 1, hoặc cũng có thể lựa chọn bài toán được trình bày trong phụ lục B của giáo trình.

Trong mỗi chủ đề, như trình bày dưới đây, chúng tôi có đề xuất thời gian dự kiến cho từng chủ đề đó cũng như các nội dung tối thiểu mà giảng viên cần phải đảm bảo. Tuy nhiên, tuỳ theo từng đối tượng học viên, giảng viên có thể thay đổi cho phù hợp. Cuối mỗi chủ đề, chúng tôi đều có phần tóm tắt các nội dung đã trình bày nhằm giúp học viên hệ thống hoá lại kiến thức đã được học. Ngoài ra phần bài tập giúp học viên hiểu sâu hơn nội dung đã trình bày, cũng như gợi mở các vấn đề thực tế.

Cụ thể, giáo trình gồm các nội dung chính sau:

Chủ đề 1: Khái niệm cơ sở dữ liệu

Trong chủ đề này, giáo trình tập trung trình bày các khái niệm cơ bản nhất về cơ sở dữ liệu và các khái niệm được sử dụng trong quá trình phân tích, thiết kế cơ sở dữ liệu. Thời lượng dành cho việc trình bày cho chủ đề này có thể không nhiều, khoảng từ 4 đến 6 tiết, với mục đích chủ yếu giúp học viên nắm bắt và hình dung được các khái niệm. Các nội dung như về mô hình dữ liệu, kho dữ liệu, giảng viên không cần đi sâu và có thể trình bày lướt qua.

Chủ đề 2: Phương pháp phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu

Đây được xem là chủ đề quan trọng nhất của giáo trình. Do vậy thời gian dành cho chủ đề này sẽ khoảng từ 14 đến 16 tiết, tuỳ từng đối tượng học viên. Nội dung của chủ đề tập trung vào 6 bước chính của quá trình phân tích, thiết kế và xây dựng hệ thống cơ sở dữ liệu: phân tích yêu cầu, thiết kế cơ sở dữ liệu mức khái niệm, thiết kế cơ sở dữ liệu mức

logic, tinh chỉnh lược đồ dữ liệu và các dạng chuẩn, thiết kế cơ sở dữ liệu mức vật lý, và bước cuối cùng là thiết kế an toàn bảo mật. Đây cũng là các nội dung mà giảng viên cần phải đảm bảo. Tuy nhiên với mục 6 của chủ đề, giảng viên có thể lựa chọn một số mục chủ yếu để giảng và lướt qua các mục còn lại, bởi một số nội dung trong mục này mang tính chất công nghệ và phụ thuộc vào phần cứng, hệ điều hành cũng như hệ quản trị cơ sở dữ liệu.

Ngoài ra, các ví dụ trong chủ đề này được chúng tôi tập trung trình bày để giải quyết một bài toán đã được giới thiệu trong mục 3 chủ đề 1. Giảng viên có thể lựa chọn thêm những ví dụ khác để minh hoạ thêm cho nội dung bài giảng.

Chủ đề 3: Quản trị hệ thống cơ sở dữ liệu

Quản trị hệ thống cơ sở dữ liệu là một phần tất yếu sau khi cơ sở dữ liệu được xây dựng đưa vào sử dụng. Việc quản trị một hệ thống cơ sở dữ liệu bao gồm rất nhiều vấn đề và thông thường, với một hệ thống cơ sở dữ liệu lớn thì cần phải có một đội ngũ chuyên gia tin học có kỹ năng. Toàn bộ nôi dung của chủ đề tập trung chủ yếu vào các vấn đề như: quản trị về mặt nội dung, về mặt kỹ thuật, về đảm bảo an ninh, phân quyền dữ liệu. Trong phần cuối của chủ đề chúng tôi dành riêng một mục để đề cập đến một số lỗi thường gặp trong quá trình quản trị một hệ thống cơ sở dữ liệu và hướng dẫn một số cách khắc phục. Thời lượng dành cho chủ đề này là từ 6 đến 8 tiết.

Trong chủ đề, đối với phần các mô hình tổ chức cơ sở dữ liệu (mục 2), giảng viên không cần tập trung nhiều thời gian và có thể cho học viên đọc trước và trên lớp chỉ cần nêu một số đặc điểm chính mang tính chất khái quát. Nội

dung tối thiểu cần phải đảm bảo trong chủ đề đó là toàn bộ mục 3, Quản trị hệ thống cơ sở dữ liệu.

Chủ đề 4: Một số hệ quản trị cơ sở dữ liệu phổ biến

Trong chủ đề này, chúng tôi tập trung trình bày về kiến trúc chung của một hệ quản trị cơ sở dữ liệu, giúp học viên có được cái nhìn tổng quát về một hệ quản trị cơ sở dữ liệu. Các phần còn lại của chủ đề được dành cho việc trình bày bốn hệ quản trị cơ sở dữ liệu thông dụng hiện nay: Access, Visual Foxpro, SQL Server, Oracle. Đối với mỗi một hệ quản trị cơ sở dữ liệu, chúng tôi trình bày lần lượt các bước để xây dựng một cơ sở dữ liệu, do vậy học viên có thể dễ dàng lựa chọn hệ quản trị cơ sở dữ liệu mà mình ưa thích để thực hành. Thời lượng dành cho chủ đề này có thể là từ 14 đến 16 tiết bao gồm cả thời gian dành cho phần thực hành.

Đối với các nội dung trong chủ đề, giảng viên cần đảm bảo giúp học viên nắm được thế nào là một hệ quản trị cơ sở dữ liệu, các thành phần thường có trong một hệ quản trị cơ sở dữ liệu. Ngoài ra, thời gian chủ yếu trong chủ đề sẽ là phần thực hành cụ thể trên một hệ quản trị cơ sở dữ liệu. Như trên đã trình bày, tuỳ từng đối tượng học viên cũng như khả năng sắp xếp thời gian, giảng viên có thể lựa chọn một trong bốn hệ quản trị cơ sở dữ liệu đã trình bày để giúp học viên thực hành. Tuy nhiên chúng tôi đề xuất là giảng viên nên lựa chọn hệ quản trị cơ sở dữ liệu Microsoft SQL Server, và nếu có điều kiện thì lựa chọn hệ quản trị cơ sở dữ liệu Oracle, bởi với hệ quản trị cơ sở dữ liệu này, giảng viên minh hoạ được cả quá trình thiết kế từ sơ đồ thực thể - liên kết, sơ đồ dữ liệu,... thông qua công cụ Oracle

Designer, phần mà ở các hệ quản trị cơ sở dữ liệu khác chưa có.

Phụ lục A: Các hệ thống CSDL sẽ được xây dựng trong khuôn khổ dự án 112

Nội dung của phụ lục này nhằm giới thiệu với các học viên một số nét chính về hệ thống các cơ sở dữ liệu mang tính chất định hướng sẽ được xây dựng trong khuôn khổ dự án 112. Tuy nhiên, giảng viên cũng nên dành thời gian cho phụ lục này khoảng từ 2 cho đến 4 tiết.

Phụ lụcB: Phân tích, thiết kế cơ sở dữ liệu cho một cửa hàng sách

Phụ lục này được xem như một bài thực hành (case study) thêm cho học viên. Bài toán được trình bày trong phụ lục theo thể loại hỏi đáp, gần gũi với học viên: từ nhu cầu của khách hàng, các luồng suy nghĩ đề giải quyết đối với người phân tích, thiết kế cơ sở dữ liệu, đến quá trình xây dựng và cài đặt cơ sở dữ liệu. Điều quan trọng hơn là bài toán đã đưa ra những vấn đề nảy sinh trong thực tế sau khi hệ cơ sở dữ liệu được đưa vào sử dụng và các hướng giải quyết, đây chính là một quá trình thường gặp với các bài toán thực tế.

Chủ đề I: Khái niệm cơ sở dữ liệu

Mục tiêu

Trong chủ đề này các bạn cần phải nắm được các khái niệm cơ bản nhất về cơ sở dữ liệu, cũng như các khái niệm được sử dụng trong quá trình phân tích, thiết kế, xây dựng cơ sở dữ liệu quan hệ như thực thể, kiểu thực thể, quan hệ, bảng dữ liệu, thuộc tính, trường dữ liệu, cùng các liên kết.

1. Giới thiệu chung

Hơn hai thập kỷ gần đây chúng ta đã được chứng kiến sự phát triển vượt bậc cả về số lượng và chất lượng của các ứng dụng dựa trên cơ sở dữ liệu, chúng ngày càng đóng vai trò quan trọng trong các hoạt động của các tổ chức, doanh nghiệp, hay cá nhân. Cơ sở dữ liệu có thể được sử dụng với những ứng dụng cài đặt riêng lẻ trong máy tính cá nhân hay dưới dạng một cơ sở dữ liệu tập trung trên mạng LAN, hoặc một cơ sở dữ liệu phân tán trên mạng diện rộng được sử dụng bởi rất nhiều các đối tượng khác nhau.

Trong chủ đề này, chúng ta sẽ tìm hiểu những khái niệm cơ bản nhất về cơ sở dữ liệu, cũng như những khái niệm sẽ được sử dụng trong quá trình phân tích, thiết kế cơ sở dữ liệu. Bên cạnh đó, chúng ta cũng sẽ xem xét các mô hình dữ liệu khác nhau đối với cơ sở dữ liệu.Tuy nhiên xin lưu ý rằng các quá trình phân tích, thiết kế, xây dựng và quản trị cơ sở dữ liệu được tập trung trình bày trong giáo trình này sẽ là cơ sở dữ liệu quan hệ, một loại cơ sở dữ liệu được xây dựng dựa trên mô hình dữ liệu quan hệ có lý thuyết toán

học vững chắc nhất. Ngoài ra, đây cũng là loại cơ sở dữ liệu phổ biến nhất, được sử dụng hầu hết trong các hệ quản trị cơ sở dữ liệu thông dụng trên thế giới.

2. Các khái niệm cơ bản

2.1. Cơ sở dữ liệu

Ngày nay, hơn bao giờ hết, sự thành công của một doanh nghiệp phụ thuộc vào khả năng có được những dữ liệu chính xác và kịp thời về các hoạt động của doanh nghiệp, để quản lý và phân tích dữ liệu một cách có hiệu quả nhằm đưa ra được chiến lược cho những hoạt động tiếp theo. Các thuật ngữ như "xa lộ thông tin" đã trở nên phổ biến và vấn đề quản lý và sử dụng thông tin dần trở thành một ngành công nghiệp có doanh số nhiều tỷ đô la.

Người ta đã coi giá trị của dữ liệu như là một thứ tài sản quý giá của doanh nghiệp với hàm lượng thông tin bao hàm trong đó. Tuy nhiên, nếu không có được sự quản lý một cách hữu hiệu đối với lượng dữ liệu ngày càng lớn này, cũng như khả năng đáp ứng yêu cầu của các câu hỏi tìm kiếm, lượng dữ liệu này sẽ trở nên vô nghĩa và khi đó doanh nghiệp sẽ mất đi một nguồn tài sản quý giá. Chính điều này đã thúc đẩy sự cần thiết có được một cơ sở dữ liệu cũng như một hệ quản trị cơ sở dữ liệu hiệu quả và linh hoạt, hỗ trợ doanh nghiệp trong công tác quản lý cũng như tìm kiếm thông tin.

Chúng ta có thể định nghĩa cơ sở dữ liệu như là một bộ sưu tập có tổ chức của các dữ liệu liên quan logic với nhau và được các hệ ứng dụng của một tổ chức cụ thể nào đó sử dụng.

Ví dụ: một người bán hàng có thể lưu danh sách khách hàng trong một cơ sở dữ liệu được cài đặt trong máy sách tay của anh ta với kích thước chỉ khoảng vài megabytes và cơ sở dữ liệu này sẽ phục vụ cho một chương trình quảng cáo sản phẩm chẳng hạn. Nhưng một doanh nghiệp có thể có một cơ sở dữ liệu vô cùng lớn, cỡ khoảng vài terabytes dữ liệu được lưu trong một máy tính lớn mainframe, và nó được sử dụng bởi một ứng dụng hỗ trợ ra quyết định cho doanh nghiệp đó.

Tuy nhiên, từ khi có máy tính, người ta đã tìm cách lưu trữ dữ liệu và xử lý chúng, một trong những hệ thống xử lý dữ liệu đầu tiên đó là các hệ thống xử lý tệp. Mặc dù ngày nay những hệ thống xử lý tệp này không còn được sử dụng rộng rãi nhưng nó vẫn tồn tại trong một số lĩnh vực hẹp. Để có thể thấy được những ưu nhược điểm của hệ thống xử lý tệp cũng như có thể hiểu sâu hơn tại sao người ta lại mong muốn xây dựng lên những cơ sở dữ liệu cùng các ưu điểm của cách tiếp cận cơ sở dữ liệu so với cách tiếp cận dựa trên hệ thống xử lý tệp, chúng ta sẽ xem xét một cách khái quát các khái niệm sau đây:.

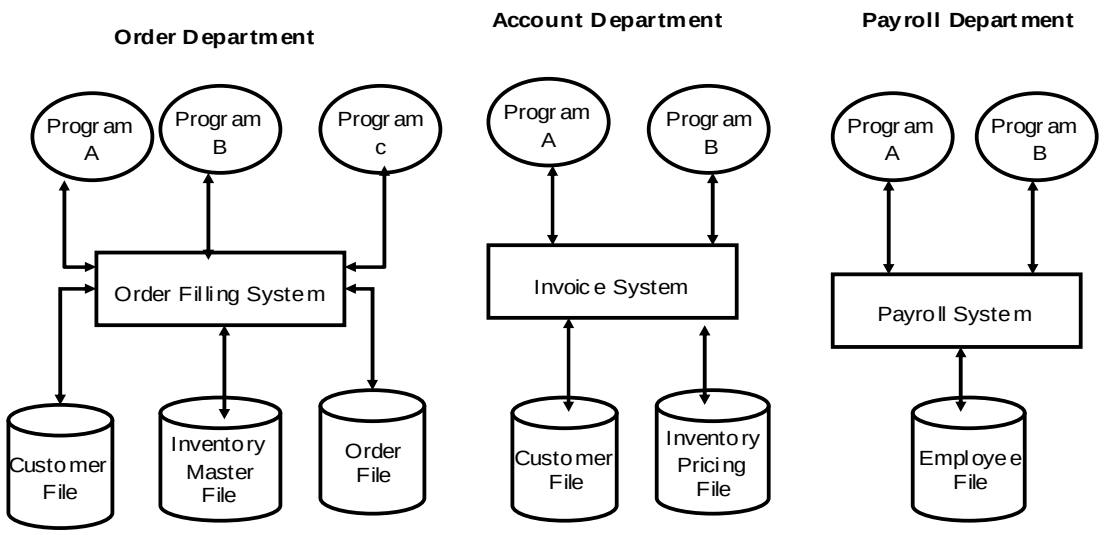
a. Các hệ thống xử lý tệp truyền thống

Như chúng ta đã biết, khi chưa có cơ sở dữ liệu, các máy tính của chúng ta phải xử lý dữ liệu thông qua các hệ thống xử lý tệp truyền thống. Ngày nay, khi khả năng của các máy tính đã tăng lên một cách đáng kể, có thể giải quyết các bài toán lớn, đòi hỏi xử lý những khối dữ liệu đồ sộ, phức tạp, các hệ thống xử lý tệp truyền thống tỏ ra không còn thích hợp và dần dần nó được thay thế bởi các hệ thống xử lý dựa trên cơ sở dữ liệu. Tuy nhiên sự hiểu biết về các hệ thống xử lý tệp vẫn là cần thiết bởi hai lý do cơ bản sau:

- Ngày nay các hệ thống xử lý tệp vẫn còn được sử dụng trong một số hệ thống, đặc biệt là các hệ thống phục vụ sao lưu cơ sở dữ liệu.
- Hiểu được các vấn đề cũng như các giới hạn của các hệ thống xử lý tệp sẽ giúp chúng ta thiết kế các cơ sở dữ liệu tốt hơn.

Ví dụ của một ứng dụng dựa trên hệ thống xử lý tệp:

Một công ty chuyên sản xuất các đồ dùng bằng gỗ đã tiến hành tin học hoá các nghiệp vụ của mình trên máy tính. Thoạt đầu, mỗi một ứng dụng trong công ty đều được xây dựng theo cách tiếp cận xử lý tệp truyền thống. Với cách tiếp cận này, các hệ thống thông tin đều tập trung vào việc xử lý dữ liệu của mỗi một phòng ban đơn lẻ, trong đó các hệ thống cố gắng đáp ứng các yêu cầu của từng nhóm người cụ thể. Do vậy, nhìn toàn bộ vào hệ thống thông tin của toàn công ty thì không có một mô hình cũng như kế hoạch hay sơ đồ tổng thể nào để có thể hoạch định được cho việc phát triển tiếp theo của các ứng dụng. Hệ thống ứng dụng dựa trên trên cách tiếp cận xử lý tệp này được minh hoạ như trong hình 1.2.1.1, đó là các hệ thống tạo lập đơn hàng Order Filling System, hệ thống xử lý hoá đơn Invoice System và hệ thống quản lý lương bổng cho nhân viên Payroll System. Nhìn vào hình vẽ chúng ta thấy rằng, các tệp dữ liệu được kết nối trực tiếp với mỗi chương trình ứng dụng, mỗi một tệp là một tập các bản ghi liên quan. Ví dụ, với hệ thống Order Filling System có 3 tệp: Customer, Inventory Master và Order. Chúng ta nhận thấy rằng có thể có những tệp được lưu lặp lại và sử dụng bởi các ứng dụng khác nhau (như Customer), đây chính là một trong những nhược điểm của các hệ thống dựa trên việc xử lý tệp.



Hình 1.2.1.1 Một hệ thống xử lý tệp truyền thống

Các nhược điểm của các hệ thống xử lý tệp:

- Tính phụ thuộc giữa chương trình và dữ liệu: Các mô tả về tệp dữ liệu được lưu trong mỗi một chương trình ứng dụng sử dụng tệp đó. Như ví dụ trên, chương trình Program A truy nhập cả hai tệp Inventory và Customer, do vậy chương trình này sẽ chứa các mô tả chi tiết của cả 2 tệp này. Điều này dẫn đến, khi có bất cứ một thay đổi nào đến cấu trúc tệp sẽ đòi hỏi cũng phải có các thay đổi tương ứng đến các mô tả trong các chương trình ứng dụng sử dụng tệp đó. Ngoài ra, như trong hình 1.2.1.1, nếu tệp Customer bị sửa đổi thì cả hai hệ thống Order System và Invoice System đều bị ảnh hưởng.
- Lưu trữ trùng lặp dữ liệu: Trong hình 1.2.1.1 chúng ta nhận thấy rằng, hệ thống Order Filling có chứa một tệp Inventory Master, trong khi đó hệ thống Invoice cũng chứa một tệp là Inventory Pricing. Rõ ràng giữa

hai tệp này sẽ có sự lưu trữ trùng lặp dữ liệu, ví dụ các thông tin về mô tả sản phẩm, giá mỗi sản phẩm cũng như số lượng đang có trong kho. Việc lưu trữ dư thừa dữ liệu này gây ra rất nhiều bất cập và đòi hỏi không gian lưu trữ lớn, nó rất dễ dẫn tới tính không toàn vẹn của dữ liệu.

- Việc chia sẻ dữ liệu bị hạn chế: Trong cách tiếp cận xử lý tệp này, mỗi một ứng dụng có các tệp dữ liệu riêng của nó và do vậy giữa những người sử dụng, họ có ít cơ hội để có thể chia sẻ dữ liệu cho nhau nếu không sử dụng chung một phân hệ ứng dụng. Như trong hình 1.2.1.1, các nhân viên trong phòng kế toán Accounting chỉ có thể truy nhập được những tệp trong ứng dụng của mình mà không thể truy nhập được những tệp trong các ứng dụng của các phòng xử l;ý đơn hàng Order hay phòng quản lý tiền lương Payroll. Do vậy, khi người lãnh đạo muốn tạo ra những báo cáo tổng hợp liên phòng thì vô cùng khó khăn bởi dữ liệu được lưu riêng lẻ cũng như có thể không thống nhất giữa các phòng.
- Tốn thời gian xây dựng: Với mỗi một ứng dụng mới, người phát triển chương trình phải bắt đầu lại từ đầu với việc xây dựng các tệp, mô tả chúng cũng như xây dựng các module chương trình để truy nhập tới các tệp đó. Quá trình này tốn khá nhiều thời gian mặc dù nó luôn lặp lại giữa các hệ thống.
- Chi phí cho bảo trì chương trình lớn: Trên thực tế, có đến 80% chi phí được dành cho giai đoạn bảo trì chương trình đối với những ứng dụng dựa trên các hệ thống xử lý tệp truyền thống này.

Tuy nhiên chúng ta cũng cần phải lưu ý rằng, những hạn chế đối với cách tiếp cận dựa trên các hệ thống xử lý tệp truyền thống này cũng có thể gặp phải trong cách tiếp cận dựa trên cơ sở dữ liệu (sẽ được trình bày dưới đây), nếu như cơ sở dữ liệu không được thiết kế tốt, cũng như chúng được xây dựng một cách riêng rẽ, không thống nhất.

b. Cách tiếp cận cơ sở dữ liệu

Cách tiếp cận cơ sở dữ liệu nhấn mạnh đến tính chia sẻ và tích hợp của dữ liệu trong toàn bộ hoạt động của một doanh nghiệp (trong giáo trình này, chúng tôi sử dụng thuật ngữ doanh nghiệp để đại diện chung cho các tổ chức có nhu cầu sử dụng cơ sở dữ liệu trong hoạt động nghiệp vụ của mình). Theo cách tiếp cận này, dữ liệu được lưu trữ có tổ chức, tập trung dưới dạng một cơ sở dữ liệu và giữa các dữ liệu có sự liên kết logic với nhau.

Quay trở lại ví dụ của công ty chuyên sản xuất các sản phẩm đồ gỗ trên, sau một thời gian hoạt động, lãnh đạo của công ty nhận thấy rằng có rất nhiều bất cập cũng như hạn chế trong cách tiếp cận dựa trên hệ thống xử lý tệp, và họ quyết định chuyển sang cách tiệp cận dựa trên cơ sở dữ liệu. Khi đó họ phải tiến hành xây dựng lên một mô hình dữ liệu thống nhất, mô tả các hoạt động nghiệp vụ cho toàn bộ doanh nghiệp, và dựa trên mô hình dữ liệu này dần dần họ đã hình thành lên một cơ sở dữ liệu và được sử dụng thống nhất trong toàn bộ công ty.

Dưới đây chúng ta sẽ xem xét các ưu điểm mà cách tiếp cận dựa trên cơ sở dữ liệu mang lại so với cách tiếp cận dựa trên các hệ thống xử lý tệp truyền thống:

- Tối thiểu hoá dư thừa dữ liệu: Mục đích thiết kế trong cách tiếp cận cơ sở dữ liệu chính là cố gắng tích hợp các tệp dữ liệu riêng rẽ (và cũng là dư thừa) trong mô hình xử lý tệp trước đây vào một cấu trúc lôgic duy nhất. Mỗi một sự kiện cụ thể sẽ được lưu trữ tại một vị trí quy định trong cơ sở dữ liệu. Ví dụ với sự kiện sản phẩm có mã Product_id bằng 3 là một chiếc bàn để máy tính có giá là 170 USD sẽ được lưu trữ tại một vị trí nằm trong bảng PRODUCT. Cũng cần lưu ý rằng, cách tiếp cận cơ sở dữ liệu có thể không loại bỏ hoàn toàn vấn đề dư thừa dữ liệu, nhưng nó đã cho phép người thiết kế có thể kiểm soát được vấn đề này. Ví dụ, đối với mỗi đơn đặt hàng trong bảng ORDER có chứa một trường Customer_id để biểu diễn mối liên kết giữa các đơn đặt hàng và các khách hàng. Tuy nhiên trong một số trường hợp, để nâng cao hiệu quả truy vấn của cơ sở dữ liệu, chúng ta có thể chấp nhận hình thức lưu trữ dư thừa dữ liệu (vấn đề này sẽ được trình bày trong chủ đề 2 của giáo trình).
- Tính độc lập chương trình - dữ liệu: Tính độc lập với dữ liệu được thể hiện ở sự tách biệt rõ ràng giữa chương trình ứng dụng và các mô tả dữ liệu (siêu dữ liệu), do vậy nó cho phép dữ liệu của một doanh nghiệp được phép sửa đổi (tất nhiên là trong một giới hạn nào đó) mà không phải thay đổi lại chương trình thao tác trên các dữ liệu đó.
- Nâng cao tính nhất quán dữ liệu: Với việc loại bỏ (chính xác hơn là kiểm soát được) tính dư thừa dữ liệu, cách tiếp cận cơ sở dữ liệu đã giảm bớt được đáng kể tính không nhất quán của dữ liệu. Ví dụ, nếu

địa chỉ của khách hàng được lưu trữ duy nhất, khi đó, nếu khách hàng thay đổi địa chỉ, chúng ta chỉ việc làm một động tác đơn giản là sửa đổi lại địa chỉ của khách hàng ở vị trí duy nhất đó, các truy vấn liên quan đến giá trị này sẽ được tự động nhận giá trị mới khi được thực hiện. Như vậy, vấn đề về cập nhật, sửa đổi dữ liệu rất đơn giản và nhất quán. Một điểm quan trọng hơn, cách tiếp cận cơ sở dữ liệu sẽ làm giảm đáng kể không gian nhớ so với phương pháp lưu trữ dữ liệu trong các hệ thống xử lý tệp truyền thống.

- Nâng cao tính chia sẻ dữ liệu: cơ sở dữ liệu được thiết kế nhằm mục đích chia sẻ dữ liệu. Mỗi một người sử dụng hợp pháp được gán những quyền sử dụng cơ sở dữ liệu khác nhau, đối với mỗi một người (hoặc nhóm người) sử dụng sẽ được cung cấp những khung nhìn sử dụng khác nhau, do vậy họ có thể chủ động trong việc chia sẻ dữ liệu thông qua các quyền kiểm soát dữ liệu.
- Nâng cao tính sản phẩm của ứng dụng: Một lợi thế đáng kể của cách tiếp cận cơ sở dữ liệu đó là giảm đáng kể chi phí và thời gian cho việc phát triển những ứng dụng nghiệp vụ. Có hai lý do quan trọng mà các ứng dụng dựa trên cơ sở dữ liệu có thể được xây dựng nhanh chóng hơn so với cách tiếp cận của các ứng dụng dựa trên tệp truyền thống, đó là: khi xây dựng các ứng dụng mới dựa trên cơ sở dữ liệu, người lập trình không phải quan tâm đến vấn đề thiết kế các tệp hay các thao tác chi tiết mức thấp. Lý do thứ hai đó là các hệ quản trị cơ sở dữ liệu thường cung cấp một loạt các công cụ hỗ trợ mức cao trong quá trình

xây dựng ứng dụng, như các công cụ sản sinh mẫu biểu sinh báo cáo hay các giao diện nhập liệu cho chương trình ứng dụng.

- Áp đặt dễ dàng các tiêu chuẩn: Với cách tiếp cận cơ sở dữ liệu, các tiêu chuẩn như quy ước đặt tên, sử dụng qui trình thống nhất trong việc truy nhập, cập nhật và bảo vệ dữ liệu dễ dàng được thực hiện. Các chức năng quản trị cơ sở dữ liệu thường cung cấp một loạt các công cụ nhằm đảm bảo những tiêu chuẩn này luôn phải được thoả mãn. Tính chất này làm cho việc bảo trì và trao đổi dữ liệu trở nên dễ dàng và thuận lợi hơn.
- Nâng cao chất lượng dữ liệu: Với cách tiếp cận cơ sở dữ liệu, người thiết kế có thể xây dựng các ràng buộc toàn vẹn nhằm đảm bảo dữ liệu khi được đưa vào cơ sở dữ liệu phải thoả mãn một số qui tắc nào đó. Hoặc đối với những cơ sở dữ liệu lớn như kho dữ liệu, thì thông thường dữ liệu được làm sạch trước khi được đưa vào kho dữ liệu.
- Nâng cao khả năng truy nhập và trả lại dữ liệu: Với các cơ sở dữ liệu quan hệ, người sử dụng không cần nhiều kỹ năng về lập trình cũng có thể dễ dàng tìm kiếm và hiển thị dữ liệu thông qua một ngôn ngữ hỏi đáp dữ liệu rất thân thiện đã được tiêu chuẩn hoá.
- Giảm chi phí bảo trì chương trình: Tính chất này là hệ quả của tính độc lập chương trình - dữ liệu. Bởi chúng ta biết rằng dữ liệu có thể bị thay đổi vì những lý do khác nhau như: chúng ta có thể bổ sung thêm những kiểu dữ liệu mới, sửa đổi các định dạng cũng

như các kiểu dữ liệu đã có,... Tính độc lập chương trình - dữ liệu cho phép xây dựng chương trình tách biệt khỏi dữ liệu ở một mức độ nhất định, do vậy khi có sự sửa đổi, chúng ta chỉ việc sửa đổi một số module thao tác trên dữ liệu mà không nhất thiết phải sửa đổi lại toàn bộ chương trình. Chính điều này đã dẫn đến việc giảm bớt được chi phí bảo trì chương trình.

Trên đây là những ưu điểm mà cách tiếp cận cơ sở dữ liệu mang lại. Một cơ sở dữ liệu thường được tích hợp vào trong một hệ quản trị cơ sở dữ liệu, đó là một chương trình máy tính (hay chính xác hơn là một bộ các chương trình máy tính) được xây dựng để quản lý một cơ sở dữ liệu. Chúng ta sẽ tìm hiểu kỹ hơn về các hệ quản trị cơ sở dữ liệu trong chủ đề 4 của giáo trình.

Phần tiếp theo chúng ta sẽ tìm hiểu các khái niệm liên quan đến cơ sở dữ liệu.

2.2. Dữ liệu và thông tin

a. Dữ liệu

Như trên chúng ta đã nhắc đến dữ liệu, vậy thì dữ liệu thực sự là gì? Trong máy tính, thuật ngữ dữ liệu được xem như là các sự kiện được biết đến mà có thể ghi lại và lưu trữ trên các thiết bị ghi nhớ của máy tính.

Ví dụ trong cơ sở dữ liệu của người bán hàng nói trên, dữ liệu bao gồm các sự kiện như tên khách hàng, địa chỉ hay số điện thoại.

Ngày nay, ngoài những dữ liệu thông thường như số và ký tự, các cơ sở dữ liệu còn được sử dụng để lưu trữ các đối tượng đặc biệt hơn như văn bản, hình ảnh, âm thanh và thậm chí cả một đoạn phim. Ví dụ, với cơ sở dữ liệu của người bán hàng trên, anh ta có thể lưu ảnh của khách hàng, thậm chí có thể là các đoạn ghi âm hay băng hình ghi lại những cuộc nói chuyện với khách hàng gần đây nhất. Để có thể phản ánh được những thực tế này, chúng ta cần hiểu rộng hơn về khái niệm dữ liệu như sau:

Dữ liệu bao gồm các sự kiện, văn bản, hình ảnh, đồ hoạ, âm thanh, đoạn phim có một giá trị nào đó đối với người sử dụng chúng và được lưu trữ, xử lý trong máy tính.

Chúng ta đã định nghĩa cơ sở dữ liệu là một sưu tập các dữ liệu được tổ chức, liên quan logic với nhau. Từ được tổ chức ở đây ngụ ý rằng dữ liệu được cấu trúc sao cho chúng có thể dễ dàng được lưu trữ, thao tác, tìm kiếm và trả về cho người sử dụng. Còn từ liên quan trong định nghĩa ngụ ý rằng dữ liệu mô tả một lĩnh vực quan tâm của một người hoặc nhóm người sử dụng nào đó và do vậy họ có thể sử dụng chung để trả lời các câu hỏi liên quan đến lĩnh vực.

Ví dụ: một cơ sở dữ liệu của một cửa hàng sửa chữa xe ô tô có thể chứa các dữ liệu về khách hàng (bao gồm các khoản mục dữ liệu như tên khách hàng, địa chỉ, số điện thoại, thẻ tín dụng), các dữ liệu về xe ô tô của khách hàng (bao gồm các khoản dữ liệu như hãng xe, đời xe, năm sản xuất) và nhật ký các lần sửa chữa cho mỗi chiếc xe (như ngày phát sinh dịch vụ, tên thợ sửa, loại hình sửa chữa, và số tiền đã thanh toán).

b. Dữ liệu và thông tin

Hai thuật ngữ dữ liệu và thông tin liên quan chặt chẽ với nhau, trên thực tế chúng thường được sử dụng thay đổi cho nhau. Tuy nhiên, giữa chúng cũng có sự khác biệt và chúng ta nên phân biệt giữa dữ liệu và thông tin.

Chúng ta quan niệm thông tin là những gì được tạo nên nhằm giảm sự không xác định. Nói cách khác chúng ta có thể hiểu thông tin như là dữ liệu đã được xử lý theo cách mà chúng có thể làm tăng hàm lượng tri thức cho người sử dụng nó.

Ví dụ: Hoá đơn N=852 của nhà cung cấp A cần phải thanh toán chậm nhất là vào ngày 10-12-1999 đó là một thông tin.

Để thể hiện thông tin người ta sử dụng dữ liệu. Thông tin được thể hiện qua các giá trị của dữ liệu. Với ví dụ trên thông tin toàn bộ được thể hiện qua ba loại dữ liệu: đó là số thứ tự hoá đơn, tên nhà cung cấp và ngày giới hạn cho việc thanh toán.

Chúng ta hãy xem thêm ví dụ sau, đây là một danh sách các sự kiện:

Nguyễn Văn M.	3245213
Trần Đức T.	7214613
Mai Văn H.	8145184
Trịnh Thị M	5524812

Những sự kiện này rõ ràng thoả mãn định nghĩa của chúng ta về dữ liệu, tuy nhiên theo cách biểu diễn trên, hầu hết mọi người đều cho rằng những dữ liệu này chưa thể mang lại thông tin. Chúng ta có thể đoán rằng đây là một danh sách bao gồm tên người cùng với số bảo hiểm xã hội của

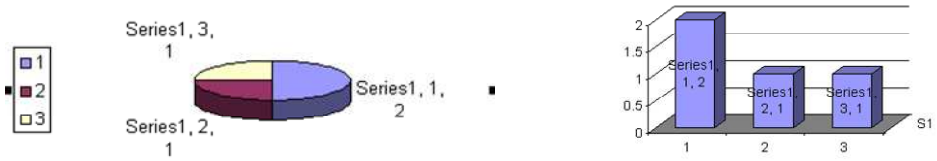
họ, nhưng thực sự chúng ta vẫn chưa có được thông tin gì về toàn bộ các số liệu này. Tuy nhiên, nếu đặt danh sách này vào một ngữ cảnh như trong hình 2.2.1a, với việc bổ sung thêm một số dữ liệu và cấu trúc của các dữ liệu đó, chúng ta sẽ nhận ra ngay rằng đó là một phần của bảng kết quả học tập cho một khoá học cụ thể. Như vậy chúng sẽ là hữu ích cho các nhân viên ghi danh hay những người quản lý lớp học chẳng hạn.

Bảng kết quả học tập

Course: KT2K38
Kinh tế
Semester: Học kỳ 2 năm 1999

Name	Id	Major	GPA
Nguyễn Văn M.	3245213	KTCT	7.5
Trần Đức T.	7214613	CNTT	6.8
Mai Văn H.	8145184	KTVM	8.2
Trịnh Thị L.	5524812	KTVM	6.9

(a) Dữ liệu được đặt trong ngữ cảnh



(b) Dữ liệu được tóm lược và biểu diễn dưới dạng trực quan

Hình 1.2.2.1 Chuyển đổi dữ liệu thành thông tin

Một cách khác để chuyển dữ liệu thành thông tin, đó là tóm lược chúng lại hoặc xử lý và trình bày chúng dưới các dạng trực quan hơn như đồ thị hay biểu đồ.

Trong hình 1.2.2.1b là các dữ liệu về kết quả học tập của các sinh viên đã được tóm lược và biểu diễn dưới dạng thông tin đồ hoạ. Rõ ràng dựa trên những thông tin này, việc quyết định có cần bổ sung thêm các khoá học mới hay thuê thêm giáo viên hay không sẽ dễ dàng được thực hiện hơn.

Trên thực tế, ngày nay các cơ sở dữ liệu có thể chứa đựng hoặc dữ liệu hoặc thông tin hoặc cả hai như trong định nghĩa chúng ta đã trình bày. Cũng có thể dữ liệu được xử lý trước và được lưu trữ dưới dạng một mẫu tóm lược trong cơ sở dữ liệu để phục vụ cho vấn đề hỗ trợ ra quyết định. Trong giáo trình này, chúng tôi sử dụng thuật ngữ cơ sở dữ liệu mà không phân biệt nội dung của nó là dữ liệu hay thông tin.

c. Siêu dữ liệu

Như chúng ta đã thấy, dữ liệu chỉ trở nên hữu ích khi chúng được đặt trong một ngữ cảnh nào đó. Cơ chế chính cung cấp ngữ cảnh cho dữ liệu đó chính là siêu dữ liệu, do vậy chúng ta có thể hiểu siêu dữ liệu là dữ liệu mô tả các tính chất hoặc đặc trưng của dữ liệu. Các tính chất này có thể là các dữ liệu về định nghĩa, về cấu trúc dữ liệu, về các luật hay các ràng buộc.

Quay trở lại ví dụ về bảng kết quả học tập trong hình 1.2.2.1a. Với mỗi khoản mục dữ liệu xuất hiện trong bảng, siêu dữ liệu cho biết các thông tin về tên, kiểu dữ liệu, độ dài, khoảng giá trị cho phép, và một số mô tả ngắn gọn đối với mỗi khoản mục dữ liệu đó.

Data Item			**Value**		
Name	**Type**	**Length**	**Mi**	**Max**	**Description**
Course	Charact	30			Course_id và
Section	Integer	1	1	9	Số hiệu học

Semeste	Charact	10			Học kỳ và
Name	Charact	30			Tên sinh viên
Id	Integer	9			Số hiệu sinh
Major	Charact	4			Môn học
GPA	Decimal	3	0.0	10.0	Điểm môn

Bảng 1.2.2.1: Ví dụ về một siêu dữ liệu cho bảng phân công lớp học

Chúng ta cần phải phân biệt rõ giữa dữ liệu và siêu dữ liệu: siêu dữ liệu là phần đã được tách khỏi dữ liệu. Cụ thể hơn, siêu dữ liệu mô tả các tính chất của dữ liệu nhưng không bao gồm dữ liệu. Do đó, siêu dữ liệu trong bảng 1.2.2.1 không bao gồm bất cứ một bộ dữ liệu nào từ bảng kết quả học tập trong hình 1.2.2.1a.

Siêu dữ liệu cho phép người thiết kế cơ sở dữ liệu và người sử dụng biết có dữ liệu gì, tồn tại ở đâu, các khoản mục dữ liệu khác nhau ở điểm gì. Việc quản lý siêu dữ liệu cũng quan trọng như việc quản lý dữ liệu liên quan với nó, bởi dữ liệu nếu không được phân biệt rõ ràng sẽ gây ra nhầm lẫn, sai sót và không thể diễn giải nổi. Trong cơ sở dữ liệu siêu dữ liệu thường được lưu trữ trong một vùng qui định mà chúng ta thường gọi là từ điển dữ liệu (data dictionary).

Dữ liệu tạo nên nội dung của cơ sở dữ liệu, tuy nhiên nó không quyết định tính hiệu quả của cơ sở dữ liệu. Bởi tính hiệu quả được thể hiện qua việc thiết kế và khai thác cơ sở dữ liệu, chúng ta cần làm quen với những khái niệm tiếp theo bởi chúng sẽ được sử dụng thường xuyên trong quá trình phân tích, thiết kế cơ sở dữ liệu.

2.3. Kiểu thực thể và bảng dữ liệu

a. Thực thể

Thực thể là một khái niệm được sử dụng thường xuyên trong quá trình phân tích, thiết kế cơ sở dữ liệu. Vậy thực thể là gì?

Chúng ta nhận thấy rằng trong thế giới thực, tồn tại rất nhiều các đối tượng khác nhau, ví dụ như một người trong một tổ chức, một sinh viên trong một trường đại học, hay thậm chí một phòng ban trong một viện nghiên cứu, tất cả những đối tượng đó đều là các thực thể khác nhau. Ngoài ra những đối tượng trừu tượng như một ngành khoa học, chúng ta cũng có thể coi là một thực thể.

Vậy ta có thể định nghĩa thực thể như là một đối tượng cụ thể hay trừu tượng trong thế giới thực mà nó tồn tại và có thể phân biệt với các đối tượng khác.

Rõ ràng một sinh viên Nguyễn Văn A là một thực thể được phân biệt hoàn toàn với một sinh viên Trần Thị B cũng là một thực thể.

b. Kiểu thực thể

Từ định nghĩa về thực thể nêu trên, chúng ta nhận thấy rằng trong thế giới thực có rất nhiều các thực thể giống nhau ở một số khía cạnh nào đó. Như các sinh viên trong một trường đại học đều là các thực thể, và họ có chung một số điểm đó là đều có mã hiệu sinh viên, có tên, ngày nhập học, số hiệu lớp. Hay với các phòng ban trong một viện nghiên cứu cũng đều là các thực thể, giữa các thực thể này đều có một điểm chung đó là chúng đều có tên, có một nhân viên

làm trưởng phòng, cũng như có một số lượng các nhân viên đang làm việc trong phòng.

Như vậy, chúng ta có thể coi một nhóm các thực thể giống nhau hình thành lên một kiểu thực thể hay còn gọi là tập thực thể.

Ví dụ: tất cả các sinh viên trong một trường đại học tạo thành một kiểu thực thể là STUDENT. Tất cả nhân viên của một doanh nghiệp hình thành lên một kiểu thực thể là EMPLOYEE. Tất cả các phòng ban trong một viện nghiên cứu tạo thành một kiểu thực thể DEPARTMENT.

Tuy nhiên chúng ta cũng nhận thấy rằng khái niệm "thực thể giống nhau" không được xác định một cách chính xác và người ta có thể đưa ra một số vô hạn các tính chất khác nhau để hình thành lên một kiểu thực thể, điều này phụ thuộc vào nhu cầu quản lý của bài toán. Như trong ví dụ về kiểu thực thể nhân viên EMPLOYEE, nếu chúng ta muốn phân biệt chi tiết hơn nữa như theo loại nhân viên là làm việc hưởng lương theo ngày công, hay làm việc hưởng lương theo hợp đồng, khi đó chúng ta phải phân nhóm tiếp và sẽ hình thành lên hai kiểu thực thể cụ thể hơn đó là kiểu thực thể nhân viên hưởng lương theo ngày EMPLOYEE_DAY và kiểu thực thể nhân viên hưởng lương theo hợp đồng EMPLOYEE_CONTRACT.

Vậy chúng ta có thể hiểu một kiểu thực thể như là một nhóm các thực thể giống nhau ở một số điểm nhất định nào đó mà chúng ta cần quan tâm.

Một trong các bước chính trong quá trình phân tích thiết kế cơ sở dữ liệu đó là: xây dựng một sơ đồ đối với thế giới thực, nói một cách cụ thể hơn tức là lựa chọn các kiểu thực

thể cần phải mô tả và mỗi quan hệ giữa các kiểu thực thể đó.

Chúng ta cần phải phân biệt sự khác nhau giữa kiểu thực thể và thể hiện của thực thể (hay thực thể):

- Một kiểu thực thể là là một tập các thực thể có chung một số tính chất hoặc đặc trưng nào đó. Mỗi một kiểu thực thể phải được đặt một tên duy nhất không trùng với các kiểu thực thể đã có.
- Còn thể hiện của thực thể là một đơn xuất hiện của kiểu thực thể.

Hình 1.2.3.1 Kiểu thực thể EMPLOYEE và 2 thể hiện thực thể của nó

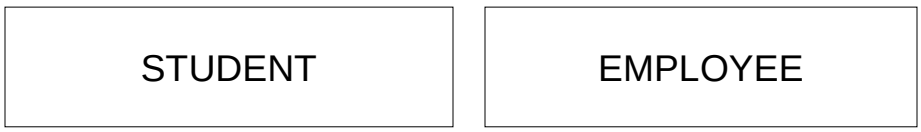
Hình 1.2.3.1 trên minh hoạ sự khác nhau giữa kiểu thực thể EMPLOYEE và 2 thể hiện thực thể của nó.

Khi phân tích thiết kế một cơ sở dữ liệu, một kiểu thực thể chỉ được mô tả một lần (sử dụng siêu dữ liệu), trong khi đó các thể hiện của kiểu thực thể đó lại được biểu diễn bởi dữ liệu lưu trong cơ sở dữ liệu.

Ví dụ có một kiểu thực thể là EMPLOYEE trong hầu hết các tổ chức, tuy nhiên có hàng trăm, thậm chí hàng nghìn thể hiện của kiểu thực thể đó được lưu trong cơ sở dữ liệu.

Trong giáo trình này, chúng tôi sẽ sử dụng thuật ngữ thực thể thay cho thuật ngữ thể hiện thực thể nếu ý nghĩa của nó là rõ ràng trong ngữ cảnh trình bày.

Để đặt tên cho kiểu thực thể, chúng ta dùng các danh từ viết liền với các chữ cái được viết hoa, bao quanh bởi một hình chữ nhật như hình 1.2.3.2 dưới đây:



Hình 1.2.3.2 Kiểu thực thể

Các kiểu thực thể được đặc tả bởi một tập các đặc trưng mà chúng ta qui ước gọi là các thuộc tính (như chúng ta sẽ thấy dưới đây). Do vậy với các thành viên tham gia vào một kiểu thực thể đòi hỏi chúng ta phải xác định được tập các thuộc tính chúng nhất giữa chúng.

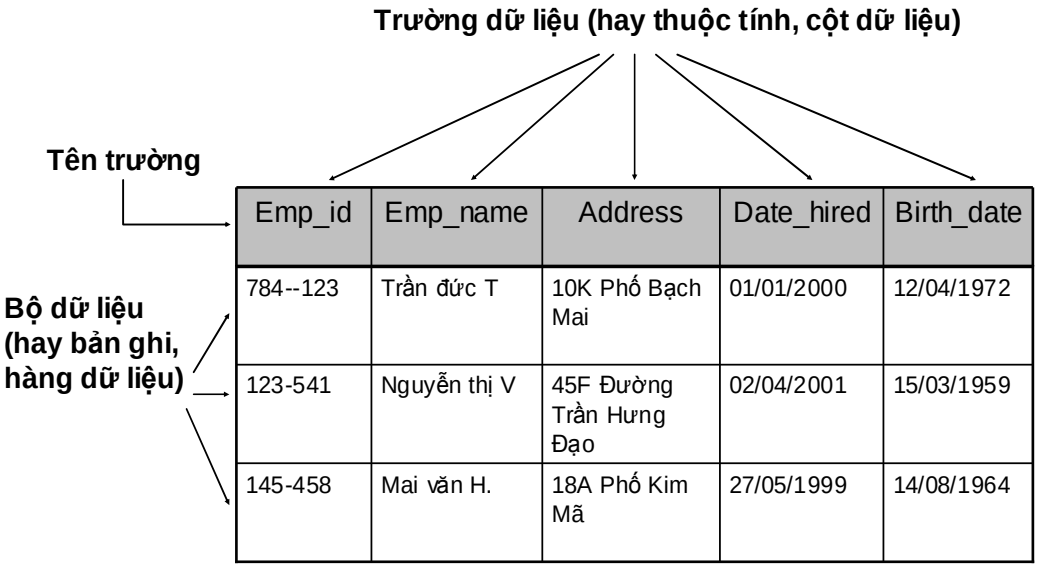
c. Bảng dữ liệu hay quan hệ

Trong quá trình phân tích cơ sở dữ liệu, chúng ta sử dụng khái niệm kiểu thực thể để minh hoạ cho một lớp các thực thể giống nhau. Bảng dữ liệu chính là hình thức thể hiện cụ

thể của kiểu thực thể khi chúng ta xây dựng cơ sở dữ liệu, và bảng dữ liệu được sử dụng để lưu dữ liệu về các thực thể trong lớp thực thể đó.

Chúng ta có thể hình dung bảng dữ liệu là một tập các bộ dữ liệu, hay bản ghi dữ liệu, mỗi bộ có cùng một số lượng thuộc tính như nhau nhưng có thể khác nhau về giá trị. Bảng dữ liệu trong cách tiếp cận cơ sở dữ liệu quan hệ, được hiểu chính xác hơn bởi cụm từ quan hệ. Một quan hệ bao gồm một lược đồ quan hệ và một thể hiện quan hệ. Thể hiện quan hệ chính là một bảng, còn lược đồ quan hệ mô tả tiêu đề các cột của bảng đó. Trong một quan hệ không thể tồn tại hai bộ dữ liệu giống nhau ở tất cả các thuộc tính. Hình 1.2.3.3 dưới đây minh hoạ cho một bảng dữ liệu, các cột của bảng tương ứng với các thuộc tính hay các trường dữ liệu, các hàng của bảng tương ứng với các bản ghi hay các bộ dữ liệu.

(Khái niệm về thuộc tính sẽ được trình bày trong phần sau).



Hình 1.2.3.3 Bảng dữ liệu hay quan hệ

Đôi khi chúng ta có thể biểu diễn các bảng dưới dạng loại bỏ các dữ liệu và chỉ đưa ra tên bảng cùng các tên cột như sau:

AUTHOR (Au_id, Au_lname, Au_fname, Address, City, Country)

TITLE (Title_id, Title, type, Price, Pub_id)

PUBLISHER (Pub_id, Pub_name, City)

AUTHOR_TITLE (Au_id, Title_id)

Mỗi bảng hay quan hệ có một số tính chất sau:

- Các dữ liệu trong bảng là các giá trị đơn, được hiểu theo nghĩa là các giá trị không cần chia nhỏ hơn nữa mà vẫn đáp ứng được nhu cầu nghiệp vụ. Ví dụ: thiết kế cơ sở dữ liệu để lưu địa chỉ của các nhân viên

trong một công ty, nếu chúng ta không cần quan tâm đến việc phân loại các nhân viên theo thành phố, quận huyện, phường xã thì với một giá trị như "Nhà số 6 phường Kim Mã quận Ba Đình thành phố Hà Nội" được chấp nhận như một giá trị đơn. Mặt khác, trong trường hợp bài toán yêu cầu quản lý địa chỉ của nhân viên đến từng cấp thành phố, quận huyện, phường xã, số nhà thì rõ ràng địa chỉ trên không phải là giá trị đơn.

- Các giá trị trong cùng một cột dữ liệu phải có cùng một kiểu. Ví dụ nếu một cột được qui định là lưu ngày vào làm việc của các nhân viên trong công ty thì cột đó không thể lưu các giá trị khác kiểu ngày tháng như tên nhân viên, tuổi tác,..
- Mỗi một hàng trong bảng là duy nhất. Trong bảng không thể tồn tại hai hàng có cùng giá trị ở tất cả các cột. Ví dụ, nếu một bảng qui ước là lưu thông tin của các nhân viên trong công ty thì không thể có một nhân viên được lưu hai lần trong cùng bảng đó.
- Thứ tự liệt kê các cột trong bảng không quan trọng.
- Thứ tự liệt kê các hàng trong bảng không quan trọng
- Mỗi một cột dữ liệu trong bảng phải có một tên duy nhất

Như trên chúng ta thấy một quan hệ là một bảng hai chiều gồm các giá trị, đó là tập hợp các bộ của quan hệ tại một thời điểm nào đó. Tuy nhiên, quan hệ lại thường được cập nhật (bổ sung, loại bỏ, điều chỉnh các bộ), do đó quan hệ có tính chất là thay đổi theo thời gian.

Tuy thay đổi về giá trị thông qua việc cập nhật, song quan hệ vẫn phải giữ một số đặc điểm về ngữ nghĩa, hay nói một cách khác nó phải tuân thủ một lược đồ quan hệ nhất đinh, như qui ước số trường trong quan hệ, kiểu dữ liệu trong quan hệ,... Chúng ta có thể hình dung một lược đồ quan hệ là một khuôn mẫu, hay nói đúng hơn là một cái lọc, cho phép thiết lập một họ các quan hệ cùng ngữ nghĩa. Trong giáo trình này, đôi lúc chúng tôi sẽ dùng thuật ngữ quan hệ thay cho thuật ngữ lược đồ quan hệ nếu ngữ cảnh trình bày cho phép.

2.4. Thuộc tính và trường dữ liệu

a. Thuộc tính

Như trong khái niệm về kiểu thực thể chúng ta đã biết, một kiểu thực thể được mô tả bằng cách sử dụng một tập các thuộc tính. Tất cả các thực thể trong một kiểu thực thể đã được xác định đều có cùng số lượng các thuộc tính đó. Chúng ta cần lưu ý rằng, việc lựa chọn các thuộc tính phản ánh mức độ chi tiết mà chúng ta muốn biểu diễn thông tin về các thực thể của kiểu thực thể đó.

Ví dụ với kiểu thực thể EMPLOYEE, chúng ta có thể lựa chọn các thuộc tính như số thẻ bảo hiểm xã hội, tên, địa chỉ, ngày bắt đầu làm việc. Tuy nhiên trong một ngữ cảnh hay bài toán khác, chúng ta có thể quan tâm thêm đến các thuộc tính khác như giới tính, tuổi hay địa chỉ của nhân viên.

Do vậy chúng ta có thể xem thuộc tính như là một tính chất hay một đặc trưng của một kiểu thực thể mà người sử dụng quan tâm.

Mỗi một kiểu thực thể luôn có một tập các thuộc tính đi kèm. Dưới đây là ví dụ về hai kiểu thực thể và các thuộc tính đi kèm với chúng:

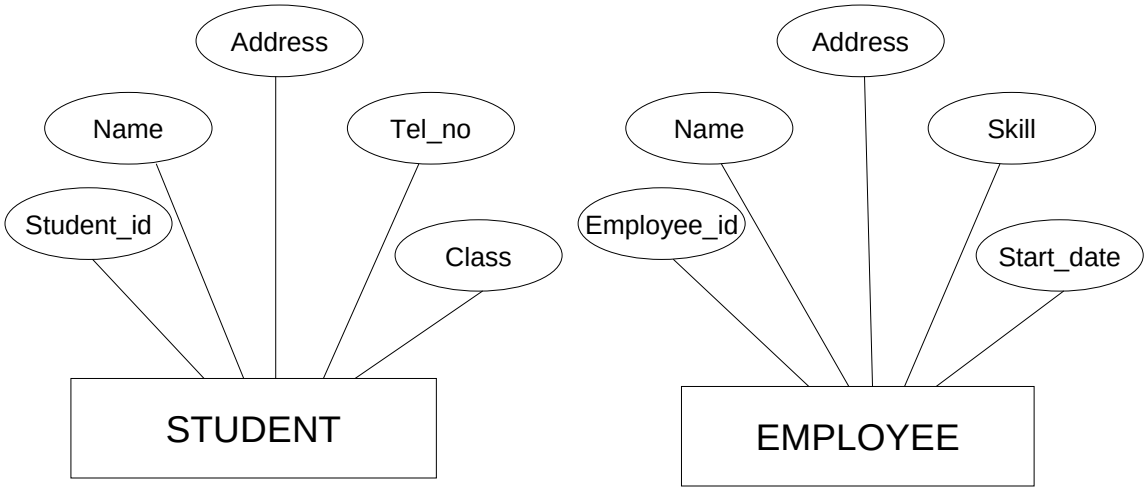
STUDENT: Student_id, Student_name, Address, Tel_no, Class

EMPLOYEE: Emp_id, Emp_name, Address, Skill, Start_date

Với mỗi thuộc tính trong kiểu thực thể, chúng ta phải xác định miền giá trị của thuộc tính đó. Ví dụ miền giá trị cho thuộc tính Emp_name của kiểu thực thể EMPLOYEE có thể là một xâu ký tự có độ dài không quá 30 ký tự. Một ví dụ khác, nếu công ty của chúng ta muốn xếp loại nhân viên theo các kỹ năng (tay nghề) từ 1 đến 7, thì miền giá trị cho thuộc tính Skill sẽ là các số nguyên có giá trị nằm trong khoảng từ 1 đến 7.

Thể hiện của thuộc tính chính là các giá trị. Ví dụ “Nguyên văn A.” là một giá trị cho thuộc tính tên nhân viên Emp_name.

Để đặt tên cho các thuộc tính, chúng ta dùng các danh từ viết liền với chữ cái đầu tiên được viết hoa, nếu một thuộc tính có tên gồm 2 từ thì chúng ta sử dụng dấu gạch dưới để nối 2 từ đó lại. Chúng ta sử dụng hình elip để biểu diễn cho một thuộc tính và một đường thẳng kết nối thuộc tính đó với kiểu thực thể mà nó mô tả, ví dụ như hình 1.2.4.1 sau là hai kiểu thực thể và các thuộc tính của chúng.



Hình 1.2.4.1 Thuộc tính và kiểu thực thể

b. Trường dữ liệu

Cũng như giữa bảng dữ liệu và kiểu thực thể, khi phân tích cơ sở dữ liệu, chúng ta sử dụng các thuộc tính để mô tả cho các kiểu thực thể, còn khi xây dựng cơ sở dữ liệu, chúng ta sử dụng các trường dữ liệu để xây dựng lên các quan hệ. Do vậy chúng ta có thể hiểu trường dữ liệu như là đơn vị nhỏ nhất có tên gọi của dữ liệu lưu trữ trong cơ sở dữ liệu. Trường dữ liệu còn được gọi là cột dữ liệu khi chúng ta sử dụng với khái niệm bảng dữ liệu.

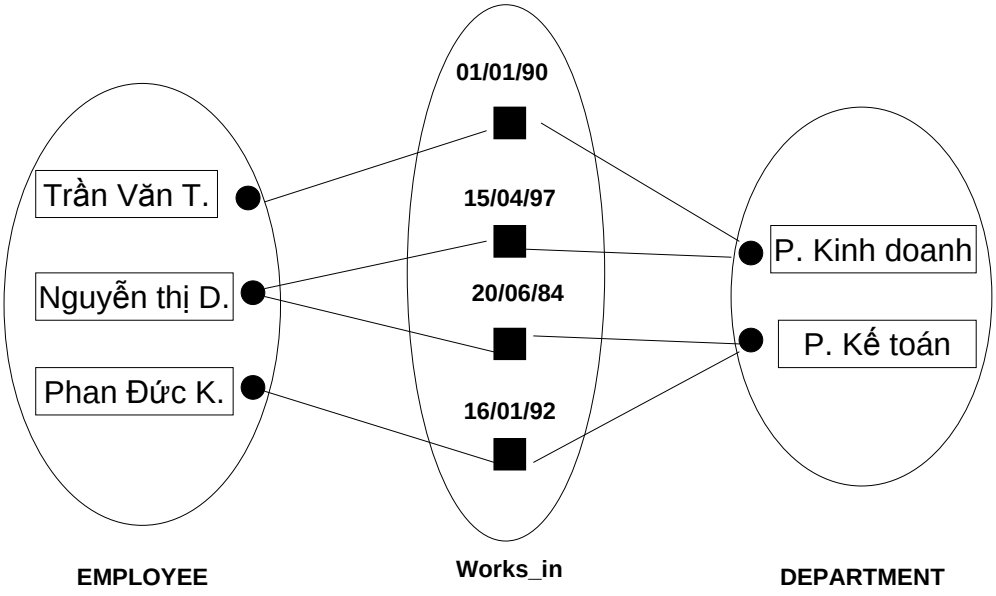
Qua đó chúng ta có thể ánh xạ các khái niệm trong khâu phân tích cơ sở dữ liệu và các khái niệm được sử dụng trong khâu thiết kế, xây dựng cơ sở dữ liệu:

- Từ kiểu thực thể chúng ta xây dựng được lên các quan hệ hay bảng dữ liệu.
- Từ các thuộc tính chúng ta xây dựng được lên các trường hay cột dữ liệu.

2.5. Liên kết giữa các thực thể

Một liên kết là một kết nối giữa các thể hiện của một hoặc nhiều kiểu thực thể. Ví dụ, chúng ta có một liên kết như nhân viên Trần Văn T. làm việc ở phòng kinh doanh; Sinh viên Nguyễn Văn A. tham gia các khoá học về C++, Visual Basic.

Chúng ta cũng có thể sử dụng các thuộc tính để mô tả cho liên kết. Các thuộc tính mô tả được sử dụng để lưu lại các thông tin về mối liên kết hơn là những thông tin về các thực thể tham gia vào liên kết đó. Ví dụ, chúng ta có thể ghi lại nhân viên Trần Văn T. làm việc ở phòng kinh doanh từ ngày 01/01/1990 thông qua một thuộc tính Start_date được đặt bên cạnh liên kết Works_in như trong hình 1.2.5.1 dưới đây:



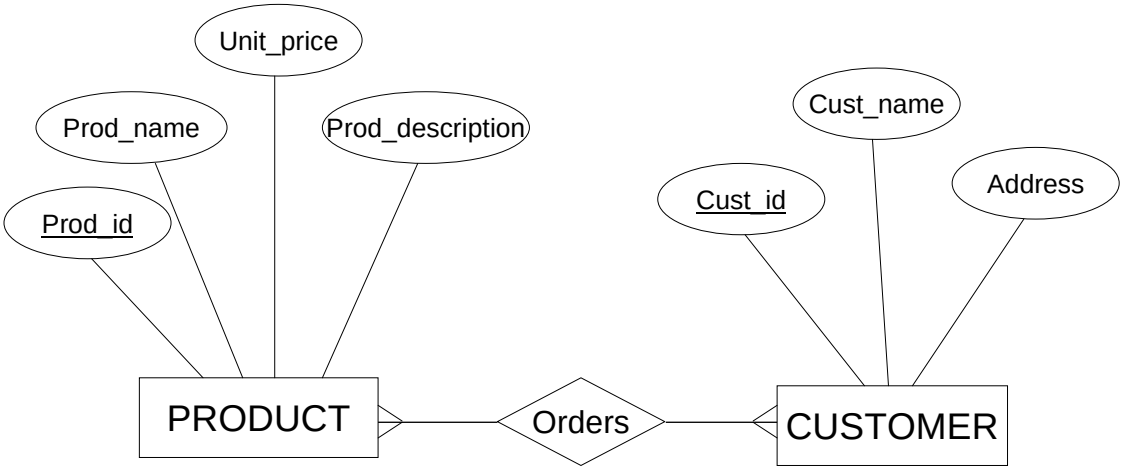
Hình 1.2.5.1 Liên kết giữa các thực thể

Một liên kết phải được xác định duy nhất bởi các thực thể tham gia liên kết mà không cần phải tham chiếu đến các thuộc tính mô tả. Như trong hình trên, mỗi một liên kết trong Works_in được xác định duy nhất bởi việc kết hợp thuộc tính đại diện Emp_name của kiểu thực thể EMPLOYEE và thuộc tính đại diện Dept_name của kiểu thực thể DEPARTMENT. Do vậy với mỗi cặp nhân viên – phòng ban trên chúng ta không thể có hơn một kết nối.

(Chú ý: trong ví dụ trên chúng ta giả thiết rằng giữa các thực thể của kiểu thực thể có thể phân biệt duy nhất với nhau thông qua các giá trị trong các thuộc tính Emp_name và Dept_name tương ứng với hai kiểu thực thể EMPLOYEE và DEPARTMENT).

Một nhóm các liên kết cùng kiểu hình thành lên một tập liên kết. Khi dùng với khái niệm kiểu thực thể, trong giáo trình này chúng tôi dùng thuật ngữ liên kết để mô tả cho sự kết nối giữa chúng thay vì sử dụng thuật ngữ tập liên kết.

Để đặt tên cho liên kết, chúng ta dùng các động từ viết liền với chữ cái đầu tiên được viết hoa, nếu một động từ có tên gồm 2 từ thì chúng ta sử dụng dấu gạch dưới để nối 2 từ đó lại. Chúng ta sử dụng hình thoi để biểu diễn cho một liên kết và các đường nối liên kết đó với các kiểu thực thể tham gia vào liên kết. Một ví dụ được cho như hình 1.2.5.2 dưới đây:



Hình 1.2.5.2 Liên kết giữa các kiểu thực thể

Hình 1.2.5.2 trên mô tả cho liên kết đặt mua "Orders" giữa hai kiểu thực thể là khách hàng CUSTOMER và mặt hàng PRODUCT với ý nghĩa: mỗi khách hàng có thể đặt mua một hoặc nhiều mặt hàng, ngược lại mỗi một mặt hàng cũng có thể được nhiều khách hàng khác nhau mua. Kiểu kết nối giữa liên kết "Orders" trên với hai kiểu thực thể còn lại sẽ được trình bày cụ thể trong chủ đề hai của giáo trình.

2.6. Khái niệm toàn vẹn dữ liệu

Như đã trình bày trong phần đầu, một trong những ưu điểm của cách tiếp cận cơ sở dữ liệu đó là dữ liệu khi được đưa vào cơ sở dữ liệu sẽ phải tuân theo một số qui định nào đó. Những qui định này được xem là những ràng buộc trên dữ liệu và nó đảm bảo tính toàn vẹn của dữ liệu. Tính toàn vẹn dữ liệu giúp cho người sử dụng có thể tin cậy ít nhất về mặt nguyên tắc đối với các dữ liệu trong cơ sở dữ liệu.

Chúng ta cũng cần lưu ý rằng toàn vẹn dữ liệu có thể đảm bảo tính hợp lệ của dữ liệu khi được lưu trong cơ sở dữ liệu,

nhưng lại khó có thể kiểm tra mỗi một giá trị như vậy về tính hợp lý. Ví dụ, khi nhập giá trị giờ làm việc của một nhân viên trong tuần, không có cách nào có thể phát hiện sự kiện là một giá trị vào bằng 35 là sai, trong khi đó thực sự phải là giá trị 38. Tuy nhiên, nếu một giá trị là 350 đối với giờ làm việc rõ ràng là sai và hệ thống sẽ không chấp nhận giá trị đó, bởi một tuần không thể có 350 giờ.

Giữ gìn tính toàn vẹn của cơ sở dữ liệu có thể xem là sự bảo vệ dữ liệu trong cơ sở dữ liệu chống lại sự cập nhật hay phá hủy không có căn cứ. Chú ý, chúng ta cũng phải phân biệt tính toàn vẹn dữ liệu với tính an toàn dữ liệu dù rằng hai khái niệm này quan hệ rất mật thiết với nhau và thực sự có thể dùng chung một cơ chế để thực hiện bảo vệ cả hai ở một mức độ nào đó.

Nói chung mỗi một bảng dữ liệu trong cơ sở dữ liệu quan hệ đều có một tập các ràng buộc toàn vẹn đi kèm với nó. Các ràng buộc toàn vẹn này cũng được lưu trữ trong từ điển dữ liệu.

Chúng ta sẽ dành cho việc tìm hiểu sâu hơn tính toàn vẹn của dữ liệu trong các chủ đề tiếp theo.

3. Ví dụ về một cơ sở dữ liệu

Trong giáo trình này, chúng tôi cố gắng đưa ra các ví dụ minh hoạ xoay quanh một bài toán xây dựng cơ sở dữ liệu cho một công ty chuyên kinh doanh đồ gỗ, chúng ta tạm đặt tên cho công ty này là Pine Furniture:

Hoạt động chung của công ty này như sau: công ty mua các nguyên vật liệu (hay bán thành phẩm) từ một số nhà cung cấp khác nhau, sau khi nhận được các nguyên vật liệu này,

công ty sẽ tiến hành sản xuất để tạo nên các sản phẩm theo các đơn đặt hàng, và gửi đến các khách hàng đã đăng ký mua các sản phẩm đó thông qua các đơn đặt hàng. Mỗi một khách hàng có thể đặt mua một hoặc nhiều loại sản phẩm khác nhau trong cùng một đơn đặt hàng.

Một số khái niệm:

- Mỗi một khách hàng CUSTOMER của công ty có thể là một cá nhân hay một tổ chức nào đó có nhu cầu đặt mua các sản phẩm của công ty. Ví dụ: công ty kinh doanh thiết bị tin học AITC muốn đặt mua 15 chiếc bàn mã hiệu MH0415 của công ty.
- Sản phẩm PRODUCT là một mặt hàng bán ra cụ thể của công ty mà khách hàng có thể đặt mua thông qua các đơn đặt hàng. Ví dụ: sản phẩm bàn máy tính mã hiệu MH0415 có kích thước chiều ngang là 2m, chiều dọc là 1.2m, chiều cao là 0.9m và được làm từ gỗ thông.
- Đơn đặt hàng ORDER là một giao dịch giữa công ty và khách hàng. Mỗi một đơn đặt hàng đều có một số hiệu duy nhất, trong mỗi đơn đặt hàng sẽ phải thể hiện được các thông tin sau: khách hàng nào đã đặt, họ đặt mua những sản phẩm nào, số lượng tương ứng cho mỗi loại sản phẩm đó và ngày đặt của đơn hàng. Ví dụ: Khách hàng là công ty A đã đặt mua các sản phẩm: giá sách loại 0600 ba chiếc; bàn làm việc loại C700 hai chiếc vào ngày 1/7/2003, số hiệu của hoá đơn đó là HD1845.
- Các nguyên vật liệu bán thành phẩm ITEM là các thành phần được sử dụng để tạo thành các sản phẩm.

Các nguyên vật liệu này sẽ do các nhà cung cấp SUPPLIER gửi đến. Ví dụ, thanh gỗ loại G200 có thể được mua từ hai nhà cung cấp là North Wood và Star Wood.

- Nhà cung cấp SUPPLIER là các công ty chuyên cung cấp các vật liệu bán thành phẩm ITEM làm nguyên liệu sản xuất cho công ty.

Một số quy tắc nghiệp vụ:

- Mỗi một nhà cung cấp SUPPLIER có thể cung cấp một hoặc nhiều nguyên liệu bán thành phẩm ITEM. Và mỗi một nguyên vật liệu ITEM phải được cung cấp ít nhất từ một nhà cung cấp.
- Mỗi một nguyên vật liệu ITEM phải được sử dụng trong ít nhất một sản phẩm PRODUCT, và tất nhiên nó thường được sử dụng trong nhiều sản phẩm.
- Một nhà cung cấp SUPPLIER có thể có nhiều giao dịch với công ty. Ngược lại, mỗi một giao dịch mua hàng chỉ được gửi đến chính xác từ một nhà cung cấp.
- Trong mỗi giao dịch với nhà cung cấp phải bao gồm một hoặc nhiều nguyên vật liệu ITEM. Và mỗi một nguyên vật liệu có thể xuất hiện trong nhiều giao dịch khác nhau.
- Một khách hàng CUSTOMER có thể đăng ký mua các sản phẩm thông qua các đơn đặt hàng ORDER. Tuy nhiên mỗi một đơn đặt hàng phải do một khách hàng duy nhất đặt.

- Trong một đơn đặt hàng ORDER, khách hàng có thể đặt mua một hoặc nhiều sản phẩm PRODUCT tạo thành những dòng đơn hàng. Và mỗi một sản phẩm có thể được đặt mua trong một hoặc nhiều đơn hàng ORDER, thậm chí có những sản phẩm không được đặt mua ở bất kỳ một đơn đặt hàng nào.

Bài toán này sẽ được sử dụng cho một số ví dụ trong chủ đề hai của giáo trình.

Trước khi chuyển sang chủ đề hai của giáo trình, trong phần sau chúng ta hãy xem xét qua một số mô hình dữ liệu.

4. Các mô hình dữ liệu

4.1. Khái quát chung

Có thể nói mô hình dữ liệu là cách biểu diễn các cấu trúc dữ liệu cho một cơ sở dữ liệu dưới dạng các khái niệm. Các cấu trúc dữ liệu bao gồm các đối tượng dữ liệu, các kết nối giữa chúng và các ràng buộc trên các đối tượng dữ liệu đó.

Bản thân thuật ngữ "mô hình dữ liệu" đã cho chúng ta thấy mô hình chỉ tập trung vào việc mô tả các dữ liệu và chúng sẽ được tổ chức như thế nào chứ không tập trung vào các thao tác sẽ được thực hiện trên các dữ liệu đó (được thực hiện bởi mô hình chức năng).

Một mô hình dữ liệu không phụ thuộc vào các ràng buộc về cả phần cứng và phần mềm. Thay vì biểu diễn dữ liệu như là một cơ sở dữ liệu, mô hình dữ liệu chủ yếu tập trung vào việc biểu diễn các dữ liệu sao cho người sử dụng có thể "nhìn thấy" được chúng trong thế giới thực.

Mô hình dữ liệu đóng vai trò như là một cầu nối giữa các khái niệm được sử dụng để mô tả cho các sự kiện và các quá trình trong thế giới thực với cách biểu diễn các khái niệm này trong một cơ sở dữ liệu.

Có hai phương pháp luận chính được sử dụng để tạo nên một mô hình dữ liệu: đó là phương pháp dựa trên cách tiếp cận thực thể - liên kết và phương pháp mô hình hoá đối tượng. Trong giáo trình này chúng ta chỉ tập trung vào cách tiếp cận thứ nhất, cách tiếp cận thực thể - liên kết bởi cơ sở dữ liệu được tập trung trình bày trong giáo trình là cơ sở dữ liệu quan hệ.

Trong quá trình phân tích, thiết kế và xây dựng cơ sở dữ liệu, có thể nói quá trình xây dựng mô hình dữ liệu chiếm nhiều thời gian và công sức nhất. Bởi mục đích của nó là đảm bảo chắc chắn rằng tất cả các đối tượng dữ liệu trong bài toán phải được trình bày đầy đủ và chính xác. Chúng ta có thể sử dụng các ngôn ngữ một cách rất tự nhiên cũng như các ký hiệu minh hoạ dễ hiểu để mô tả cho các đối tượng dữ liệu, do đó người sử dụng cuối cùng có thể dễ dàng hiểu và cho ý kiến về mô hình. Mặt khác người thiết kế cơ sở dữ liệu cũng có thể sử dụng mô hình dữ liệu như một bản thiết kế chi tiết để có thể tiến hành thiết kế một cơ sở dữ liệu mức vật lý.

Dưới đây chúng ta sẽ xem xét một số mô hình dữ liệu, trong đó đáng lưu ý nhất là mô hình dữ liệu quan hệ. Mặc dù được xây dựng khá lâu, nhưng đến nay mô hình dữ liệu quan hệ vẫn được xem như là một mô hình quan trọng nhất trong tất cả các hệ quản trị cơ sở dữ liệu phổ biến.

Ngoài mô hình dữ liệu quan hệ, chúng ta cũng sẽ lướt qua một số mô hình dữ liệu khác như mô hình dữ liệu mạng,

mô hình dữ liệu phân cấp là những "tiền thân" của mô hình dữ liệu quan hệ.

4.2. Mô hình dữ liệu quan hệ

Mô hình dữ liệu quan hệ được Codd đề xuất năm 1970, được xem là một mô hình có cơ sở toán học vững chắc nhất. Tại thời điểm đó, hầu hết các hệ cơ sở dữ liệu đều được xây dựng dựa trên hai mô hình trước đó là mô hình dữ liệu phân cấp và mô hình dữ liệu mạng. Sau khi mô hình dữ liệu quan hệ ra đời, nó đã tạo ra một cuộc cách mạng mới trong lĩnh vực cơ sở dữ liệu và nhanh chóng thay thế các mô hình dữ liệu trước đó.

Ngày nay, mô hình dữ liệu quan hệ vẫn tiếp tục đóng một vai trò thống trị và là nền tảng cho hàng loạt các hệ quản trị cơ sở dữ liệu nổi tiếng và phổ biến như Oracle, DB2, MS SQLServer, Access, ... Các hệ cơ sở dữ liệu quan hệ này có mặt ở khắp mọi nơi và đã trở thành một trong những ngành công nghiệp có doanh thu cao nhất.

Mô hình dữ liệu quan hệ tương đối đơn giản và dễ hiểu. Một cơ sở dữ liệu quan hệ là một tập của một hoặc nhiều quan hệ, trong đó mỗi một quan hệ là một bảng với các hàng và các cột. Cách biểu diễn đơn giản theo bảng gồm các hàng và cột này khiến cho người mới làm quen với cơ sở dữ liệu cũng có thể hiểu được nội dung của cơ sở dữ liệu. Hơn thế nữa, mô hình dữ liệu quan hệ đã cung cấp các khái niệm chặt chẽ được hình thức hoá cao, cho phép áp dụng các công cụ toán học, các thuật toán tối trên dữ liệu.

Mô hình dữ liệu quan hệ được trừu tượng hoá cao nhưng nó chỉ dừng ở mức logic, nghĩa là độc lập với mức vật lý, với

sự cài đặt, với các thiết bị lưu trữ. Nhờ đó, mô hình dữ liệu quan hệ đảm bảo được tính độc lập giữa dữ liệu và chương trình ứng dụng. Ngoài ra, do sự biểu diễn dữ liệu thống nhất, mô hình dữ liệu quan hệ có thể cho phép sử dụng các ngôn ngữ thao tác dữ liệu ở mức cao, dễ sử dụng và có thể chuẩn hoá. Đây cũng chính là một trong những ưu điểm lớn nhất của một mô hình dữ liệu quan hệ so với các mô hình dữ liệu trước đây

4.3. Một số mô hình dữ liệu khác

a. Mô hình dữ liệu mạng và mô hình dữ liệu phân cấp

Trong mô hình mạng và mô hình phân cấp, các thực thể được biểu diễn dưới dạng các bản ghi và các liên kết được biểu diễn bởi các móc nối hay con trỏ. Về cơ bản, các móc nối có khả năng biểu diễn các liên kết giữa một thực thể và nhiều thực thể của các kiểu thực thể liên quan. Tuy nhiên, với cách nhìn dữ liệu theo kiểu phân cấp bởi cấu trúc cây, các bản ghi dữ liệu chỉ có ý nghĩa khi nó được xem xét trong một ngữ cảnh cụ thể, nói cách khác các bản ghi phụ thuộc không thể tồn tại trong cấu trúc cây nếu không có phần tử bậc trên nó. Tuy nhiên, với mô hình phân cấp, chúng ta có một cách rất tự nhiên để mô hình hoá những cấu trúc phân cấp thực sự của thế giới thực.

Mô hình dữ liệu mạng có cấu trúc tổng quát hơn so với mô hình dữ liệu phân cấp bởi một bản ghi dữ liệu đã cho có thể có một số bất kỳ các phần tử bậc trên trực tiếp cũng như một số bất kỳ các phần tử phụ thuộc trực tiếp. Cấu trúc nội tại của mô hình mạng phức tạp hơn so với mô hình phân

cấp nhưng mô hình mạng lại đối xứng hơn mô hình phân cấp. Điểm bất lợi chủ yếu của mô hình này đó là sự quá phức tạp trong bản thân mô hình và trong các thao tác xử lý dữ liệu. Do vậy, những người sử dụng và người viết chương trình sẽ phải hao tốn nhiều thời gian vào việc giải quyết những vấn đề nảy sinh trong quá trình phát triển và bảo trì chương trình.

b. Mô hình dữ liệu hướng đối tượng

So với các mô hình dữ liệu khác, mô hình dữ liệu hướng đối tượng là một mô hình cho phép biểu diễn các đối tượng khá tự nhiên và sát với thực tế hơn cả. Các đối tượng phức tạp và các liên kết giữa chúng được phản ánh đầy đủ và giàu ngữ nghĩa. Do có khả năng biểu diễn dữ liệu khá tốt, nên mô hình dữ liệu hướng đối tượng cũng hỗ trợ các ngôn ngữ thao tác dữ liệu hướng đối tượng mạnh, cho phép người sử dụng dễ dàng biểu diễn cả các truy vấn khá phức tạp. Tuy nhiên, cho đến nay, chưa có một sự thống nhất và chuẩn hoá đối với các khái niệm của mô hình hướng đối tượng cũng như chưa có một cơ sở toán học tốt hình thức hoá ở mức cao, chặt chẽ đối với mô hình này, do vậy nó vẫn được tiếp tục nghiên cứu và phát triển trong tương lại.

Ngày nay, người ta đã nghĩ đến một mô hình dữ liệu kết hợp giữa mô hình dữ liệu quan hệ và mô hình dữ liệu hướng đối tượng gọi tắt là mô hình dữ liệu quan hệ - hướng đối tượng nhằm tận dụng được các ưu điểm của mỗi mô hình trên. Nhưng đây thực sự là một vấn đề khá khó khăn nên mô hình này vẫn đang trong giai đoạn nghiên cứu và thử nghiệm.

5. Kho dữ liệu

Như chúng ta đã biết, cơ sở dữ liệu đã được sử dụng từ những năm 60 và chúng ngày càng chứng tỏ là những thành phần không thể thiếu trong các hoạt động của doanh nghiệp. Qua năm tháng, lượng dữ liệu tác nghiệp này ngày càng nhiều và người ta đã tìm cách khai thác chúng một cách hiệu quả hơn, chủ yếu dựa vào tính lịch sử của chúng. Một xu hướng mới đã xuất hiện và phát triển khá nhanh trong các hệ cơ sở dữ liệu đó là các cơ sở dữ liệu hỗ trợ việc phân tích và khai thác dữ liệu, người ta gọi đó là những kho dữ liệu, chúng chủ yếu được phục vụ cho các ứng dụng hỗ trợ ra quyết định DSS (Decision Support System). Dưới đây là những đặc điểm cơ bản của kho dữ liệu:

- Tính tích hợp: dữ liệu tập hợp trong kho được thu thập từ nhiều nguồn và được tích hợp với nhau tạo thành một thể thống nhất.
- Tính hướng chủ đề: Các dữ liệu được thu thập, trích lọc và làm sạch để đưa vào kho theo những chủ đề nhất định.
- Gắn thời gian: Mỗi hàng thông tin trong kho dữ liệu đều chứa một yếu tố thời gian như một phần của khoá chính để đảm bảo tính đơn nhất của mỗi hàng và cung cấp một đặc trưng về thời gian cho dữ liệu.
- Tính lịch sử: Các thông tin trong kho dữ liệu được tập trung theo thời gian và thể hiện một khung nhìn về một lĩnh vực chủ đề trong một giai đoạn dài.
- Tính chỉ đọc: Dữ liệu trong kho là dữ liệu chỉ đọc và không thể bị sửa đổi.

- Tính bất biến: Dữ liệu được lưu trữ lâu dài trong kho, mặc dù những dữ liệu mới vẫn tiếp tục được thu thập nhưng dữ liệu cũ trong kho vẫn không bị xoá, bị sửa đổi, điều đó cho phép cung cấp thông tin về một khoảng thời gian dài, đủ số liệu cần thiết cho các mô hình nghiệp vụ phân tích, dự báo.
- Bao gồm cả dữ liệu tổng hợp và chi tiết: Dữ liệu chi tiết là thông tin mức thấp nhất được lưu trữ trong kho. Dữ liệu tổng hợp được tích hợp lại qua nhiều giai đoạn khác nhau.

6. Kết luận

Trong chủ đề này, chúng ta đã làm quen với các khái niệm cơ bản liên quan đến việc phân tích, thiết kế các hệ cơ sở dữ liệu.

Các khái niệm cần phải nắm được trong chủ đề này đó là: các khái niệm về cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu, các khái niệm về dữ liệu, siêu dữ liệu, thông tin, từ điển dữ liệu, các khái niệm về thực thể, kiểu thực thể, thuộc tính, trường dữ liệu, liên kết, bảng dữ liệu, quan hệ, lược đồ quan hệ. Những khái niệm này được trình bày lần lượt, dễ hiểu và được minh hoạ bằng các ví dụ kèm theo. Cũng cần lưu ý rằng các khái niệm này không độc lập hoàn toàn với nhau mà giữa chúng có một sự liên quan nhất định.

Ngoài ra một số mô hình dữ liệu cũng đã được đề cập, trong đó quan trọng nhất là mô hình dữ liệu quan hệ, và việc phân tích, thiết kế các cơ sở dữ liệu được trình bày

trong giáo trình sẽ là những cơ sở dữ liệu quan hệ được xây dựng dựa trên chính mô hình dữ liệu này.

7. Bài tập

1. Hãy cho biết thế nào là một cơ sở dữ liệu

2. Hãy phân biệt giữa dữ liệu, thông tin và siêu dữ liệu

3. Thực thể là gì? Kiểu thực thể là gì? Lấy ví dụ trong thực tế cho hai khái niệm này

4. Quan hệ là gì? Quan hệ (hay bảng dữ liệu) trong khái niệm cơ sở dữ liệu khác với các bảng thông thường khác ở điểm gì?

5. Thuộc tính là gì? Hãy lấy ví dụ về một kiểu thực thể và các thuộc tính mô tả cho kiểu thực thể đó.

6. Lựa chọn một bài toán trên thực tế gần với nghiệp vụ hàng ngày của bạn, liệt kê các kiểu thực thể cùng các thuộc tính cần quản lý tương ứng trong bài toán đó.

Chủ đề II: Phương pháp phân tích, thiết kế và xây dựng các hệ thống CSDL

Mục tiêu:

Mục tiêu chính của chủ đề này là giúp người đọc hình dung, nắm bắt và làm chủ được các bước cần phải thực hiện khi phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu quan hệ. Các bước đó bao gồm:

- Phân tích yêu cầu
- Thiết kế cơ sở dữ liệu mức khái niệm
- Thiết kế cơ sở dữ liệu mức logic
- Tinh chỉnh sơ đồ dữ liệu và các dạng chuẩn
- Thiết kế cơ sở dữ liệu mức vật lý
- Thiết kế an toàn bảo mật cho cơ sở dữ liệu

Trong mỗi bước, các bạn sẽ được hướng dẫn cụ thể các tình huống cần phải xử lý. Các kinh nghiệm thực tế sẽ được trình bày thông qua các ví dụ minh hoạ.

1. Khái quát chung

Phân tích, thiết kế và xây dựng các hệ cơ sở dữ liệu là một trong những bước quan trọng nhất trong quá trình phân tích, thiết kế, xây dựng các hệ thống thông tin. Các cơ sở dữ liệu được xem là nền tảng của hệ thống thông tin, do vậy chúng ta có thể nói rằng một hệ thống thông tin chỉ mang lại hiệu quả khi cơ sở dữ liệu được thiết kế tốt.

Việc nắm vững các bước trong quá trình phân tích, thiết kế và xây dựng các hệ thống cơ sở dữ liệu là một đòi hỏi bắt buộc đối với mỗi một người phân tích, thiết kế các hệ thống thông tin. Ngoài ra đối với những người quản trị hệ thống, người phát triển chương trình, việc hiểu rõ các bước này cũng là điều kiện cần trong công việc của họ. Bởi chỉ có nắm vững được các công đoạn phân tích, xây dựng cơ sở dữ liệu này, họ mới có thể chủ động và thực hiện tốt công việc của mình.

Trong quá trình phân tích cơ sở dữ liệu, chúng ta sẽ phải sử dụng đến một mô hình, đó là mô hình thực thể - liên kết thông qua việc biểu diễn các đối tượng trong bài toán bằng những sơ đồ thực thể - liên kết.

Trong quá trình thiết kế cơ sở dữ liệu, chúng ta lại phải sử dụng đến các mô hình dữ liệu, mà ở đây chính là mô hình dữ liệu quan hệ được thể hiện qua việc xây dựng các sơ đồ dữ liệu mức logic với các quan hệ.

2. Phân tích yêu cầu

Bước đầu tiên trong quá trình phân tích, thiết kế một hệ thống cơ sở dữ liệu luôn luôn là bước phân tích yêu cầu.

Trong bước này chúng ta cần tìm hiểu dữ liệu gì sẽ phải được lưu trữ trong cơ sở dữ liệu, ứng dụng nào sẽ được xây dựng dựa trên cơ sở dữ liệu đó, và các thao tác gì sẽ được thực hiện trên những dữ liệu đó. Nói một cách tổng quát, chúng ta phải tìm hiểu người sử dụng họ muốn gì từ hệ thống cơ sở dữ liệu mà chúng ta sẽ xây dựng.

Quá trình phân tích yêu cầu của người sử dụng thường là quá trình thảo luận với khách hàng, nghiên cứu môi trường

tác nghiệp hiện thời và các mong muốn của khách hàng đối với hệ thống cơ sở dữ liệu sẽ được xây dựng.

Để có thể nắm được chi tiết một lĩnh vực dự định sẽ được xây dựng, chúng ta cần phải tìm hiểu hiện trạng, phân tích các yêu cầu của các đối tượng sử dụng từ những người thừa hành, đến những người quản lý và lãnh đạo.

Nếu những người thừa hành có thể cho chúng ta biết các thông tin chi tiết về một công việc cụ thể, người quản lý có thể cho chúng ta biết một chức năng, một lĩnh vực trong hệ thống thì người lãnh đạo lại có thể cho chúng ta một cách tổng thể cùng các mục tiêu trung và dài hạn của toàn bộ hệ thống.

Kỹ thuật và công cụ thường được sử dụng nhiều nhất trong giai đoạn này là phỏng vấn. Bổ sung cho kỹ thuật cơ bản này có thể là các phiếu điều tra, các bảng câu hỏi gửi tới các đối tượng mà chúng ta muốn nghiên cứu, tìm hiểu. Dù có khá nhiều kỹ thuật có thể áp dụng để tiến hành công việc phân tích yêu cầu, song để có được những hiểu biết cơ bản thì phỏng vấn vẫn là công cụ và kỹ thuật hữu hiệu nhất.

Mục đích chính của khâu phân tích yêu cầu, đó là:

- Xác định các yêu cầu dữ liệu của cơ sở dữ liệu và trình bày chúng dưới dạng các đối tượng.
- Phân loại và mô tả các thông tin về các đối tượng đó.
- Xác định và phân loại các mối liên kết giữa các đối tượng
- Xác định các giao dịch sẽ được thực hiện trên các đối tượng dữ liệu đó.

- Xác định các qui tắc nghiệp vụ (luật) đảm bảo tính toàn vẹn của dữ liệu.

Khâu phân tích yêu cầu thường được tiến hành song song với khâu mô hình hoá dữ liệu. Khi thu thập được các thông tin, chúng ta cũng nên tiến hành xác định và phân loại các thông tin đó, chúng sẽ là các kiểu thực thể, các thuộc tính, hay là các liên kết. Chúng ta hãy đặt tên cho chúng và cố gắng sử dụng các từ ngữ thật gần gũi với người sử dụng. Những đối tượng này sau đó sẽ được mô hình hoá và phân tích bằng cách sử dụng sơ đồ thực thể - liên kết, bước phân tích thiết kế cơ sở dữ liệu mức khái niệm.

Chúng ta cần phải lưu ý rằng, sơ đồ thực thể - liên kết cũng phải được xem xét bởi người sử dụng, họ có thể sẽ phát hiện ra những chỗ mà chúng ta thể hiện còn thiếu hay chưa đúng, khi đó chúng ta cần phải có kế hoạch để tiếp tục tiến hành khảo sát, phân tích bổ sung.

Có ba điểm chính cần phải ghi nhớ trong bước phân tích yêu cầu của người sử dụng, đó là:

- Khi trao đổi với người sử dụng, chúng ta phải sử dụng các thuật ngữ trong thế giới thực. Người sử dụng thường không có các khái niệm thế nào là một kiểu thực thể, thế nào là một thuộc tính, thế nào là liên kết nhưng họ lại có thể hiểu về một người, một sự kiện, mối quan hệ giữa chúng và về các bước thực hiện để giải quyết một vấn đề mà họ tiếp xúc hàng ngày.
- Chúng ta hãy bỏ thời gian để nghiên cứu về mô hình cùng các hoạt động của doanh nghiệp. Khi hiểu các

qui trình nghiệp vụ của doanh nghiệp thì chúng ta cũng sẽ dễ dàng hơn trong việc mô hình hoá nó.

- Những người sử dụng khác nhau thường có cách suy xét và nhìn nhận dữ liệu theo các góc độ khác nhau, tương ứng với chức năng và nhiệm vụ của họ. Do vậy, việc phỏng vấn đầy đủ các đối tượng sử dụng là điều hết sức quan trọng.

2.1. Xác định dữ liệu cần lưu trữ

Đối với các hệ thống cơ sở dữ liệu, điểm cốt yếu khi tìm hiểu hệ thống đó là phải xác định được các dữ liệu nào sẽ được lưu trữ trong cơ sở dữ liệu. Nếu chúng ta xác định sai hoặc thiếu sẽ dẫn tới những thay đổi khôn lường sau khi cơ sở dữ liệu đi vào hoạt động, và khi đó vấn đề sửa đổi lại thiết kế sẽ tốn khá nhiều thời gian và công sức. Do đó, chúng ta cần phải xác định càng chi tiết đối với dữ liệu cần lưu trữ bao nhiêu sẽ càng đảm bảo tính thành công của cơ sở dữ liệu sẽ được xây dựng bấy nhiêu.

Đối với mỗi dữ liệu thu thập được chúng ta cần thể hiện đầy đủ các mục sau:

- Tên dữ liệu: chọn tên phù hợp với ngôn ngữ thường dùng của bài toán, sau đó gán cho nó một tên tượng trưng tương ứng.
- Định nghĩa: chúng ta hãy cố gắng định nghĩa sao cho người sử dụng có thể hiểu được.
- Kiểu dữ liệu: qua cấu trúc dữ liệu, chúng ta sẽ phải xác định chúng thuộc kiểu nào trong các kiểu sau: Ký

tự, số, ngày tháng, logic,... đồng thời chỉ rõ kích cỡ cần thiết hay là độ dài của dữ liệu.

- Loại dữ liệu: nó là loại sơ cấp (không thể chia nhỏ được nữa), loại kết nối (được gộp lại từ việc ghép nhiều dữ liệu với nhau), loại tính toán (được suy ra từ một số dữ liệu khác).
- Định lượng: chúng ta phải ước tính được số các giá trị khác nhau mà dữ liệu có thể nhận được.
- Ghi chú thêm: cho phép chúng ta dùng hình thức diễn đạt tự do để có thể nêu thêm các dữ kiện bổ sung cho dữ liệu.

Ví dụ về một phiếu mô tả dữ liệu:

- Tên: tên nhà cung cấp (Supplier_name)
- Định nghĩa: dùng để đặt tên, thường tóm tắt, cho phép xác định nhà cung cấp
- Cấu trúc: Kiểu ký tự C(30)
- Loại: sơ cấp
- Định lượng: khoảng 50
- Ví dụ: công ty SUNIMEX, công ty IMEXCO
- Ghi chú: có thể chỉ sử dụng 15 ký tự.

2.2. Xác định ứng dụng sẽ được cài đặt trên cơ sở dữ liệu

Như phần đầu của chủ đề đã trình bày, việc xây dựng cơ sở dữ liệu là nền tảng cho một hệ thống thông tin. Ngoài cơ sở dữ liệu trong hệ thống còn có một thành phần cũng khá

quan trọng, đó là các ứng dụng. Giữa hai thành phần này của một hệ thống thông tin có sự quan hệ mật thiết với nhau, bởi các ứng dụng phải được xây dựng và cài đặt trên một cơ sở dữ liệu cụ thể. Do vậy, trong quá trình phân tích yêu cầu, chúng ta cũng phải xác định được những ứng dụng nào sẽ được xây dựng, những đối tượng người dùng nào sẽ sử dụng, để từ đó có được một cái nhìn tổng quan hơn về cơ sở dữ liệu cần xây dựng.

2.3. Xác định các thao tác thường xuyên được thực hiện

Việc xác định các thao tác thường xuyên được thực hiện trên dữ liệu đóng một vai trò quan trọng, yếu tố này đôi khi lại quyết định tính thành công của một cơ sở dữ liệu. Một cơ sở dữ liệu được thiết kế chính xác về mặt nghiệp vụ nhưng lại không quan tâm đến các thao tác thường xuyên được thực hiện thì chưa thể được đánh giá là một thiết kế đầy đủ.

Trên thực tế, khi lượng dữ liệu tăng lên thì thời gian thực hiện các thao tác trên các dữ liệu cũng tăng lên. Do vậy, nếu trong khâu khảo sát, phân tích yêu cầu, chúng ta không xác định được những thao tác thường xuyên được thực hiện thì không thể đưa ra được những thiết kế hiệu quả, không thể tạo ra được những chỉ mục phù hợp cũng như cách phân chia dữ liệu hợp lý.

Bởi vậy, trong quá trình phân tích tìm hiểu yêu cầu, chúng ta cũng cần phải khảo sát kỹ đến vấn đề này, mặc dù không được mô tả ngay trong các sơ đồ thực thể - liên kết nhưng các thông tin thu thập được sẽ lại đóng vai trò quan trọng

trong quá trình tinh chỉnh sơ đồ cơ sở dữ liệu mức logic cũng như thiết kế cơ sở dữ liệu mức vật lý.

2.4. Một số kỹ thuật phân tích yêu cầu

a. Kỹ thuật phỏng vấn

Khi tìm hiểu yêu cầu của cơ sở dữ liệu dự định xây dựng, một trong những kỹ thuật phân tích yêu cầu hữu hiệu nhất đó là kỹ thuật phỏng vấn. Trong kỹ thuật phỏng vấn, chúng ta cũng phải phân tích để lựa chọn đúng các đối tượng cần phỏng vấn.

Phỏng vấn nên được tiến hành tuần tự theo cấu trúc phân cấp: đầu tiên là ban lãnh đạo, tiếp đến là các bộ phận, phòng ban. Trong quá trình phỏng vấn chúng ta cố gắng càng nắm được nhiều hồ sơ (đặc biệt là các quy trình nghiệp vụ), nhiều thống kê báo cáo càng tốt.

Kết quả của phỏng vấn sẽ là: các mục tiêu chính mà cơ sở dữ liệu dự định xây dựng phải được đáp ứng.

Phỏng vấn ban lãnh đạo:

Mục tiêu: Tìm hiểu được bức tranh tổng thể của doanh nghiệp trong từng lĩnh vực, từng bộ phận.

Phỏng vấn các vị trí làm việc:

Mục tiêu: Tiếp thu được tất cả các thao tác nghiệp vụ cùng những thông tin cần phải quản lý thuộc các lĩnh vực đã phỏng vấn Ban lãnh đạo.

Chúng ta cũng cần phải phỏng vấn những người phụ trách từng vị trí làm việc có liên quan đến lĩnh vực quan tâm. Ví dụ, với một bài toán quản lý sản phẩm: chúng ta cần phải phỏng vấn các vị trí sau: người phụ trách giao dịch với nhà

cung cấp, người phụ trách giao dịch với khách hàng, người quản lý các sản phẩm làm ra, …

Khi tiến hành phỏng vấn, chúng ta cố gắng tạo ra các cuộc phỏng vấn gồm những người đại diện cho mỗi phòng ban nghiệp vụ. Cố gắng ghi lại đầy đủ tất cả các thông tin trong các cuộc phỏng vấn đó dưới các dạng như tài liệu, băng ghi âm,…

Cuối cùng, chúng ta có thể tiến hành phỏng vấn từng người tham gia trực tiếp vào từng công việc cụ thể. Để có thể liệt kê và mô tả toàn bộ các công việc phải thực hiện, đối với mỗi công việc cần phải nắm bắt được những sự kiện khởi động, chu kỳ thực hiện, thời lượng thực hiện, những dữ kiện có liên quan, khối lượng các dữ kiện này, các qui tắc cần phải áp dụng để thực hiện công việc, quan sát sự luân chuyển thông tin giữa các vị trí làm việc.

Ở giai đoạn phỏng vấn, chúng ta có thể sử dụng các phiếu phỏng vấn trong đó phải thể hiện được các thông tin như: tên công việc, các điều kiện khởi động, kết quả đưa ra, tần suất, thời lượng thông tin, dữ liệu , các quy tắc nghiệp vụ cần thực hiện…

Sau các buổi phỏng vấn chúng ta cần phải xem lại tất cả những gì đã thu thập được, hệ thống sắp xếp lại dưới dạng hồ sơ khảo sát, liệt kê đầy đủ các thông tin. Việc xác định các đối tượng dữ liệu, các qui tắc nghiệp vụ ngoài việc chuẩn bị cho các giai đoạn phân tích dữ liệu tiếp theo còn nhằm để phát hiện được những vấn đề còn chưa rõ để đề ra các câu hỏi phụ cho những người đối thoại trong các buổi phỏng vấn kế tiếp.

b. Kỹ thuật sử dụng phiếu khảo sát

Có thể nói đây là một hình thức phỏng vấn gián tiếp. Các câu hỏi được liệt kê trong một phiếu điều tra, khảo sát. Người được khảo sát sẽ ghi lại các câu trả lời vào trong các phiếu điều tra, khảo sát đó.

Việc sử dụng các loại câu hỏi cũng như thứ tự các câu hỏi ở đây cũng giống như khi phỏng vấn trực tiếp, tuy nhiên thứ tự này chỉ có ý nghĩa về mặt sắp xếp vấn đề mà ít có tác dụng dẫn dắt tư duy của người được hỏi. Vì người được khảo sát có thể đọc qua một lượt các câu hỏi, rồi tuỳ tiện trả lời các câu hỏi theo bất cứ trật tự nào mà họ muốn, thậm chí họ có thể không trả lời một số câu hỏi.

So với kỹ thuật phỏng vấn trực tiếp, chỗ yếu chính của phiếu khảo sát là thiếu sự giao tiếp giữa chúng ta và khách hàng. Nhưng ngược lại, phiếu khảo sát lại có chỗ mạnh là có thể mở rộng diện khảo sát và ít tốn kém.

c. Kỹ thuật phân tích tài liệu

Các loại sổ sách, tài liệu, tệp dưới dạng điện tử,… thường là những nguồn thông tin để khảo sát về các loại dữ liệu, luồng dữ liệu và giao dịch. Kỹ thuật phân tích khai thác nguồn tài liệu này được tiến hành như sau: đầu tiên lập một danh sách các tài liệu qua sự tìm hiểu từ phía các đối tượng sử dụng, sau đó nghiên cứu từng tài liệu để phát hiện ra các dữ liệu cơ bản và các dữ liệu cấu trúc. Kỹ thuật này có thể cho phép chúng ta phát hiện ra các dữ liệu trùng lặp hoặc sự thiếu nhất quán trong lưu trữ, tên gọi.

Ngoài ra, nếu doanh nghiệp đã có một hệ thống cơ sở dữ liệu và họ muốn nâng cấp hay thay đổi toàn bộ thì chúng ta

có thể xem xét lại các tài liệu liên quan đến cơ sở dữ liệu đó, điều này sẽ giúp cho chúng ta biết thêm khá nhiều thông tin và hiểu được tại sao doanh nghiệp lại muốn xây dựng một hệ thống cơ sở dữ liệu mới.

3. Thiết kế cơ sở dữ liệu mức khái niệm

Các thông tin thu thập được trong bước phân tích yêu cầu trên sẽ được sử dụng để xây dựng lên những mô tả đầu tiên về dữ liệu cũng như các ràng buộc cần phải thoả mãn đối với dữ liệu. Bước này thường được thực hiện thông qua một mô hình mà chúng ta gọi là mô hình thực thể - liên kết.

Mô hình thực thể - liên kết thường được sử dụng như là một công cụ kết nối giữa người thiết kế cơ sở dữ liệu và người dùng cuối trong quá trình phân tích cơ sở dữ liệu. Nó được sử dụng để xây dựng lên mô hình dữ liệu mức khái niệm, mô tả cấu trúc và các ràng buộc của dữ liệu trong cơ sở dữ liệu, mô hình này độc lập với cả hệ quản trị cơ sở dữ liệu cũng như mô hình dữ liệu sẽ được sử dụng đề cài đặt nó.

Mô hình thực thể - liên kết thường được biểu diễn dưới dạng một sơ đồ thực thể - liên kết thông qua các hình vẽ mô tả cho 3 đối tượng chính, đó là kiểu thực thể, thuộc tính và liên kết. Do vậy, kết quả của bước thiết kế cơ sở dữ liệu mức khái niệm sẽ là một sơ đồ thực thể - liên kết đầy đủ.

Khi xây dựng một sơ đồ thực thể - liên kết, chúng ta phải tiến hành các lựa chọn sau:

- Một khái niệm sẽ được mô hình hoá thành một kiểu thực thể hay một thuộc tính?
- Nếu là kiểu thực thể thì nó sẽ là loại nào?

- Nếu là thuộc tính thì nó sẽ là loại nào?
- Các thuộc tính đã đủ để minh hoạ cho kiểu thực thể chưa?
- Một khái niệm sẽ được mô hình hoá thành một kiểu thực thể hay một liên kết?
- Nếu là một liên kết thì chúng kết nối những kiểu thực thể nào? Nên sử dụng loại liên kết nào?

Để có thể trả lời các câu hỏi đó, chúng ta phải phân tích kỹ các thông tin thu thập được trong bước phân tích yêu cầu cũng như kinh nghiệm thiết kế các hệ cơ sở dữ liệu của chính mình.

3.1. Lựa chọn giữa kiểu thực thể mạnh và kiểu thực thể yếu

a. Kiểu thực thể mạnh

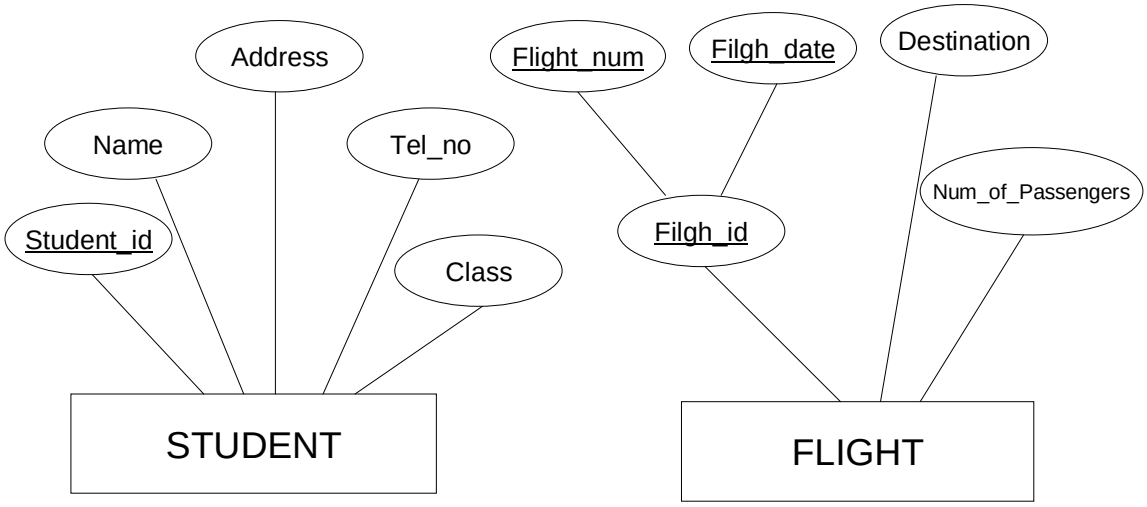
Hầu hết các kiểu thực thể được xác định trong một bài toán xây dựng cơ sở dữ liệu là các kiểu thực thể mạnh. Kiểu thực thể mạnh là một kiểu thực thể có thể tồn tại độc lập với các kiểu thực thể khác. Ví dụ như các kiểu thực thể EMPLOYEE, STUDENT trong các ví dụ đã nêu.

Trong kiểu thực thể mạnh, mỗi thực thể của nó luôn tồn tại một đặc trưng duy nhất được gọi là định danh. Định danh này có thể bao gồm một hoặc gộp của một số thuộc tính, mỗi giá trị xuất hiện trong định danh sẽ xác định duy nhất một thực thể trong kiểu thực thể. Các thuộc tính tham gia vào định danh được gọi là các thuộc tính định danh hay thuộc tính khoá. Chúng ta sẽ tìm hiểu kỹ hơn về loại thuộc

tính này trong phần lựa chọn các thuộc tính được trình bày trong các phần sau.

Lưu ý: chúng ta thường sử dụng khái niệm định danh với kiểu thực thể, còn khái niệm khoá với quan hệ.

Ví dụ chúng ta có hai kiểu thực thể như trong hình 2.3.1.1 sau:



Hình 2.3.1.1 Kiểu thực thể mạnh

Cả hai kiểu thực thể trên đều là kiểu thực thể mạnh, chúng được ký hiệu bằng một hình chữ nhật. Trong kiểu thực thể sinh viên STUDENT: định danh là thuộc tính số hiệu sinh viên Student_id. Còn trong kiểu thực thể chuyến bay FLIGHT thì định danh bao gồm kết hợp của hai thuộc tính đó là số hiệu chuyến bay Flight_num và ngày Fligh_date, bởi để xác định một chuyến bay cụ thể chúng ta cần kết hợp các giá trị trong cả hai thuộc tính này.

b. Kiểu thực thể yếu

Ngược lại, kiểu thực thể yếu là một kiểu thực thể mà sự tồn tại của nó phụ thuộc vào một kiểu thực thể khác. Trong sơ

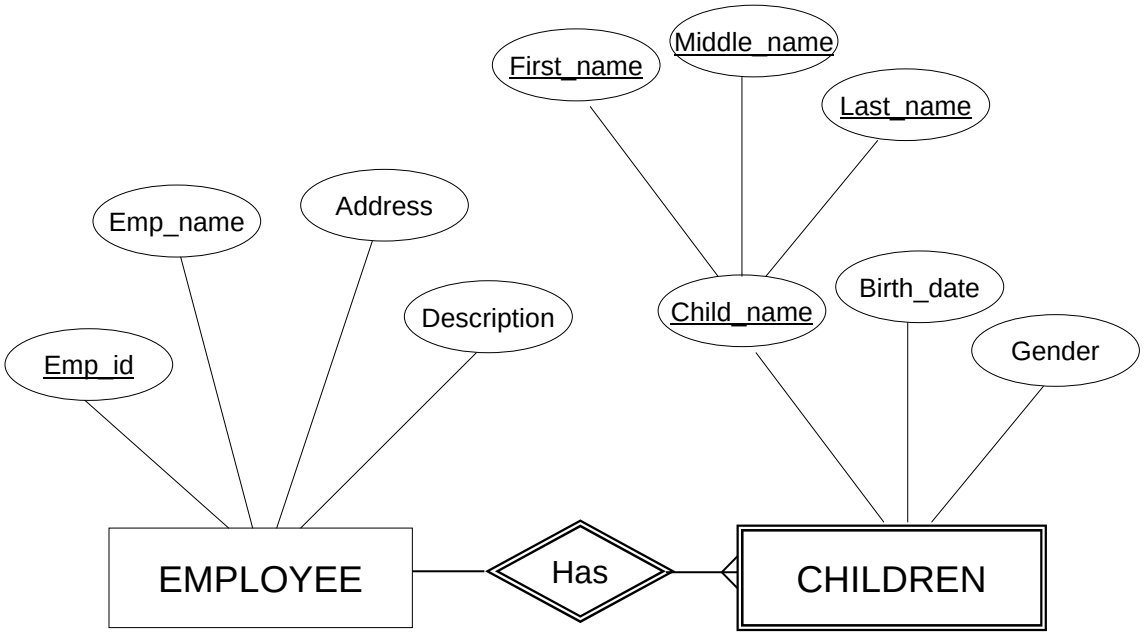
đồ thực thể - liên kết, một kiểu thực thể yếu luôn luôn phải được biểu diễn cùng với kiểu thực thể mà nó phụ thuộc. Kiểu thực thể mà kiểu thực thể yếu phụ thuộc vào được gọi là kiểu thực thể sở hữu (owner).

Trong kiểu thực thể yếu không tồn tại định danh đầy đủ. Ngoài các thuộc tính mô tả, trong kiểu thực thể yếu chỉ tồn tại một số thuộc tính được xem là định danh bộ phận. Định danh bộ phận này sẽ được kết hợp với định danh của kiểu thực thể sở hữu để có thể xác lập duy nhất một thực thể trong nó.

Chúng ta ký hiệu một kiểu thực thể yếu bởi một hình chữ nhật có nét kép như trong hình 2.3.1.2 dưới đây.

Liên kết giữa kiểu thực thể yếu và kiểu thực thể sở hữu nó được gọi là liên kết định danh và được ký hiệu bằng một hình thoi có nét kép.

Ví dụ: Chúng ta muốn lưu những thông tin về tình hình các cháu nhỏ là con của các nhân viên làm việc trong doanh nghiệp. Khi đó chúng ta phải sử dụng hai kiểu thực thể, một kiểu thực thể là EMPLOYEE mô tả các thông tin về các nhân viên và một kiểu thực thể là CHILDREN mô tả các thông tin về tình trạng con cái của họ. Rõ ràng EMPLOYEE là một kiểu thực thể mạnh với định danh là thuộc tính mã hiệu nhân viên Emp_id. Còn CHILDREN là một kiểu thực thể yếu, nó phụ thuộc vào kiểu thực thể EMPLOYEE, thuộc tính Child_name được gọi là định danh bộ phận. Và liên kết “Has” là một liên kết định danh, nó được ký hiệu bằng một hình thoi có hai nét. Ví dụ này được minh họa như hình 2.3.1.2 sau:



Hình 2.3.1.2 Kiểu thực thể yếu

3.2. Lựa chọn giữa các loại thuộc tính

a. Thuộc tính đơn và gộp

Thuộc tính đơn (hay nguyên tử) là thuộc tính không thể chia nhỏ thành các thành phần riêng biệt. Ví dụ như các thuộc tính số hiệu sinh viên Student_id, màu sắc Color, Số lượng Quantity,...

Ngược lại, có một số thuộc tính mà giá trị của nó có thể chia được ra thành nhiều phần có ý nghĩa. Ví dụ như thuộc tính địa chỉ Address, thuộc tính này có thể được chia thành các phần như: Country, Province, City và Street. Hay trong hình 2.3.1.2 trên, thuộc tính Child_name là thuộc tính gộp của 3 thuộc tính First_name, Middle_name và Last_name.

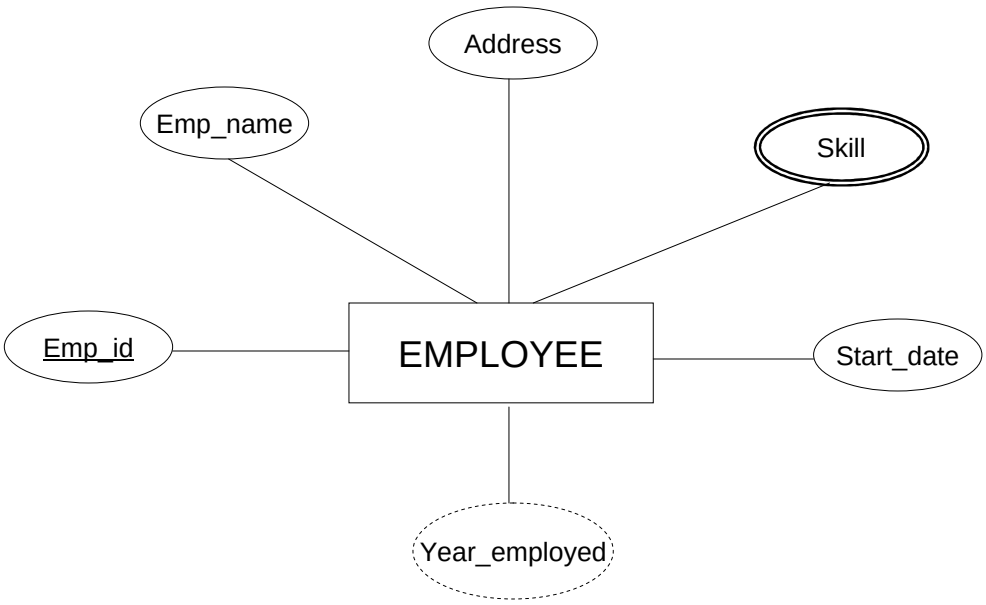
Thuộc tính gộp chỉ mang tính chất minh hoạ bởi nó được biểu diễn bởi các thuộc tính đơn. Do đó khi xây dựng sơ đồ thực thể - liên kết, chúng ta cố gắng trình bày các thuộc tính dưới dạng các thuộc tính đơn.

b. Thuộc tính đơn trị và thuộc tính đa trị

Trong quá trình phân tích bài toán đề tìm kiếm các thuộc tính cho một kiểu thực thể, chúng ta sẽ gặp những trường hợp mà một thuộc tính có thể có nhiều hơn một giá trị đối với một số thực thể.

Ví dụ: trong kiểu thực thể EMPLOYEE, chúng ta nhận thấy rằng thuộc tính kỹ năng Skill của mỗi một thực thể có thể nhận nhiều hơn một giá trị, bởi trên thực tế một nhân viên lập trình có thể biết cả ngôn ngữ C++ và ngôn ngữ Basic.

Như vậy, một thuộc tính đa trị là một thuộc tính có thể chứa một hoặc một vài giá trị. Chúng ta ký hiệu thuộc tính đa trị bằng một hình elip có một vòng tròn kép như thuộc tính Skill trong hình 2.3.2.1 sau:



Hình 2.3.2.1 Thuộc tính kép

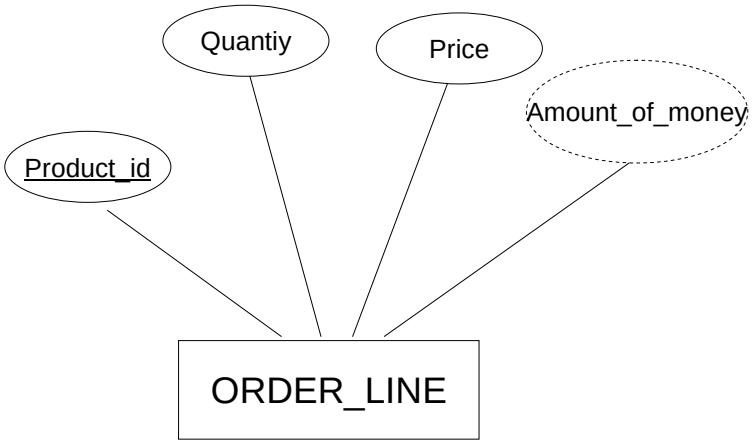
c. Thuộc tính lưu trữ và thuộc tính dẫn xuất

Có một số thuộc tính giá trị của nó có thể được tính toán hoặc suy ra từ một số giá trị của các thuộc tính liên quan.

Ví dụ: với kiểu thực thể EMPLOYEE, chúng ta có thuộc tính ngày bắt đầu làm việc Start_date cho biết ngày một nhân viên được nhận vào làm việc, rõ ràng nếu muốn biết một nhân viên nào đó đã làm việc được bao nhiêu lâu thì chúng ta có thể dựa trên giá trị trong thuộc tính này và ngày hiện thời để tính ra giá trị đó. Hay trong các bài toán kế toán, với kiểu thực thể dòng đơn hàng ORDER_LINE chúng ta biết rằng giá trị trong thuộc tính tổng tiền Amount_of_money luôn bằng giá trị trong thuộc tính đơn giá Unit_price nhân với giá trị trong thuộc tính số lượng Quantity.

Như vậy một thuộc tính dẫn xuất là một thuộc tính mà các giá trị của nó có thể được suy ra từ các giá trị của các thuộc tính liên quan (cũng có thể giá trị đó không lưu trong cơ sở dữ liệu, như giá trị ngày hiện thời như trong ví dụ trên).

Chúng ta ký hiệu thuộc tính dẫn suất bằng một hình elip có nét đứt như trong các hình 2.3.2.1 và 2.3.2.2:



Hình 2.3.2.2 Thuộc tính dẫn xuất

d. Thuộc tính khoá hay thuộc tính định danh

Như đã trình bày, trong cùng một kiểu thực thể chúng ta cần phải phân biệt được giữa các thực thể với nhau. Để làm được điều này chúng ta phải quy ước chọn ra một hoặc một số thuộc tính sao cho giá trị xuất hiện trong các thuộc tính đó sẽ xác định duy nhất một thực thể, chúng ta gọi các thuộc tính này là một định danh và các thuộc tính tham gia vào định danh đó là các thuộc tính khoá hay thuộc tính định danh.

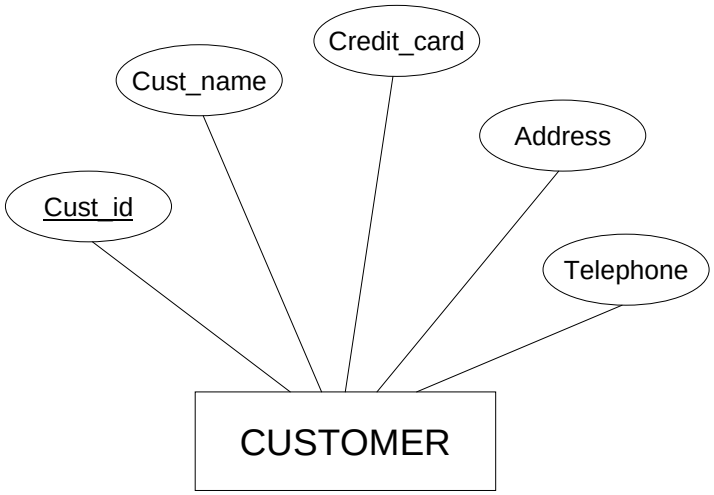
Ví dụ, định danh cho kiểu thực thể STUDENT sẽ là thuộc tính số hiệu sinh viên Student_id. Tuy nhiên, một thuộc

tính khác như tên sinh viên Student_name chẳng hạn thì không thể là định danh, bởi có thể có một số sinh viên có tên trùng nhau (bao gồm cả họ và tên), mặt khác tên của một sinh viên cũng có thể thay đổi.

Đôi khi chúng ta sẽ gặp trường hợp trong cùng một kiểu thực thể có thể có nhiều hơn một (hoặc một số) thuộc tính có thể được sử dụng làm định danh, khi đó dựa trên bài toán cùng kinh nghiệm, chúng ta phải chọn lấy một trong số các lựa chọn đó làm định danh duy nhất.

Ví dụ: với kiểu thực thể khách hàng CUSTOMER, chúng ta có các thuộc tính: Cust_id, Cust_name, Address, Tel_no, Credit_card. Rõ ràng với kiểu thực thể này thuộc tính số hiệu khách hàng Cust_id và số thẻ tín dụng Credit_card đều có thể chọn là định danh, bởi mỗi giá trị trong các thuộc tính này đều xác định duy nhất một thực thể khách hàng Tuy nhiên theo kinh nghiệm, chúng ta vẫn nên chọn thuộc tính số hiệu khách hàng Cust_id làm định danh duy nhất cho kiểu thực thể này bởi có thể có những khách hàng không sử dụng thẻ tín dụng.

Chúng ta phân biệt thuộc tính định danh với các thuộc tính khác bằng một đường gạch chân dưới thuộc tính đó như hình 2.3.2.3 sau:



Hình 2.3.2.3 Thuộc tính định danh

Dưới đây là một số gợi ý mà chúng ta có thể sử dụng để lựa chọn được một định danh duy nhất cho mỗi kiểu thực thể:

- Chọn một định danh mà giá trị của nó sẽ không bị thay đổi theo thời gian. Ví dụ trong kiểu thực thể EMPLOYEE, chúng ta có thể chọn hai thuộc tính tên nhân viên Name và địa chỉ Address kết hợp làm định danh duy nhất vì thực sự chúng luôn là duy nhất. Tuy nhiên lựa chọn này là một lựa chọn không tối ưu bởi rõ ràng một nhân viên có thể thay đổi chỗ ở của mình trong quá trình làm việc tại công ty.
- Chọn định danh gồm các thuộc tính mà giá trị của chúng không được phép mang giá trị null (giá trị không xác định). Trong trường hợp định danh là một thuộc tính gộp thì các thuộc tính con của nó cũng không được phép rỗng. Ví dụ như với kiểu thực thể chuyến bay FILGHT như hình 2.3.1.1 trên, đối với định danh gộp, số hiệu chuyến bay Flight_id, là kết hợp của 2 thuộc tính số hiệu chuyến bay Flight_num

và ngày bay Flight_date đều không thể mang giá trị null.

- Tránh sử dụng những thuộc tính mà giá trị của nó thể hiện thông tin, hay cấu trúc của nó thể hiện sự phân loại, vị trí,... làm định danh. Ví dụ nếu chúng ta có kiểu thực thể kho hàng STORE, các giá trị trong thuộc tính mã kho Store_code có qui ước một ký tự đầu được thể hiện cho địa danh đặt kho, hai ký tự sau thể hiện cho số thứ tự kho trong địa danh đó. Rõ ràng theo thời gian, mã các kho hàng này có thể thay đổi như khi số lượng các địa danh có thể đặt các kho tăng lên nhiều hơn 9 ví trí, tức là khi chúng ta cần phải có tới hai ký tự để thể hiện cho vị trí đặt kho, và khi đó các mã kho cũng phải thay đổi theo. Một ví dụ khác mà trên thực tế, một số bài toán đã gặp phải, đó là qui ước cách đánh mã số các địa bàn hành chính sử dụng bộ mã 7 ký tự, cách đánh mã số này qui ước như sau: 1 ký tự đầu ký hiệu cho vùng, 2 ký tự sau ký hiệu cho tỉnh thuộc vùng đó, 2 ký tự tiếp theo ký hiệu cho huyện thuộc tỉnh và 2 ký tự cuối cùng ký hiệu cho xã thuộc huyện. Khi nảy sinh các vấn đề tách tỉnh, gộp tỉnh, tách huyên, gộp huyện, tách xã gộp xã thì các mã này rõ ràng cũng sẽ bị sửa đổi theo qui tắc đánh mã như trên.
- Cố gắng lựa chọn những thuộc tính đơn làm định danh trong khả năng có thể thay vì sử dụng kết hợp của một số thuộc tính. Ví dụ nếu ta có một cơ sở dữ liệu về một giải bóng đá, trong đó có một kiểu thực thể là trận đấu GAME, chúng ta nên lựa chọn thuộc tính số thứ tự trận đấu Game_no làm định danh thay

vì sử dụng kết hợp hai thuộc tính tên đội khách Visiting_team và tên đội nhà Home_team làm định danh.

3.3. Lựa chọn giữa các loại liên kết

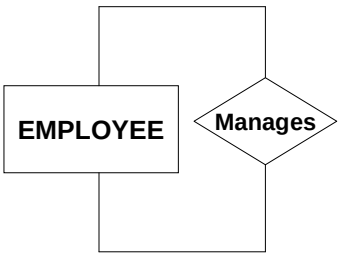
Để mô hình hoá thế giới thực, chúng ta cũng phải sử dụng đến các liên kết để minh hoạ cho mối quan hệ giữa các kiểu thực thể. Và để có thể lựa chọn được đúng loại liên kết chúng ta cần phải nắm được 3 khái niệm cơ bản sẽ được dùng trong mỗi một liên kết.

a. Ba khái niệm cơ bản về liên kết

Việc phân chia ra các loại liên kết được dựa trên 3 khái niệm:

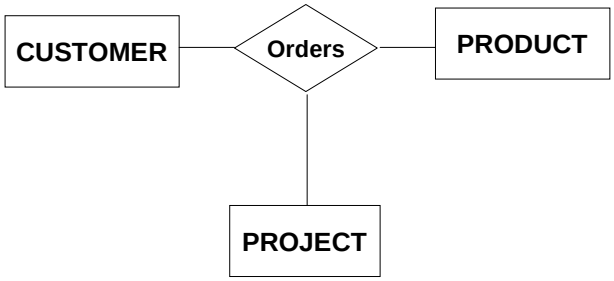
- Cấp độ của liên kết: được thể hiện bằng việc có bao nhiêu kiểu thực thể tham gia vào liên kết. Có thể liên kết kết nối giữa cùng một kiểu thực thể, hoặc cũng có thể liên kết kết nối giữa hai kiểu thực thể hay cũng có thể liên kết kết nối giữa nhiều kiểu thực thể. Tuy nhiên, trên thực tế liên kết giữa hai kiểu thực thể thường được sử dụng nhiều nhất và trong trường hợp chúng ta gặp phải những liên kết giữa nhiều kiểu thực thể thì cố gắng đưa chúng về loại liên kết giữa hai kiểu thực thể.

Chúng ta ký hiệu các cấp độ này như hình 2.3.3.1 sau:



(a) Liên kết giữa cùng một kiểu thực thể

(b) Liên kết giữa hai kiểu thực thể



(c) Liên kết giữa nhiều kiểu thực thể

Hình 2.3.3.1 Cấp độ của liên kết

Ví dụ: như trong hình 2.3.3.1 trên:

Hình (a) thể hiện cho mối liên kết trong cùng một kiểu thực thể EMPLOYEE, liên kết này cho biết một nhân viên có thể quản lý trực tiếp một số nhân viên khác, chẳng hạn giám đốc sẽ quản lý trực tiếp các trưởng phòng, các trưởng phòng lại quản lý trực tiếp các trưởng nhóm, còn các trưởng nhóm lại quản lý trực tiếp các nhân viên. Liên kết này còn được gọi là liên kết đệ qui.

Hình (b) thể hiện cho mối liên kết giữa hai kiểu thực thể là khách hàng CUSTOMER và mặt hàng PRODUCT, liên kết này cho biết khách hàng có thể đặt mua các mặt hàng.

Hình (c) thể hiện cho mối liên kết giữa ba kiểu thực thể là khách hàng CUSTOMER, mặt hàng PRODUCT và

dự án PROJECT. Liên kết này cho biết khách hàng mua các mặt hàng gì và để phục vụ cho dự án nào. Thông thường trong hầu hết các trường hợp, với những liên kết giữa ba kiểu thực thể, chúng ta nên chuyển đổi liên kết đó thành một kiểu thực thể kết nối (sẽ được trình bày dưới đây).

- Lực lượng của liên kết: khái niệm này thể hiện số lượng các thực thể của mỗi một kiểu thực thể tham gia vào liên kết đó như thể nào. Các trường hợp có thể xảy ra như sau: nếu khi mỗi một thực thể của kiểu thực thể này chỉ có thể kết nối với mỗi một thực thể của kiểu thực thể kia thì đó là liên kết một-một. Còn nếu với mỗi một thực thể của kiểu thực thể này có thể kết nối với nhiều thực thể của kiểu thực thể kia thì chúng ta có liên kết một-nhiều. Cuối cùng, nếu mỗi một thực thể của kiểu thực thể này có thể kết nối với nhiều thực thể của kiểu thực thể kia và ngược lại thì khi đó chúng ta có liên kết nhiều-nhiều.

Để phân biệt với đầu liên kết phía một, chúng ta ký hiệu đầu liên kết phía nhiều có ba đoạn thẳng nhỏ xuất phát từ một điểm như trong hình 2.3.3.2 sau:

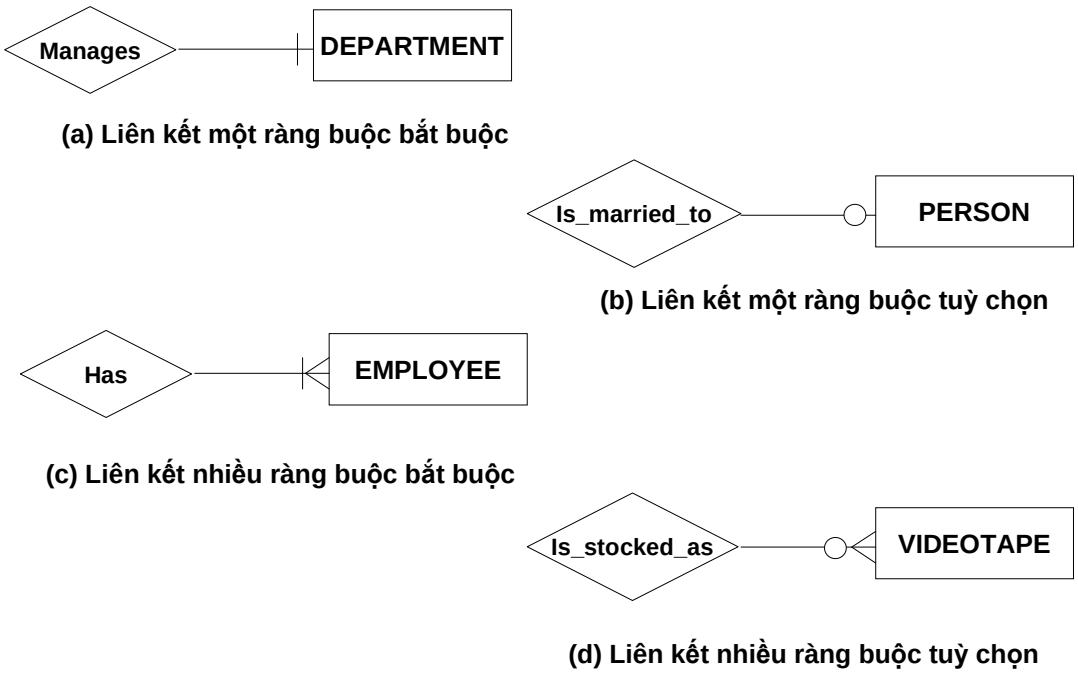
(a) Đầu liên kết phía một **(b) Đầu liên kết phía nhiều**

Hình 2.3.3.2 Lực lượng của liên kết

Chúng ta sẽ tìm hiểu kỹ hơn khái niệm lực lượng của liên kết trong các phần tiếp theo.

- Ràng buộc tham gia của liên kết: khái niệm này thể hiện cung cách của các kiểu thực thể tham gia vào liên kết. Cụ thể, nếu mỗi một thực thể của kiểu thực thể A khi xuất hiện và tham gia vào liên kết đều phải được kết nối với một hoặc một số thực thể của kiểu thực thể B khi đó chúng ta nói ràng buộc tham gia vào liên kết của kiểu thực thể A là bắt buộc. Ngược lại, khi một thực thể của kiểu thực thể A xuất hiện mà có thể có hoặc không có một thực thể nào của kiểu thực thể B cùng tham gia vào liên kết với A thì ta nói ràng buộc tham gia vào liên kết của A là tuỳ chọn.

Chúng ta ký hiệu các ràng buộc này như hình 2.3.3.3 sau:



Hình 2.3.3.3 Ràng buộc tham gia của liên kết

Như trong hình 2.3.3.3 trên chúng ta có:

Hình (a) mô tả cho liên kết một có ràng buộc tham gia là bắt buộc. Ví dụ mỗi một phòng ban phải có duy nhất một người quản lý, khi đó rõ ràng kiểu thực thể

DEPARTMENT khi tham gia vào liên kết phải là loại ràng buộc bắt buộc.

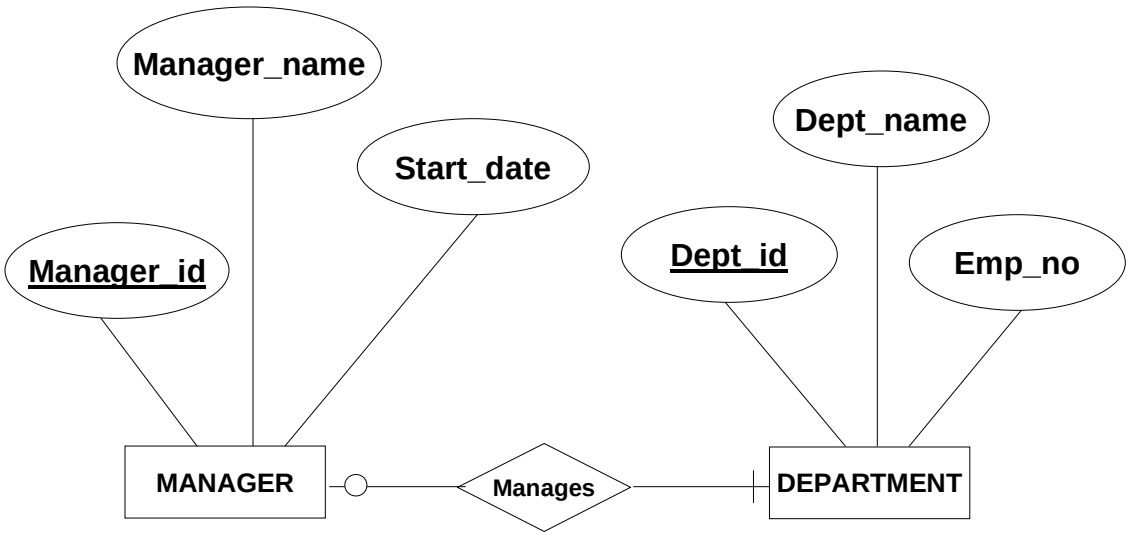
Hình (b) mô tả cho liên kết một có ràng buộc tham gia là tuỳ chọn. Ví dụ, mỗi một người chỉ có thể lập gia đình với một người khác và cũng có thể có những người không hoặc chưa lập gia định. Như vậy chúng ta thấy rằng kiểu thực thể PERSON tham gia vào liên kết một theo kiểu ràng buộc tuỳ chọn.

Hình (c) mô tả cho một liên kết nhiều với ràng buộc tham gia là bắt buộc. Ví dụ một nhân viên phải được bố trí vào làm việc ở một phòng nhất định và có thể có nhiều nhân viên cùng làm việc trong một phòng. Như vậy chúng ta thấy rằng ràng buộc tham gia của kiểu thực thể EMPLOYEE vào liên kết nhiều phải là loại bắt buộc

Hình (d) mô tả cho một liên kết nhiều với ràng buộc tham gia là tuỳ chọn. Ví dụ mỗi một hoặc một số cuốn băng sẽ được sử dụng để ghi lại cho một bộ phim. Tuy nhiên cũng có thể có những cuốn băng vẫn chưa được sử dụng nhưng chúng ta vẫn phải quản lý. Khi đó kiểu thực thể VIDEOTAPE tham gia vào liên kết theo loại ràng buộc là tuỳ chọn.

b. Các liên kết một-một (1:1)

Dạng đơn giản nhất và hiếm gặp nhất của liên kết trên hai kiểu thực thể là một-một, có nghĩa là với mỗi thực thể trong kiểu thực thể A có nhiều nhất một thực thể trong kiểu thực thể B kết nối với nó và ngược lại.



Hình 2.3.3.4 Liên kết một - một

Ví dụ như liên kết quản lý "Manages" giữa 2 kiểu thực thể DEPARTMENT và MANAGER trong hình 2.3.3.4 trên có thể được khai báo là liên kết một-một. Như vậy, chúng ta thấy rằng không bao giờ có một phòng cùng được quản lý bởi nhiều người, và ngược lại cũng không bao giờ thấy một người quản lý 2 hay nhiều phòng. Từ hình vẽ chúng ta cũng có thể thấy một người quản lý có thể ở một thời điểm nào đó không quản lý phòng nào nhưng một phòng thì luôn luôn có một người quản lý.

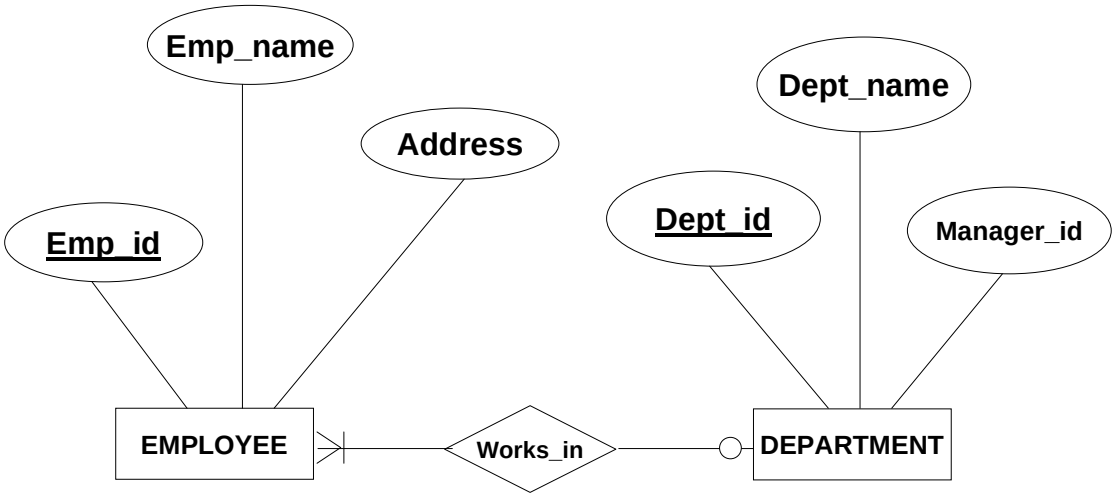
Một ví dụ khác: mỗi một nhân viên trong công ty được bố trí làm việc ở một bàn làm việc nhất định. Cũng như vậy, ở mỗi một bàn làm việc chỉ có thể có một nhân viên ngồi làm việc mà thôi.

c. Các liên kết một-nhiều (1:N)

Đây là loại liên kết quan trọng nhất và phổ biến nhất. Trong liên kết một-nhiều, một thực thể trong kiểu thực thể B được

kết nối với không, một hay nhiều thực thể trong kiểu thực thể A, nhưng mỗi thực thể trong kiểu thực thể A chỉ được kết nối với nhiều nhất một thực thể trong kiểu thực thể B. Liên kết này được gọi là liên kết một-nhiều từ B đến A. Điều này có nghĩa liên kết là một hàm bộ phận từ B đến A.

Ví dụ liên kết làm việc "Works_in" giữa hai kiểu thực thể DEPARTMENT và EMPLOYEE trong hình 2.3.3.5 dưới đây, mỗi một nhân viên được bố trí làm việc ở nhiều nhất một phòng, và mỗi một phòng có thể gồm nhiều nhân viên hoặc chưa có nhân viên nào.



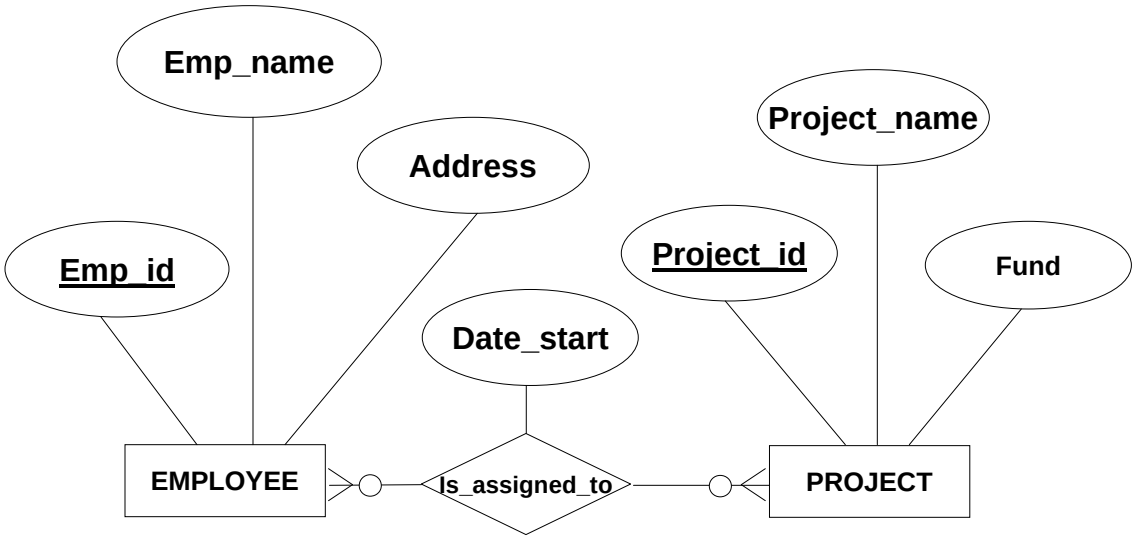
Hình 2.3.3.5 Liên kết một - nhiều

d. Các liên kết nhiều-nhiều (M:N)

Chúng ta cũng bắt gặp các liên kết nhiều-nhiều, đôi khi còn được gọi là các liên kết không cụ thể. Liên kết nhiều-nhiều là liên kết khi với mỗi một thực thể trong kiểu thực thể A, nó được kết nối với không, một hoặc nhiều thực thể trong kiểu thực thể B và ngược lại, với mỗi thực thể B trong kiểu

thực thể B, nó được kết nối với không, một hoặc nhiều thực thể trong kiểu thực thể A.

Ví dụ: Một nhân viên tại cùng một thời điểm có thể cùng tham gia vào hai hay nhiều dự án. Ngược lại mỗi một dự án có thể được thực hiện bởi một hay nhiều nhân viên khác nhau. Như vậy một nhân viên có thể được kết nối với nhiều dự án, ngược lại, một dự án cũng có thể được kết nối với nhiều nhân viên. Hình 2.3.3.6 mô tả cho liên kết tham gia này:



Hình 2.3.3.6 Liên kết nhiều - nhiều

Trong khi các liên kết nhiều-nhiều xuất hiện khá thường xuyên trong thực tế, chúng ta phải cẩn thận khi xem xét các liên kết này được biểu thị như thế nào trong mô hình khái niệm của cơ sở dữ liệu.

Đôi khi liên kết nhiều-nhiều không được chuyển đổi trực tiếp thành các bảng, thay vào đó nó sẽ được chuyển đổi

thành hai hay nhiều liên kết một-nhiều bằng cách sử dụng thêm kiểu thực thể kết nối.

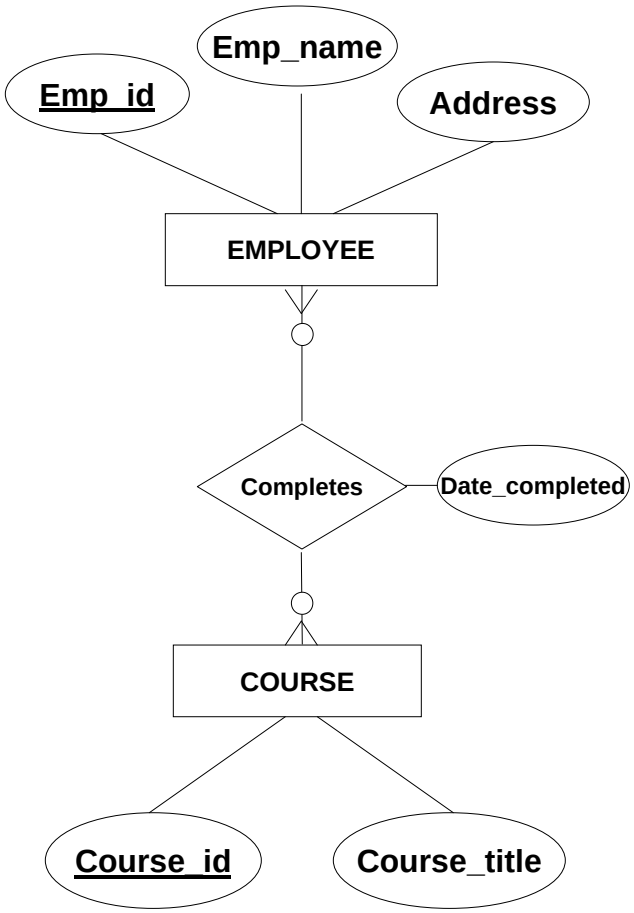
e. Thuộc tính cho liên kết

Các thuộc tính cũng có thể được sử dụng để mô tả cho các liên kết.

Ví dụ trong hình 2.3.3.7a là một số dữ liệu về tình hình tốt nghiệp các khoá học nâng cao trình độ của một số nhân viên. Giá trị trong thuộc tính ngày tháng Date_completed cho biết ngày tháng mà các nhân viên đã hoàn thành các khoá học tương ứng.

Emp_name	Course	Date_ completed
Trần Văn T.	C++	06/2003
Trần Văn T.	Java	09/2003
Mai Văn H.	Basic	05/2003
Mai Văn H.	SQL	07/2003
Lê Đức T.	C++	02/2003
Nguyễn Thị K.	C++	02/2003

(a) Bảng dữ liệu



(b) Thuộc tính cho liên kết

Hình 2.3.3.7 Thuộc tính trên liên kết

Vậy thì trong sơ đồ thực thể - liên kết, thuộc tính Date_completed sẽ được đặt ở vị trí nào? Chúng ta không thể đặt thuộc tính này trong các kiểu thực thể EMPLOYEE cũng như COURSE. Bởi Date_completed mô tả cho liên kết "Completes" chứ không phải cho các kiểu thực thể đó. Mặt khác chúng ta cũng thấy rằng, mỗi một thể hiện của liên kết "Completes" cũng tồn tại đi kèm một giá trị của thuộc tính Date_completed. Như bảng trong hình 2.3.3.7a, chúng ta thấy rằng Trần Văn T. đã hoàn thành khoá học về C++ vào tháng 06/2003.

Do vậy chúng ta phải sử dụng thuộc tính Date_completed để mô tả cho liên kết hoàn thành "Completes" giữa hai kiểu thực thể nhân viên EMPLOYEE và lớp học COURSE như trong hình 2.3.3.7b. Trên thực tế cũng có thể có một số thuộc tính khác đi kèm với liên kết "Completes" này ví dụ như kết quả tốt nghiệp Course_Grade, hay nơi tốt nghiệp Location chẳng hạn.

3.4. Lựa chọn giữa kiểu thực thể và thuộc tính

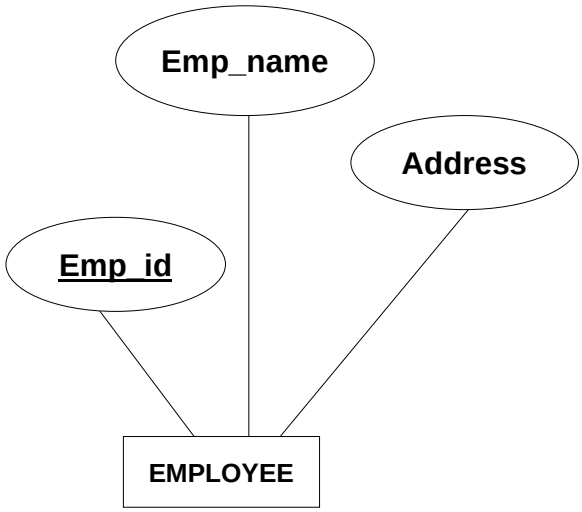
Trên đây chúng ta đã biết cách lựa chọn giữa các kiểu thực thể với nhau, giữa các liên kết với nhau, giữa các thuộc tính với nhau. Nhưng chúng ta vẫn chưa biết khi nào thì hình thành một thuộc tính hay khi nào thì hình thành một kiểu thực thể? Trong phần này chúng tôi sẽ giúp các bạn thực hiện điều đó.

Trong quá trình xác định các thuộc tính cho một tập thực thể, đôi lúc chúng ta sẽ gặp vấn đề là liệu một tính chất nào đó nên được mô hình hoá thành một thuộc tính hay thành một kiểu thực thể và nếu vậy thì nó quan hệ với các kiểu thực thể khác thông qua các liên kết như thế nào?

Ví dụ: Với bài toán quản lý doanh nghiệp, khi xem xét đến thông tin về địa chỉ của nhân viên, chúng ta phải đứng trước hai lựa chọn: chúng ta sẽ chỉ mô hình hoá nó như là một thuộc tính Address nằm trong kiểu thực thể EMPLOYEE?; hay chúng ta sẽ mô hình hoá nó thành một kiểu thực thể địa chỉ ADDRESS, và nếu vậy nó sẽ quan hệ với kiểu thực thể EMPLOYEE như thế nào?

Lựa chọn thứ nhất: Nếu bài toán không yêu cầu chúng ta cần phải quản lý đến các lần thay đổi địa chỉ của nhân viên

thì khi đó có thể coi địa chỉ như là một thuộc tính mô tả nằm trong kiểu thực thể nhân viên EMPLOYEE (hình 2.3.4.1).

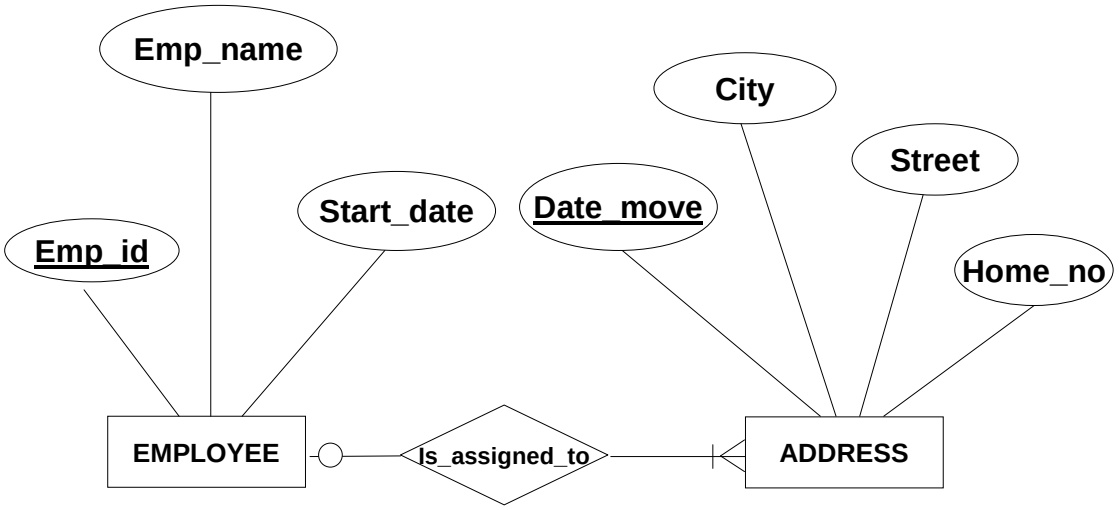


Hình 2.3.4.1 Hình thành một thuộc tính trong kiểu thực thể

Lựa chọn thứ hai: Ngược lại, nếu bài toán yêu cầu chúng ta phải quản lý cả những lần nhân viên thay đổi địa chỉ chỗ ở cũng như thời gian thay đổi thì khi đó chúng ta nên hình thành một kiểu thực thể mới, đặt tên là ADDRESS chẳng hạn, và kiểu thực thể này sẽ được kết nối với kiểu thực thể nhân viên EMPLOYEE theo kiểu liên kết một - nhiều (hình 2.3.4.2).

Như vậy, chúng ta sẽ chọn lựa chọn thứ hai này khi phải đáp ứng 2 tình huống sau:

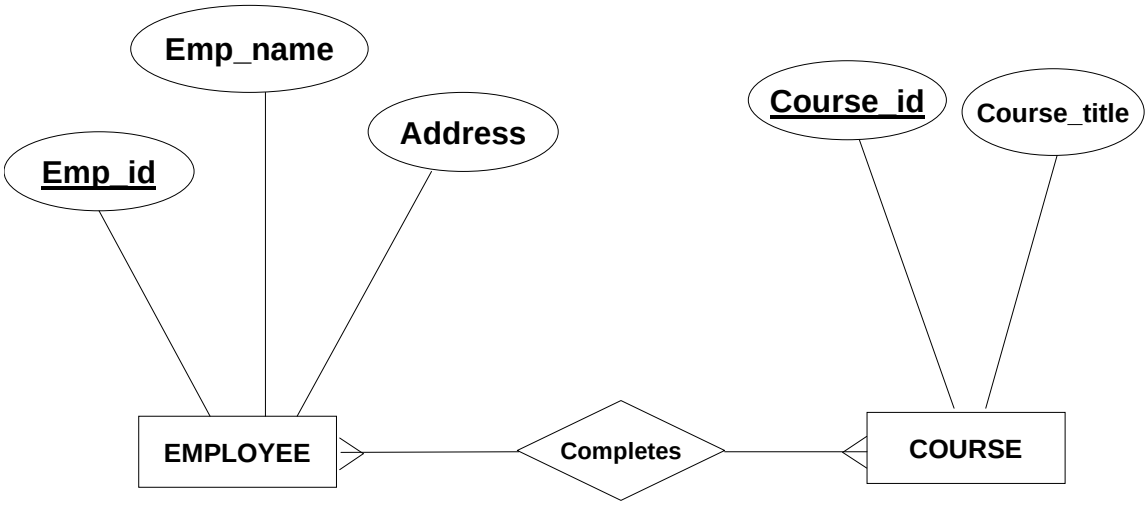
- Trên thực tế có những trường hợp chúng ta phải lưu trữ nhiều hơn một địa chỉ đối với mỗi một nhân viên.
- Chúng ta muốn quản lý chi tiết hơn nữa đối với thông tin địa chỉ như ngày chuyển đến địa chỉ mới, …



Hình 2.3.4.2 Hình thành một kiểu thực thể mới

3.5. Lựa chọn giữa kiểu thực thể và liên kết

Để bắt đầu, chúng ta hãy xét một ví dụ sau: Trong bài toán quản lý nhân sự cho công ty, chúng ta muốn quản lý về tình hình nâng cao trình độ của các nhân viên trong công ty. Giả sử chúng ta đã có hai kiểu thực thể nhân viên EMPLOYEE và khoá học COURSE. Rõ ràng giữa hai kiểu thực thể này tồn tại một liên kết mô tả cho biết các nhân viên đã hoàn tất những khoá học nào. Chúng ta tạm đặt tên liên kết này là hoàn thành "Completes". Ngoài ra cũng nhận thấy rằng để có thể mô tả được đầy đủ các trường hợp, chúng ta phải sử dụng loại liên kết nhiều - nhiều, bởi mỗi một nhân viên có thể hoàn tất và tốt nghiệp nhiều khoá học, ngược lại mỗi một khoá học có thể có nhiều nhân viên đã tốt nghiệp. Chúng ta thể hiện bằng hình 2.3.5.1 sau:



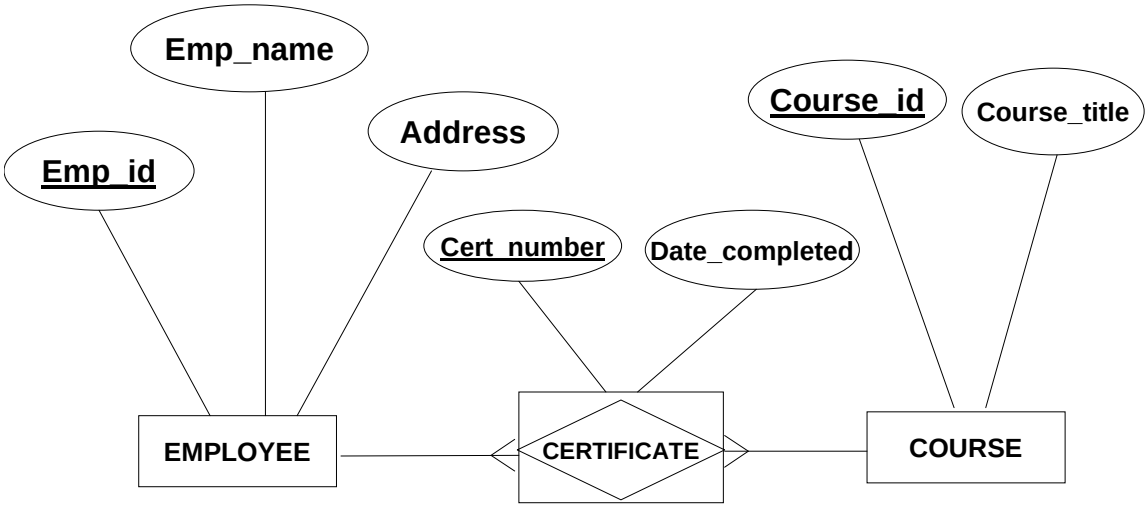
Hình 2.3.5.1

Nếu nhu cầu quản lý thông tin về tình hình nâng cao trình độ của nhân viên được mở rộng thêm, như chúng ta phải lưu lại số chứng chỉ mà mỗi một nhân viên đã nhận, cũng như ngày đã tốt nghiệp của mỗi một khoá học, thì khi đó ta phải sử dụng thêm các thuộc tính để mô tả cho liên kết "Completes" trên.

Trong liên kết nhiều - nhiều, khi phải xác định các thuộc tính cho liên kết, chúng ta hãy nghĩ đến trường hợp chuyển đổi liên kết đó thành một kiểu thực thể mới, được gọi là kiểu thực thể kết nối, nó vừa đóng vai trò là một liên kết giữa hai kiểu thực thể lại vừa đóng vai trò là một kiểu thực thể thực sự.

Quay trở lại ví dụ trên, với liên kết "Completes", rõ ràng chúng ta phải sử dụng hai thuộc tính là số chứng chỉ Cert_number và ngày tốt nghiệp Date_completed để mô tả cho liên kết này, và do vậy chúng ta nên hình thành một kiểu thực thể chứng chỉ CERTIFICATE cùng với hai thuộc tính đó.

Chúng ta biểu diễn kiểu thực thể này bằng một hình thoi (như một liên kết) và bên ngoài là một hình chữ nhật (như một kiểu thực thể) như hình 2.3.5.2 dưới đây:



Hình 2.3.5.2 Hình thành kiểu thực thể kết nối

Vậy thì khi nào thì chúng ta nên chuyển đổi một liên kết thành một kiểu thực thể kết nối? Dưới đây là bốn điều kiện có thể xem xét để chuyển đổi:

- Liên kết giữa các kiểu thực thể là liên kết nhiều-nhiều.
- Kiểu thực thể kết nối được chuyển đổi thường được xác định bởi một định danh đơn (gồm một thuộc tính).
- Trong liên kết có thêm một số thuộc tính mô tả, ngoài thuộc tính định danh.
- Khi hình thành lên kiểu thực thể kết nối thì kiểu thực thể này độc lập với các kiểu thực thể được nối bởi liên kết đó.

Như ví dụ trên, việc chuyển đổi liên kết “Completes” thành một kiểu thực thể kết nối CERTIFICATE cho biết mỗi một nhân viên hoàn thành một khoá học đều được cấp một chứng chỉ, và mỗi một chứng chỉ đều có một số Cert_number duy nhất, do vậy thuộc tính này sẽ trở thành định danh cho kiểu thực thể kết nối mới, cùng với thuộc tính mô tả khác như Date_completed.

Chúng ta cũng phải lưu ý rằng, khi chuyển đổi liên kết thành kiểu thực thể kết nối thì ký hiệu cho liên kết cũng thay đổi theo. Khi đó chiều liên kết nhiều xuất hiện ở hai kiểu thực thể ban đầu được chuyển thành các liên kết nhiều xuất phát từ kiểu thực thể kết nối vừa hình thành đến hai kiểu thực thể còn lại.

3.6. Các phân cấp “là-một”

Trên thực tế, đôi khi chúng ta gặp những trường hợp mà một tập thực thể A là một tập con của một tập thực thể B, nó có những tính chất chung với tập thực thể B, nhưng lại có một số tính chất của riêng nó. Khi đó chúng ta gọi kiểu thực thể B là kiểu thực thể gốc (supertype), kiểu thực thể A là kiểu thực thể con (subtype).

Ví dụ, trong bài toán quản lý doanh nghiệp, chúng ta có kiểu thực thể nhân viên EMPLOYEE, tuy nhiên trong số những thực thể thuộc kiểu EMPLOYEE này có những nhân viên làm việc hưởng lương theo hợp đồng hàng năm, có những nhân viên làm việc hưởng lương theo giờ, lại có những nhân viên làm việc theo hình thức là chuyên gia tư vấn. Họ đều là nhân viên của công ty, đều có số hiệu nhân viên, ngày bắt đầu tham gia vào làm việc tại công ty. Nhưng đối với mỗi hình thức làm việc đó, họ lại có thể

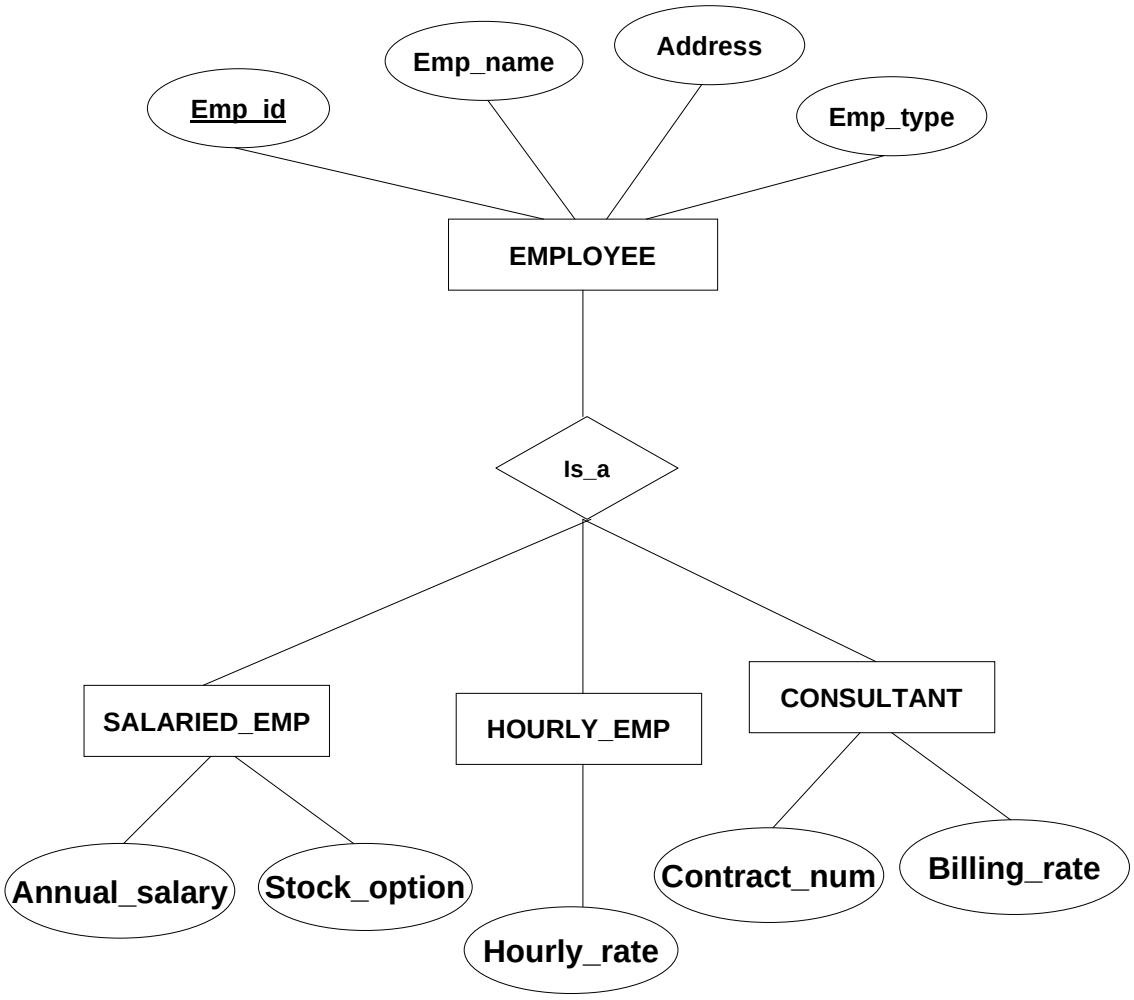
hình thành lên những kiểu thực thể mới và kết nối với kiểu thực thể EMPLOYEE theo một liên kết mà chúng ta qui ước gọi là liên kết “là-một” hay phân cấp “là-một”.

Một cách hình thức, chúng ta nói A “là-một” B nếu kiểu thực thể B là sự tổng quát hoá của kiểu thực thể A và kiểu thực thể A là sự chuyên biệt hoá của kiểu thực thể B.

Khi chúng ta hình thành liên kết “là-một” giữa hai kiểu thực thể A và B thì A sẽ kế thừa các thuộc tính của B, nhưng A cũng có thể bổ sung thêm một số thuộc tính của riêng mình mà B không có, tức là các thực thể thuộc B nhưng không thuộc A không có những thuộc tính này.

Về mặt hình thức, một thực thể a trong kiểu thực thể A có quan hệ với chính xác một thực thể b trong kiểu thực thể B, sao cho a và b thực sự là cùng một thực thể trong thế giới thực. Và chúng ta cũng cần lưu ý rằng, không có một thực thể b nào trong tập B có quan hệ với hai thực thể khác nhau của A. Các thuộc tính khoá của kiểu thực thể A thực sự cũng là các thuộc tính khoá của kiểu thực thể B, và các giá trị của những thuộc tính này đối với một thực thể a trong A được lấy từ b tương ứng trong B.

Quay trở lại ví dụ trên, một cách tốt nhất đối với việc thiết kế đó là chúng ta có thêm các kiểu thực thể mới: nhân viên hưởng lương theo hợp đồng hàng năm SALARIED_EMP, nhân viên hưởng lương theo giờ HOURLY_EMP và cộng tác viên CONSULTANT, tất cả các kiểu thực thể này đều liên kết “là-một” với kiểu thực thể EMPLOYEE như hình 2.3.6.1 sau:



Hình 2.3.6.1 Phân cấp hay liên kết "là-một"

Với kiểu thực thể SALARIED_EMP, chúng ta có thêm các thuộc tính như mức lương hàng năm Annual_salary, số cổ phần nắm giữ trong công ty Stock_option. Với kiểu thực thể HOURLY_EMP có thêm thuộc tính mức lương tính theo từng giờ Hourly_rate. Với kiểu thực thể CONSULTANT chúng ta có thêm các thuộc tính thuê theo số hợp đồng Contract_num, mức lương theo hợp đồng Billing_rate.

Một ví dụ khác: Trong công ty chúng ta có một kiểu thực thể nhân viên EMPLOYEE với các thuộc tính như Emp_id, Emp_name, Address, Start_date. Nếu công ty có một đội bóng thì rõ ràng một số nhân viên trong công ty sẽ là những cầu thủ của đội bóng, và họ có thể có thêm một số thuộc tính riêng như thể lực Physical_Strength, tốc độ Speed, vị trí Position,... mà những nhân viên khác không có. Nếu chúng ta thêm những thuộc tính này vào kiểu thực thể EMPLOYEE trên thì sẽ có rất nhiều những nhân viên khác do không tham gia vào đội bóng sẽ không có các giá trị ở những thuộc tính này. Do vậy, cách tốt nhất đối với việc thiết kế cơ sở dữ liệu là chúng ta nên có thêm một kiểu thực thể mới là BALL_PLAYER và một liên kết "là-một" kết nối đến kiểu thực thể EMPLOYEE.

3.7. Một sơ đồ thực thể liên kết

Với các hướng dẫn trên, đến đây chúng ta đã có thể tiến hành xây dựng một sơ đồ thực thể - liên kết hoàn chỉnh cho quá trình thiết kế cơ sở dữ liệu mức khái niệm.

Quay trở lại ví dụ về công ty kinh doanh đồ gỗ đã trình bày trong chủ đề 1, trong phần này chúng ta sẽ tiến hành xây dựng một sơ đồ thực thể - liên kết đầy đủ cho nó với việc áp dụng các nguyên tắc đã trình bày trong các phần trên.

Qua phân tích bài toán chúng ta thấy hình thành lên các kiểu thực thể sau:

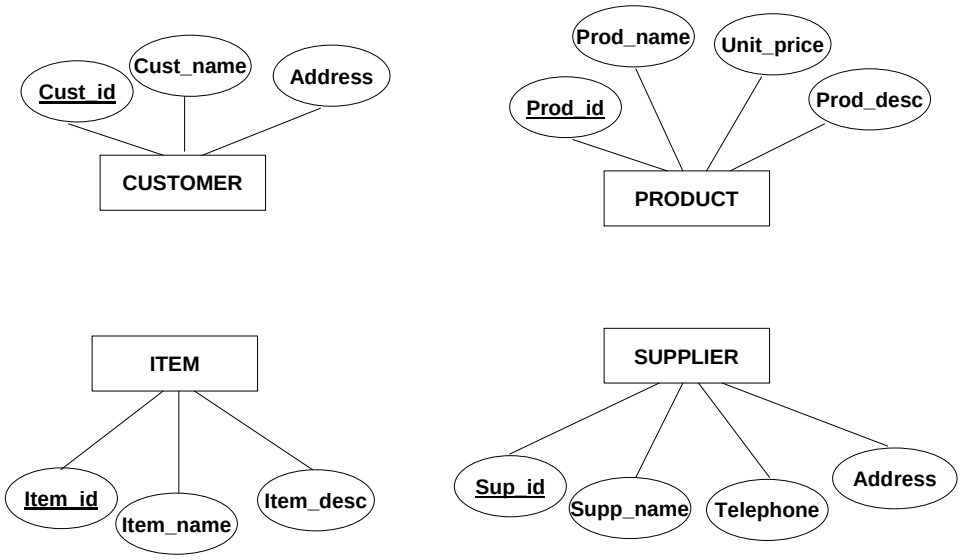
- Rõ ràng các khách hàng là những đối tác chính của công ty và công ty cũng có nhu cầu quản lý họ. Do vậy chúng ta sẽ có một kiểu thực thể đầu tiên đó là CUSTOMER, với mỗi khách hàng chúng ta cần phải

quản lý những thuộc tính sau: Tên khách hàng Cust_name, địa chỉ Address, số thẻ tín dụng Credit_card, và đối với mỗi một khách hàng chúng ta phải đặt cho họ một mã số duy nhất, chúng ta gọi là Cust_id, thuộc tính này sẽ là thuộc tính khoá của kiểu thực thể CUSTOMER.

- Công ty sản xuất và bán các sản phẩm khác nhau cho khách hàng, do vậy các sản phẩm cũng sẽ trở thành một kiểu thực thể, chúng ta đặt tên là PRODUCT. Với mỗi sản phẩm chúng ta cần quản lý tên sản phẩm Prod_name, đơn giá Unit_price, mô tả về sản phẩm Prod_desc. Cũng như kiểu thực thể CUSTOMER, với mỗi sản phẩm chúng ta cũng cần phải phân biệt chúng, do vậy chúng ta sẽ sử dụng thêm một thuộc tính Prod_id làm thuộc tính khoá cho PRODUCT.
- Để sản xuất ra các sản phẩm, công ty phải thu mua các nguyên vật liệu từ các nhà cung cấp khác nhau. Nên chúng ta sẽ có hai kiểu thực thể là nhà cung cấp SUPPLIER và nguyên vật liệu ITEM. Với kiểu thực thể SUPPLIER chúng ta cần quản lý mã nhà cung cấp Sup_id, tên nhà cung cấp Sup_name, địa chỉ Address, số điện thoại Telephone, trong đó thuộc tính Sup_id là thuộc tính khoá để phân biệt giữa các nhà cung cấp với nhau. Với kiểu thực thể ITEM chúng ta cần quản lý mã nguyên vật liệu Item_id, tên nguyên vật liệu Item_name và mô tả Item_desc, chúng ta cũng xác định thuộc tính Item_id sẽ là thuộc tính khoá của kiểu thực thể này.

Như vậy qua việc phân tích trên, chúng ta tạm thời xác định được 4 kiểu thực thể ban đầu là: CUSTOMER, PRODUCT,

SUPPLIER và ITEM cùng các thuộc tính của chúng như hình 2.3.1.7 sau:

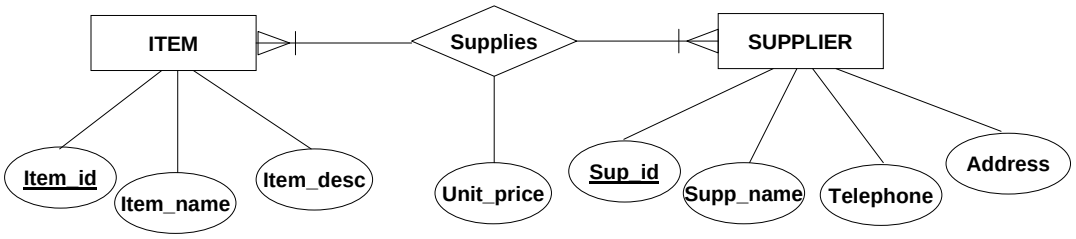


Hình 2.3.7.1 Bốn kiểu thực thể ban đầu

Chúng ta hãy tiếp tục tiến hành phân tích các mối quan hệ giữa chúng, ở đây xin lưu ý, trong quá trình phân tích để tìm ra các liên kết chũng ta vẫn có thể hình thành lên các kiểu thực thể mới như các phần trên đã trình bày.

- Để đơn giản, chúng ta hãy bắt đầu từ việc công ty mua các nguyên vật liệu từ các nhà cung cấp. Chúng ta nhận thấy rằng đây sẽ là một liên kết nhiều-nhiều, bởi mỗi một nguyên vật liệu, công ty có thể mua từ một số các nhà cung cấp khác nhau, ngược lại với mỗi một nhà cung cấp, họ có thể cung cấp cho công ty một số nguyên vật liệu. Chúng ta cũng nhận thấy rằng với liên kết nhiều-nhiều này, sẽ có một thuộc tính mô tả đi kèm, đó là thuộc tính đơn giá Unit_price do có thể cùng một nguyên vật liệu nhưng các nhà

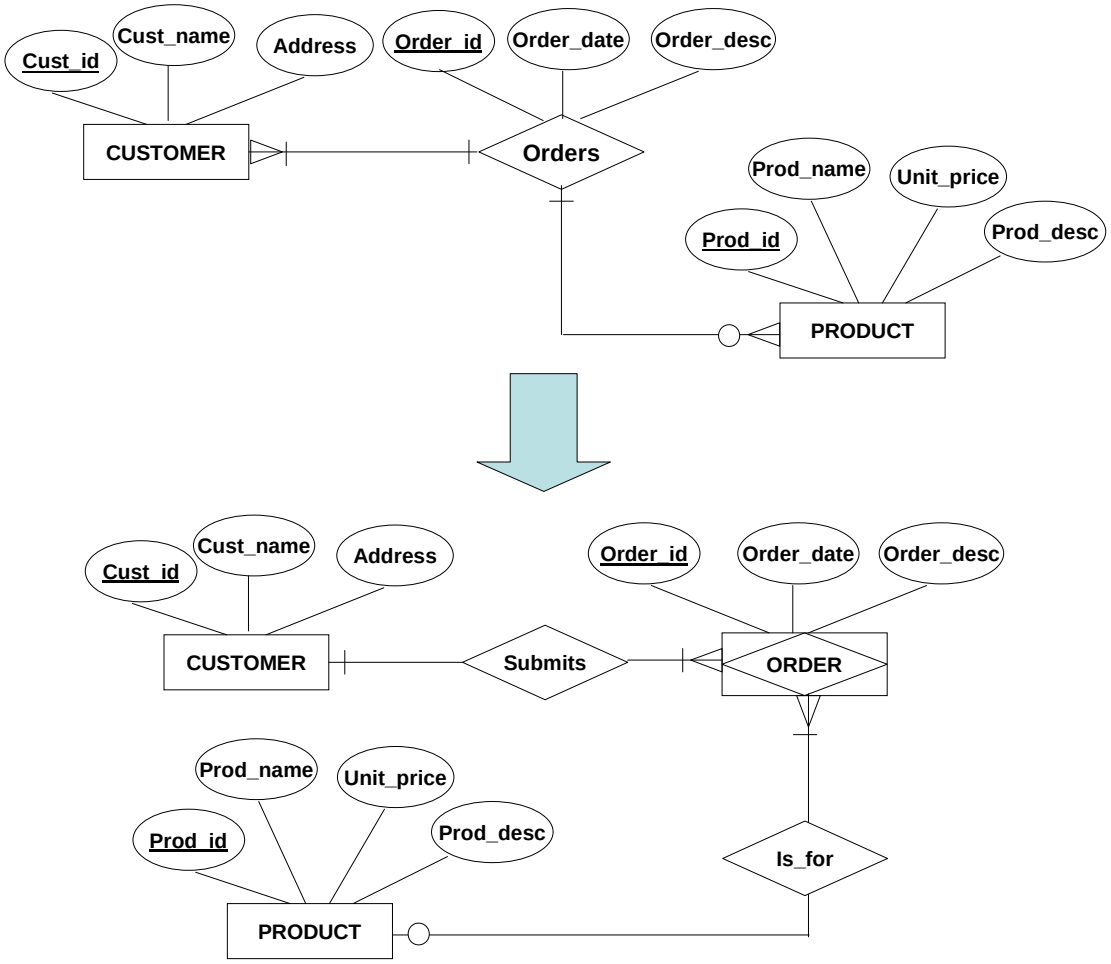
cung cấp khác nhau có thể có các giá chào bán khác nhau. Chúng ta đặt tên cho liên kết giữa hai kiểu thực thể SUPPLIER và ITEM này là cung cấp "Supplies". Về ràng buộc của liên kết, chúng ta phân tích và lựa chọn như sau: bởi công ty sẽ không quản lý những nhà cung cấp mà không bán một nguyên vật liệu nào cho công ty nên đầu liên kết với nhà cung cấp SUPPLIER sẽ là loại ràng buộc bắt buộc. Ngược lại, với các nguyên vật liệu, có thể sẽ có những thứ công ty không phải mua từ các nhà cung cấp, đó có thể là những nguyên vật liệu bán thành phẩm được tạo ra trong quá trình sản xuất ra sản phẩm cuối cùng, do vậy đầu liên kết với ITEM sẽ là loại ràng buộc tuỳ chọn. Hình 2.3.7.2 sau minh hoạ cho bước phân tích này:



Hình 2.3.7.2 Liên kết "Supplies"

- Bây giờ chúng ta hãy phân tích với một quan hệ phức tạp hơn, đó là quan hệ giữa khách hàng và công ty. Rõ ràng khách hàng muốn mua hàng của công ty thì họ phải đặt mua các sản phẩm đó. Như vậy chúng ta sẽ có một liên kết đặt mua "Orders" giữa khách hàng và sản phẩm. Chúng ta cũng nhận thấy rằng đây là một liên kết nhiều-nhiều bởi một khách hàng có thể

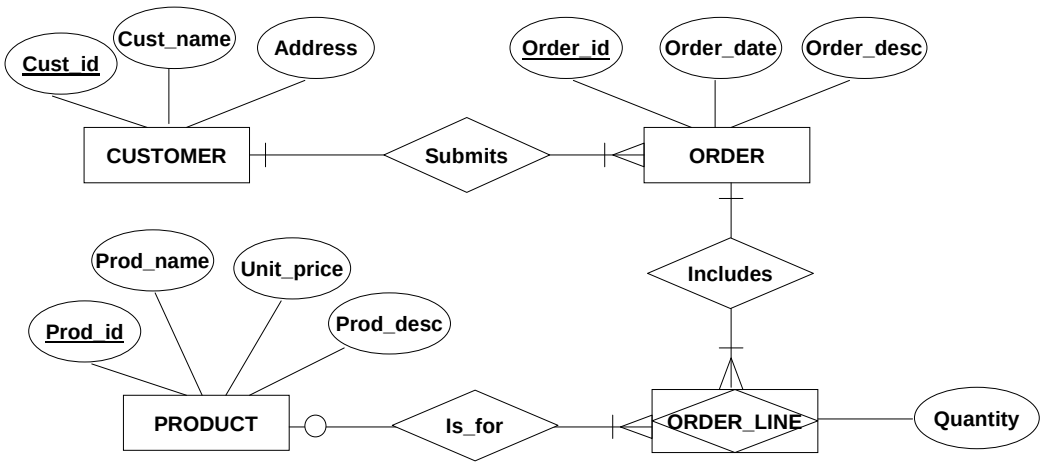
đặt mua nhiều sản phẩm cũng như một sản phẩm có thể được nhiều khách hàng khác nhau cùng mua. Cụ thể hơn, trên thực tế, đối với mỗi một giao dịch mua sản phẩm, công ty đều phải sử dụng một hoá đơn bán hàng, mỗi một hoá đơn hàng đều có một số hiệu duy nhất, cùng ngày tháng của đơn hàng. Do vậy, như trên đã trình bày, đến đây chúng ta có thể nghĩ ngay đến trường hợp chuyển đổi liên kết nhiều-nhiều này thành một kiểu thực thể kết nối, và đặt tên nó là đơn hàng ORDER như hình 2.3.7.3 sau:



Hình 2.3.7.3 Hình thành kiểu thực thể kết nối ORDER

- Phân tích tiếp nghiệp vụ này, trong mỗi hoá đơn bán hàng, khách hàng sẽ liệt kê các sản phẩm mà họ muốn đặt mua. Đi kèm với mỗi một sản phẩm sẽ là đơn giá, số lượng và thành tiền (trong đó thành tiền là thuộc tính dẫn xuất từ hai thuộc tính đơn giá và số lượng). Chúng ta thấy rằng nếu thiết kế như trên chỉ với một kiểu thực thể ORDER thì với mỗi sản phẩm mà khách hàng đặt mua, giá trị hai thuộc tính số hiệu đơn hàng và ngày đơn hàng sẽ bị lặp lại ở mỗi một

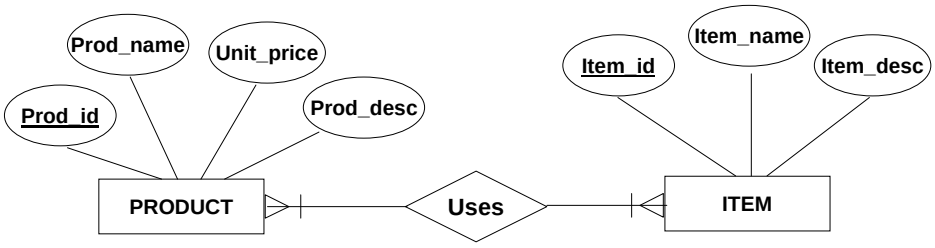
dòng đặt mua sản phẩm, mặc dù nó là duy nhất đối với mỗi một đơn hàng. Do vậy chúng ta nên tách kiểu thực thể này thành hai kiểu thực thể, một là đơn hàng gốc ORDER lưu số hiệu đơn hàng Order_id, ngày đơn hàng Order_date (có thể có thêm thuộc tính diễn giải cho đơn hàng Order_desc), và một kiểu thực thể là dòng đơn hàng ORDER_LINE. Lưu ý là trong ORDER_LINE chúng ta không hình thành thuộc tính đơn giá bởi trong kiểu thực thể PRODUCT chúng ta đã có thuộc tính này rồi, do vậy khi khách hàng lựa chọn một sản phẩm đơn giá đi theo sản phẩm đó sẽ xuất hiện, cũng như thuộc tính thành tiền là thuộc tính dẫn xuất nên chúng ta có thể lựa chọn tạo ra hoặc không cũng được. (Cũng cần lưu ý rằng, nếu giá các sản phẩm có sự thay đổi theo thời gian và công ty có nhu cầu quan tâm đến những đơn hàng đã kết thúc thì chúng ta nên tạo thêm hai thuộc tính này). Ở đây để đơn giản, chúng ta giả thiết kiểu thực thể kết nối ORDER_LINE chỉ có một thuộc tính mô tả, đó là số lượng Quantity. Mô tả bước phân tích này được thể hiện trong hình 2.3.7.4 sau:



Hình 2.3.7.4 Hình thành kiểu thực thể dòng đơn hàng ORDER_LINE

- Rõ ràng liên kết giữa ORDER và ORDER_LINE là liên kết một-nhiều như phân tích trên, còn với loại ràng buộc của liên kết thì chúng ta đều phải lựa chọn loại ràng buộc là bắt buộc ở cả hai đầu của liên kết, bởi một thực thể là đơn hàng gốc a trong kiểu thực thể ORDER luôn phải có ít nhất một thực thể b là sản phẩm đặt mua trong kiểu thực thể ORDER_LINE kết nối với nó. Ngược lại, mỗi một dòng đặt mua sản phẩm b phải thuộc một đơn hàng a. Chúng ta đặt tên liên kết này là bao gồm “Includes”.
- Liên kết giữa khách hàng và đơn hàng chúng ta đặt tên là đặt mua “Submits”, và do công ty cũng sẽ không quản lý những khách hàng mà không bao giờ đặt mua hàng của công ty nên đầu liên kết phía kiểu thực thể CUSTOMER sẽ có kiểu ràng buộc là bắt buộc. Và ngược lại, một đơn hàng cũng phải có một khách hàng đặt nên đầu liên kết phía kiểu thực thể ORDER cũng sẽ có kiểu ràng buộc là bắt buộc.

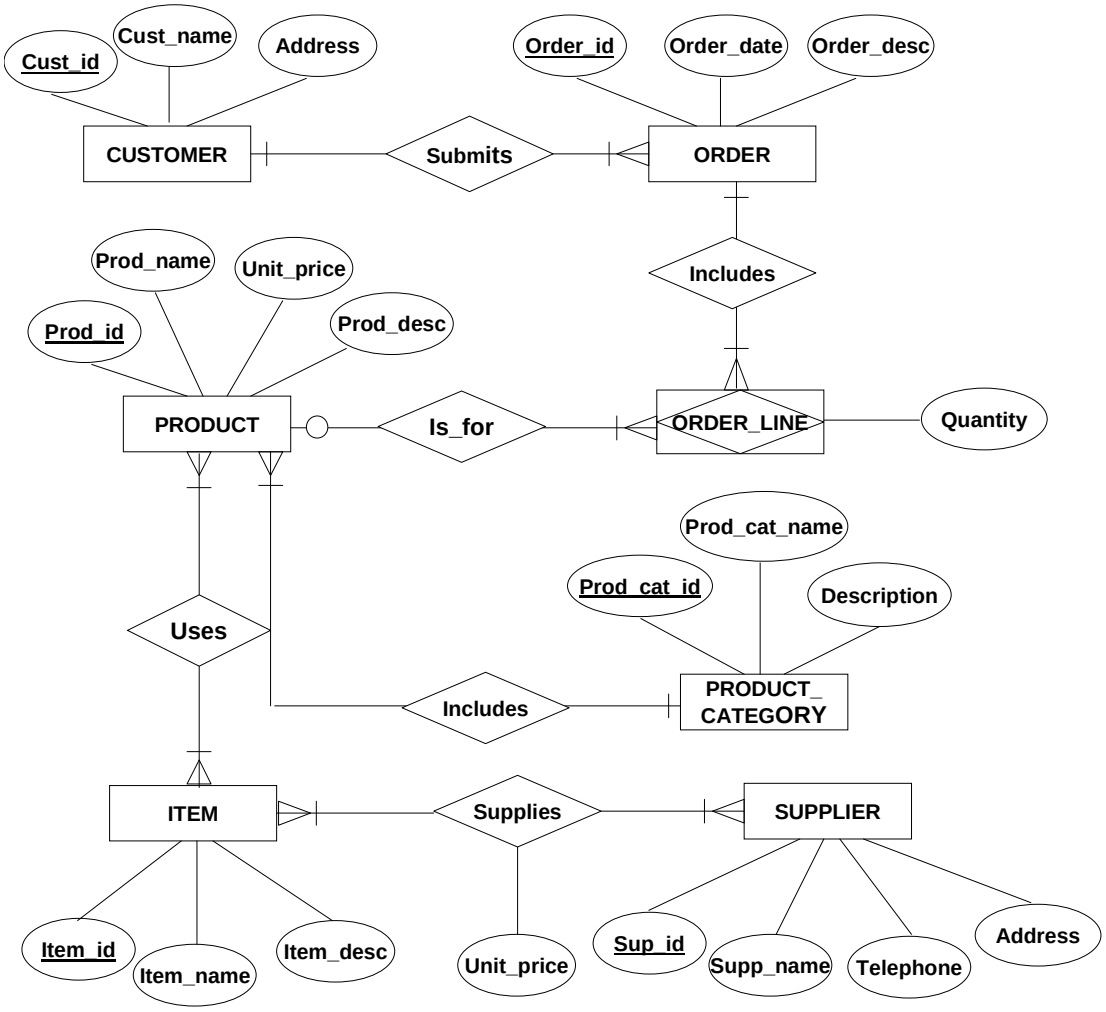
- Do chúng ta đã chuyển đổi liên kết đặt mua hàng “Orders” trên thành kiểu thực thể kết nối (ORDER và ORDER_LINE), nên liên kết “Is_for” giữa kiểu thực thể PRODUCT và dòng đơn hàng ORDER_LINE sẽ được chuyển thành liên kết một-nhiều. Rõ ràng với một dòng đơn hàng trong ORDER_LINE phải tồn tại một sản phẩm kết nối từ PRODUCT nên loại ràng buộc cho đầu liên kết với ORDER_LINE sẽ là bắt buộc. Ngược lại, ở đầu liên kết phía PRODUCT, do có thể có những sản phẩm không được đặt mua ở bất kỳ một đơn hàng nào nên loại ràng buộc sẽ là tuỳ chọn. Đến đây chúng ta đã chuyển đổi xong cho liên kết nhiều-nhiều giữa hai kiểu thực thể khách hàng CUSTOMER và sản phẩm PRODUCT thành một kiểu thực thể kết nối.
- Tiếp theo chúng ta hãy phân tích mối quan hệ giữa sản phẩm PRODUCT và nguyên vật liệu ITEM. Chúng ta nhận thấy rằng đây cũng là một liên kết nhiều-nhiều, bởi mỗi một sản phẩm có thể được làm từ nhiều nguyên vật liệu khác nhau và ngược lại, mỗi một nguyên vật liệu có thể được sử dụng trong nhiều sản phẩm khác nhau. Chúng ta đặt tên cho liên kết này là sử dụng “Uses”. Loại ràng buộc đối với hai đầu của liên kết cũng sẽ là loại ràng buộc bắt buộc, bởi mỗi một nguyên vật liệu mua về công ty đều phải sử dụng đến để tạo ra các sản phẩm, ngược lại mỗi một sản phẩm đều phải được tạo từ các nguyên vật liệu khác nhau.



Hình 2.3.7.5 Liên kết giữa PRODUCT và ITEM

- Nếu công ty có nhu cầu phân loại các sản phẩm bán ra theo từng nhóm, như các sản phẩm chuyên về bàn làm việc, các sản phẩm chuyên về giá sách, các sản phẩm chuyên về ghế ngồi,… thì chúng ta sẽ có thêm một kiểu thực thể mới đó là PRODUCT_CATEGORY, kiểu thực thể này sẽ liên kết với kiểu thực thể PRODUCT trên theo dạng liên kết một-nhiều. Do mỗi một sản phẩm phải thuộc vào một nhóm nhất định và mỗi một nhóm có thể có nhiều sản phẩm cùng loại. Chúng ta đặt tên liên kết này là bao gồm “Includes”. Hai đầu của liên kết đều là loại ràng buộc bắt buộc, bởi công ty qui định mỗi một sản phẩm phải thuộc vào một nhóm, cũng như trong một nhóm phải có ít nhất một sản phẩm.

Cuối cùng, chúng ta có một sơ đồ thực thể - liên kết đầy đủ như hình 2.3.7.6 sau:



Hình 2.3.7.6 Sơ đồ thực thể - liên kết đầy đủ

3.8. Mô tả các ràng buộc trên dữ liệu

Việc xác định và mô tả các ràng buộc trên dữ liệu sẽ được thực hiện dựa trên sơ đồ thực thể - liên kết và các luật nghiệp vụ

Trong chủ đề 1, chúng ta đã biết ràng buộc dữ liệu là những luật mà không thể bị xâm phạm bởi người sử dụng cơ sở dữ liệu. Các ràng buộc sẽ đảm bảo tính toàn vẹn và nhất quán của dữ liệu.

Ví dụ: Một sinh viên chỉ có thể đăng ký học tiếp một học phần khác nếu anh ta đã hoàn thành xong một số học phần nào đó. Hay một khách hàng sẽ được giảm giá 10% khi anh ta không bị quá hạn thanh toán. Chúng ta thấy rằng, một số giá trị của một thuộc tính nào đó sẽ phải phụ thuộc vào các giá trị của một số thuộc tính khác.

Ví dụ: Bậc lương của mỗi một nhân viên chỉ có thể nhận các giá trị từ 1 đến 7. Trong luật nghiệp vụ này, chúng ta thấy rằng các giá trị mà bậc lương chỉ có thể nhận là các giá trị số nằm giữa 1 và 7 mà không thể nhận các giá trị là ký tự.

Đối với mỗi một kiểu thực thể, thông thường sẽ có một tập các luật nghiệp vụ đi kèm, các luật nghiệp vụ này sẽ trở thành các ràng buộc toàn vẹn trong cơ sở dữ liệu. Những ràng buộc này về lý thuyết có thể có độ phức tạp bất kỳ, tuy rằng trong thực tế, nhiều ràng buộc lại rất đơn giản. Trong cơ sở dữ liệu quan hệ, các ràng buộc toàn vẹn thường được định nghĩa thông qua các câu lệnh SQL (sẽ được trình bày trong bước sau).

Tại bước này, chúng ta cần phải mô tả các giá trị mà mỗi một thuộc tính có thể nhận, hay nói một cách khác, đó chính là miền giá trị của thuộc tính để có thể đảm bảo rằng các giá trị đưa vào thuộc tính đó phải thoả mãn. Cũng như các luật nghiệp vụ mà chúng phải thoả mãn dưới dạng các tài liệu mô tả đi kèm với sơ đồ thực thể - liên kết.

Mỗi một thuộc tính trong sơ đồ thực thể - liên kết phải được khai báo miền xác định bao gồm các thông tin như sau:

- Kiểu dữ liệu: như kiểu số nguyên, số thập phân, kiểu ký tự, kiểu ngày tháng,… Hầu hết các cơ sở dữ liệu đều hỗ trợ các kiểu dữ liệu này.
- Độ dài: Đối với mỗi một kiểu dữ liệu xác định cho thuộc tính, chúng ta cần phải xác định thêm độ dài cho kiểu dữ liệu đó. Ví dụ nếu thuộc tính tuổi có kiểu số nguyên thì các giá trị cho thuộc tính đó có thể nhận giá trị tối đa là 3 chữ số. Hay nếu một thuộc tính tên có kiểu là ký tự thì giá trị của nó có thể nhận tối đa là 30 ký tự.
- Khoảng giá trị: xác định các giá trị lớn nhất và nhỏ nhất đối với các giá trị của thuộc tính, thường được áp dụng với các kiểu dữ liệu là số và ngày tháng. Như ví dụ trên, thuộc tính bậc lương sẽ được mô tả là kiểu số có độ dài là 1 và nhận các giá trị trong khoảng từ 1 đến 7.
- Hỗ trợ giá trị null (không xác định): cho biết các giá trị của một thuộc tính có được phép nhận giá trị null hay không. Ví dụ với các thuộc tính khoá thì rõ ràng các giá trị của nó không thể nhận giá trị null. Hay với thuộc tính tên, mặc dù không tham gia vào làm khoá chính nhưng chúng ta vẫn phải mô tả cho thuộc tính này không được phép nhận giá trị null.
- Các định dạng dữ liệu: mô tả dữ liệu của thuộc tính khi đưa vào cơ sở dữ liệu phải ở định dạng nào. Ví dụ như với thuộc tính có kiểu là ngày tháng, các định dạng có thể là dd/mm/yyyy hay mm/dd/yyyy. Với thuộc tính có kiểu dữ liệu là ký tự thì có cần viết hoa tất cả các ký tự hay không.

Tất cả những thông tin mô tả ràng buộc toàn vẹn cho dữ liệu này cùng với sơ đồ thực thể - liên kết sẽ là những thông tin đầu vào cho bước thiết kế cơ sở dữ liệu mức logic. Chúng sẽ được thể hiện trong các câu lệnh định nghĩa các đối tượng cho cơ sở dữ liệu.

Như vậy, mô hình thực thể - liên kết thường được biểu diễn bằng một sơ đồ thực thể - liên kết bao gồm các kiểu thực thể (trong đó mỗi đối tượng trong thế giới thực được biểu diễn bằng một thực thể), thuộc tính và liên kết.

Một kiểu thực thể là một tập các thực thể hay đối tượng có cấu trúc giống nhau. Các thực thể được mô tả bằng các thuộc tính. Mỗi một kiểu thực thể có một hoặc một tập các thuộc tính được gọi là khoá hay định danh nhằm xác định duy nhất mỗi một thực thể. Giữa các kiểu thực thể có sự kết nối với nhau thông qua các liên kết. Các liên kết cũng chính là các ràng buộc khoá giữa các kiểu thực thể

Trong phần này này, chúng ta đã xem xét các bước mô hình hoá nhằm lựa chọn giữa các kiểu đối tượng khác nhau trong sơ đồ thực thể - liên kết. Chúng ta cần phải nắm vững được những lựa chọn này để khi đứng trước các trường hợp phức tạp có thể lựa chọn được đúng đắn đối tượng cần mô hình hoá.

Kết quả của giai đọan thiết kế cơ sở dữ liệu mức khái niệm là sơ đồ thực thể - liên kết cùng với các mô tả ràng buộc dữ liệu trên các thuộc tính. Trong các hệ quản trị cơ sở dữ liệu lớn, như Oracle, SQL Server, thường tích hợp luôn các bộ công cụ cho phép chúng ta thể hiện chung cả hai kết quả đó, do vậy khi làm quen với các hệ quản trị cơ sở dữ liệu này, chúng ta cũng nên làm quen luôn với những bộ công cụ đó,

bởi chúng hoàn toàn thích hợp và thuận lợi cho giai đoạn thiết kế mà trước đây chúng ta vẫn phải làm trên giấy.

4. Thiết kế cơ sở dữ liệu mức logic

Trong bước này, chúng ta phải tiến hành lựa chọn hệ quản trị cơ sở dữ liệu để cài đặt cho cơ sở dữ liệu, chuyển đổi các thiết kế cơ sở dữ liệu mức khái niệm ở bước trên thành các lược đồ cơ sở dữ liệu của hệ quản trị cơ sở dữ liệu đã được lựa chọn.

Nhiệm vụ chính của thiết kế cơ sở dữ liệu mức logic là chuyển đổi sơ đồ thực thể liên kết đã được thực hiện ở phần thiết kế cơ sở dữ liệu mức khái niệm thành sơ đồ cơ sở dữ liệu quan hệ mức logic.

Cụ thể, chúng ta sẽ phải tập trung vào các vấn đề sau:

- Dữ liệu sẽ được biểu diễn như thế nào?
- Loại ràng buộc toàn vẹn nào sẽ được sử dụng?
- Các bước chuyển đổi từ mô hình thiết kế mức khái niệm sang mô hình thiết kế mức logic?

4.1. Giới thiệu chung về ngôn ngữ SQL

SQL (Structured Query LanguAge) là ngôn ngữ hỏi đáp dữ liệu có cấu trúc được xây dựng đầu tiên trong hệ quản trị cơ sở dữ liệu quan hệ R-System của IBM. Ngày nay, SQL trở thành một ngôn ngữ được sử dụng rộng rãi nhất trong các hệ quản trị cơ sở dữ liệu quan hệ.

Người ta sử dụng SQL để tạo lập, thao tác và truy vấn dữ liệu. Khi số lượng các nhà phát triển tham gia vào việc xây

dựng SQL tăng lên đáng kể, người ta nhận thấy rằng cần phải tiêu chuẩn hoá SQL.

Chuẩn SQL đầu tiên được xây dựng vào năm 1986 bởi Viện tiêu chuẩn quốc gia Hoa Kỳ (ANSI) và khi đó nó được gọi là chuẩn SQL-86. Có một phiên bản phụ xuất hiện vào năm 1989 và chính thức sau đó vào năm 1992 và được gọi là SQL-92. Tổ chức tiêu chuẩn quốc tế ISO đã hợp tác với ANSI để xây dựng nên chuẩn này. Hầu hết các hệ quản trị cơ sở dữ liệu thương mại ngày nay đều hỗ trợ chuẩn SQL-92. Trong SQL người ta chia ngôn ngữ (hay các câu lệnh) ra làm các loại: ngôn ngữ định nghĩa dữ liệu DDL (Data Definition LanguAge) như các câu lệnh CREATE, ALTER, DROP; ngôn ngữ thao tác dữ liệu DML (Data Manipulation LanguAge) như các câu lệnh SELECT, INSERT, UPDATE, DELETE. Các câu lệnh liên quan đến việc phân quyền dữ liệu như GRANT, REVOKE được gọi là ngôn ngữ điều khiển dữ liệu DML (Data Control LanguAge), tuy nhiên nó cũng có thể được coi là một phần của ngôn ngữ DML.

4.2. Ràng buộc toàn vẹn

Một ràng buộc toàn vẹn là một điều kiện xác định trên một lược đồ cơ sở dữ liệu nhằm giới hạn các dữ liệu có thể được lưu trữ trong quan hệ mà nó mô tả.

Cùng với việc định nghĩa một lược đồ cơ sở dữ liệu, chúng ta phải xây dựng các ràng buộc toàn vẹn để khai báo trên các thuộc tính.

Cũng cần phải lưu ý rằng, để có thể tiến hành xây dựng các khai báo cho các thuộc tính sẽ phải sử dụng đến những mô

tả dữ liệu ở các bước trên: như thuộc tính đó có là thuộc tính khoá không?, nó có thể nhận những giá trị gì? null, kiểu ký tự, kiểu số, kiểu ngày tháng..., độ dài tối đa là bao nhiêu? hệ quản trị cơ sở dữ liệu mà chúng ta đã lựa chọn có hỗ trợ các kiểu dữ liệu đó không?

Để trả lời được các câu hỏi đó, các ràng buộc toàn vẹn phải tuân theo các qui định sau:

- Sau khi một ràng buộc được định nghĩa, bất kỳ một câu lệnh thao tác dữ liệu trên bảng không thoả mãn ràng buộc đó thì câu lệnh sẽ không được thực hiện thành công và trả lại một lỗi.
- Các ràng buộc toàn vẹn được định nghĩa trên các bảng dữ liệu và được lưu như là một phần của định nghĩa bảng trong từ điển dữ liệu, do vậy các ứng dụng thao tác trên các bảng dữ liệu đều phải tuân thủ các ràng buộc toàn vẹn đó. Cũng như vậy, khi thay đổi các ràng buộc toàn vẹn, thì chúng ta chỉ cần thay đổi ở mức cơ sở dữ liệu.
- Nếu một ràng buộc toàn vẹn được tạo ra cho một bảng và trong bảng đã tồn tại một số dữ liệu không thoả mãn ràng buộc đó thì có nghĩa ràng buộc đó không thể có hiệu lực.

Những ràng buộc toàn vẹn sau thường được hỗ trợ bởi các hệ quản trị cơ sở dữ liệu phổ biến trên thị trường hiện này:

a. Ràng buộc miền

Đó là các ràng buộc về kiểu dữ liệu mà hệ quản trị cơ sở dữ liệu hỗ trợ, ví dụ như với hệ quản trị cơ sở dữ liệu Oracle,

nó hỗ trợ khá nhiều kiểu dữ liệu, một số kiểu quen thuộc như sau:

Kiểu dữ liệu	Mô tả
VARCHAR2(n)	Kiểu ký tự chứa tối đa n ký tự. Giá trị n tối đa là bằng 4000 và tối thiểu là 1
NVARCHAR2(n)	Tương tự như kiểu dữ liệu trên nhưng được xác định theo bảng mã quốc gia.
NUMBER(p,s)	Kiểu số, trong đó p là độ dài của số và s là phần thập phân. Giá trị p có thể nhận được là từ 1 đến 38
LONG	Kiểu ký tự có thể có độ dài tới 2Gigabyte
DATE	Kiểu ngày tháng
TIMESTAMP	Kiểu ngày giờ

Các ràng buộc miền là một dạng cơ bản nhất của các ràng buộc toàn vẹn. Chúng được kiểm tra một cách dễ dàng bởi hệ thống khi có một khoản mục dữ liệu mới được đưa vào cơ sở dữ liệu.

Có thể có một số thuộc tính có cùng miền giá trị. Ví dụ các thuộc tính tên khách hàng Cust_name, tên nhà cung cấp Sup_name trong các ví dụ trên có cùng một miền giá trị, chúng đều được khai báo là các xâu ký tự có độ dài 50.

b. Ràng buộc khoá

Tất cả các khái niệm khoá: khoá chính, khoá ngoại lai, khoá duy nhất có thể được khai báo bởi các lệnh của ngôn ngữ định nghĩa dữ liệu trong SQL.

- NOT NULL: Khi một trường trong quan hệ được định nghĩa ràng buộc này thì tất cả các giá trị đưa vào trường này của quan hệ đều không được mang giá trị rỗng. Tất cả các trường khoá chính đều luôn phải thoả mãn mệnh đề ràng buộc này.
- UNIQUE KEY: Ràng buộc không cho phép có 2 giá trị giống y hệt nhau trên cột hoặc tập các cột được định nghĩa bởi mệnh đề ràng buộc loại này.
- PRIMARY KEY: Ràng buộc không cho phép 2 giá trị hoàn toàn giống nhau cũng như giá trị null được đưa vào cột hoặc tập các cột được định nghĩa thoả mãn mệnh đề ràng buộc này.
- FOREIGN KEY: Ràng buộc đòi hỏi mỗi giá trị trong cột hoặc tập các cột được định nghĩa bởi ràng buộc này phải bằng một giá trị lấy từ quan hệ liên kết tương ứng, và cột trong quan hệ liên kết tương ứng cũng phải thoả mãn các ràng buộc toàn vẹn là Unique key hoặc Primary key. Ràng buộc này đôi khi còn được gọi là ràng buộc tham chiếu. Cột được khai báo loại ràng buộc này có thể nhận giá trị null nếu nó không được khai báo cùng với mệnh đề NOT NULL.
- CHECK: Ràng buộc kiểm tra điều kiện dữ liệu đưa vào phải thoả mãn một biểu thức logic được định nghĩa trong ràng buộc.

Khoá được sử dụng trong định nghĩa của một số kiểu của ràng buộc toàn vẹn. Một khoá là một cột hoặc một tập các cột được bao gồm trong định nghĩa của một số kiểu ràng buộc toàn vẹn cụ thể nào đó. Khoá mô tả mối liên kết giữa các bảng dữ liệu khác nhau và các các cột của một cơ sở dữ liệu quan hệ. Các giá trị riêng rẽ trong một khoá được gọi là giá trị khoá.

Ví dụ: Khi tạo lập quan hệ EMPLOYEE_SKILL, chúng ta định nghĩa các ràng buộc khoá chính PRIMARY KEY cho hai thuộc tính Emp_id và Skill, ràng buộc khoá ngoại lai cho thuộc tính Emp_id tham chiếu đến khoá chính trong quan hệ EMPLOYEE như câu lệnh sau:

```
CREATE TABLE EMPLOYEE_SKILL (Emp_id   char(10),
                    Skill        Number(1) NOT NULL
                    PRIMARY KEY (Emp_id,Skill),
                    FOREIGN      KEY      (Emp_id)
REFERENCES EMPLOYEE)
```

c. Các kích hoạt

Ví dụ, một luật nghiệp vụ qui định rằng, việc giảm giá cho một khách hàng không được phép vượt quá 10% giá trí mặt hàng, nếu có một thao tác thêm mới hoặc sửa đổi vượt quá giá trị này sẽ phải được phòng tránh. Hay các luật qui định một sinh viên chỉ được phép ghi danh học tiếp một môn mới ở khoá tiếp theo sau khi anh ta đã hoàn tất ít nhất 5 môn học của khoá trước. Rõ ràng, với những luật nghiệp vụ này thì những loại ràng buộc trên là rất khó đáp ứng. Hầu hết các hệ quản trị cơ sở dữ liệu ngày nay đều cung cấp một

cơ chế để định nghĩa các loại ràng buộc thao tác này thông qua các kích hoạt (trigger).

Như vậy chúng ta có thể hiểu kích hoạt là một chỉ thị được thực hiện một cách tự động bởi hệ thống như một tác động phụ của một thay đổi đối với cơ sở dữ liệu. Các kích hoạt đôi khi còn được gọi là các luật sự kiện-điều kiện-hành động ECA, chúng khác biệt với các kiểu ràng buộc được trình bày ở trên như sau:

- Các kích hoạt chỉ được kiểm tra khi các sự kiện, được đặc tả bởi người thiết kế cơ sở dữ liệu, xuất hiện. Các kiểu sự kiện được cho phép thường là thêm, xoá hay sửa đổi đối với quan hệ.
- Thay cho việc ngăn chặn ngay lập tức sự kiện vừa được phát ra, một kích hoạt kiểm tra sẽ được thực hiện, nếu điều kiện này không thoả mãn thì không một hành động nào kết hợp với kích hoạt này xảy ra đáp lại sự kiện đó. Ngược lại, nếu điều kiện của kích hoạt được thoả mãn, hành động kết hợp với kích hoạt được thực hiện bởi hệ quản trị cơ sở dữ liệu. Hành động này có thể ngăn cấm sự kiện đó hoặc thực hiện một loạt các hành động khác đi kèm với sự kiện đó.

Các kích hoạt được chia ra thành hai loại chính: loại được thực hiện trước khi thao tác trên dữ liệu xảy ra BEFORE và loại được thực hiện sau khi thao tác trên dữ liệu xảy ra AFTER.

Các thao tác trên dữ liệu mà các kích hoạt được phép can thiệp đó là các lệnh: INSERT, UPDATE, DELETE.

Cũng cần phải lưu ý rằng, mặc dù các kích hoạt cho phép chúng ta định nghĩa và áp đặt các luật toàn vẹn dữ liệu,

song chúng không giống hoàn toàn như một ràng buộc toàn vẹn, bởi các kích hoạt không thể kiểm tra được những dữ liệu đã được đưa vào bảng trước đó. Do vậy, chúng ta chỉ nên sử dụng đến các kích hoạt khi mà không thể sử dụng được các loại ràng buộc toàn vẹn trên.

4.3. Lựa chọn hệ quản trị CSDL

Việc lựa chọn hệ quản trị cơ sở dữ liệu sẽ phải được thực hiện ngay từ đầu trong giai đoạn thiết kế mức logic, bởi vì các hệ quản trị cơ sở dữ liệu ngày nay tuy đều hỗ trợ ngôn ngữ tiêu chuẩn SQL-92 nhưng đối với mỗi một hệ quản trị cơ sở dữ liệu cụ thể, sẽ có thể sử dụng một số cú pháp khác nhau.

Ngoài ra, đối với mỗi một hệ quản trị cơ sở dữ liệu khác nhau, chúng có thể sẽ hỗ trợ những kiểu dữ liệu khác nhau ngoài những kiểu cơ bản như ký tự, số, ngày tháng,..., ngay cả đối với các kiểu cơ bản này, chúng cũng sẽ có cú pháp cũng như độ dài hỗ trợ tối đa khác nhau tuỳ thuộc vào từng hệ quản trị cơ sở dữ liệu.

Ví dụ, với Visual Foxpro có hỗ trợ kiểu dữ liệu memo dùng để lưu những trường có giá trị vô cùng lớn, như một tệp văn bản chẳng hạn, trong khi đó hệ quản trị cơ sở dữ liệu Oracle lại hỗ trợ một kiểu tương tự là kiểu long có thể chứa tới 2Gigabyte dữ liệu.

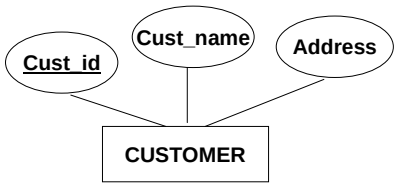
Với Oracle có hỗ trợ thêm các kiểu NVARCHAR2 để lưu các trường kiểu ký tự nhưng theo các vùng lãnh thổ khác nhau tối đa là 4000 ký tự phụ thuộc vào tập ký tự được sử dụng. Các kiểu CLOB, BLOB có thể lưu các tệp kiểu ký tự, nhị phân lên đến 4GB.

Sau khi lựa chọn xong hệ quản trị cơ sở dữ liệu chúng ta sẽ tiến hành chuyển đổi sơ đồ thực thể - liên kết sang mô hình dữ liệu quan hệ, cụ thể là các quan hệ logic sẽ được tạo ra trong bước này. Ngôn ngữ SQL sử dụng thuật ngữ bảng "TABLE" để thay cho thuật ngữ quan hệ "RELATION".

4.4. Bước một: Chuyển đổi các kiểu thực thể thông thường

Mỗi một kiểu thực thể trong sơ đồ thực thể liên kết được chuyển đổi thành một quan hệ. Tên đặt cho quan hệ có thể lấy trùng với tên đã đặt cho kiểu thực thể. Mỗi một thuộc tính của kiểu thực thể sẽ trở thành một thuộc tính (hay trường) của quan hệ. Các thuộc tính tham gia vào định danh của kiểu thực thể sẽ trở thành thuộc tính khoá chính trong quan hệ.

Hình 2.4.4.1 sau minh hoạ một ánh xạ từ kiểu thực thể CUSTOMER thành quan hệ CUSTOMER tương ứng.



CUSTOMER

Cust_id	Cust_name	Address

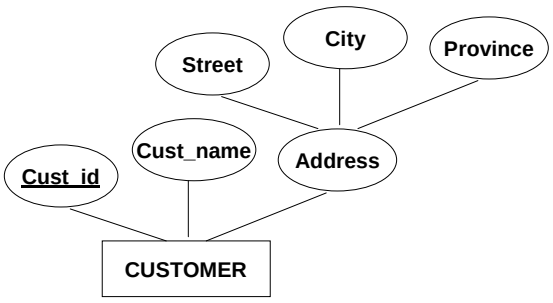
Hình 2.4.4.1 Chuyển đổi từ kiểu thực thể CUSTOMER thành quan hệ CUSTOMER

Câu lệnh SQL sau sẽ thể hiện bước chuyển đổi trên (bao gồm cả các ràng buộc về miền giá trị đối với mỗi thuộc tính và khoá):

```
CREATE TABLE CUSTOMER (Cust_id          char(10),
```

```
                    Cust_name       char(30),
                    Address     char(150)
                    PRIMARY KEY (Cust_id))
```

Nếu trong kiểu thực thể có một thuộc tính kết hợp được đại diện bằng các thuộc tính đơn, thì những thuộc tính đơn này sẽ có mặt trong quan hệ tạo thành.. Hình 2.4.4.2 minh hoạ cho trường hợp này, trong đó Address được biểu diễn như là một thuộc tính kết hợp của các thuộc tính đơn như Street, City, Province. Kiểu thực thể này được ánh xạ thành quan hệ tương ứng chỉ chứa các thuộc tính đơn.



CUSTOMER

Cust_id	Cust_name	Street	City	Province

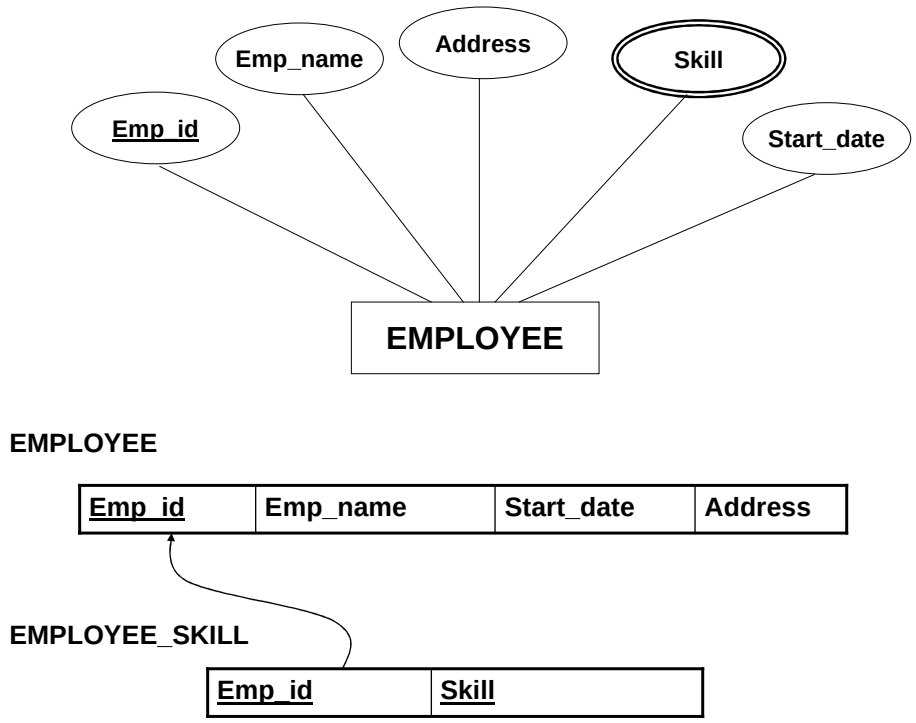
Hình 2.4.4.2 Chuyển đổi từ kiểu thực thể có thuộc tính kết hợp thành quan hệ

Câu lệnh tạo bảng như sau:

```
CREATE TABLE CUSTOMER (Cust_id          char(10),
                    Cust_name       char(30),
                    Street          char(50),
                    City        char(50),
                    Province    char(50)
                    PRIMARY KEY (Cust_id))
```

Nếu trong kiểu thực thể có chứa các thuộc tính đa trị, thì khi ánh xạ thành quan hệ, kiểu thực thể này sẽ tạo thành 2 quan hệ mới: quan hệ thứ nhất chứa tất cả các thuộc tính của kiểu thực thể ngoại trừ thuộc tính đa trị; và quan hệ thứ hai chỉ chứa hai thuộc tính, thuộc tính thứ nhất là khoá chính của quan hệ thứ nhất vừa tạo, và thuộc tính thứ hai là thuộc tính đa trị, cả hai thuộc tính này mới tạo thành khoá chính trong quan hệ thứ hai. Như vậy trong quan hệ thứ hai không tồn tại các thuộc tính mô tả mà chỉ gồm các thuộc tính tham gia vào khoá chính.

Hình 2.4.4.3 sau là một ví dụ minh hoạ cho kiểu thực thể EMPLOYEE của chúng ta. Như trong các ví dụ trước, thuộc tính Skill của kiểu thực thể này là một thuộc tính đa trị, tức là một nhân viên có thể có nhiều kỹ năng khác nhau, nhưng anh ta luôn chỉ có một mã duy nhất. Do vậy khi chuyển đổi kiểu thực thể này sang quan hệ chúng ta phải tạo thành hai quan hệ: Quan hệ thứ nhất vẫn được gọi là EMPLOYEE chứa khoá chính Emp_id cùng các thuộc tính đơn trị khác như Emp_name, và Address. Quan hệ thứ hai được đặt tên là EMPLOYEE_SKILL sẽ bao gồm khoá chính Emp_id và thuộc tính đa trị Skill, hai thuộc tính này kết hợp với nhau mới tạo thành khoá chính trong quan hệ mới. Liên kết giữa khoá chính và khoá ngoại lai của hai quan hệ được chỉ ra bởi đường mũi tên như trong hình.



Hình 2.4.4.3 Chuyển đổi từ kiểu thực thể có thuộc tính đa trị thành quan hệ

Quan hệ EMPLOYEE_SKILL không chứa các thuộc tính mô tả, tuy nhiên cũng có thể nhận thấy rằng, chúng ta có thể bổ sung thêm các thuộc tính đó vào cho quan hệ mới này như Certificate_Date để ghi nhận lại một nhân viên A đã thi đạt được kỹ năng B ngày nào, từ đó có thể suy ra anh ta đã có bao nhiêu năm kinh nghiệm với kỹ năng đó.

Câu lệnh tạo hai bảng này được mô tả như sau:

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
                      Emp_name        char(30) NOT NULL,
                      Start_date date  NOT NULL,
                      Address     char(150)
                      PRIMARY KEY (Emp_id))
```

```
CREATE TABLE EMPLOYEE_SKILL (Emp_id   char(10),
                    Skill        Number(1) NOT NULL
                    PRIMARY KEY (Emp_id,Skill),
                    FOREIGN        KEY        (Emp_id)
REFERENCES EMPLOYEE)
```

4.5. Bước hai: Chuyển đổi các kiểu thực thể yếu

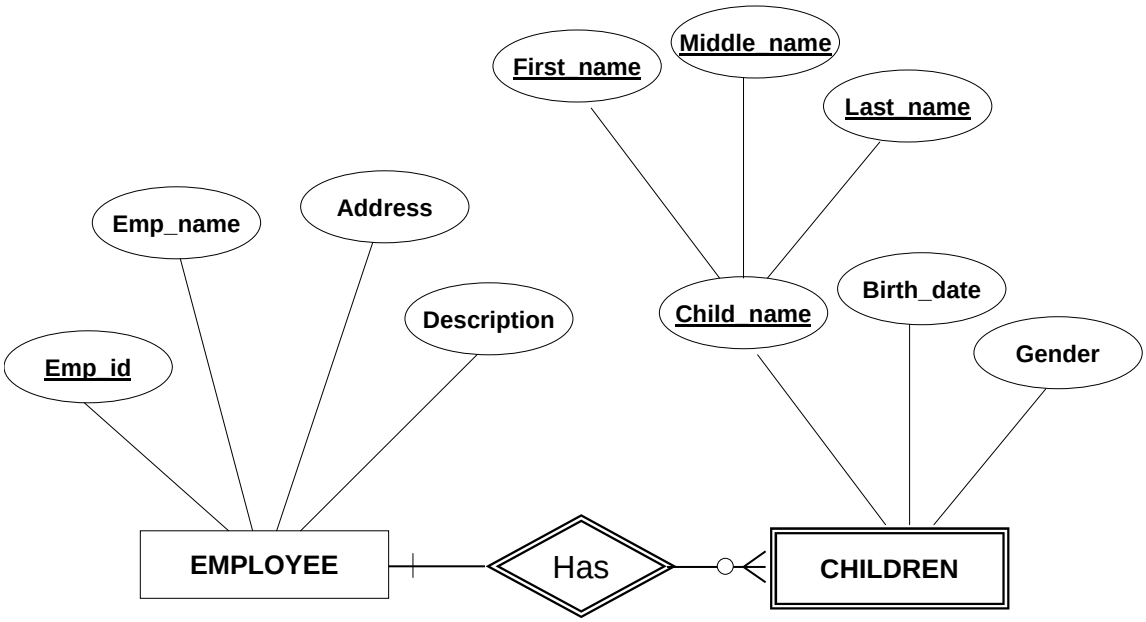
Như chúng ta đã biết, một kiểu thực thể yếu không được phép tồn tại độc lập mà luôn phụ thuộc vào một kiểu thực thể khác thông qua một liên kết định danh. Một kiểu thực thể yếu không có một định danh đầy đủ, nhưng nó cũng có một đinh danh mà chúng ta qui ước gọi là định danh bộ phận để có thể phân biệt được các thực thể trong kiểu thực thể yếu này khi biết một thực thể tương ứng trong kiểu thực thể sở hữu nó.

Khi chúng ta muốn chuyển đổi một kiểu thực thể yếu sang quan hệ, chúng ta phải chuyển đổi các kiểu thực thể sở hữu nó thành các quan hệ trước theo bước 1, sau đó mới có thể tiến hành chuyển đổi nó như sau:

Đối với mỗi kiểu thực thể yếu, chúng ta tạo một quan hệ tương ứng với nó bằng cách chuyển đổi tất cả các thuộc tính thành các thuộc tính trong quan hệ. Tất nhiên, nếu trong kiểu thực thể yếu tồn tại những thuộc tính kết hợp thì nó cũng sẽ được chuyển sang thành các thuộc tính trong quan hệ theo nguyên tắc như trong bước 1, tức là chỉ chuyển đổi các thuộc tính đơn hợp thành thuộc tính kết hợp đó. Ngoài ra, trong quan hệ mới thì khoá chính trong quan hệ sở hữu tương ứng của nó cũng sẽ phải có mặt trong quan hệ mới và trở thành các thuộc tính đóng vai trò khoá ngoại

lai. Cuối cùng, khoá chính của quan hệ mới sẽ là hợp của các khoá ngoại lai và các thuộc tính tham gia vào định danh bộ phận của kiểu thực thể yếu.

Hình 2.4.5.1 sau là một ví dụ minh hoạ cho bước thứ hai này, trong đó kiểu thực thể CHILDREN là một kiểu thực thể yếu, nó phụ thuộc vào kiểu thực thể sở hữu EMPLOYEE và kết nối thông qua liên kết “Has” (chúng ta có thể tham khảo thêm hình 2.3.1.2 trong phần trước). Ở đây cũng thấy rằng thuộc tính kết hợp Child_name bao gồm các thuộc tính đơn là First_name, Middle_name và Last_name, chúng sẽ có mặt trong quan hệ mới, đồng thời nó cũng đóng vai trò là định danh bộ phận trong kiểu thực thể yếu, nên khi chuyển sang quan hệ nó sẽ tham gia vào khoá chính.



EMPLOYEE

Emp_id	Emp_name	Address

CHILDREN

First_name	Middle_name	Last_name	Emp_id	Birth_date	Gender

Hình 2.4.5.1 Chuyển đổi từ kiểu thực thể yếu thành quan hệ

Như vậy khoá chính của quan hệ CHILDREN sẽ bao gồm 4 thuộc tính: First_name, Middle_name, Last_name và thuộc tính khoá ngoại lai Emp_ID từ quan hệ sở hữu EMPLOYEE tương ứng. Các thuộc tính ngày sinh Birth_date, giới tính Gender là các thuộc tính mô tả. Mũi tên trong hình minh hoạ cho liên kết giữa thuộc tính khoá ngoại lai với khoá chính.

Câu lệnh tạo hai bảng này được mô tả như sau:

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
```

```
                    Emp_name      char(30) NOT NULL,
                    Address    char(150)
                    PRIMARY KEY (Emp_id))
CREATE TABLE CHILDREN (First_name  char(10),
                Middle_name    char(10),
                Last_name      char(10),
                Emp_id         char(10),
                Birth_date date NOT NULL,
                Gender         char(1)
                PRIMARY KEY (First_name, Middle_name,
Last_name,Emp_id),
                FOREIGN KEY (Emp_id) REFERENCES
EMPLOYEE)
```

4.6. Bước ba: Chuyển đổi các liên kết giữa hai kiểu thực thể

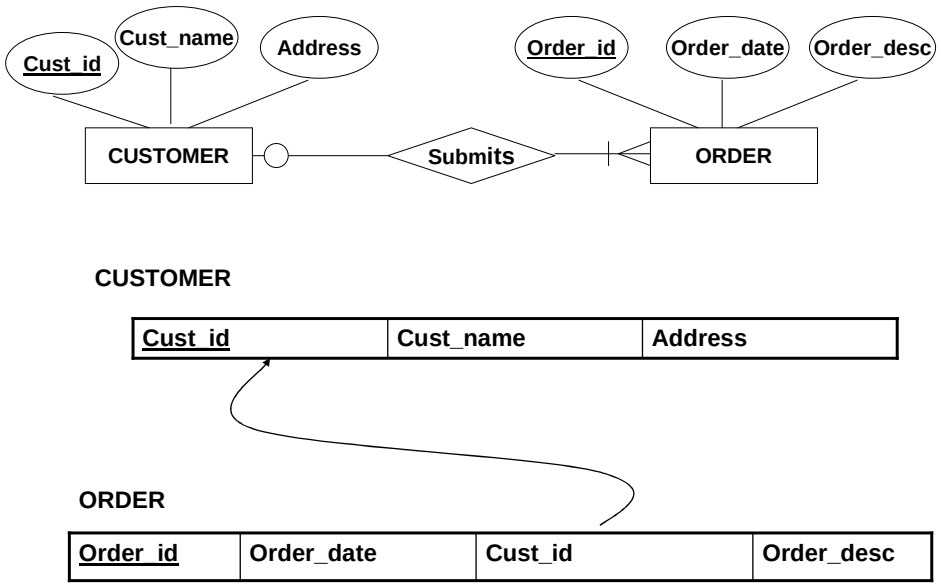
Việc chuyển đổi các liên kết này phụ thuộc vào kiểu liên kết giữa các kiểu thực thể: liên kết đơn trong cùng một kiểu thực thể, liên kết đôi giữa hai kiểu thực thể hay liên kết đa giữa nhiều kiểu thực thể; cấp độ của liên kết: liên kết một-một, một-nhiều hay liên kết nhiều-nhiều và các ràng buộc tham gia của liên kết. Chúng ta sẽ lần lượt xem xét việc chuyển đổi của từng loại một như sau:

a. Chuyển đổi các liên kết một-nhiều

Đối với loại liên kết này, đầu tiên chúng ta phải ánh xạ hai kiểu thực thể tham gia trong liên kết thành hai quan hệ

tương ứng như trong hai bước 1 và 2 trên. Tiếp theo đưa các thuộc tính khoá chính của quan hệ phía liên kết một sang thành các thuộc tính khoá ngoại lại trong quan hệ phía liên kết nhiều. Nếu đầu liên kết phía nhiều có dạng ràng buộc là bắt buộc thì các khoá ngoại lai sẽ tham gia vào làm khoá chính của quan hệ.

Ví dụ: hình 2.4.6.1 sau minh hoạ cho quá trình này, trong hình a là một liên kết một-nhiều. Hình b là kết quả của phép chuyển đổi, trong đó chúng ta thấy rằng khoá chính Cust_id trong quan hệ phía liên kết một CUSTOMER đã được chuyển sang thành thuộc tính khoá ngoại lai Cust_id trong quan hệ ORDER phía liên kết nhiều. Liên kết khoá ngoại lai giữa hai quan hệ được minh hoạ bằng đường mũi tên.



Hình 2.4.6.1 Chuyển đổi các liên kết một - nhiều

Câu lệnh tạo hai bảng này được mô tả như sau:

CREATE TABLE CUSTOMER(Cust_id char(10),

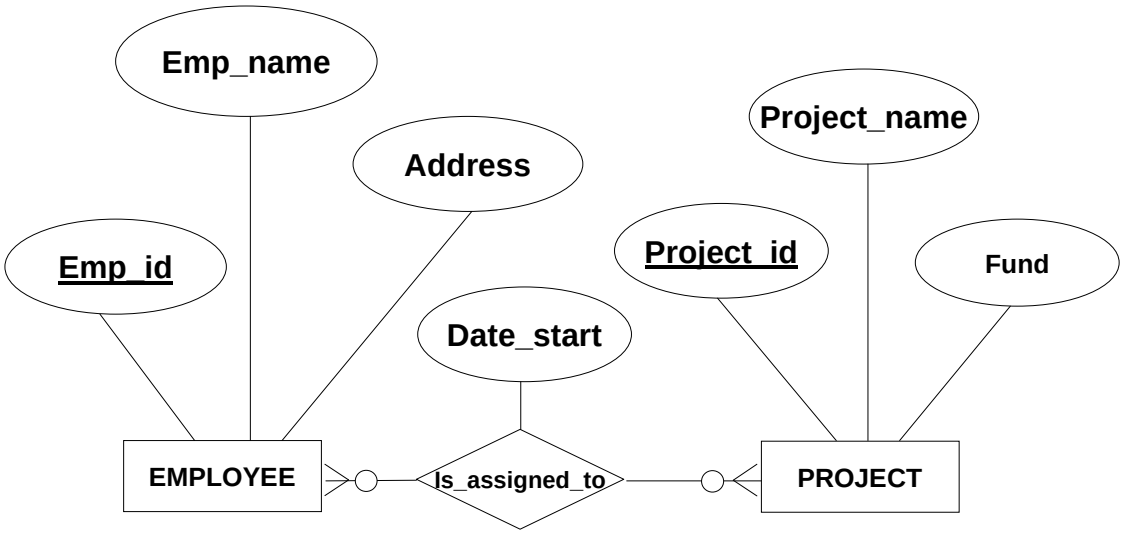
```
                    Cust_name       char(30) NOT NULL,
                    Address    char(150)
                    PRIMARY KEY (Cust_id))
CREATE TABLE ORDER(Order_id    char(10),
                Order_date      date  NOT NULL,
                Cust_id         char(10) NOT NULL,
                Order_desc      char(200)
                PRIMARY KEY (Order_id),
                FOREIGN  KEY  (Cust_id)  REFERENCES
CUSTOMER)
```

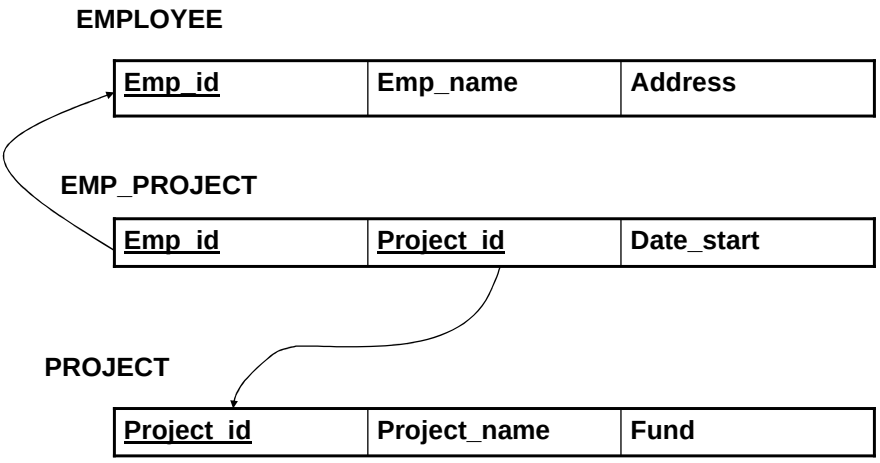
b. Chuyển đổi các liên kết nhiều-nhiều

Nếu giữa hai kiểu thực thể A và B có một liên kết kiểu nhiều-nhiều. Khi đó chúng ta phải tạo thêm một quan hệ mới C. Trong quan hệ mới C này, khoá chính của nó sẽ được tạo lập từ các khoá chính của hai quan hệ A và B. Như vậy chúng ta thấy rằng để có được quan hệ C, đầu tiên ta phải chuyển đổi được hai quan hệ A và B từ hai kiểu thực thể A và B tương ứng như trong các bước trước. Các thuộc tính mô tả cho liên kết nhiều-nhiều sẽ trở thành các thuộc tính tương ứng trong quan hệ C mới.

Ví dụ: Hình 2.4.6.2 sau là một ví dụ cho qui tắc chuyển đổi này. Liên kết “Is_assigned_to” kết nối giữa hai kiểu thực thể EMPLOYEE và PROJECT, đây là một liên kết loại nhiều-nhiều, bởi một nhân viên có thể tham gia vào nhiều dự án và một dự án có thể bao gồm nhiều nhân viên tham gia.. Thuộc tính ngày bắt đầu tham gia dự án Date_start là

thuộc tính mô tả cho liên kết này. Do đó khi chuyển đổi sang các quan hệ, hai kiểu thực thể EMPLOYEE và PROJECT được chuyển đổi đầu tiên, tiếp đó liên kết "Is_assigned_to" được chuyển đổi thành một quan hệ mới mà chúng ta đặt tên là EMP_PROJECT, trong quan hệ này chúng ta thấy rằng các thuộc tính khoá chính Emp_id và Project_id trong hai quan hệ EMPLOYEE và PROJECT tương ứng đã có mặt trong quan hệ mới và tham gia vào làm khoá chính trong quan hệ EMP_PROJECT này, còn thuộc tính Date_start trở thành thuộc tính mô tả trong quan hệ mới. Các liên kết của khoá ngoại lai được minh hoạ bằng các mũi tên giữa các quan hệ.





Hình 2.4.6.2 Chuyển đổi các liên kết nhiều - nhiều

Câu lệnh tạo các bảng này được mô tả như sau:

```
CREATE TABLE EMPLOYEE(Emp_id        char(10),
                      Emp_name      char(30) NOT NULL,
                      Address       char(150)
                      PRIMARY KEY (Emp_id))
CREATE TABLE PROJECT (Project_id    char(10),
                      Project_name  char(100) NOT NULL,
```

```
                    Fund        number(10)
                    PRIMARY KEY (Project_id))
CREATE TABLE EMP_PROJECT(Emp_id char(10),
                    Project_id  char(10),
                    Date_start date
                PRIMARY KEY (Emp_id, Project_id),
                FOREIGN KEY (Emp_id) REFERENCES
EMPLOYEE
                FOREIGN KEY (Project_id) REFERENCES
            PROJECT)
```

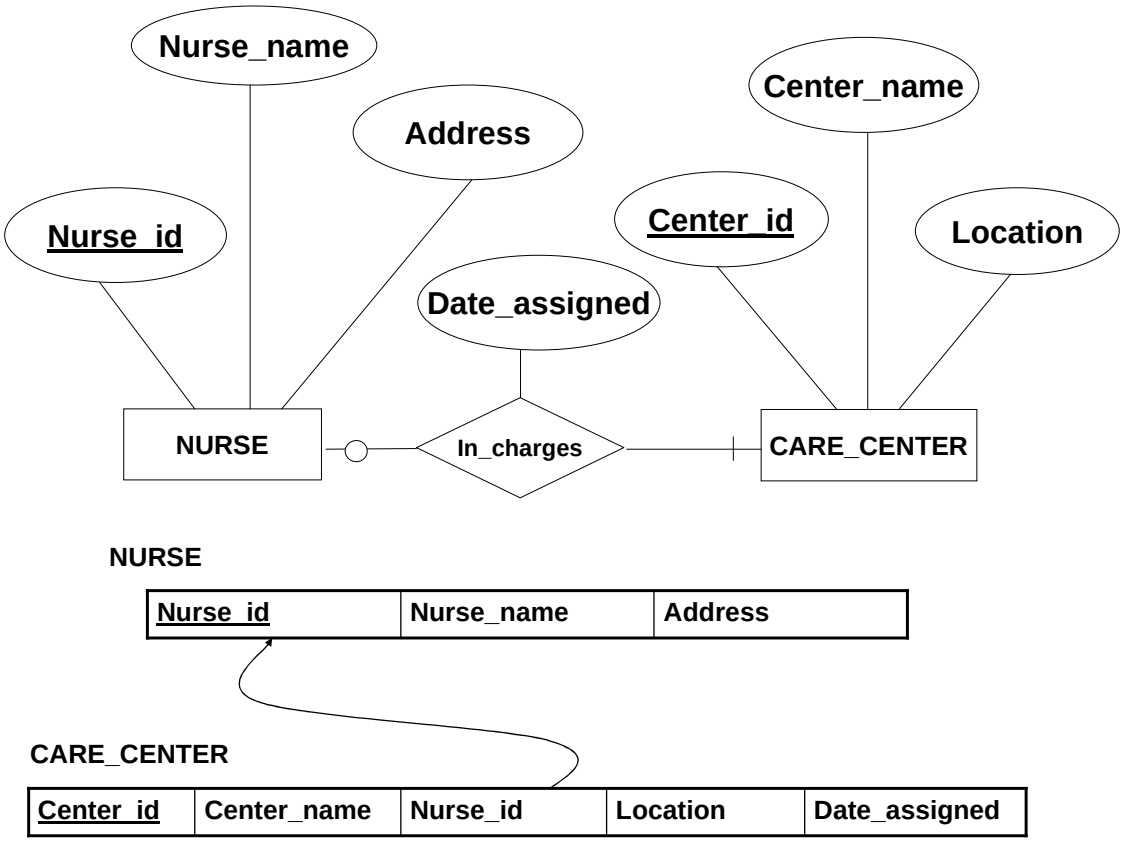
c. Chuyển đổi các liên kết một-một

Liên kết một-một có thể xem như là một trường hợp đặc biệt của liên kết một-nhiều. Quá trình chuyển đổi liên kết này qua hai giai đoạn: Đầu tiên hai quan hệ được chuyển đổi từ hai kiểu thực thể tương ứng tham gia vào liên kết. Tiếp theo khoá chính trong một quan hệ sẽ được ánh xạ thành khoá ngoại lai trong quan hệ còn lại.

Trong liên kết một-một, một đầu của liên kết luôn luôn là loại tuỳ chọn (optional), đầu còn lại luôn luôn là bắt buộc (mandatory). Do vậy khi chuyển đổi các kiểu thực thể thành các quan hệ thì kiểu thực thể ở phía liên kết bắt buộc sẽ trở thành quan hệ trong đó có khoá ngoại lai lấy từ khoá chính của quan hệ ánh xạ từ kiểu thực thể phía liên kết tuỳ chọn. Như vậy chúng ta thấy rằng, sẽ tránh được trường hợp phải lưu giá trị rỗng trong các thuộc tính khoá ngoại lai. Các thuộc tính mô tả còn lại cũng sẽ có mặt trong các quan

hệ tương ứng như các qui tắc trên. Ngoài ra nếu trong liên kết có các thuộc tính mô tả cho liên kết đó thì nó cũng sẽ trở thành các thuộc tính mô tả cho quan hệ phía liên kết tuỳ chọn.

Ví dụ minh hoạ: Hình 4.6.3 minh hoạ cho bước chuyển đổi này, giữa hai kiểu thực thể NURSE và CARE_CENTER tồn tại một liên kết một-một “In_charge”. Mỗi một trung tâm chăm sóc phải có một y tá đảm nhiệm. Do đó kết nối từ CARE_CENTER đến NURSE phải là kết nối bắt buộc, tức là không thể có một trung tâm chăm sóc nào mà không có một y tá đảm nhận. Nhưng ngược lại, kết nối từ NURSE đến CARE_CENTER lại là kết nối tuỳ chọn, bởi có thể có những y tá không đảm nhận một trung tâm chăm sóc nào. Thuộc tính Date_assigned là thuộc tính mô tả cho liên kết giữa hai kiểu thực thể này nó ghi lại ngày mà y tá bắt đầu đảm nhận trung tâm chăm sóc.



Hình 2.4.6.3 Chuyển đổi các liên kết một - một

Kết quả của phép chuyển đổi liên kết một-một này cho ta thấy rằng hai quan hệ NURSE và CARE_CENTER đã được tạo ra từ hai kiểu thực thể tương ứng. Quan hệ NURSE được hình thành như trong bước 1, còn đối với quan hệ CARE_CENTER, do tham gia vào liên kết với tư cách là một kết nối bắt buộc, do vậy khóa ngoại lai từ quan hệ NURSE sẽ có mặt trong quan hệ này, ngoài ra thuộc tính Date_assigned mô tả cho liên kết cũng sẽ được bổ sung vào quan hệ.

Câu lệnh tạo bảng như sau:

```
CREATE TABLE NURSE  (Nurse_id  char(10),
                     Nurse_name    char(100) NOT NULL,
```

```
                    Address     char(200)
                    PRIMARY KEY (Nurse_id))
CREATE TABLE CARE_CENTER(Center_id     char(10),
                    Center_name    char(100) NOT NULL,
                    Nurse_id   char(10) NOT NULL,
                    Location    char(200)
                    Date_assigned  date
                PRIMARY KEY (Center_id),
                FOREIGN KEY (Nurse_id) REFERENCES
NURSE)
```

4.7. Bước bốn: Chuyển đổi các kiểu thực thể kết nối

Như đã trình bày trong phần 3, khi tiến hành mô hình hoá dữ liệu với sơ đồ thực thể - liên kết, nếu chúng ta gặp phải một liên kết nhiều-nhiều, chúng ta có thể lựa chọn để mô hình hoá liên kết đó thành một kiểu thực thể kết nối. Cách tiếp cận này cũng giúp người sử dụng khi tham gia vào quá trình thiết kế dễ hình dung hơn. Việc chuyển đổi kiểu thực thể kết nối này cũng bao gồm các các khâu chuyển đổi tương tự như chuyển đổi liên kết nhiều-nhiều ở bước 3 nêu trên. Cụ thể các khâu chuyển đổi như sau:

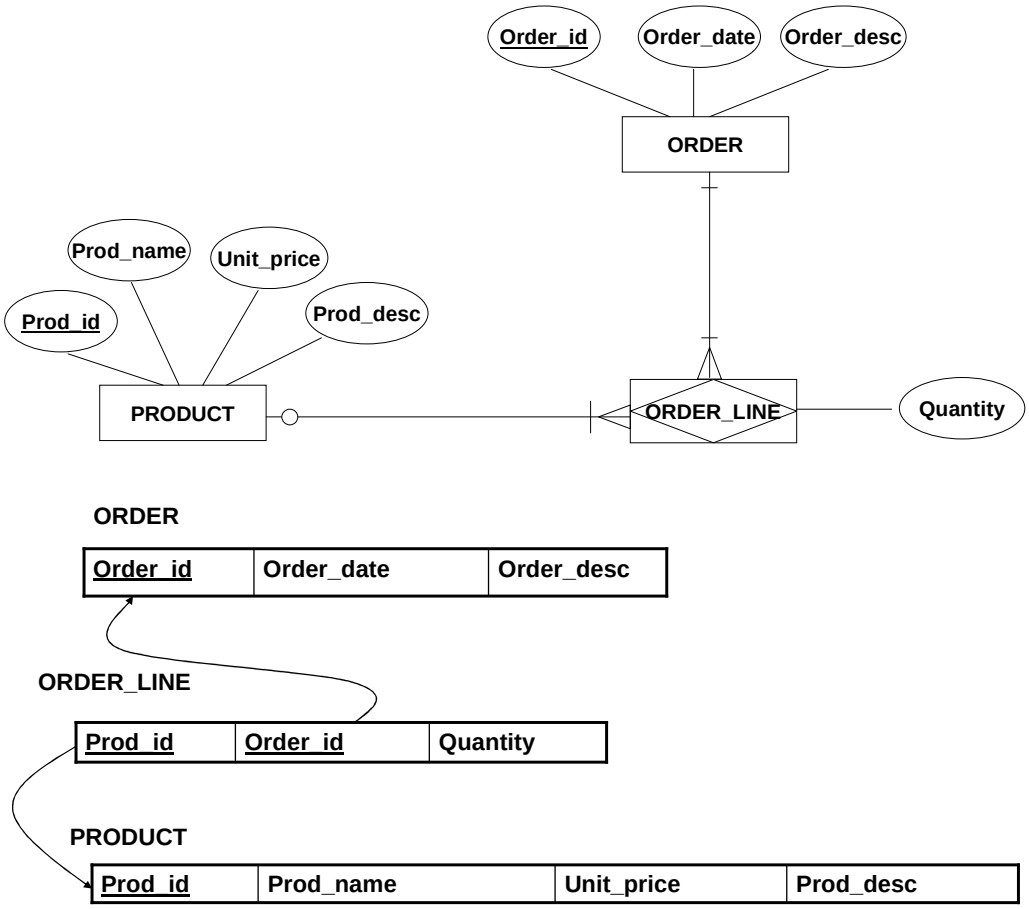
Đầu tiên chúng ta tạo ra 3 quan hệ: hai quan hệ đầu tương ứng với hai kiểu thực thể tham gia, quan hệ thứ ba được tạo nên từ kiểu thực thể kết nối, và chúng ta gọi nó là quan hệ kết nối. Ngoài ra, bước chuyển đổi còn phụ thuộc vào việc

trong kiểu thực thể kết nối có các thuộc tính định danh hay không. Để tiện theo dõi, chúng ta chia ra hai trường hợp:

a. Trường hợp thuộc tính định danh không xác định

Khi các thuộc tính định danh không được gán, khoá chính mặc định cho quan hệ kết nối sẽ bao gồm các thuộc tính tham gia vào làm khoá chính của hai quan hệ còn lại. Do đó mỗi khoá sẽ là một khoá ngoại lai trong quan hệ mới kết nối đến hai quan hệ còn lại.

Ví dụ, Hình 2.4.7.1 là một ví dụ minh hoạ cho bước chuyển đổi này. Kiểu thực thể kết nối ORDER_LINE liên kết hai kiểu thực thể ORDER và PRODUCT. Chúng ta có thể nhận thấy rằng nó tương tự như ví dụ trong trường hợp của liên kết nhiều-nhiều trong hình 2.4.6.2.



Hình 2.4.7.1 Chuyển đổi kiểu thực thể kết nối với thuộc tính định danh không xác định

Câu lệnh tạo bảng này được mô tả như sau:

```
CREATE TABLE ORDER(Order_id        char(10),
                   Order_date      date NOT NULL,
                   Order_desc      char(150)
                   PRIMARY KEY (Order_id))
CREATE TABLE PRODUCT(Prod_id       char(10),
                   Prod_name       char(30) NOT NULL,
                   Unit_price number(10) NOT NULL,
                   Prod_descchar(10)
                   PRIMARY KEY (Prod_id))
```

```
CREATE TABLE ORDER_LINE(Order_id  char(10),
                        Prod_id          char(10),
                        Quantity    number(10) NOT NULL
                        PRIMARY KEY (Order_id,Prod_id),
                    FOREIGN KEY (Order_id) REFERENCES
ORDER,
                    FOREIGN KEY (Prod_id) REFERENCES
ORDER)
```

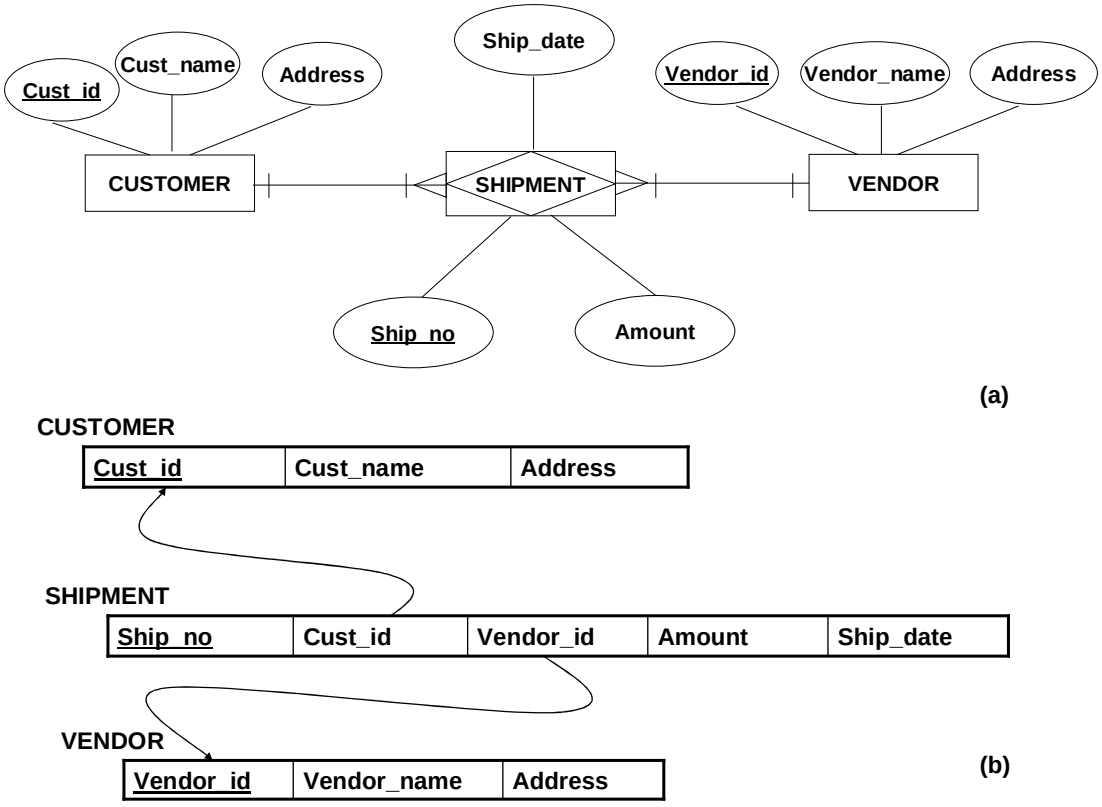
Hãy lưu ý đến thứ tự các câu lệnh tạo bảng, đầu tiên hai quan hệ ORDER và PRODUCT phải được tạo trước, tiếp theo mới đến quan hệ ORDER_LINE.

b. Trường hợp thuộc tính định danh đã xác định

Đôi khi trong quá trình mô hình hoá dữ liệu với sơ đồ thực thể liên kết, chúng ta đã xác định được định danh cho kiểu thực thể kết nối (chúng được gọi là khoá hay định danh đại diện). Có hai lý do khiến chúng ta có thể xác định được định danh này:

- Khi định danh dễ dàng xác định và phù hợp với bài toán thực tế. Ví dụ số hiệu đơn hàng luôn là duy nhất nên nó là định danh cho kiểu thực thể kết nối đơn hàng (nối hai kiểu thực thể là khách hàng và đại lý cung cấp).
- Khi định danh mặc định (xác định được bằng việc kết hợp hai định danh của hai kiểu thực thể được kết nối) không thể xác định duy nhất cho các thể hiện của kiểu thực thể kết nối.

Khi đó quá trình chuyển đổi kiểu thực thể kết nối sẽ tiến hành như sau: một quan hệ kết nối vẫn phải được tạo ra tương ứng với kiểu thực thể kết nối đó (như trường hợp nêu trên), tuy nhiên khoá chính của quan hệ kết nối mới bây giờ là khoá đại diện. Các khoá chính trong hai quan hệ còn lại vẫn có mặt trong quan hệ kết nối mới nhưng nó chỉ đóng vai trò là các khoá ngoại lai mà không tham gia vào làm khóa chính.



Hình 2.4.7.2 Chuyển đổi kiểu thực thể kết nối với thuộc tính định danh xác định

Hình 2.4.7.2 trên là ví dụ minh hoạ cho bước chuyển đổi trong trường hợp khoá định danh đã xác định. Trong hình a chúng ta thấy rằng kiểu thực thể kết nối SHIPMENT liên kết hai kiểu thực thể CUSTOMER và VENDOR. Trong

kiểu thực thể kết nối này chúng ta đã có thuộc tính Ship_no đóng vai trò định danh bởi hai lý do:

- Thứ nhất số hiệu hoá đơn vận chuyển Ship_no là có thực và người sử dụng đã quen với giá trị này, nó không thể mang các giá trị trùng nhau.
- Thứ hai, định danh mặc định ban đầu cho quan hệ kết nối mới sẽ là hai thuộc tính Cust_id và Vendor_id không thể xác định duy nhất cho một hoá đơn vận chuyển, bởi đối với mỗi một khách hàng và với mỗi một đại lý có thể có rất nhiều các hoá đơn vận chuyển, do vậy chúng ta không thể sử dụng kết hợp của hai thuộc tính định danh này thành khoá chính cho quan hệ kết nối mới.

Trên thực tế, với mỗi một nhà vận chuyển có rất nhiều hoá đơn vận chuyển đến của cùng một khách hàng, thậm chí nếu chúng ta cho rằng có thể sử dụng thêm thuộc tính ngày vận chuyển Ship_date vào cùng tham gia làm khoá chính thì cũng không thể thoả mãn, bởi trong cùng một ngày vẫn có thể phát sinh nhiều hoá đơn đến của cùng một khách hàng từ cùng một nhà đại lý. Tuy nhiên nếu sử dụng khoá đại diện Ship_no thì luôn xác định duy nhất một hoá đơn vận chuyển. Hai thuộc tính mô tả Ship_date và Amount vẫn sẽ xuất hiện trong quan hệ kết nối mới.

Kết quả của phép chuyển đổi này được thể hiện trong hình 2.4.7.2, quan hệ kết nối mới được đặt tên là SHIPMENT. Khoá chính của quan hệ này là Ship_no, hai thuộc tính Cust_id và Vendor_id đóng vai trò là khoá ngoại lai liên kết với hai quan hệ còn lại.

Câu lệnh mô tả:

```
CREATE TABLE CUSTOMER(Cust_id        char(10),
                      Cust_name      char(30) NOT NULL,
                      Address        char(150)
                      PRIMARY KEY (Cust_id))
CREATE TABLE VENDOR(Vendor_id        char(10),
                    Vendor_name      char(30) NOT NULL,
                    Address          char(150)
                    PRIMARY KEY (Vendor_id))
CREATE TABLE SHIPMENT(Ship_no        char(10),
                      Cust_id        char(10) NOT NULL,
                      Vendor_id char(10) NOT NULL,
                      Amount         number(10)    NOT
                      NULL,
                      Ship_date date NOT NULL
                  PRIMARY KEY (Ship_no),
                  FOREIGN KEY (Cust_id) REFERENCES
CUSTOMER,
                  FOREIGN KEY (Vendor_id) REFERENCES
VENDOR)
```

4.8. Bước năm: Chuyển đổi các liên kết đơn

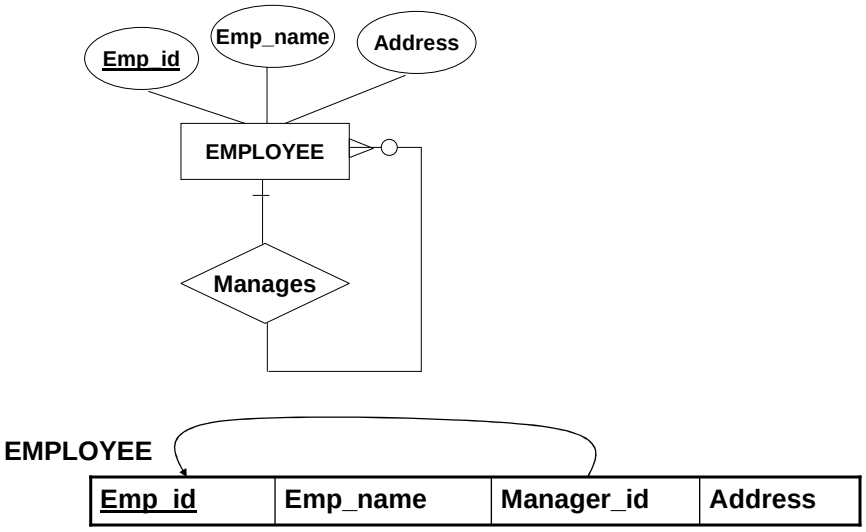
Như chúng ta đã biết liên kết đơn là liên kết giữa hai thực thể trong cùng một kiểu thực thể. Các liên kết đơn đôi khi còn được gọi là các liên kết đệ qui. Hai trường hợp quan trọng nhất của kiểu liên kết này đó là liên kết đơn một-nhiều và liên kết đơn nhiều-nhiều. Chúng ta sẽ xem xét hai

trường hợp này riêng rẽ bởi cách tiếp cận để chuyển đổi chúng là khá khác nhau:

a. Chuyển đổi các liên kết đơn một-nhiều

Trước hết, kiểu thực thể trong liên kết đơn vẫn được chuyển đổi thành một quan hệ như trong bước 1. Sau đó một thuộc tính đóng vai trò là khoá ngoại lai sẽ được đưa thêm vào quan hệ và sẽ liên kết với các khoá chính, do đó miền giá trị cho khoá ngoại lai này cũng sẽ là miền giá trị của khoá chính và trên thực tế, nó sẽ được lấy từ tập các giá trị trong khoá chính. Khoá ngoại lai này đôi khi còn được gọi là khoá ngoại lai đệ qui bởi nó tham chiếu đến chính khoá chính trong cùng quan hệ.

Ví dụ: Hình 2.4.8.1 dưới đây minh hoạ cho một liên kết đơn một-nhiều có tên là “Manages”, liên kết này cho biết một nhân viên có thể là người quản lý của một số nhân viên khác nhưng họ đều là nhân viên của công ty. Mỗi một nhân viên có duy nhất một người quản lý trực tiếp. Quan hệ EMPLOYEE được chuyển đổi tương ứng từ kiểu thực thể EMPLOYEE và liên kết đơn một-nhiều được cho trong hình b. Khoá ngoại lai đệ qui trong quan hệ mới được đặt tên là Manage_id. Miền xác định của khoá này cũng chính là miền xác định của khoá chính Emp_id. Mỗi một bản ghi của quan hệ này sẽ lưu dữ liệu về một nhân viên bao gồm: mã hiệu nhân viên Emp_id, tên nhân viên Emp_name, ngày sinh Birth_date, và mã hiệu người quản lý trực tiếp của nhân viên đó Manager_id. Lưu ý rằng vì là khoá ngoại lai nên Manager_id sẽ tham chiếu trực tiếp đến Emp_id, và chúng ta thấy rằng khoá chính của quan hệ mới vẫn là Emp_id.



Hình 2.4.8.1 Chuyển đổi các liên kết đơn một - nhiều

```
CREATE TABLE EMPLOYEE(Emp_id          char(10),
                      Emp_name        char(30) NOT NULL,
                      Manager_id      char(10),
                      Address         char(150)
                     PRIMARY KEY (Emp_id),
                     FOREIGN KEY (Manager_id)
REFERENCES EMPLOYEE)
```

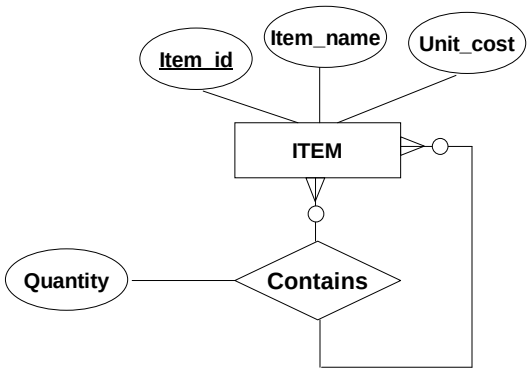
Trong thực tế chúng ta gặp liên kết đơn kiểu một-nhiều này rất nhiều. Ví dụ, để mô tả một kiểu thực thể địa bàn hành chính, chúng ta phải sử dụng liên kết này, bởi mỗi một xã sẽ thuộc một huyện nhất định, tiếp đến mỗi một huyện sẽ

phải thuộc một tỉnh thành nhất định và cuối cùng mỗi một tỉnh thành lại thuộc một vùng nhất định.

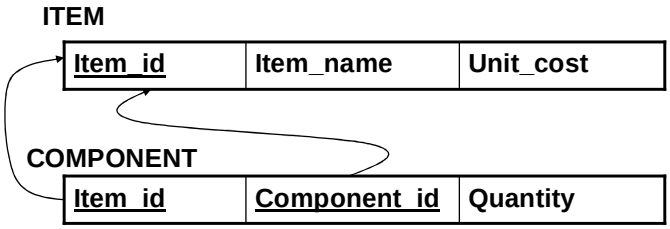
b. Chuyển đổi các liên kết đơn nhiều-nhiều

Với kiểu liên kết này, sẽ có đến hai quan hệ sẽ được tạo ra: một quan hệ được tạo ra tương ứng với chính kiểu thực thể hiện thời, và một quan hệ khác được sinh ra từ mối liên kết nhiều-nhiều và được gọi là quan hệ kết nối. Khoá chính của quan hệ kết nối sẽ bao gồm hai thuộc tính, giá trị trong cả hai thuộc tính này đều lấy chung từ các giá trị trong thuộc tính khoá chính của quan hệ còn lại. Mọi thuộc tính kết nối của liên kết đều được đặt trong quan hệ kết nối mới.

Ví dụ minh hoạ: Một liên kết đơn kiểu nhiều-nhiều được thể hiện trong hình 2.4.8.2. Mỗi một bộ phận có thể được bao gồm từ các bộ phận khác, và ngược lại một bộ phận có thể tham gia như một thành phần trong nhiều bộ phận khác, do đó liên kết đơn “Contains” này là một liên kết nhiều-nhiều. Các quan hệ ánh xạ từ kiểu thực thể cùng với liên kết đơn này được chỉ ra trong phần b của hình 2.4.8.2. Quan hệ ITEM được ánh xạ trực tiếp từ kiểu thực thể ITEM. Còn quan hệ COMPONENT là một quan hệ kết nối với khoá chính bao gồm hai thuộc tính là Item_id và Component_id. Thuộc tính mô tả liên kết số lượng Quantity cũng sẽ xuất hiện trong quan hệ kết nối này, nó cho biết số lượng của các thành phần Component_id được sử dụng trong một bộ phận Item_no. Lưu ý rằng cả hai thuộc tính Item_no và Component_id đều tham chiếu đến khoá chính Item_no của quan hệ ITEM.



(a)



(b)

Hình 2.4.8.2 Chuyển đổi các liên kết đơn nhiều - nhiều

```
CREATE TABLE ITEM(Item_id        char(10),
                  Item_name      char(30) NOT NULL,
                  Unit_cost      number(10)
                  PRIMARY KEY (Item_id))
CREATE TABLE COMPONENT(Item_id   char(10),
                       Component_id   char(10),
                       Quantity   number(5)
                  PRIMARY KEY (Item_id, Component_id),
                  FOREIGN        KEY        (Component_id)
REFERENCES ITEM)
```

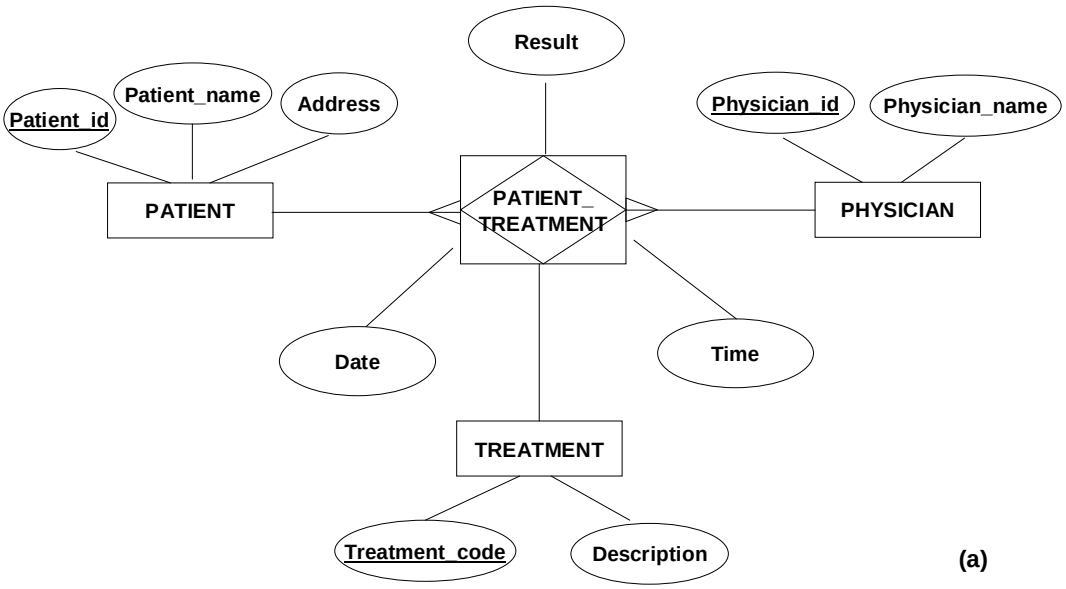
4.9. Bước sáu: Chuyển đổi các liên kết giữa nhiều kiểu thực thể

Trong phần thiết kế cơ sở dữ liệu mức khái niệm, chúng ta đã đề cập đến các liên kết giữa nhiều kiểu thực thể (liên kết đa nguyên), đó là loại liên kết kết nối ít nhất giữa 3 kiểu thực thể. Ở đây, chúng ta nên chuyển đổi loại liên kết này thành kiểu thực thể kết nối để có thể biểu diễn các ràng buộc bộ phận một cách chính xác hơn.

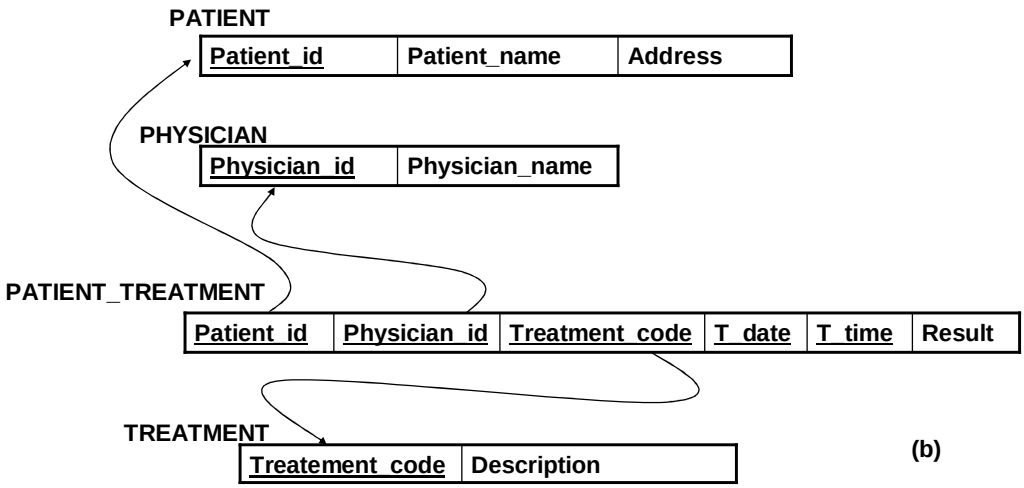
Giả thiết có 3 kiểu thực thể tham gia vào liên kết với kiểu thực thể kết nối. Khi đó, trước tiên chúng ta phải tiến hành chuyển đổi 3 kiểu thực thể đó thành các quan hệ tương ứng như trong các bước trên. Tiếp theo, khi chuyển đổi kiểu thực thể kết nối thành quan hệ, khoá chính mặc định cho nó sẽ là kết hợp của tất cả các thuộc tính tham gia vào làm khoá chính trong 3 quan hệ đã tạo trên. Tuy nhiên, có thể trong một số trường hợp chúng ta vẫn cần phải bổ sung thêm thuộc tính để hình thành lên một khoá chính duy nhất. Những thuộc tính này cũng đóng vai trò là khoá ngoại lai tham chiếu tương ứng đến các thuộc tính khoá chính trong ba quan hệ còn lại. Mỗi một thuộc tính trong kiểu thực thể kết nối sẽ trở thành một thuộc tính mô tả trong quan hệ kết nối mới. Để dễ hiểu hơn, chúng ta hãy xem ví dụ sau:

Ví dụ minh hoạ: Một kiểu thực thể kết nối liên kết với ba kiểu thực thể được thể hiện trong hình 2.4.9.1. Kiểu thực thể này mô tả cho mối quan hệ giữa một bệnh nhân nhận được một phương pháp điều trị từ một bác sỹ, chúng ta đặt tên cho kiểu thực thể này là PATIENT_TREATMENT, gồm các thuộc tính: ngày điều trị T_date, giờ điều trị T_time, và kết quả điều trị Result.

Kết quả của phép chuyển đổi các kiểu thực thể này được thể hiện trong phần b của hình 2.4.9.1. Các thuộc tính khoá chính Patient_id, Physician_id và Treatment_code trở thành các khoá ngoại lai trong quan hệ PATIENT_TREATMENT. Ở đây chúng ta nhận thấy rằng, chúng không xác định duy nhất một thể hiện trong quan hệ kết nối mới, bởi một bệnh nhân có thể nhận được cùng một phương pháp điều trị từ cùng một bác sỹ nhưng ở các thời điểm khác nhau. Do vậy trong quan hệ kết nối PATIENT_TREATMENT mới, chúng ta phải kết hợp thêm các thuộc tính thời gian vào để có thể xác định duy nhất một thể hiện trong quan hệ. Kết quả của phép ánh xạ này cũng được thể hiện trong hình 2.4.9.1.



(a)



(b)

Hình 2.4.9.1 Chuyển đổi các liên kết giữa nhiều kiểu thực thể

```
CREATE TABLE PATIENT(Patient_id      char(10),
                     Patient_name    char(30) NOT NULL,
                     Address     char(150)
                     PRIMARY KEY (Patient_id))
CREATE TABLE PHYSICIAN(Physiciant_id      char(10),
                     Physiciant_name      char(30) NOT NULL
```

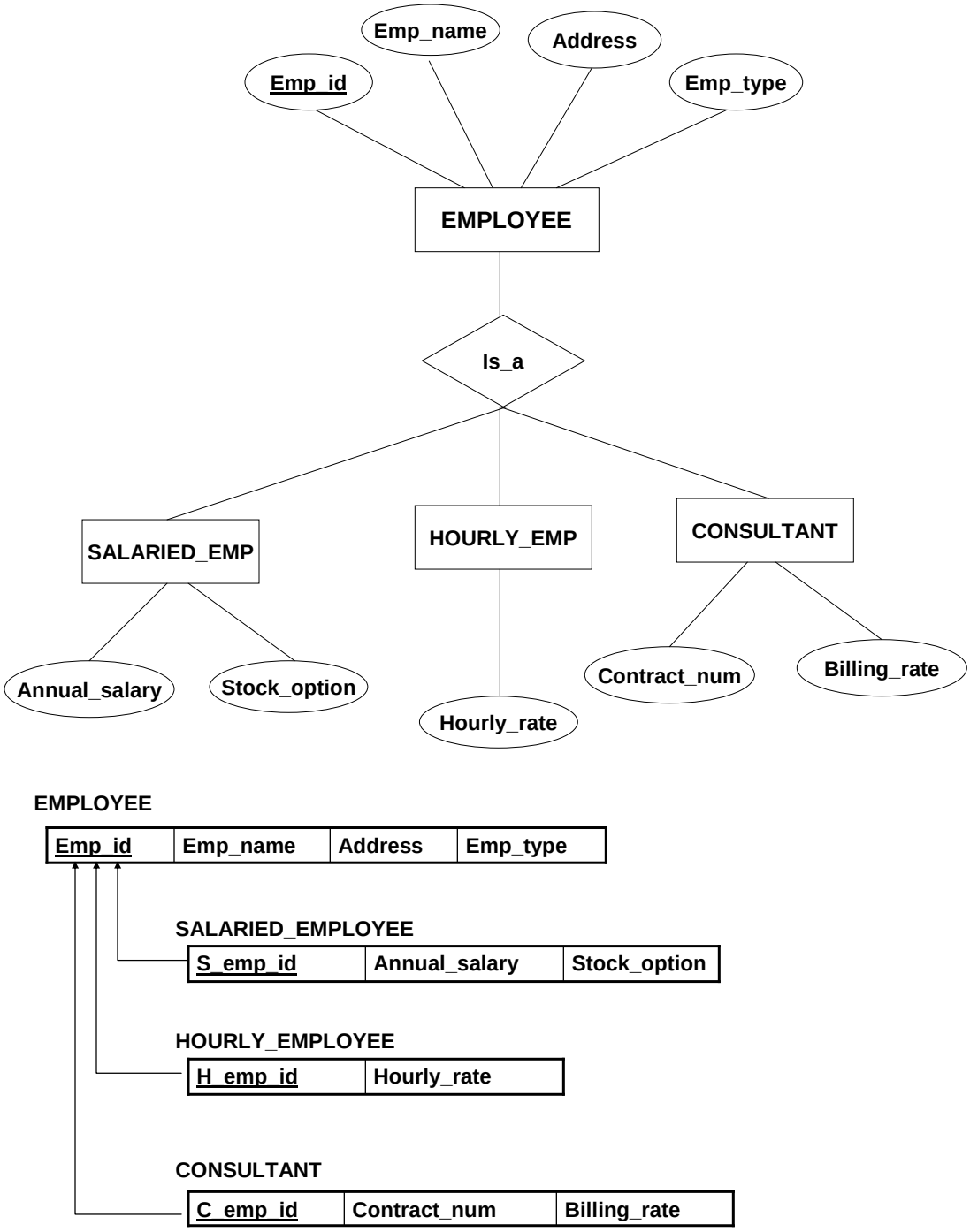
```
                PRIMARY KEY (Physician_id))
CREATE TABLE TREATMENT(Patient_id char(10),
                Physiciant_id       char(10),
                Treatment_code      char(10),
                T_date          date,
                T_time          time,
               Result                  char(300) NOT NULL
PRIMARY KEY (Patient_id, Physician_id,Treatment_code,
T_date, T_time),
FOREIGN KEY (Patient_id) REFERENCES PATIENT),
FOREIGN KEY (Physician_id) REFERENCES
PHYSICIAN),
FOREIGN KEY (Treatment_code) REFERENCES
TREATMENT))
```

4.10. Bước bảy: Chuyển đổi các phân cấp "là một"

Trong mô hình dữ liệu quan hệ không hỗ trợ trực tiếp các phân cấp "là-một" hay liên kết "là-một". Tuy nhiên, chúng ta có thể sử dụng một số chiến lược để có thể biểu diễn được kiểu liên kết này. Một trong số những chiến lược hay được sử dụng được tiến hành như sau:

- Tạo mỗi quan hệ tách biệt cho kiểu thực thể gốc (supertype) và cho mỗi kiểu thực thể con của nó.
- Với kiểu thực thể gốc, quan hệ được tạo ra như cách áp dụng cách tạo quan hệ ở các bước trên.

- Với các kiểu thực thể con, các thuộc tính khoá chính trong mỗi quan hệ con được lấy từ tập thuộc tính khoá chính trong quan hệ gốc.
- Các thuộc tính còn lại trong mỗi kiểu thực thể con sẽ trở thành các thuộc tính mô tả tương ứng trong các quan hệ con.



Hình 2.4.10.1 Chuyển đổi các liên kết "là một"

Ví dụ minh hoạ: Hình 2.4.10.1 biểu diễn một kiểu thực thể supertype EMPLOYEE cùng với các kiểu thực thể subtype của nó là SALARIED_EMPLOYEE, HOURLY_EMPLOYEE và CONSULTANT. Khoá chính

của kiểu thực thể EMPLOYEE là Emp_id và thuộc tính Emp_type được gọi là thuộc tính phân biệt cho các kiểu thực thể con.

Kết quả của phép chuyển đổi các kiểu thực thể này cũng được thể hiện trong hình 2.4.10.1. Một quan hệ được tạo ra tương ứng với kiểu thực thể EMPLOYEE. Đối với mỗi kiểu thực thể subtype còn lại cũng sẽ hình thành các quan hệ tương ứng. Khoá chính của tất cả các quan hệ này đều là Emp_id. Tiền tố phía trước trong mỗi thuộc tính khoá chính trong mỗi quan hệ con chỉ mang tính chất phân biệt. Mỗi một thuộc tính khoá chính này sẽ đóng cả vai trò là khoá ngoại lai tham chiếu đến khoá chính trong quan hệ EMPLOYEE, chúng được thể hiện bằng các đường mũi tên trong hình. Các thuộc tính mô tả còn lại trong kiểu thực thể con trở thành các thuộc tính mô tả trong quan hệ tương ứng.

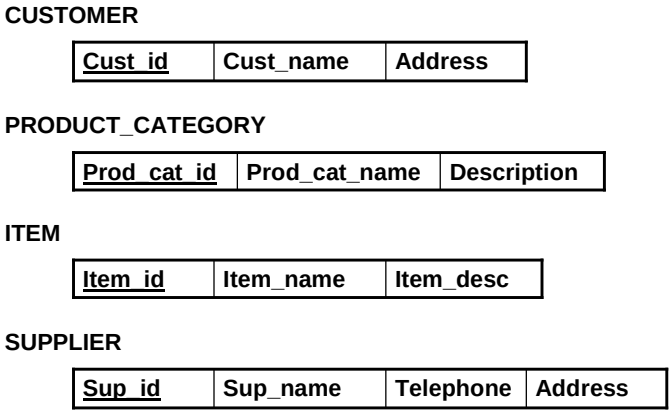
```
CREATE TABLE EMPLOYEE(Emp_id      char(10),
                      Emp_name    char(30) NOT NULL,
                      Emp_type    char(30) NOT NULL,
                      Address     char(150)
                      PRIMARY KEY (Emp_id))
CREATE TABLE SALARIED_EMPLOYEE(S_emp_id
    char(10),
                      Annual_salary   number(7) NOT NULL,
                      Stock_option    char(100)
                      PRIMARY KEY (S_emp_id),
                      FOREIGN KEY (S_emp_id) REFERENCES
                  EMPLOYEE))
```

```
CREATE TABLE HOURLY_EMPLOYEE(H_emp_id
    char(10),
            Hourly_rate     number(5) NOT NULL
            PRIMARY KEY (H_emp_id),
            FOREIGN KEY (H_emp_id) REFERENCES
        EMPLOYEE))
CREATE TABLE CONSULTANT(C_emp_id    char(10),
            Contract_num    char(10) NOT NULL,
            Billing_rate    number(8) NOT NULL
            PRIMARY KEY (C_emp_id),
            FOREIGN KEY (C_emp_id) REFERENCES
        EMPLOYEE))
```

4.11. Một sơ đồ cơ sở dữ liệu mức logic

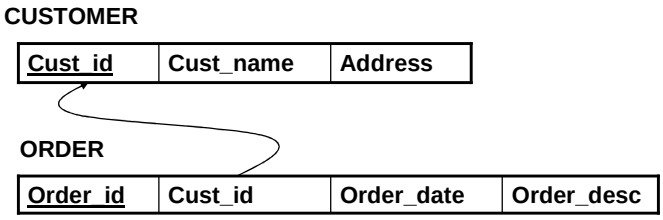
Trong phần này chúng ta tiếp tục xem xét việc chuyển đổi sơ đồ thực thể - liên kết trong phần 3 thành sơ đồ dữ liệu mức logic:

Như chúng ta thấy, các kiểu thực thể CUSTOMER, SUPPLIER, ITEM, PRODUCT_CATEGORY là những kiểu thực thể thông thường nên sẽ được chuyển đổi thành những quan hệ tương ứng theo nguyên tắc đã trình bày: Mỗi một thuộc tính của kiểu thực thể sẽ trở thành một trường trong quan hệ tương ứng. Các định danh của kiểu thực thể sẽ trở thành khoá chính trong quan hệ như trong hình 2.4.11.1 sau:



Hình 2.4.11.1 Các quan hệ chuyển đổi từ các kiểu thực thể thông thường

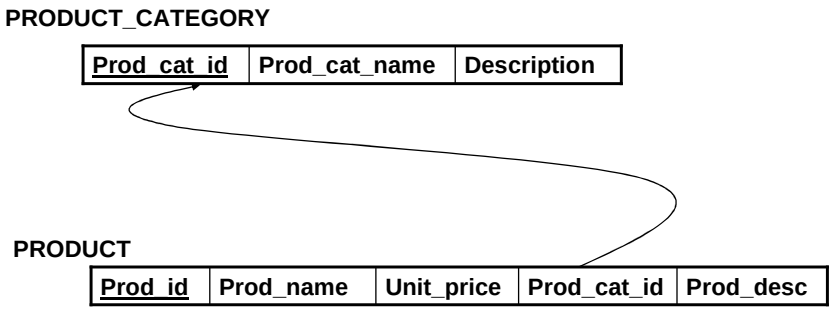
Tiếp theo chúng ta hãy xét đến kiểu thực thể ORDER. Do kiểu thực thể này có liên kết một - nhiều với kiểu thực thể CUSTOMER nên khoá chính trong quan hệ CUSTOMER sẽ được chuyển thành khoá ngoại lai trong quan hệ ORDER. Thuộc tính Order_id vẫn sẽ trở thành khoá chính trong quan hệ ORDER. Do vậy trong quan hệ ORDER chúng ta sẽ có thêm một thuộc tính khoá ngoại lai Cust_id tham chiếu đến thuộc tính Cust_id của quan hệ CUSTOMER. Kết quả của bước chuyển đổi này được thể hiện trong hình 2.4.11.2:



Hình 2.4.11.2 Các quan hệ chuyển đổi từ các kiểu thực thể CUSTOMER và ORDER

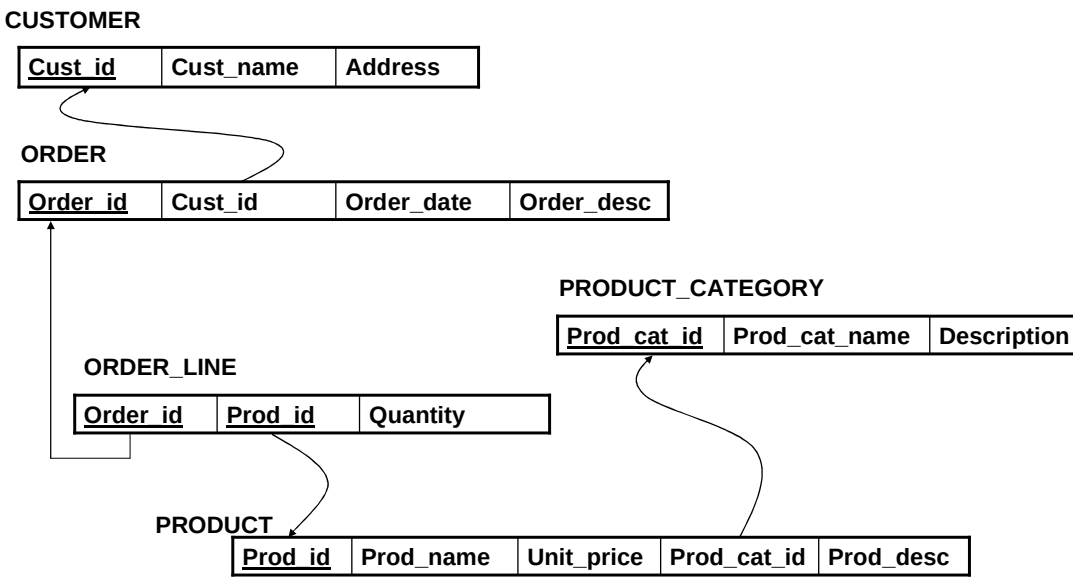
Với kiểu thực thể PRODUCT, do có liên kết một nhiều với kiểu thực thể PRODUCT_CATEGORY, nên khi chuyển

đổi thành quan hệ cho kiểu thực thể này, cũng như trường hợp trên, thuộc tính Prod_cat_id sẽ xuất hiện thêm và trở thành khoá ngoại lai tham chiếu đến khoá chính Prod_cat_id trong quan hệ PRODUCT_CATEGORY như hình 2.4.11.3 sau:



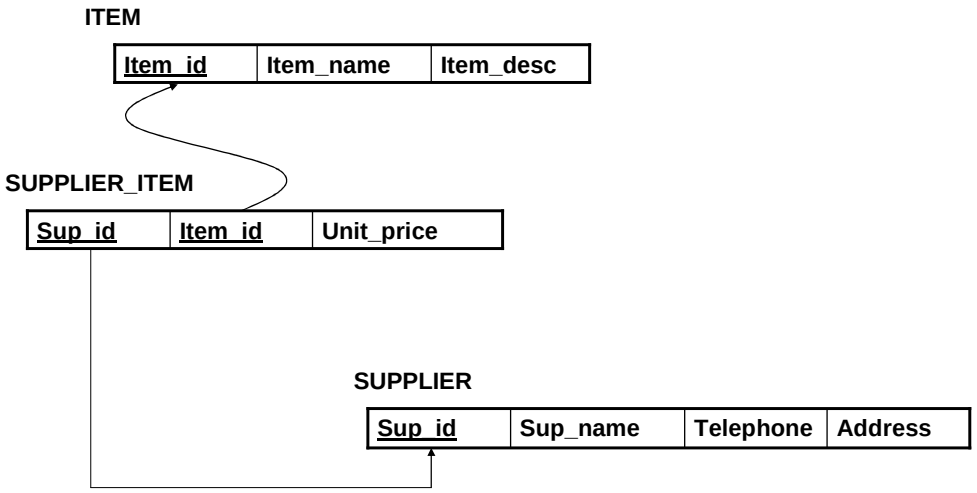
Hình 2.4.11.3 Các quan hệ chuyển đổi từ kiểu thực thể PRODUCT_CATEGORY và PRODUCT

Với kiểu thực thể liên kết ORDER_LINE, do có kết nối hai kiểu thực thể là ORDER và PRODUCT theo liên kết một-nhiều nên khi hình thành quan hệ, khoá chính của kiểu thực thể này sẽ được hình thành từ các khoá chính của hai quan hệ ORDER và PRODUCT, Order_id tham chiếu đến khoá chính của quan hệ ORDER và Prod_id tham chiếu đến khoá chính của quan hệ PRODUCT. Thuộc tính mô tả cho liên kết Quantity cũng sẽ trở thành một thuộc tính trong quan hệ mới. Hình 2.4.11.4 sau minh hoạ cho bước chuyển đổi này:



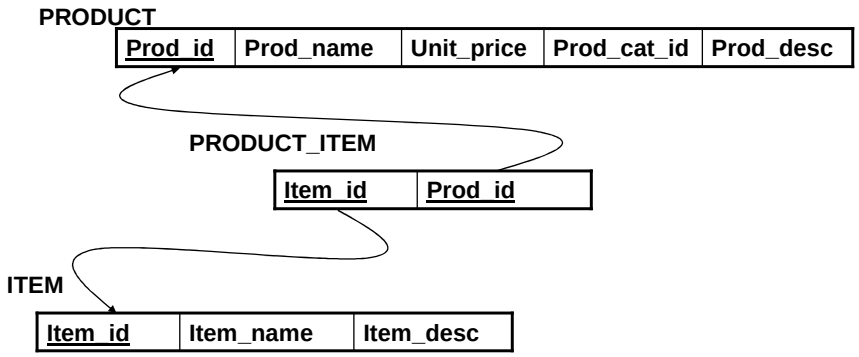
Hình 2.4.11.4 Chuyển đổi thành quan hệ cho kiểu thực thể ORDER_LINE

Liên kết nhiều-nhiều "Supplies" kết nối hai kiểu thực thể SUPPLIER và ITEM nên cũng sẽ tạo thành một quan hệ, chúng ta đặt tên là SUPPLIER_ITEM. Khoá cho quan hệ này sẽ là kết hợp khoá của hai quan hệ trong liên kết, bao gồm hai trường Sup_id tham chiếu đến khoá chính của quan hệ SUPPLIER và trường Item_id tham chiếu đến khoá chính của quan hệ ITEM, ngoài ra thuộc tính Unit_price cũng sẽ trở thành một trường quan hệ như trong hình 2.4.11.5:



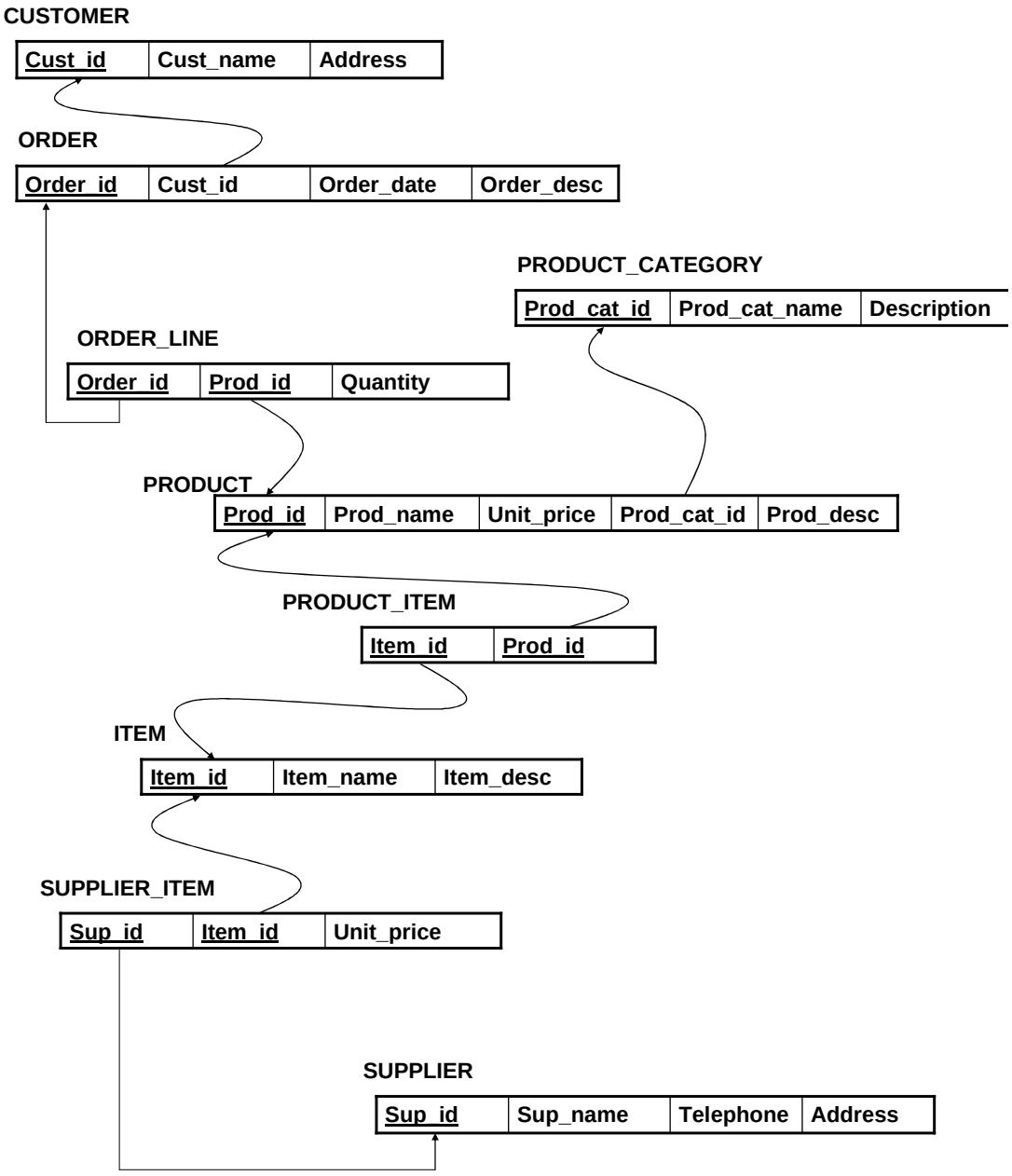
Hình 2.4.11.5 Chuyển đổi liên kết nhiều-nhiều "Supplies" thành quan hệ

Cuối cùng, liên kết nhiều-nhiều "Uses" giữa hai kiểu thực thể PRODUCT và ITEM cũng sẽ được tạo thành một quan hệ như trong trường hợp trên, chúng ta đặt tên cho quan hệ này là PRODUCT_ITEM. Tuy nhiên ở quan hệ này, mọi trường đều tham gia vào làm khoá chính, Item_id tham chiếu đến khoá chính của quan hệ ITEM, Prod_id tham chiếu đến khoá chính của quan hệ PRODUCT.



Hình 2.4.11.6 Chuyển đổi liên kết nhiều nhiều "Uses" thành quan hệ

Cuối cùng chúng ta có sơ đồ cơ sở dữ liệu mức logic như hình 2.4.11.7 sau:



Hình 2.4.11.7 Sơ đồ cơ sở dữ liệu mức logic

4.12. Kết luận

Bước thiết kế cơ sở dữ liệu mức logic là quá trình chuyển đổi từ mô hình dữ liệu mức khái niệm thành mô hình dữ liệu mức logic được mô tả dưới dạng một sơ đồ. Mô hình

dữ liệu quan hệ biểu diễn dữ liệu dưới dạng các quan hệ hay còn được gọi là các bảng. Một quan hệ là một bảng dữ liệu hai chiều và được đặt tên sao cho không trùng với các quan hệ khác trong cùng cơ sở dữ liệu. Tính chất khoá của các quan hệ đóng vai trò vô cùng quan trọng và nó cho biết không thể có hai bản ghi trong cùng một quan hệ có giá trị giống nhau ở tất cả các thuộc tính khoá . Và đặc biệt, trong quan hệ không thể chứa các thuộc tính đa trị.

Trong phần này, chúng ta đã xem xét lần lượt các trường hợp chuyển đổi từ sơ đồ thực thể - liên kết (được thực hiện ở bước 3) thành sơ đồ cơ sở dữ liệu mức logic.

Mỗi một kiểu thực thể trong sơ đồ thực thể liên kết được chuyển đổi thành một quan hệ. Một liên kết một-nhiều được biểu diễn với việc bổ sung thêm một khoá ngoại lai, chính là khoá chính trong quan hệ phía liên kết một, vào quan hệ chuyển đổi từ kiểu thực thể phía liên kết nhiều. Với liên kết nhiều-nhiều, chúng ta đã phải chuyển đổi nó thành một quan hệ mới riêng rẽ với hai quan hệ còn lại tham gia vào liên kết. Khoá chính của quan hệ mới này được tạo thành từ việc kết hợp hai khoá chính trong hai quan hệ mà nó kết nối. Chúng ta cũng đã xem xét khi nào thì khoá ngoại lai sẽ tham gia vào khoá chính trong quan hệ, khi nào thì nó chỉ đóng vai trò là tham chiếu.

Các câu lệnh mô tả cho quan hệ đã bước đầu được hình thành. Chúng sẽ tiếp tục được phân tích và sửa đổi trong hai bước sau: Tinh chỉnh sơ đồ dữ liệu và thiết kế cơ sở dữ liệu mức vật lý. Kết hợp với các câu lệnh tạo chỉ mục, ràng buộc, chúng sẽ trở thành kết quả của giai đoạn thiết kế mức vật lý và sẵn sàng được sử dụng để cài đặt trên một hệ quản trị cơ sở dữ liệu đã lựa chọn.

5. Tinh chỉnh sơ đồ dữ liệu và các dạng chuẩn

Bước thứ tư trong quá trình thiết kế cơ sở dữ liệu đó là phân tích tập các quan hệ trong sơ đồ cơ sở dữ liệu quan hệ mức logic để xác định các vấn đề còn chưa ổn và tiến hành tinh chỉnh chúng. Việc tinh chỉnh sơ đồ dữ liệu được thực hiện dựa trên lý thuyết chuẩn hoá quan hệ.

Thiết kế cơ sở dữ liệu mức khái niệm và logic cho chúng ta một sơ đồ các quan hệ và các ràng buộc toàn vẹn, chúng được xem như như những bước khởi đầu tốt cho quá trình thiết kế cơ sở dữ liệu sau này. Trong bước này, chúng ta cần phải xem xét kỹ hơn các ràng buộc toàn vẹn so với trong mô hình thực thể liên kết. Chúng ta sẽ tập trung vào một lớp ràng buộc quan trọng, đó là phụ thuộc hàm. Ngoài phụ thuộc hàm còn có một số loại ràng buộc toàn vẹn khác như phụ thuộc đa trị, phụ thuộc kết nối, chúng được sử dụng để loại bỏ các dư thừa trong các trường hợp phụ thuộc hàm không thể thực hiện được, tuy nhiên trong giáo trình này chúng ta chỉ dừng ở mức sử dụng các phụ thuộc hàm, bởi nó đã cho phép chúng ta loại bỏ sự dư thừa dữ liệu ở hầu hết các trường hợp thường gặp trong các bài toán thực tế.

5.1. Tinh chỉnh các lược đồ quan hệ

Việc tinh chỉnh sơ đồ dữ liệu chính là quá trình chuẩn hoá để đưa các lược đồ quan hệ về các dạng chuẩn, nhằm loại bỏ tối đa sự lưu trữ dư thừa dữ liệu. Như chúng ta đã biết, lưu trữ dư thừa, trùng lặp dữ liệu là nguyên nhân chính dẫn đến các vần đề về không toàn vẹn dữ liệu và dẫn đến các vấn đề sau:

- Dị thường khi cập nhật: Cùng một dữ liệu, nếu chúng được lưu trữ lặp lại ở nhiều nơi thì khi cập nhật sửa đổi một chỗ, chúng ta phải tìm tất cả những chỗ khác để cập nhật cho đồng bộ, tuy nhiên công việc này khá phức tạp và rất dễ gây ra sự bỏ sót, do đó sẽ dẫn đến tính không nhất quán của dữ liệu.
- Dị thường khi thêm mới: Đôi khi việc chèn thêm dữ liệu mới không thể thực hiện được do cũng phải chèn thêm những dữ liệu khác mà tại thời điểm thêm mới chưa có.
- Dị thường khi xoá bỏ: Khi xoá bỏ một số dữ liệu trong quan hệ, chúng ta cũng có thể đánh mất những dữ liệu khác mặc dù chủ tâm chúng ta không muốn như vậy, do đó sẽ gây ra sự mất mát thông tin.

Tất cả những vấn đề đó khiến chúng ta phải tiến hành chuẩn hoá các quan hệ. Chuẩn hoá được xem là một kỹ thuật để củng cố và nâng cao cho quá trình thiết kế logic, thông qua việc thoả mãn các ràng buộc nhằm loại bỏ các dư thừa dữ liệu. Một quan hệ được xem là ở một dạng chuẩn nào đó nếu nó thoả mãn một tập các ràng buộc nhất định. Ngày nay, trong lý thuyết chuẩn hoá người ta đã xác định được 5 dạng chuẩn, song trên thực tế chúng ta chỉ xét đến 3 dạng chuẩn đầu tiên.

Mục đích của chúng ta trong bước này đó là tạo ra được một cơ sở dữ liệu bao gồm một tập các quan hệ ở dạng chuẩn 3. Các dạng chuẩn 1 và 2 được xem là những bước trung gian để có thể đưa quan hệ đến dạng chuẩn 3.

Để có thể hiểu rõ hơn đối với các dạng chuẩn, chúng ta hãy bắt đầu bằng việc tìm hiểu thế nào là một phụ thuộc hàm.

5.2. Phụ thuộc hàm

Trong một quan hệ R, chúng ta có các thuộc tính X và Y, ta nói thuộc tính Y phụ thuộc hàm vào thuộc tính X trong quan hệ R nếu và chỉ nếu mỗi giá trị của X trong R luôn xác định rõ ràng một giá trị của Y tại mọi thời điểm. X và Y cũng có thể bao gồm nhiều thuộc tính.

Chúng ta kí hiệu phụ thuộc hàm bằng một mũi tên như sau: $X \rightarrow Y$: X xác định Y hay Y phụ thuộc hàm vào X.

Việc thuộc tính Y phụ thuộc hàm vào thuộc tính X sẽ cho chúng ta biết: khi có các giá trị của thuộc tính X thì sẽ xác định được các giá trị của thuộc tính Y.

Ví dụ: Trong quan hệ nhân viên EMPLOYEE, nếu chúng ta biết số hiệu của một nhân viên nào đó thì chúng ta có thể dễ dàng tìm ra tên của anh ta là gì, địa chỉ của anh ta ở đâu. Như vậy các thuộc tính tên Emp_name, địa chỉ Address sẽ phụ thuộc hàm vào thuộc tính số hiệu nhân viên Emp_id. Bởi như chúng ta đã phân tích, mỗi một nhân viên sẽ chỉ có một số hiệu duy nhất. Nhưng ngược lại, nếu chúng ta biết tên của một nhân viên thì chưa chắc chúng ta đã tìm ra được duy nhất số hiệu cũng như địa chỉ của anh ta bởi có thể trong công ty có rất nhiều người cùng có tên như vậy. Vậy trong quan hệ này chúng ta có một phụ thuộc hàm sau: $Emp\_id \rightarrow \{Emp\_name, Address\}$.

Ví dụ: Trong quan hệ EMPLOYEE_PROJECT, chúng ta có các thuộc tính số hiệu nhân viên Emp_id, mã dự án Project_id, ngày bắt đầu tham gia dự án Date_Start. Nếu biết số hiệu nhân viên là NV_15721, mã dự án là DA_1534 thì chúng ta có thể biết ngay được ngày nhân viên NV_15721 tham gia vào dự án DA_1543 là ngày nào.

Nhưng nếu chúng ta chỉ biết số hiệu nhân viên là NV_15721 thì chúng ta không thể biết duy nhất giá trị ngày tham gia dự án bởi nhân viên đó có thể tham gia vào nhiều dự án trong những ngày khác nhau. Cũng như vậy, nếu chúng ta chỉ biết mã dự án là DA_1543 thì chúng ta cũng không thể biết duy nhất một ngày dự án bởi trong dự án đó có nhiều nhân viên bắt đầu tham gia vào những ngày khác nhau. Còn nếu chúng ta biết một ngày nào đó, ví dụ 14/05/2004 thì chúng ta cũng không thể xác định duy nhất các giá trị như mã số dự án, số hiệu nhân viên tham gia vào dự án, bởi cùng ngày hôm đó có thể có nhiều nhân viên tham gia vào nhiều dự án khác nhau. Như vậy chúng ta nhận thấy rằng chỉ khi cùng biết hai giá trị số hiệu nhân viên và mã số dựa án thì chúng ta mới có thể biết duy nhất ngày tham gia vào dự án, vậy chúng ta có phụ thuộc hàm sau: {Emp_id, Project_id} $\rightarrow$ Date_start.

Ví dụ: Cũng tương tự như ví dụ trên, để biết số lượng chính xác (duy nhất) của một mặt hàng đã được bán trong một đơn hàng nào đó trong quan hệ dòng đơn hàng ORDER_LINE, chúng ta cần phải biết số hiệu đơn hàng và số hiệu sản phẩm. Vậy chúng ta có phụ thuộc hàm sau: {Order_id, Product_id} $\rightarrow$ Quantity.

Nhận xét: chúng ta có thể nhận ra ngay một điểm quan trọng sau: Nếu X là khoá chính, khi đó mọi thuộc tính còn lại trong quan hệ đều phải phụ thuộc hàm vào nó. Lưu ý là X có thể là một thuộc tính hoặc một tập thuộc tính.

Sau khi hiểu thế nào là phụ thuộc hàm, chúng ta hãy tiếp tục xem các quan hệ sẽ được đưa về các dạng chuẩn như thế nào.

5.3. Các dạng chuẩn

a. Dạng chuẩn 1NF

Chúng ta nói rằng một quan hệ ở dạng chuẩn 1 nếu tất cả các giá trị các thuộc tính của nó là sơ cấp (tức là chỉ chứa các giá trị được coi là không thể phân chia nhỏ hơn được nữa).

Ví dụ: Để dễ minh hoạ cho quá trình chuẩn hoá ở các phần tiếp sau, ở đây giả thiết chúng ta có lược đồ quan hệ EMP_PROJECT bao gồm những thuộc tính như sau:

EMP_PROJECT (Emp_id, Emp_name, Department, Mgr_id, Mgr_name, Project)

Và một quan hệ của nó:

Eno	Enam	Depart	Mgr	Mgrna	Project
7902	*Sửu*	*10_Sales*	*7988*	*Dậu*	*15_Feasibility,*
7988	*Dậu*	*20_Marketing*	*7699*	*Thìn*	*15_Feasibility,*
7562	*Hợi*	*10_Sal*	*769*	*Thìn*	*25_Anal*

Quan hệ này rõ ràng là chưa ở dạng chuẩn 1. Vì lý do như sau:

Mỗi nhân viên có thể tham gia vào nhiều dự án, và số lượng dự án mà mỗi nhân viên tham gia là có thể khác nhau. Như ta thấy Dậu và Sửu tham gia vào 3 dự án, trong khi đó Hợi chỉ tham gia vào một dự án mà thôi. Do đó, trong trường hợp này, để đưa quan hệ về dạng chuẩn 1 ta phải thêm các bộ mới. Cụ thể ứng với mỗi dự án trong thuộc

tính Project của một nhân viên, ta phải thêm một dòng mới, với các trường còn lại được giữ nguyên ở dòng mới, còn các dự án (trong thuộc tính Project) phải được tách ra thành từng dự án.

Tuy nhiên, ta nhận thấy rằng trong quan hệ còn có 2 thuộc tính chưa ở dạng nguyên tố, đó là thuộc tính Department, bao gồm mã và tên bộ phận, và thuộc tính vừa tách Project (bao gồm cả mã và tên dự án). Do đó, ta tách như sau:

- Thuộc tính Department sẽ được tách thành 2 thuộc tính Dno và Dname, trong đó Dno sẽ lưu mã bộ phận và Dname lưu tên bộ phận.
- Thuộc tính Project cũng sẽ được tách thành 2 thuộc tính Pno và Pname, trong đó thuộc tính Pno lưu mã dự án và thuộc tính Pname lưu tên dự án.

Đến đây ta được quan hệ EMPLOYEE đã ở dạng chuẩn 1, để phục vụ cho việc minh hoạ ở các bước sau, chúng ta thêm vào quan hệ hai thuộc tính nữa, đó là: ngày bắt đầu tham gia dự án của nhân viên Start, và số tiền của dự án mà mỗi nhân viên được trả Bill. Ta có:

En	**Ena**	**D**	**Dnam**	**Mg**	**Mgr**	**P**	**Pnam**	**Start**	**Bi**
79	*Sửu*	*1*	*Sales*	*798*	*Dâu*	*15*	*Feasi*	*10/09*	*10*
79	*Sửu*	*1*	*Sales*	*798*	*Dậu*	*35*	*Testin*	*20/09*	*10*
79	*Sửu*	*1*	*Sales*	*798*	*Dậu*	*45*	*Hand*	*20/10*	*15*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*	*15*	*Feasi*	*15/09*	*20*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*	*25*	*Analy*	*14/09*	*25*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*	*45*	*Hand*	*20/10*	*20*
75	*Hợi*	*1*	*Sales*	*769*	*Thìn*	*25*	*Analy*	*20/05*	*15*

Như trong các phần trước, trong khi thực hiện các bước ánh xạ từ mô hình thực thể liên kết sang quan hệ, chúng ta đã trình bày quy trình để loại bỏ các thuộc tính đa trị từ các kiểu thực thể. Do vậy nếu chúng ta đã thực hiện bước thiết kế logic này rồi thì ở đây chúng ta có thể không cần quan tâm đến việc liệu một quan hệ có ở dạng chuẩn một hay không.

b. Dạng chuẩn 2NF

Một quan hệ được gọi là ở dạng chuẩn hai nếu quan hệ đó đã ở dạng chuẩn một và mọi thuộc tính không khoá đều phụ thuộc hàm đầy đủ vào khoá chính. Tức là không một thuộc tính không khoá nào phụ thuộc hàm một phần vào khoá chính.

Một quan hệ ở dạng chuẩn một sẽ ở dạng chuẩn hai nếu một trong những điều kiện sau được áp dụng:

- Khoá chính chỉ chứa duy nhất một thuộc tính. Ví dụ như thuộc tính Emp_id trong quan hệ EMPLOYEE. Thuộc tính Project_id trong quan hệ PROJECT. Thuộc tính Supplier_id trong quan hệ SUPPLIER.
- Không tồn tại các thuộc tính không khoá trong quan hệ. Tức là tất cả các thuộc tính trong quan hệ đều tham gia vào khoá chính. Ví dụ như trong quan hệ ITEM_COMPONENT, nếu chỉ bao gồm hai thuộc tính Itemp_id và Component_id thì cả 2 thuộc tính này đều là khoá của quan hệ.
- Mọi thuộc tính không khoá đều phụ thuộc hàm đầy đủ vào tất cả các thuộc tính khoá. Ví dụ, trong quan hệ SUPPLIER_ITEM, thuộc tính đơn giá Unit_price phụ thuộc vào cả hai thuộc tính mã nhà cung cấp

Sup_id và số hiệu vật liệu Item_id nên nó ở dạng chuẩn hai.

Quay trở lại quan hệ EMP_PROJECT trên của chúng ta, rõ ràng khoá của quan hệ này là kết hợp của hai thuộc tính Emp_id và Pno. Chúng ta cũng nhận thấy, mỗi một tên dự án chỉ phụ thuộc hàm vào mã dự án, tức là phụ thuộc hàm bộ phận vào một thuộc tính của khoá {Emp_id, Pno}. Cũng tương tự như vậy, các thuộc tính {Ename, Dno, Dname, Mgr_id, Mgrname} chỉ phụ thuộc hàm bộ phận vào thuộc tính Eno, là một thuộc tính của khoá chính K. Chính điều này đã làm xuất hiện những vấn đề sau khi cập nhật và lưu trữ dữ liệu:

- Phép thay đổi: Việc lưu trữ thông tin về một nhân viên, ví dụ như nhân viên Sửu phải lặp lại 3 lần tên phòng, 3 lần tên người quản lý. Khi cần thay đổi phòng làm việc của Sửu chẳng hạn, cần phải thay đổi 3 lần các thông tin này, rõ ràng là tốn kém thời gian hơn. Hơn nữa khi thay đổi thông tin về một nhân viên lại không liên quan gì tới thông tin dự án. Để xác định những thông tin về một nhân viên, chỉ cần thuộc tính Eno là xác định được duy nhất thông tin về họ.
- Phép chèn: Trong quan hệ EMPLOYEE chỉ chứa thông tin về những nhân viên đã tham gia một dự án nào đó. Nếu muốn chèn thêm một nhân viên vào quan hệ thì không được vì không rõ nhân viên đó đã tham gia vào dự án nào chưa. Bất lợi này chỉ được khắc phục nếu loại bỏ những thông tin về dự án ra khỏi quan hệ .

- Phép loại bỏ: Giả sử vì một lý do nào ta đó cần huỷ bỏ dự án Analysis ra khỏi quan hệ, khi đó rõ ràng là thông tin liên quan đến nhân viên Hợi sẽ bị mất.

Để khắc phục những bất lợi trên, ta tách quan hệ EMPLOYEE về 3 quan hệ ở dạng chuẩn 2 như sau:

Quan hệ thứ nhất: PROJECT:

P	Pnam
15	*Feasi*
25	*Analy*
35	*Testin*
45	*Hand*

Trong quan hệ này, thuộc tính Pno sẽ làm khoá chính.

Quan hệ thứ 2: ASSIGNMENT:

En	P	Start	Bi
79	*15*	*10/09*	*10*
79	*35*	*20/09*	*10*
79	*45*	*20/10*	*15*
79	*15*	*15/09*	*20*
79	*25*	*14/09*	*25*
79	*45*	*20/10*	*20*
75	*25*	*20/05*	*15*

Trong quan hệ này, tập thuộc tính {Eno,Pno} vẫn làm khoá chính.

Quan hệ thứ 3: EMPLOYEE:

En	Ena	D	Dnam	Mg	Mgr
79	*Sửu*	*1*	*Sales*	*798*	*Dậu*
79	*Dậu*	*2*	*Marke*	*769*	*Thìn*
75	*Hợi*	*1*	*Sales*	*769*	*Thìn*

Trong quan hệ này, thuộc tính Eno sẽ làm khoá chính.

Đến đây ta đã có 3 quan hệ đã ở dạng chuẩn 2.

c. Dạng chuẩn 3NF

Một quan hệ đã là 2NF được xem là ở dạng chuẩn 3NF nếu tất cả các phụ thuộc hàm giữa khoá chính và các thuộc tính khác của nó đều là trực tiếp.

Hay nói cách khác, mọi thuộc tính không nằm trong khoá chính đều không phụ thuộc hàm vào một thuộc tính không phải là khoá chính.

Để phân rã các quan hệ ở dạng chuẩn 2 thành các quan hệ ở dạng chuẩn 3, ta phải thực hiện các bước sau:

- Tìm ra các phụ thuộc hàm bắc cầu vào khoá chính có dạng : K $\rightarrow$ X, X $\rightarrow$ Y, trong đó K là tập các thuộc tính khoá, X và Y là tập các thuộc tính không khoá.
- Nếu tìm thấy phụ thuộc hàm bắc cầu như trên thì tiến hành tách quan hệ hiện thời thành hai quan hệ, quan hệ thứ nhất bao gồm các thuộc tính X và Y, quan hệ thứ hai bao gồm các thuộc tính trong quan hệ ban đầu trừ những thuộc tính trong Y.

- Kiểm tra với các quan hệ con xem đã ở dạng chuẩn 3 hay chưa, nếu chưa lại thực hiện tách tiếp như bước trên.

Ví dụ: quay trở lại các quan hệ đã ở dạng chuẩn 2NF trên chúng ta phân tích tiếp:

Hai quan hệ PROJECT và ASSIGMENT rõ ràng là đã ở dạng chuẩn 3. Vì ở quan hệ Projects, ngoài khoá chính ra không còn có thuộc tính nào khác.

Trong quan hệ ASSIGMENT, hai thuộc tính mô tả còn lại Start và Bill không phụ thuộc vào nhau, do đó quan hệ này cũng ở dạng chuẩn 3.

Trong quan hệ còn lại: EMPLOYEE, ta nhận thấy rằng thuộc tính Dname chỉ phụ thuộc hàm vào thuộc tính Dno, mà không phụ thuộc hàm trực tiếp vào khoá chính K = Eno, rõ ràng Dname đã phụ thuộc hàm bắc cầu vào khoá chính K thông qua thuộc tính Dno. Cũng tương tự như vậy đối với thuộc tính Mgrname, nó chỉ phụ thuộc hàm vào thuộc tính Mgr_id mà không phụ thuộc hàm trực tiếp vào thuộc tính Eno.

Do đó ở quan hệ này, ta phải tách thành 3 quan hệ là DEPARTMENT, MANAGER và EMPLOYEE ở dạng chuẩn 3 như sau:

Quan hệ thứ nhất: DEPARTMENT

D	**Dnam**
2	*Marke*
1	*Sales*

Với khoá là thuộc tính: Dno

Quan hệ thứ hai: MANAGER

Mgr	Mgr
798	*Dậu*
769	*Thìn*

Với khoá là thuộc tính: Mgr_id

Và quan hệ thứ ba: EMPLOYEE

En	Ena	D	Mgr	Mgr
79	*Sửu*	*1*	*798*	*Dâu*
79	*Dậu*	*2*	*769*	*Thìn*
75	*Hợi*	*1*	*790*	*Dần*

Với khoá là thuộc tính: Eno

Tiếp tục chuẩn hoá với 2 quan hệ trên:

Trong các quan hệ DEPARTMENT và MANAGER, vì ngoài khoá chính ra, không còn có thuộc tính nào khác, do đó chúng đã ở dạng chuẩn 3.

Trong quan hệ EMPLOYEE, lần lượt tìm phụ thuộc hàm bắc cầu vào khoá chính Eno của các thuộc tính còn lại, ta không tìm thấy phụ thuộc hàm nào thỏa phụ thuộc bắc cầu, do đó quan hệ cũng ở dạng chuẩn 3.

Như vậy kết hợp với 2 quan hệ còn lại là:

Quan hệ PROJECT:

P	Pnam
15	*Feasi*
25	*Analy*
35	*Testin*
45	*Hand*

và quan hệ ASSIGNMENT:

En	P	Start	Bi
79	*15*	*10/09*	*10*
79	*35*	*20/09*	*10*
79	*45*	*20/10*	*15*
79	*15*	*15/09*	*20*
79	*25*	*14/09*	*25*
79	*45*	*20/10*	*20*
75	*25*	*20/05*	*15*

Ta có 5 quan hệ đã ở dạng chuẩn 3NF.

Quay trở lại ví dụ chung của công ty kinh doanh thiết bị đồ gỗ, nếu trong bước thiết kế cơ sở dữ liệu mức khái niệm (ở các hình 2.3.7.3 và 2.3.7.4) chúng ta không tiến hành phân tách liên kết "Orders" thành hai kiểu thực thể là ORDER và ORDER_LINE thì khi hình thành quan hệ ở bước thiết kế cơ sở dữ liệu mức logic chúng ta sẽ có một quan hệ ORDER với các thuộc tính sau:

ORDER(Order_id, Order_date, Order_desc, Cust_id, Prod_it, Quantity).

Trong đó 2 thuộc tính Order_id và Prod_id là các thuộc tính khoá của quan hệ.

Rõ ràng quan hệ ORDER trên không ở dạng chuẩn 3NF bởi các thuộc tính Order_date, Order_desc và Prod_id chỉ phụ thuộc vào thuộc tính Order_id mà thôi, còn trường Quantity thì phụ thuộc đầy đủ vào toàn bộ các thuộc tính khoá của quan hệ (Order_id, Prod_id).

Do vậy nếu trong các bước đó chúng ta chưa phân tích kỹ được như đã trình bày thì tại bước tinh chỉnh và chuẩn hoá các quan hệ này chúng ta phải tiến hành tách quan hệ ORDER trên thành hai quan hệ mới đó là:

ORDER (Order_id, Order_date, Order_desc, Prod_id)

ORDER_LINE (Order_id, Prod_id, Quantity)

Đến đây chúng ta cũng đã có hai quan hệ đã ở dạng chuẩn 3NF như trong các bước trên. Như vậy có thể nhận thấy rằng, chỉ khi kết thúc bước này, chúng ta mới có được một sơ đồ thiết kế cơ sở dữ liệu mức khái niệm hoàn chỉnh theo nghĩa mọi quan hệ đã ở dạng chuẩn 3NF và đã tối thiểu hoá được việc lưu trữ dư thừa dữ liệu.

Trong lý thuyết thiết kế cơ sở dữ liệu, ngoài 3 dạng chuẩn trình bày trên, người ta còn phát triển tiếp các dạng chuẩn khác như dạng chuẩn BCNF, dạng chuẩn 4 với khái niệm phụ thuộc đa trị và dạng chuẩn 5 với khái niệm phụ thuộc kết nối . Tuy nhiên trong giáo trình này, chúng ta chỉ dừng ở dạng chuẩn 3, vì như trên đã trình bày, trên thực tế khi các quan hệ được chuẩn hoá ở dạng chuẩn 3, chúng đã hạn chế khá tốt tính lưu trữ dư thừa dữ liệu. Ngoài ra, cũng nhận thấy rằng, trong quá trình tiến hành chuẩn hoá các quan hệ như trên, tính toàn vẹn của dữ liệu vẫn được đảm bảo.

6. Thiết kế cơ sở dữ liệu vật lý

Trong bước này, chúng ta phải xem xét, cân nhắc kỹ các vấn đề liên quan đến khối lượng dữ liệu mà cơ sở dữ liệu đang xây dựng cần phải đáp ứng, ngoài ra, cũng trong bước này có thể chúng ta phải tiến hành tinh chỉnh thêm thiết kế các quan hệ để đảm bảo chắc chắn rằng nó đáp ứng được các mục tiêu về hiệu năng truy suất.

Bước này có thể đơn giản chỉ là việc xây dựng các chỉ số cho một số bảng dữ liệu, phân cụm một số bảng, hoặc cũng có thể bao gồm cả bước thiết kế lại một phần của sơ đồ cơ sở dữ liệu logic có được trong các bước thiết kế trước.

Chúng ta cũng cần lưu ý rằng thiết kế cơ sở dữ liệu mức vật lý không bao gồm quá trình cài đặt cơ sở dữ liệu đó.

Để có thể thực hiện được bước này, chúng ta cần phải có sự hiểu biết thấu đáo về hệ quản trị cơ sở dữ liệu sẽ được lựa chọn để cài đặt cho cơ sở dữ liệu, đặc biệt là:

- Các đối tượng trong cơ sở dữ liệu mà hệ quản trị cơ sở dữ liệu được lựa chọn hỗ trợ, cấu trúc vật lý và các tệp cần thiết cho các đối tượng đó.
- Cách thức mà hệ quản trị cơ sở dữ liệu đó hỗ trợ cho các đối tượng: chỉ mục, toàn vẹn tham chiếu, các ràng buộc, các kiểu dữ liệu,...
- Ưu nhược điểm đối với mỗi một phiên bản của hệ quản trị cơ sở dữ liệu đã lựa chọn.
- Các tham số cấu hình cho hệ quản trị cơ sở dữ liệu.
- Các câu lệnh định nghĩa dữ liệu để có thể chuyển đổi thành các đối tượng dữ liệu mức vật lý

6.1. Phân tích kích thước và nhu cầu sử dụng dữ liệu

Kích thước dữ liệu và việc thống kê những dữ liệu thường xuyên được sử dụng là những tiêu chuẩn quan trọng ở đầu vào cho quá trình thiết kế cơ sở dữ liệu mức vật lý. Do vậy chúng ta cần phải đánh giá được kích thước của cơ sở dữ liệu cũng như việc sử dụng dữ liệu.

Kích thước và nhu cầu sử dụng dữ liệu thường được phát hiện ra trong quá trình phân tích hệ thống. Kích thước của dữ liệu đại diện cho độ phức tạp của hoạt động nghiệp vụ, và nên được đánh giá trong một khoảng thời gian vài năm tuỳ thuộc vào từng ứng dụng cơ sở dữ liệu cụ thể. Nhu cầu sử dụng hay tần xuất truy nhập dữ liệu được đánh giá dựa trên thời gian của các sự kiện, độ lớn của các giao dịch, các hoạt động liên quan đến báo cáo, truy vấn tìm kiếm dữ liệu. Ngày nay, một số cơ sở dữ liệu lớn thường hỗ trợ các truy vấn dạng ad hoc, những truy vấn này chiếm đáng kể thời gian thực hiện bởi chúng phải xử lý trên những lượng dữ liệu vô cùng lớn, do vậy đôi khi với những cơ sở dữ liệu có sử dụng truy vấn này chúng ta nên để ý đến kích thước của dữ liệu hơn là tần xuất sử dụng các truy vấn đó.

Trong quá trình phân tích kích thước và nhu cầu sử dụng dữ liệu, chúng ta nên giải thích được tính tự nhiên của các truy nhập thông qua các hình vẽ mô tả các đường truy nhập.

Ví dụ chúng ta có thể mô tả trong số 200 truy nhập đến dữ liệu trong bảng đơn hàng ORDER, có 150 truy nhập dựa trên khoá chính Order_id (như hỏi thông tin về một đơn hàng cụ thể), 50 truy nhập còn lại là dựa trên các giá trị của trường ngày đơn hàng Order_date.

Những mô tả rõ ràng này sẽ giúp cho công đoạn lựa chọn chỉ mục, như sẽ trình bày trong phần dưới đây, thuận tiện hơn, chính xác hơn và phù hợp với thực tế. Cũng tương tự như vậy, chúng ta nên mô tả cả các kết quả trả về của một truy nhập: đó sẽ là một thao tác tạo mới, thao tác sửa đổi, xoá bỏ hay trả về của dữ liệu.

6.2. Thiết kế các trường

Trường là đơn vị nhỏ nhất của một dữ liệu ứng dụng mà phần mềm hệ thống có thể nhận ra, phần mềm hệ thống có thể là ngôn ngữ lập trình hay hệ quản trị cơ sở dữ liệu. Một trường tương ứng với một thuộc tính đơn trong mô hình dữ liệu logic.

Đối với mỗi trường dữ liệu, chúng ta phải quan tâm đến những vấn đề sau: kiểu dữ liệu (hay kiểu lưu trữ) sẽ được sử dụng để biểu diễn các giá trị của trường đó, các kiểm soát toàn vẹn dữ liệu, và làm thế nào để một hệ quản trị cơ sở dữ liệu có thể quản lý được vấn đề mất mát giá trị trong trường đó.

a. Lựa chọn kiểu dữ liệu

Mỗi một trường dữ liệu được lưu trữ trong cơ sở dữ liệu bắt buộc phải thuộc một kiểu dữ liệu mà hệ quản trị cơ sở dữ liệu đó hỗ trợ. Việc lựa chọn một kiểu dữ liệu phải thoả mãn 4 mục đích sau:

- Tối thiểu hoá không gian lưu trữ
- Biểu diễn được tất cả các khả năng của giá trị cần lưu trữ
- Nâng cao tính toàn vẹn dữ liệu

- Hỗ trợ tất cả các thao thác trên dữ liệu

Tuy nhiên, những mục đích này có thể có mức độ quan trọng khác nhau tuỳ thuộc vào từng bài toán cụ thể.

Lựa chọn đúng kiểu dữ liệu cho một trường, trong không gian tối thiểu, nhằm biểu diễn mọi giá trị có thể xảy ra (tất nhiên là loại trừ những giá trị không hợp lệ) cho thuộc tính tương ứng và có thể hỗ trợ các thao tác trên dữ liệu (ví dụ các kiểu dữ liệu số phải đáp ứng được các thao tác là các hàm số học tính toán, các kiểu dữ liệu ký tự phải đáp ứng được các thao tác ghép xâu, tách xâu). Trong mô hình dữ liệu mức khái niệm chúng ta đã xây dựng các ràng buộc miền cho các thuộc tính, trong bước này, những thông tin đó cũng sẽ giúp cho chúng ta khá nhiều trong việc lựa chọn kiểu dữ liệu cho các thuộc tính đó.

Việc đạt được 4 mục tiêu trên hoàn toàn không dễ dàng và chúng ta cần phải nắm bắt được toàn bộ các trường hợp có thể xảy ra. Ví dụ, chúng ta có kiểu dữ liệu SmallInt có độ rộng lớn nhất là 2 bytes. Giả sử kiểu dữ liệu này là đủ để biểu diễn cho trường số lượng bán Quantity, song nếu chúng ta lại muốn tính toán trên kiểu dữ liệu này như tính tổng, thì có thể kết quả tính sẽ vượt qua độ rộng cho phép, và do vậy có thể dẫn đến những sai sót.

Chúng ta cũng cần lưu ý rằng với những kiểu dữ liệu cụ thể sẽ có những khả năng thao tác cụ thể, ví dụ với kiểu dữ liệu DATE chỉ cho phép những phép toán số học cụ thể trên trường DATE mà thôi.

b. Kiểm soát các toàn vẹn dữ liệu

Trong nhiều hệ quản trị cơ sở dữ liệu, kiểm soát các toàn vẹn dữ liệu (tức là quản lý các giá trị có thể nhận của một

trường dữ liệu) có thể được xây dựng thành những cấu trúc vật lý cho các trường.

Kiểu dữ liệu là một loại kiểm soát toàn vẹn, bởi nó giới hạn kiểu của dữ liệu mà một trường có thể nhận được. Ví dụ nếu một trường được khai báo có kiểu dữ liệu là kiểu số thì khi đó không thể chèn các dữ liệu kiểu ký tự vào trường đó được.

Độ rộng của trường dữ liệu cũng là một kiểu kiểm soát toàn vẹn dữ liệu, bởi nó giới hạn kích thước của dữ liệu đưa vào. Ví dụ nếu một trường lưu bậc lương của nhân viên có giá trị không vượt quá một số nguyên thì không thể lưu những giá trị bằng hoặc lớn hơn 10 được. Hay trường tên của một nhân viên không bao giờ vượt quá 30 ký tự chẳng hạn.

Một số kiểm soát toàn vẹn dữ liệu khác:

- Giá trị mặc định: là giá trị được gán cho một trường nếu một bản ghi mới được tạo ra, trường được định nghĩa giá trị mặc định sẽ mang giá trị đó trừ khi người sử dụng thay đổi. Việc gán giá trị mặc định cho một trường sẽ giúp giảm thời gian nhập dữ liệu cho người sử dụng, và nó cũng giảm tỷ lệ mắc lỗi khi nhập dữ liệu (một trong những lỗi hay xảy ra nhất). Ví dụ, mỗi một chứng từ mua hàng được tạo ra đều mang giá trị là ngày hiện thời thì chúng ta có thể đặt giá trị mặc định cho thuộc tính này là ngày hệ thống (sysdate).
- Điều kiện khoảng: Một điều kiện khoảng sẽ giới hạn tập các giá trị cho phép đối với một trường dữ liệu. Một khoảng có thể là một khai báo về cận trên và cận dưới đối với một trường dữ liệu kiểu số nào đó có thể

được nhận giá trị. Nói một cách khác, giá trị của dữ liệu đưa vào được gọi là hợp lệ nếu nó nằm trong khoảng giữa cận trên và cận dưới đó. Có một lưu ý nhỏ đó là nên đặt các điều kiện khoảng này trong khi định nghĩa các bảng dữ liệu của cơ sở dữ liệu, bởi ta cũng có thể giới hạn các giá trị nhập vào thông qua chương trình ứng dụng, và khi đó thì rất dễ gây ra lỗi không nhất quán, triệt để và khó khăn trong quá trình tìm và thay đổi khi chúng ta muốn sửa đổi giá trị khoảng này. Ví dụ như nếu chúng ta qui ước thuộc tính bậc lương của nhân viên có kiểu dữ liệu là Smallint, có độ rộng là 1 và nằm trong khoảng từ 1 đến 7 thì ta có thể khai báo giá trị khoảng cho thuộc tính này có cận dưới bằng 1 và cận trên bằng 7. Khi đó cơ sở dữ liệu sẽ ngăn cản được những giá trị đưa vào không hợp lệ như các giá trị 8, 9, mặc dù độ dài của chúng thoả mãn điều kiện ràng buộc về độ rộng của trường.

- Điều kiện giá trị Null: một giá trị null được định nghĩa như là một giá trị không xác định. Đối với mỗi một thuộc tính tham gia vào khoá chính phải được đảm bảo cấm được các giá trị null. Ngoài ra một số thuộc tính khác cũng có thể khai báo điều kiện này. Ví dụ, trong quan hệ nhân viên EMPLOYEE, ngoài thuộc tính mã hiệu nhân viên Emp_id là khoá chính không được phép mang giá trị null, thuộc tính Emp_name cũng nên được khai báo không được phép mang giá trị null, bởi nếu chúng ta biết mã hiệu của một nhân viên mà không biết tên của anh ta là ai thì cũng vô nghĩa.

- Toàn vẹn tham chiếu: Toàn vẹn tham chiếu là một dạng của toàn vẹn theo điều kiện giá trị khoảng, bởi những giá trị hợp lệ phải được lấy từ (tham chiếu đến) một quan hệ nào đó được trường đó tham chiếu đến. Các khoá ngoại lai là một trường hợp phổ biến của dạng kiểm soát này. Ví dụ trong quan hệ EMPLOYEE, chúng ta có thuộc tính Dept_id là thuộc tính khoá ngoại lai, nó được tham chiếu đến thuộc tính khoá chính Dept_id của quan hệ DEPARTMENT. Do vậy mọi giá trị mà thuộc tính Dept_id trong quan hệ EMPLOYEE phải được lấy từ tập giá trị trong thuộc tính Dept_id của quan hệ DEPARTMENT.

6.3. Tổ chức các bản ghi vật lý và phi chuẩn

a. Tổ chức các bản ghi vật lý

Trong mô hình dữ liệu logic, chúng ta đã nhóm vào một quan hệ các thuộc tính được xác định bởi cùng một khoá chính. Trái lại, một bản ghi vật lý là một nhóm các trường được lưu trữ liên tiếp trong các vị trí của bộ nhớ và được trả về dưới dạng một đơn vị đọc/ghi. Việc thiết kế các bản ghi vật lý bao gồm việc lựa chọn thứ tự các trường được lưu trữ kế tiếp nhằm đạt được hai mục đích: sử dụng hiệu quả vùng lưu trữ thứ cấp và tăng tốc độ xử lý dữ liệu.

Vấn để sử dụng hiệu quả vùng lưu trữ thứ cấp bị ảnh hưởng bởi cả kích thước của bản ghi vật lý lẫn cấu trúc của vùng lưu trữ thứ cấp. Các hệ điều hành máy tính đọc dữ liệu từ ổ cứng dưới các dạng đơn vị được gọi là trang (page), chứ không phải là các bản ghi vật lý. Một trang là lượng dữ liệu

được đọc hoặc viết vào bộ nhớ thứ cấp trong một phép vào hoặc ra. Kích thước của một trang là được xác định trước và được lựa chọn sao cho đạt hiệu quả cao nhất khi sử dụng RAM. Phụ thuộc vào từng hệ thống máy tính, một bản ghi vật lý có thể được phép hoặc không được phép nằm trên hai trang khác nhau. Nếu kích thước của một trang không là bội số của kích thước bản ghi vật lý, khi đó một số vùng cuối của trang sẽ bị lãng phí. Số lượng các bản ghi vật lý có thể lưu trong một trang được gọi là hệ số khối (blocking factor). Nếu không gian lưu trữ khan hiếm và các bản ghi vật lý không thể nằm trên các trang khác nhau, việc có nhiều bản ghi vật lý trong một quan hệ logic sẽ tối thiểu hoá việc tiêu tốn không gian lưu trữ.

Việc thiết kế một bản ghi vật lý để bảo toàn không gian lưu trữ sẽ dễ dàng hơn nếu mọi trường dữ liệu là có độ dài cố định. Trong trường hợp này, các trường được đặt theo thứ tự trường này nối tiếp trường kia. Với các trường kiểu ký tự, mỗi giá trị sẽ được gán thêm các giá trị trống vào cuối, còn với trường số nó sẽ được gán thêm các số 0 vào trước. Với các trường có kích thước thay đổi, việc lưu trữ phức tạp hơn. Ví dụ kiểu trường VARCHAR2 mà hệ quản trị cơ sở dữ liệu Oracle hỗ trợ là một kiểu trường có kích thước thay đổi.

Lưu ý: Ngày nay hầu hết các hệ quản trị cơ sở dữ liệu lớn đều hỗ trợ việc tổ chức các bản ghi vật lý thông qua các tham số thiết lập cho cơ sở dữ liệu, do vậy, chúng ta không cần đi quá sâu vào việc xem xét các bản ghi được lưu trữ và truy suất vật lý như thế nào mà nên quan tâm đến kích thước của chúng tương ứng với mỗi khối dữ liệu mà hệ quản trị cơ sở dữ liệu đó hỗ trợ. Ví dụ như với hệ quản trị

cơ sở dữ liệu Oracle, kích thước phổ biến của mỗi database block mà nó hỗ trợ là 8kb, tuy nhiên nếu các bản ghi có kích thước khá lớn thì chúng ta có thể sửa đổi tham số này thành 16kb để có thể tận dụng được thêm không gian lưu trữ, giảm thiểu sự phân mảnh.

b. Phi chuẩn

Những vấn đề được trình bày trên đây về thiết kế các bản ghi vật lý chủ yếu tập trung vào việc sử dụng hiệu quả không gian lưu trữ. Trong hầu hết các trường hợp, mục đích thứ 2 của việc thiết kế các bản ghi vật lý - tức là xử lý hiệu quả dữ liệu - chiếm hầu hết thời gian thiết kế cơ sở dữ liệu mức vật lý. Việc thiết kế cơ sở dữ liệu vật lý nhằm xử lý dữ liệu một cách hiệu quả, giống như việc sắp xếp sao cho có thể tìm kiếm một cuốn sách hiệu quả nhất trong một thư viện, phụ thuộc rất nhiều vào việc sắp xếp gần nhau những dữ liệu (sách) liên quan. Thông thường, tất cả các thuộc tính xuất hiện trong một quan hệ không được sử dụng đồng thời, và dữ liệu từ các quan hệ khác nhau thường được kết hợp lại để trả lời cho một truy vấn hay để tạo ra một báo cáo. Do vậy, mặc dù các quan hệ đã được chuẩn hoá có thể giải quyết được các dị thường trong lưu trữ, cập nhật, bảo trì cơ sở dữ liệu, nhưng nếu chúng được thiết kế quá chặt chẽ thì có thể dẫn tới việc xử lý dữ liệu không hiệu quả.

Một số kết quả so sánh trên một cơ sở dữ liệu được chuẩn hoá hoàn và một cơ sở dữ liệu chỉ được chuẩn hoá một phần. Kết quả cho thấy rằng, việc xử lý trên những cơ sở dữ liệu chỉ được chuẩn hoá một phần còn tỏ ra nhanh hơn trên những cơ sở dữ liệu được chuẩn hoá hoàn toàn. Mặc dù kết quả này phụ thuộc rất lớn vào cơ sở dữ liệu cũng

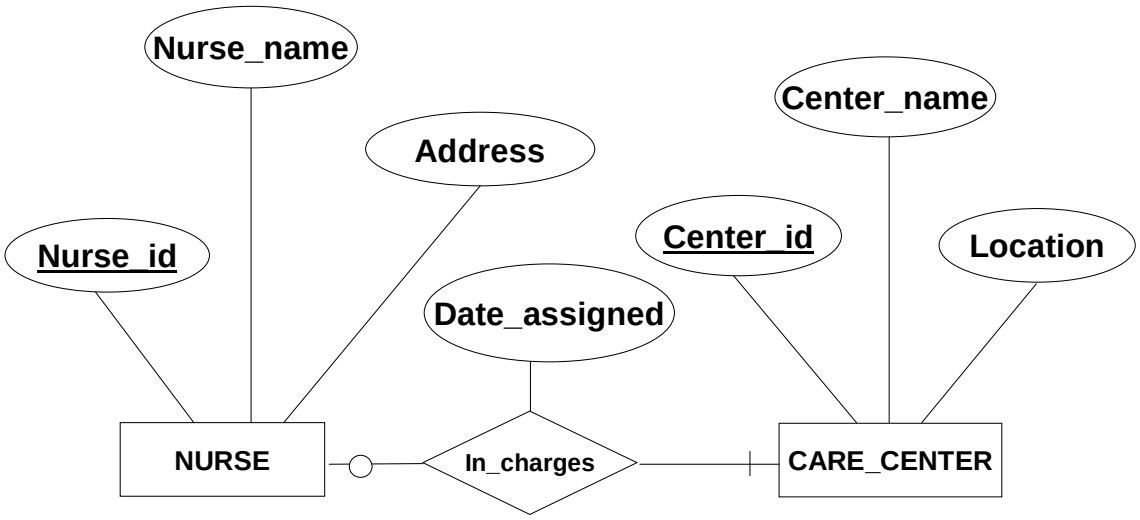
như loại xử lý dữ liệu, nhưng nó cũng cho thấy rằng chúng ta nên cân nhắc cẩn thận khi thiết kế các bản ghi vật lý với việc chuẩn hoá các quan hệ cho cơ sở dữ liệu.

Phi chuẩn là một kỹ thuật chuyển đổi các quan hệ đã được chuẩn hoá ở mức cao thành những dạng chuẩn ở mức thấp hơn nhằm tăng tốc độ truy nhập trong cơ sở dữ liệu. Tuy nhiên chúng ta cũng cần phải lưu ý rằng, việc thiết kế phi chuẩn sẽ làm gia tăng khả năng xảy ra lỗi cũng như vấn đề không nhất quán dữ liệu và khiến cho việc lập trình ứng dụng phức tạp hơn.

Dưới đây chúng ta sẽ xem xét một số trường hợp có thể tiến hành phi chuẩn, từ đó có thể rút ra kinh nghiệm trong quá trình thiết kế cơ sở dữ liệu mức vật lý:

- Trường hợp hai quan hệ được hình thành từ hai kiểu thực thể tham gia vào liên kết một - một: Nếu việc truy suất giữa hai quan hệ diễn ra thường xuyên thì chúng ta nên kết hợp chúng thành một quan hệ.

Ví dụ: Hình 2.6.3.1 dưới đây cho thấy, một quan hệ NURSE liên kết với một quan hệ CARE_CENTER thông qua khoá Nurse_id, tức là một y tá có thể đảm nhiệm một trung tâm chăm sóc bệnh nhân. Chúng ta có thể kết hợp 2 quan hệ này thành một, trong đó giá trị trong các trường Center_name, Location, Date_assigned có thể mang giá trị null đối với những y tá không phụ trách trung tâm nào. Chúng ta cũng nhận thấy rằng khi kết nối hai quan hệ như vậy thì chúng ta cũng không cần đến thuộc tính Center_id.



Hai quan hệ được chuẩn hoá

NURSE

Nurse_id	Nurse_name	Address

CARE_CENTER

Center_id	Center_name	Nurse_id	Location	Date_assigned

Quan hệ được phi chuẩn hoá

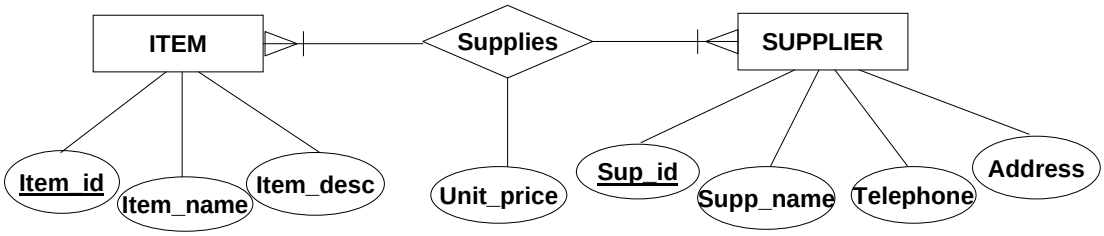
NURSE

Nurse_id	Nurse_name	Address	Centre_name	Location	Date_assigned

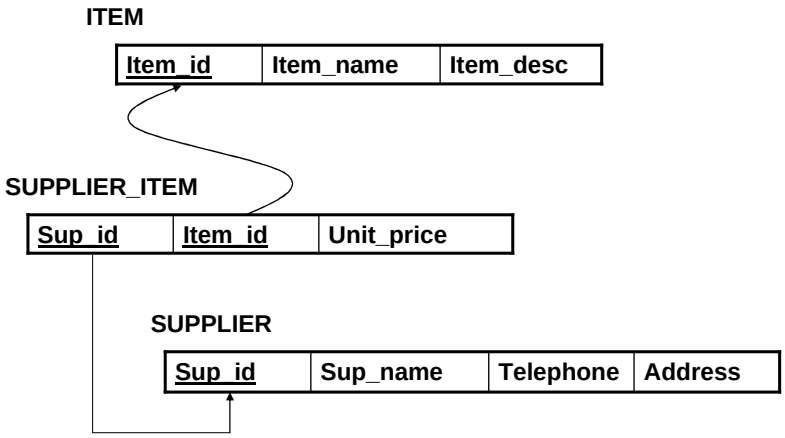
Hình 2.6.3.1 Phi chuẩn trong trường hợp liên kết một - một

- Trường hợp các quan hệ được hình thành từ một liên kết nhiều-nhiều với các thuộc tính trong liên kết không phải là khoá: Thay vì phải hình thành 3 quan hệ chúng ta có thể kết hợp thành 2 quan hệ chính. Chúng ta gộp các thuộc tính trong một quan hệ vào trong quan hệ được hình thành từ liên kết.

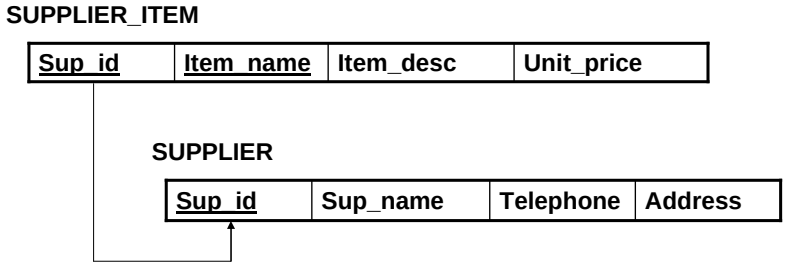
Ví dụ: Hình 2.6.3.2 sau là một ví dụ về các quan hệ và liên kết minh hoạ cho nghiệp vụ liên kết báo giá cho các mặt hàng khác nhau của các nhà cung cấp khác nhau. Trong trường hợp này, các trường của các quan hệ ITEM và SUPPLIER_ITEM có thể được gộp vào trong quan hệ hình thành từ liên kết nhiều-nhiều nhằm tránh việc phải kết nối 3 quan hệ. Chú ý rằng, chính việc kết hợp này cũng tạo ra sự lưu trữ dư thừa dữ liệu, như trong ví dụ này là các giá trị nằm trong trường Item_desc bị lặp lại trong mỗi bản ghi báo giá.



Ba quan hệ được chuẩn hoá



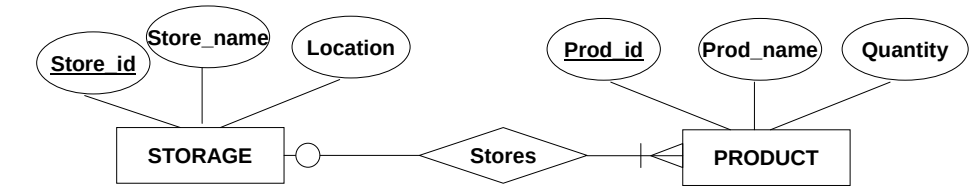
Quan hệ được phi chuẩn hoá



Hình 2.6.3.2 Phi chuẩn trong trường hợp liên kết nhiều - nhiều

- Trường hợp các quan hệ được hình thành từ liên kết một-nhiều và kiểu thực thể phía liên kết đầu một không tham gia vào bất cứ một liên kết nào khác. Khi đó chúng ta có thể cân nhắc đến việc kết hợp hai quan hệ hình thành từ hai kiểu thực thể này thành một quan hệ phi chuẩn như ví dụ sau:

Ví dụ: Giả thiết chúng ta có liên kết một-nhiều sau: mỗi một kho STORAGE sẽ được sử dụng để lưu một số mặt hàng cụ thể PRODUCT, ngược lại mỗi một mặt hàng chỉ được phép lưu trong một kho. Khi đó chúng ta có thể gộp hai quan hệ trên thành một quan hệ duy nhất là ITEM như trong hình 2.6.3.3 sau:



Quan hệ được chuẩn hoá

STORAGE

Store_id	Store_name	Location

PRODUCT

Prod_id	Prod_name	Store_id	Quantity

Quan hệ được phi chuẩn hoá

PRODUCT

Prod_id	Prod_name	Store_name	Location	Quantity

Hình 2.6.3.3 Phi chuẩn trong trường hợp liên kết một - nhiều

Trên đây là một số trường hợp mà chúng ta có thể gộp các quan hệ nhằm tránh việc phải thường xuyên kết nối giữa chúng. Ngược lại, phi chuẩn cũng có thể được sử dụng để tách các quan hệ bằng cách phân mảng một quan hệ thành nhiều bảng riêng biệt. Cả 2 phương pháp phân mảng theo chiều dọc và theo chiều ngang đều có thể được sử dụng.

- Phân mảng theo chiều ngang là chia một quan hệ thành nhiều vùng khác nhau có cùng cấu trúc các

trường. Chúng ta có thể hình dung công việc này như trong thư viện, ta đặt các sách khoa học vào một ngăn riêng biệt, các sách văn học vào một ngăn riêng biệt khác. Trong cơ sở dữ liệu cũng vậy, việc phân mảng theo chiều ngang tỏ ra khá hiệu quả khi các bản ghi trong quan hệ được xử lý riêng rẽ theo từng vùng. Ví dụ với quan hệ khách hàng CUSTOMER, chúng ta có thể chia quan hệ này thành 4 phân mảng khác nhau dựa trên các giá trị trong thuộc tính Region. Khi đó, các nghiệp vụ xử lý sẽ nhanh hơn do được tiến hành dựa trên các mảng khác nhau, bởi thực sự máy tính chỉ phải thao tác trên những tệp tách biệt nhau, số lượng các bản ghi cần thao tác được giảm đi đáng kể. Việc phân mảng theo chiều ngang cũng tỏ ra an toàn hơn, bởi bản chất chúng là những tệp riêng biệt, chúng ta có thể tiến hành các thao tác phân quyền đảm bảo an toàn mức tệp để giới hạn việc sử dụng đối với người sử dụng. Phương pháp này cũng giúp việc khôi phục dữ liệu dễ dàng hơn. Khi một phân mảng bị hỏng, chúng ta có thể tiến hành khôi phục riêng phân mảng đó mà không làm ảnh hưởng đến các phân mảng khác cũng như toàn bộ quan hệ. Ngay cả khi một phân mảng bị hỏng thì các phân mảng khác vẫn có thể hoạt động thay vì cả quan hệ sẽ bị hỏng. Cuối cùng, mỗi một phân mảng có thể được lưu trữ trên những ổ cứng khác nhau, điều này giúp cho việc tăng đáng kể tốc độ xử lý dữ liệu trong quan hệ.

Cũng cần phải lưu ý rằng, việc phân mảng theo chiều ngang gần tương tự như việc tạo các liên kết kiểu “là - một” bởi các thực thể khác nhau sẽ tham gia vào những liên

kết khác nhau, do vậy chúng cũng sẽ tham gia vào các quá trình xử lý khác nhau.

- Phân mảng theo chiều dọc chia các cột của một quan hệ thành những tệp khác nhau và thuộc tính tham gia vào làm khoá chính sẽ được lưu trữ ở tất cả các tệp được tách ra đó. Ví dụ trong một quan hệ chúng ta có thể tách các cột số liệu liên quan đến kế toán thành một bảng riêng, các số liệu liên quan đến kỹ thuật thành một bảng riêng và các số liệu liên quan đến bán hàng thành một bảng riêng. Khi các số liệu này cần được xử lý cùng nhau thì những bảng này có thể dễ dàng kết hợp lại (do có cùng chung các trường khoá chính), do vậy nó được xử lý như một quan hệ.

Trong một số trường hợp, chúng ta cũng có thể kết hợp cả 2 phương pháp phân mảng trên cho một quan hệ, điều này thường được áp dụng khi mà cơ sở dữ liệu khá lớn và được lưu trên những máy tính khác nhau.

Dưới đây là các ưu, nhược điểm của các phương pháp phân mảng và tạo bản sao dữ liệu:

Ưu điểm:

- Tính hiệu quả: Dữ liệu được xử lý cùng nhau (có chung một đặc điểm) sẽ được sắp xếp vật lý gần nhau, ngược lại dữ liệu được xử lý khác nhau sẽ được lưu ở những vị trí cách nhau.
- Tối ưu hoá cục bộ: Mỗi một phân mảng của dữ liệu có thể được lưu trữ khác nhau nhằm tối ưu hoá việc xử lý dựa trên đặc trưng sử dụng của chúng.
- Tính an toàn: Khi sử dụng phân mảng, mỗi một người hoặc nhóm người dễ dàng được phân các

quyền thao tác, sử dụng khác nhau trên mỗi phân mảng đó, do vậy trong cùng một quan hệ, với phân mảng này, người sử dụng A có thể cập nhật, sửa đổi dữ liệu, trong khi với các phân mảng khác anh ta lại không được phép sửa đổi thậm chí được phép nhìn thấy.

- Tính dễ khôi phục: Khi chia thành các phân mảng, mỗi một phân mảng sẽ có kích thước nhỏ hơn rất nhiều so với cả quan hệ, do vậy khôi phục lại một phân mảng sẽ nhanh hơn. Và khi một phân mảng bị hỏng, thì các phân mảng khác vẫn hoạt động được, do vậy các lỗi xảy ra đối với mỗi phân mảng sẽ được cô lập.
- Tính cân bằng tải: Các phân mảng do bản chất là những tệp riêng biệt do vậy có thể được lưu trữ trên những vùng khác nhau (các ổ cứng hoặc các thiết bị lưu trữ) và làm tăng tốc độ xử lý do chúng có thể được thực hiện song song.

Nhược điểm:

- Tốc độ truy nhập không nhất quán: Các phân mảng khác nhau sẽ được truy nhập với các tốc độ khác nhau, do vậy dễ làm cho người sử dụng hiểu lầm. Cũng như vậy, khi có các thao tác đòi hỏi kết hợp các phân mảng lại sẽ làm cho tốc độ xử lý giảm đi đáng kể.
- Phức tạp: Việc phân chia thành các phân mảng có thể khiến người lập trình phải tốn công sức viết chương trình xử lý hơn, bởi bản chất các phân mảng là những tệp vật lý khác nhau.

- Dị thường: Các dị thường trong các thao tác thêm mới, sửa đổi, xoá bỏ có thể xảy ra và do vậy cần phải cẩn thận để tránh những trường hợp này.
- Cần nhiều không gian lưu trữ hơn và thời gian cập nhât: Dữ liệu có thể được lưu trữ trùng lặp trên các phân mảng, do vậy sẽ cần nhiều không gian lưu trữ hơn, khi cập nhật cũng phải cập nhật ở tất cả những nơi lưu trữ trùng lặp đó.

6.4. Tổ chức các tệp vật lý

Ngày nay, hầu hết các hệ quản trị cơ sở dữ liệu đều quản lý các tệp dữ liệu tập trung trong một cơ sở dữ liệu, do vậy trong mục này chúng tôi sẽ không trình bày cách thức lưu trữ các bản ghi vật lý trong mỗi tệp. Mục đích chính của mục này là giúp chúng ta hiểu được các cách tổ chức tệp mà các hệ quản trị cơ sở dữ liệu ngày nay thường dùng, trên cơ sở đó có thể lựa chọn các hình thức tệp phù hợp với các quan hệ đã được xây dựng.

Một tệp vật lý là một vị trí được đặt tên trong bộ nhớ thứ cấp được cấp phát cho mục đích lưu trữ các bản ghi vật lý. Một số hệ điều hành cho phép cắt một tệp vật lý thành những đoạn riêng biệt, tuy nhiên nó vẫn là trong suốt đối với người thiết kế cơ sở dữ liệu.

Tổ chức tệp là lựa chọn kỹ thuật sắp xếp vật lý các bản ghi của một tệp dữ liệu trên thiết bị lưu trữ thứ cấp. Khi lựa chọn tổ chức lưu trữ cho một tệp dữ liệu cụ thể trong cơ sở dữ liệu, có 7 yếu tố thường được xem xét đến, đó là:

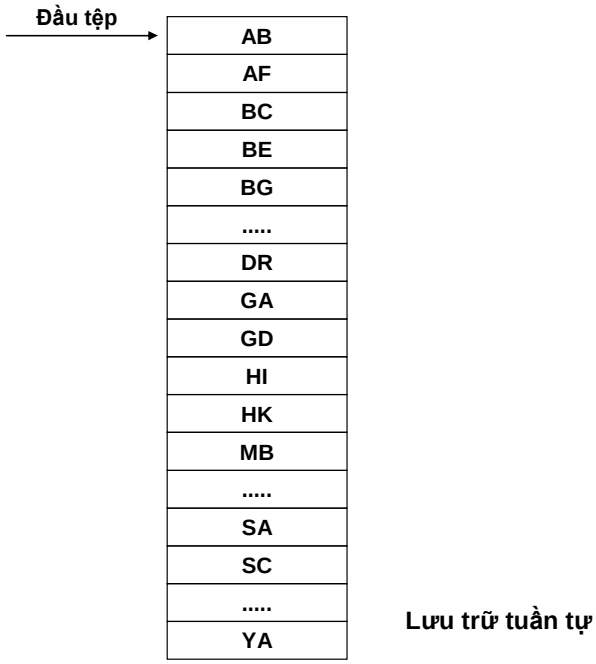
- Nâng cao tốc độ trả về dữ liệu.

- Xử lý nhanh các dữ liệu đưa vào cũng như các giao tác
- Sử dụng hiệu quả không gian lưu trữ
- Bảo vệ dữ liệu khỏi bị mất mát và lỗi
- Giảm thiểu việc phải tái tổ chức lại
- Dự trù được việc cấp phát không gian lưu trữ tiếp theo
- Đảm bảo an toàn đối với các sử dụng bất hợp pháp

Dưới đây là một số cách tổ chức tệp phổ biến trong các hệ quản trị cơ sở dữ liệu:

a. Tổ chức tệp tuần tự

Trong một tệp được tổ chức theo cơ chế tuần tự, các bản ghi trong cùng một tệp được lưu trữ tuần tự theo các giá trị của trường khoá chính. Để cấp phát vị trí cho một bản ghi, hệ quản trị cơ sở dữ liệu phải quét từ đầu tệp để tìm ra vị trí phù hợp cho bản ghi đó. Các tệp này thường được sử dụng khi bảng dữ liệu có khoá chính là các giá trị tăng dần và các bản ghi được đưa vào bảng theo đúng thứ tự tăng dần đó.



Hình 2.6.4.1 Tổ chức tệp tuần tự

b. Tổ chức tệp chỉ mục

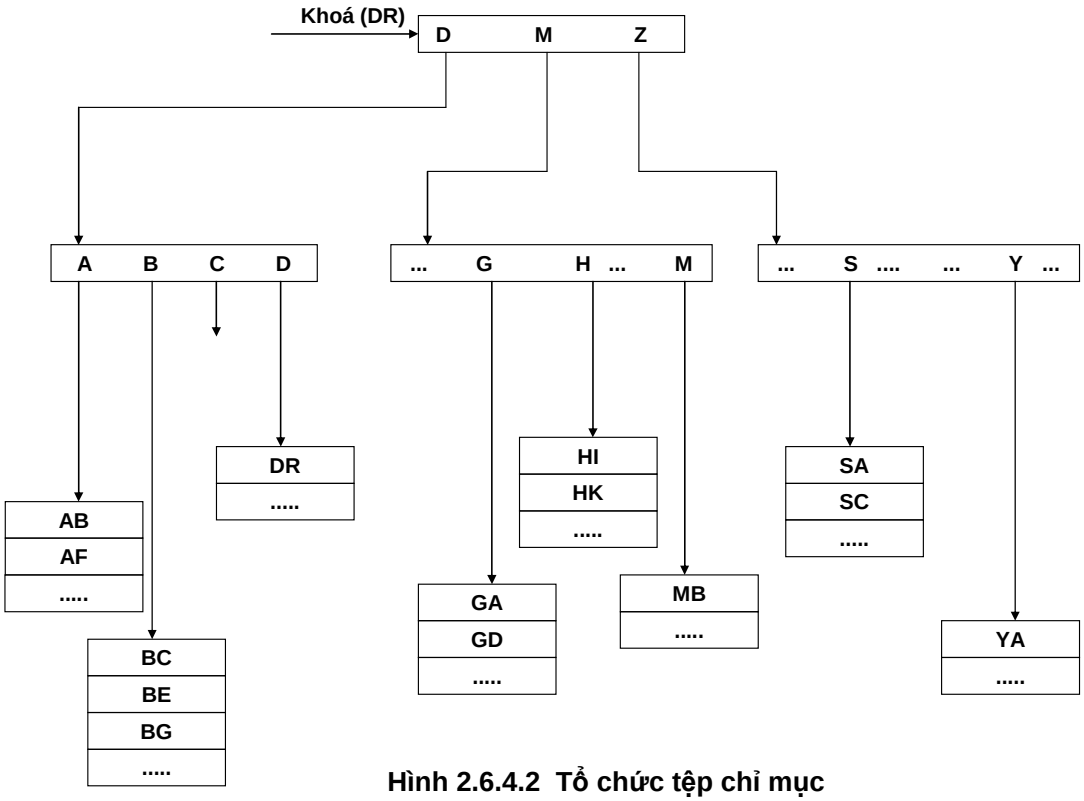
Trong phương pháp tổ chức tệp này, các bản ghi có thể được lưu trữ tuần tự hoặc không tuần tự và khi một bản ghi được tạo ra thì đi kèm với nó là một chỉ mục cũng được tạo ra tương ứng nhằm tạo ra một cơ chế để tìm kiếm nhanh chóng đến bản ghi đó. Cách tổ chức này giống cách tổ chức các ô phích tìm kiếm sách trong thư viện, thay vì độc giả phải tìm kiếm trên những tủ sách đồ sộ, họ có thể tìm kiếm trên các phiếu được sắp xếp có thứ tự trong các ô phích, trong mỗi phiếu đều lưu các giá trị mà độc giả có thể tìm kiếm cùng vị trí lưu trữ của quyển sách tương ứng. Tuy nhiên, trong máy tính có thể hơi khác, bởi mỗi một chỉ mục có thể được ánh xạ tới một giá trị khoá của một hoặc một số bản ghi. Một chỉ mục có thể trỏ tới duy nhất một bản ghi (như trong trường hợp sử dụng khoá chính làm chỉ mục) hoặc chỉ tới một số các bản ghi khi chúng có chung một giá

trị nào đó, ta gọi chỉ mục này là chỉ mục khoá thứ cấp. Các chỉ mục khoá thứ cấp đóng vai trò quan trọng trong việc hỗ trợ trong các truy vấn kiểu ad hoc đòi hỏi tốc độ cũng như trong các yêu cầu tạo báo cáo.

Một ví dụ về loại chỉ mục này đó là chỉ mục trên trường Order_date của quan hệ ORDER. Rõ ràng sẽ có một tập các bản ghi có chung giá trị trong trường Order_date bởi một ngày có thể có rất nhiều đơn hàng được đặt, và do vậy khi Order_date được sử dụng làm chỉ mục chúng ta sẽ dễ dàng tìm kiếm đến những đơn hàng được tạo ra trong một khoảng thời gian đưa vào.

Một số hệ quản trị cơ sở dữ liệu cũng hỗ trợ khả năng việc sử dụng chỉ mục trên các chỉ mục để tạo ra một phân cấp các chỉ mục. Bởi chúng ta biết rằng, một chỉ mục bản chất của nó cũng là một tệp dữ liệu, do vậy nếu nó quá lớn thì cũng có thể tổ chức theo phương pháp chỉ mục phân cấp này, tức là tạo chỉ mục của chỉ mục. Như trong hình 2.6.4.2, mỗi một chỉ mục có một giá trị khoá và một con trỏ trỏ tới một chỉ mục khác hoặc trỏ thẳng tới bản ghi dữ liệu. Ví dụ, để tìm bản ghi có khoá là ‘DR’, cấu trúc tệp được bắt đầu từ chỉ mục trên cùng và lấy con trỏ sau giá trị D, con trỏ này trỏ tới một chỉ mục khác với tất cả các khoá bắt đầu bằng chữ A đến chữ D theo thứ tự từ điển. Tiếp theo, chương trình tìm kiếm lại sử dụng tiếp con trỏ sau giá trị D trong chỉ mục này, con trỏ này trỏ đến một dãy các bản ghi có trường khoá bắt đầu bằng chữ D. Cuối cùng, việc tìm kiếm được tiến hành trên dãy các chỉ mục này để có thể chỉ ra được bản ghi có giá trị khóa là ‘DR’ hay là không có bản ghi đó.

Cấu trúc chỉ mục trong hình 2.6.4.2 được gọi là cấu trúc chỉ mục tuần tự. Trong cấu trúc này, các bản ghi chỉ mục được lưu trữ tuần tự theo giá trị của trường khoá sử dụng làm chỉ mục. Trong hình 2.6.4.2, các bản ghi chỉ mục được lưu trữ tuần tự trong mỗi khối dữ liệu, mỗi khối dữ liệu này có thể không được cấp phát gần nhau, nhưng cấu trúc chỉ mục tuần tự này có thể được sử dụng để lưu trữ các bản ghi dữ liệu của một tệp tuần tự. Một cách tổ chức khác đối với các tệp chỉ mục tuần tự này đó là sử dụng con trỏ liên kết giữa các khối, tức là thay vì được tổ chức phân cấp như trên, phần cuối mỗi khối đều có thêm một con trỏ để trỏ đến vị trí bắt đầu của khối tiếp theo. Ví dụ như trong hình 2.6.4.2, ở các khối trong mức thứ 2 chúng ta sẽ tổ chức thêm một con trỏ để trỏ sang khối tiếp theo. Loại tổ chức tệp được chỉ mục này được sử dụng với cả các tệp dữ liệu được tổ chức tuần tự hoặc không tuần tự (các bản ghi dữ liệu được lưu trữ tại những vị trí bất kỳ trong thiết bị lưu trữ thứ cấp). So với các tệp chỉ mục không tuần tự, các tệp chỉ mục tuần tự cho phép việc sử dụng không gian lưu trữ hiệu quả hơn và việc xử lý dữ liệu tuần tự rõ ràng có tốc độ nhanh hơn.



Hình 2.6.4.2 Tổ chức tệp chỉ mục

Một trong những thế mạnh của cách tổ chức tệp chỉ mục đó là khả năng có thể tạo ra các đa chỉ mục. Như trong thư viện, các đa chỉ mục có thể xây dựng là các chỉ mục dựa trên tiêu đề, dựa trên tên tác giả và thậm chí dựa trên chủ đề của cuốn sách. Do vậy, chúng ta có thể tạo ra một tệp chỉ mục khác trên cùng một tập bản ghi dữ liệu.

Ví dụ, nếu chúng ta sử dụng chỉ mục Sup_id để tìm ra địa chỉ của các bản ghi với giá trị mã nhà cung cấp bằng "CTY005", và sử dụng chỉ mục Unit_price để tìm ra những bản ghi có giá trị tại trường đơn giá nhỏ hơn 500$, sau đó có thể sử dụng giao của 2 tập bản ghi này để tìm ra những bản ghi thoả mãn cả 2 điều kiện trên.

Khả năng sử dụng nhiều chỉ mục trong cùng một truy vấn đóng vai trò quan trọng trong việc thực hiện các truy vấn

trong cơ sở dữ liệu quan hệ. Các phép toán logic AND, OR và NOT dễ dàng được thực hiện thông qua việc sử dụng các chỉ mục.

Cấu trúc tổng quát của một chỉ mục phân cấp như trong hình 2.6.4.2 được gọi là cây (có một nút trên cùng được gọi là gốc và các nút lá ở dưới cùng). Việc xử lý các nút trên cây thông qua các tính chất của cây dựa trên các cấu trúc khác nhau, do vậy việc xử lý cũng phụ thuộc vào từng cấu trúc cây đó. Tuy nhiên, hầu hết các hệ quản trị cơ sở dữ liệu đều xử dụng một cấu trúc cây đặc biệt được gọi là cây cân bằng (B-tree). Trong cấu trúc cây B-tree này, tất cả các nút lá (các bản ghi dữ liệu và các con trỏ tương ứng) đều có cùng khoảng cách đến nút gốc. Trong hình 2.6.4.2, tất cả các nút lá đều cách nút gốc 2 mức.

Tuy nhiên, như đã trình bày ở trên, mặc dù chúng ta là người thiết kế cơ sở dữ liệu mức vật lý, chúng ta cũng không cần phải can thiệp vào cấu trúc cũng như các thuật toán xử lý trên các cấu trúc cây này, bởi chúng đã được quản lý bởi hệ điều hành cũng như hệ quản trị cơ sở dữ liệu.

Ngoài ra, ngày nay phổ biến một loại chỉ mục mới, đó là chỉ mục bitmap. Trong cấu trúc này, mỗi một giá trị khoá được gán một bitmap. Ví dụ, giả sử trong quan hệ PRODUCT chúng ta có 10 bản ghi và trong thuộc tính Price chúng ta có 4 giá trị phân biệt trong 10 bản ghi đó. Một chỉ mục bitmap trên trường Price này sẽ có 4 bản ghi với 10bits cho mỗi bản ghi đó. Mỗi một bit trong bản ghi bitmap đó tương ứng với một hàng trong quan hệ Product. Giả sử, nếu bit thứ 8 tương ứng với Price có giá bằng 400$, thì điều đó cũng có nghĩa rằng bản ghi thứ 8 trong quan hệ

Product có giá trị bằng 400$ (trong mỗi một cột bit sẽ chỉ có một bit được mang giá trị 1).

Cấu trúc chỉ mục bitmap này rất thích hợp cho những trường có giá trị khác nhau ít, như các trường về giới tính, màu sắc. Một ưu điểm nữa đó là cấu trúc chỉ mục kiểu bitmap đòi hỏi rất ít không gian lưu trữ (thường khoảng bằng 25%) so với các chỉ mục theo cấu trúc cây thông thường. Sở dĩ như vậy bởi một đặc trưng cơ bản của chỉ mục bitmap đó là một bitmap có thể được sử dụng cho nhiều giá trị khoá. Ví dụ, chúng ta có thể thêm một tập các hàng vào cuối bảng trong hình 2.6.4.3 cho tất cả các giá trị của thuộc tính Room. Khi đó, mỗi cột bit có thể có 2 giá trị bằng 1, tương ứng với các giá trị cho Price và Room. Nếu một truy vấn muốn tìm kiếm các sản phẩm có giá bằng 100$ và phòng là phòng ăn có thể kết hợp giao của các bit của hàng Price 100$ và của Room dinner để tìm ra các bản ghi phù hợp với điều kiện đó. Đặc biệt thao tác với các bit tỏ ra vô cùng hiệu quả.

Price	1	2	3	4	5	6	7	8	9	10
100	0	0	1	0	1	0	0	0	0	0
200	1	0	0	0	0	0	0	0	0	0
300	0	1	0	0	0	0	0	0	0	0
400	0	0	0	1	0	1	0	1	1	0

Các sản phẩm số 3 và 5 có giá là 100 USD
Sản phẩm 1 có giá là 200 USD
Các sản phẩm 2,7 và 10 có giá là 300 USD
Các sản phẩm 4,6,8 và 9 có giá là 400 USD

HÌnh 2.6.4.3 Chỉ mục bitmap trên thuộc tính Price của quan hệ PRODUCT

c. Tổ chức các tệp băm

Trong kiểu tổ chức này, địa chỉ của mỗi bản ghi được xác định thông qua một thuật toán băm, đó là một thủ tục chuyển đổi giá trị của khoá chính thành địa chỉ của bản ghi.

Một thuật toán băm điển hình đó là sử dụng kỹ thuật chia giá trị khoá chính cho một số nguyên tố và sau đó sử dụng phần dư của phép chia này để gán cho vị trí lưu trữ tương đối. Ví dụ, giả sử chúng ta phải tổ chức cho một tệp lưu trữ khoảng 1000 bản ghi là nhân viên của một công ty, khi đó số nguyên tố phù hợp nhất đó là 997, bởi nó gần 1000 nhất. Giả sử ta cần tìm vị trí của bản ghi có giá trị khoá chính là 12396. Khi đó chúng ta chia giá trị này cho 997 và được số dư là 432. Do vậy bản ghi của nhân viên này sẽ được lưu trữ tại vị trí 432 trong tệp. Tuy nhiên, như chúng ta thấy, dù sao thì cũng tồn tại trường hợp có 2 bản ghi khi chia cho 997 cúng có số dư là 432, ví dụ nhân viên có giá trị khoá là 10429, khi đó phải tồn tại những thuật toán băm tiếp để xử lý những trường hợp xung đột này. Trên thực tế, phương pháp tổ chức tệp băm cũng tỏ ra khá có hiệu quả.

d. Tổ chức các tệp được phân cụm (clustering)

Một số hệ quản trị cơ sở dữ liệu cho phép lưu trữ các bản ghi của những tệp khác nhau liền kề nhau trong không gian bộ nhớ thứ cấp. Khi dó, một vùng đĩa liên tiếp không chứa các bản ghi có cùng cấu trúc và nó hình thành các phân cụm (hay phân nhóm). Ví dụ, trong Oracle, các hàng của một, hai hoặc nhiều hơn của các bảng liên quan với nhau được lưu trữ trong cùng một vùng lưu trữ trong đĩa cứng nếu chúng thường xuyên được kết nối với nhau. Như vậy chúng ta thấy rằng một phân nhóm được xác định bởi các

bảng với các cột mà giữa các bảng đó thường xuyên có sự kết nối với nhau.

Ví dụ bảng Customer và bảng Customer_Order thường xuyên được kết nối thông qua giá trị của trường Customer_ID, hoặc các hàng của bảng Price_Quote (lưu trữ báo giá của các mặt hàng của các nhà cung cấp khác nhau) có thể được phân nhóm với nhau thông qua trường Item_ID.

Việc phân nhóm giảm thời gian truy nhập đến các bản ghi liên quan với nhau so với các phương pháp cấp phát thông thường khác (các tệp khác nhau được lưu trữ ở những vị trí khác nhau).

Các lệnh định nghĩa dưới đây của cơ sở dữ liệu Oracle sẽ cho thấy các phân nhóm được định nghĩa như thế nào và các bảng được gán vào trong các phân nhóm đó ra sao. Đầu tiên, phân nhóm được xác định:

```
CREATE CLUSTER ORDERING (CLUSTERKEY
CHAR(25))
```

ORDERING là tên được đặt cho vùng phân nhóm, thuật ngữ CLUSTERKEY là bắt buộc, tuy nhiên nó sẽ không được sử dụng lại.

Sau đó các bảng được gán vào phân nhóm trên như sau:

```
CREATE TABLE CUSTOMER(
Cust_id   VARCHAR2(10) NOT NULL,
Cust_name      VARCHAR2(50) NOT NULL,
Cust_address  VARCHAR2(200))
CLUSTER ORDERING (Cust_id)
```

```
CREATE TABLE ORDER(
Order_id  VARCHAR2(10) NOT NULL
Cust_id   VARCHAR2(25) NOT NULL,
Order_date      DATE)
CLUSTER ORDERING (Cust_id)
```

Truy nhập đến các bản ghi trong phân nhóm được thông qua một chỉ mục trên khoá phân nhóm hay thông qua các hàm băm trên khoá phân nhóm. Các bản ghi phân nhóm được sử dụng hiệu quả nhất khi chúng ít bị thay đổi. Ngược lại, khi các bản ghi liên tục được thêm vào, sửa đổi, xoá bỏ thì không gian nhớ rác rất dễ sinh ra và do vậy khó khăn trong việc định vị các bản ghi liên kết với nhau. Tuy nhiên, kỹ thuật phân nhóm cũng vẫn là một lựa chọn cho người thiết kế các tệp vật lý nhằm nâng cao hiệu năng sử dụng các bảng.

6.5. Lựa chọn và sử dụng các loại chỉ mục

Việc xây dựng các chỉ mục đóng vai trò quan trọng trong quá trình thiết kế cơ sở dữ liệu mức vật lý. Các chỉ mục được sử dụng phải dựa trên các truy vấn (bao gồm cả những câu lệnh cập nhật dữ liệu vì thực sự chúng cũng là những câu lệnh truy vấn). Các thuộc tính được lựa chọn làm chỉ mục phải dựa trên các điều kiện trong mệnh đề where của câu lệnh truy vấn. Với những chỉ mục hợp lý, chúng sẽ làm tăng đáng kể tốc độ của câu lệnh truy vấn dữ liệu.

Tuy nhiên, khi xây dựng các chỉ mục cho một quan hệ dữ liệu, chúng ta cũng cần phải cân nhắc đến sự ảnh hưởng

của chúng đến tốc độ cập nhật dữ liệu trên quan hệ đó. Mặc dù một chỉ mục có thể làm tăng tốc độ truy vấn, nhưng khi cập nhật một giá trị trong các thuộc tính được sử dụng làm chỉ mục đó, hệ thống sẽ phải cập nhật lại cho tất cả các chỉ mục đã sử dụng thuộc tính đó, do vậy nó sẽ làm chậm quá trình cập nhật dữ liệu.

Như vậy, việc lựa chọn được một tập chỉ mục đòi hỏi sự hiểu biết thấu đáo về các kỹ thuật chỉ mục cũng như cơ chế tối ưu hóa truy vấn. Dưới đây là một số nguyên tắc giúp chúng ta có thể lựa chọn khi xây dựng các chỉ mục:

- Nguyên tắc 1 (khi nào thì xây dựng chỉ mục): Chúng ta không nên xây dựng một chỉ mục mà không có một câu lệnh truy vấn dữ liệu nào sử dụng chúng. Bất cứ khi nào có thể, chúng ta hãy lựa chọn những chỉ mục có thể làm tăng ít nhất nhiều hơn một câu lệnh truy vấn. Với những câu lệnh thường xuyên được thực hiện trên quan hệ có số lượng bản ghi lớn hầu như nhất thiết chúng ta phải tạo các chỉ mục cho chúng.
- Nguyên tắc 2 (lựa chọn khoá tìm kiếm): Những thuộc tính xuất hiện trong mệnh đề where sẽ là những thuộc tính ứng cử cho việc xây dựng các chỉ mục. Đối với những điều kiện lựa chọn giá trị chính xác, chúng ta nên cân nhắc đến những chỉ mục kiểu băm cho những thuộc tính xuất hiện trong các đối sánh đó. Còn đối với những điều kiện lựa chọn giá trị trong khoảng, chúng ta có thể lựa chọn chỉ mục loại cây cân bằng B-tree trên những thuộc tính đó. Loại chỉ mục này rất phù hợp trong trường hợp các quan hệ được chỉ mục ít bị cập nhật.

- Nguyên tắc 3 (khoá tìm kiếm đa thuộc tính): Chúng ta lựa chọn chỉ mục được xây dựng trên nhiều thuộc tính khi trong mệnh đề where xuất hiện so sánh trên nhiều thuộc tính. Khi xây dựng các chỉ mục trên các thuộc tính khoá tìm kiếm này, chúng ta nên cân nhắc thứ tự liệt kê các điều kiện so sánh, bởi chúng cũng ảnh hưởng đến việc phân tích và sử dụng chỉ mục.
- Nguyên tắc 4 (khi nào thì sử dụng phân nhóm - cluster): chú ý rằng đối với mỗi một quan hệ chỉ có nhiều nhất một chỉ mục kiểu phân nhóm, và việc sử dụng cơ chế phân nhóm sẽ làm ảnh hưởng đáng kể đến tốc độ xử lý trên quan hệ, do vậy việc lựa chọn chỉ mục loại phân nhóm cần phải được xem xét kỹ lưỡng. Thông thường các truy vấn dựa trên giá trị khoảng thường được hưởng lợi nhiều nhất từ cơ chế phân nhóm.
- Nguyên tắc 5 (lựa chọn chỉ mục băm hay cây cân bằng): Một chỉ mục B-tree thường được sử dụng nhiều hơn bởi nó phù hợp cho cả những truy vấn kiểu giá trị nằm trong một khoảng nào đó và cả những truy vấn có điều kiện bằng chính xác một giá trị cụ thể nào đó. Chỉ mục kiểu băm tỏ ra phù hợp hơn trong những trường hợp sau: Quan hệ được chỉ mục là quan hệ trong, và khoá tìm kiếm nằm trong điều kiện kết nối giữa các quan hệ. Hoặc trong các truy vấn sử dụng điều kiện bằng một giá trị chính xác nào đó.
- Nguyên tắc 6 (cân đối chi phí cho việc bảo trì chỉ mục): Sau khi xây dựng xong một chỉ mục, chúng ta cần phải xem xét đến sự ảnh hưởng của chúng đến

các công đoạn cập nhật dữ liệu: Nếu vấn đề bảo trì các chỉ mục làm chậm các phép cập nhật, hơn nữa những phép này lại thường xuyên được thực hiện thì khi đó chỉ mục tỏ ra không hiệu quả và chúng ta nên cân nhắc đến khả năng loại bỏ chúng.

Dưới đây ta hãy xét đến một ví dụ cụ thể, ví dụ này sẽ minh hoạ cho chúng ta nên lựa chọn chỉ mục như thế nào trong quá trình thiết kế cơ sở dữ liệu vật lý: Giả thiết, chúng ta có 2 lược đồ quan hệ gồm các trường như sau:

EMPLOYEE (Eno, Ename, Age, Sal, Hobby)

DEPARTMENT (Dno, Dname, Mgr)

Chúng ta hãy xem xét đến một câu lệnh truy vấn đơn giản sau:

```
Select    E.Ename,
          D.Mgr
From          EMPLOYEE E,
          DEPARTMENT D
Where     D.Dname = 'Toy' and
          E.Dno = D.Dno
```

Đầu tiên chúng ta hãy xem xét kế hoạch đánh giá cho truy vấn trên sẽ được thực hiện như thế nào. Đầu tiên, những bộ nào thoả mãn điều kiện D.Dname = 'Toy' được thực hiện, tiếp theo, trong số những bộ đó, sẽ được so sánh qua điều kiện thứ hai E.Dno = D.Dno để lựa chọn ra tiếp những bộ thoả mãn.

Chúng ta thấy cả 2 điều kiện trong mệnh đề where đều là điệu kiện bằng. Do vậy chúng ta sẽ phải nghĩ ngay đến việc

sử dụng chỉ mục kiểu băm trên những thuộc tính này. Điều kiện D.Dname = 'Toy' cho chúng ta biết nên xây dựng một chỉ mục cho trường Dname của quan hệ DEPARTMENT. Còn đối với điều kiện E.Dno = D.Dno, chúng ta đứng trước lựa chọn: nên xây dựng chỉ mục cho thuộc tính Dno của quan hệ EMPLOYEE hay thuộc tính Dno của quan hệ DEPARTMENT hay thậm chí cho cả 2 quan hệ đó? Chúng ta thấy rằng, với việc sử dụng chỉ mục trên thuộc tính Dname, chỉ có một số ít các bộ được trả về thỏa mãn điều kiện Dname = 'Toy', với những bộ này, chúng ta cần tìm ra những bộ tương ứng bên quan hệ EMPLOYEE thoả mãn điều kiện thứ 2 E.Dno = D.Dno, do vậy chúng ta nên xây dựng một chỉ mục trên trường Dno của quan hệ EMPLOYEE. Cũng cần phải lưu ý rằng, nếu chúng ta tiếp tục xây dựng một chỉ mục khác trên trường Dno của quan hệ DEPARTMENT thì cũng không làm tăng tốc độ của kết quả trả về, bởi những bộ thoả mãn điều kiện tìm kiếm trên trong quan hệ DEPARTMENT đã được trả về bằng cách sử dụng chỉ mục của trường Dname.

Như vậy các phương án đánh giá truy vấn đã giúp chúng ta có thể xác định được việc lựa chọn các chỉ mục. Đây là công việc mà chúng ta cần phải lưu tâm trong quá trình thiết kế cơ sở dữ liệu mức vật lý.

Tiếp theo, chúng ta sửa đổi câu lệnh trên đi một chút, giả sử mệnh đề where bây giờ sẽ là:

```
Where   D.Dname = 'Toy' and
        E.Dno = D.Dno and
        E.Age = 25.
```

Chúng ta hãy xem xét phương án đánh giá cho câu truy vấn trên. Một phương án tốt, đó là trả về những bộ trong quan hệ DEPARTMENT thoả mãn điều kiện trên trường Dname và đối sánh với những bộ trong quan hệ EMPLOYEE thông qua chỉ mục trên trường Dno, sau đó việc lựa chọn những bộ thoả mãn điều kiện trên trường Age mới được áp dụng trên những bộ này. Trong truờng hợp này, chúng ta thực sự không cần đến chỉ mục trên trường Dno của quan hệ EMPLOYEE nếu chúng ta đã có một chỉ mục trên trường Age. Những bộ trong quan hệ DEPARTMENT sẽ được trả về thông qua việc sử dụng chỉ mục trên trường Dname và những bộ trong quan hệ EMPLOYEE sẽ được trả về thông qua việc sử dụng chỉ mục trên trường Age sẽ được kết nối với nhau. Vì số lượng những bộ thoả mãn 2 điều kiện này không nhiều, nên vấn đề đối sánh giữa chúng không chiếm nhiều bộ nhớ cũng như thời gian của máy tính. Với phương án đánh giá này, chúng ta nhận thấy rằng nó cũng đạt được mục đích của câu lệnh truy vấn, tuy nhiên nó có vẻ không được hợp lý lắm nếu xét trên khía cạnh thực tế thiết kế thì sử dụng chỉ mục trên trường Dno tỏ ra hợp lý hơn, bởi đây là khoá ngoại lai được lấy từ quan hệ DEPARTMENT sang quan hệ EMPLOYEE, và nó có thể sẽ được sử dụng thêm ở những quan hệ khác khi liên kết với quan hệ EMPLOYEE.

Tiếp theo chúng ta hãy xem xét đến các câu lệnh truy vấn có điều kiện tìm kiếm dựa trên giá trị khoảng, bằng cách sửa đổi câu truy vấn trên như sau:

```
Select    E.Ename,
          D.Dname
From      EMPLOYEE E,
```

```
                DEPARTMENT D
Where    E.Sal between 10.000 and 20.000 and
         E.Hobby = ‘Stamps’ and
         E.Dno = D.Dno
```

Câu truy vấn này đã sử dụng một điều kiện tìm kiếm trên giá trị khoảng thông qua toán tử Between. Nó cũng tương đương với điều kiện:

```
10.00 <= E.Sal and E.Sal <= 20.000
```

Tuy nhiên, chúng ta nên sử dụng toán tử Beween bởi nó dễ hiểu và bộ tối ưu hoá truy vấn dễ phân tích hơn.

Quay trở lại câu lệnh truy vấn trên, cả 2 điều kiện E.Sal between 10.000 and 20.000 và E.Hobby = ‘Stamps’ đều dựa trên quan hệ EMPLOYEE. Do vậy, một kế hoạch đánh giá tốt sẽ là sử dụng quan hệ EMPLOYEE làm quan hệ ngoài và quan hệ DEPARTMENT làm quan hệ trong và chúng ta nên xây dựng một chỉ mục trên thuộc tính Dno của quan hệ DEPARTMENT. Còn đối với quan hệ EMPLOYEE, chúng ta sẽ lựa chọn chỉ mục nào để xây dựng? Một chỉ mục kiểu B-tree trên thuộc tính Sal sẽ là hợp lý, đặc biết nếu nó được phân cụm. Và một chỉ mục kiểu băm trên thuộc tính Hobby cũng giúp ích cho điều kiện tìm kiếm lựa chọn bằng. Nếu một trong 2 chỉ mục này đã có thì những bộ trong quan hệ EMPLOYEE sẽ được trả về thông qua chỉ mục đó, còn nếu có cả 2 chỉ mục trên thì bộ tối ưu hoá truy vấn sẽ lựa chọn chỉ mục sẽ trả về số lượng bộ thoả mãn ít hơn (hoặc điều kiện khoảng dựa trên Sal hoặc điều kiện bằng dựa trên Hobby), và điều này phụ thuộc vào dữ liệu trong quan hệ. Nếu chỉ có một số ít nhân viên có mức lương nằm trong khoảng tìm kiếm trên và có

nhiều nhân viên có thú vui chơi tem thì chỉ mục B-tree sẽ được lựa chọn, ngược lại chỉ mục băm trên thuộc tính Hobby sẽ được lựa chọn.

Đối với những hệ quản trị cơ sở dữ liệu lớn phổ biến hiện nay như Oracle hay SQL Server thường có các thống kê về dữ liệu trên các quan hệ được sử dụng thường xuyên, khi đó việc lựa chọn một trong 2 chỉ mục trên sẽ được dựa trên những thống kê này, ngược lại, nếu không có thống kê thì chỉ mục theo điều kiện bằng thường được lựa chọn trước, tất nhiên là đối với những truy vấn có giá trị lựa chọn tường minh như ví dụ trên. Còn đối với những điều kiện lựa chọn có giá trị không tường minh, thường dựa vào các biến truyền vào, hoặc các câu lệnh truy vấn động, thì bộ tối ưu hoá sẽ tự động phân tích để lựa chọn chỉ mục tại thời điểm chạy câu lệnh (run-ime).

Với những chỉ mục được phân cụm, những câu lệnh truy vấn dựa trên giá trị khoảng thường cho tốc độ trả về tốt hơn so với những chỉ mục không được phân cụm. Ví dụ như với truy vấn sau:

```
Select    E.Dno
From          EMPLOYEE E
Where     E.Age > 40
```

Trên thực tế, có khá nhiều truy vấn tìm kiếm trên nhiều trường khác nhau. Ví dụ, nếu chúng ta muốn trả về những bộ trong quan hệ EMPLOYEE sao cho có Age = 30 và Sal = 4000, vậy thì sử dụng một chỉ mục dựa trên khoá tìm kiếm <Sal,Age> (hoặc <Age,Sal>) sẽ là tốt hơn so với việc chỉ sử dụng một chỉ mục trên khoá tìm kiếm <Sal> hoặc một chỉ mục trên khoá tìm kiếm <Age>. Hay giả sử chúng

ta cùng có 2 chỉ mục dựa trên 2 khoá tìm kiếm Sal và Age độc lập nhau, việc sử dụng đồng thời cả 2 chỉ mục đó liệu có cho kết quả tốt hơn không? Chúng ta sẽ lựa chọn chỉ mục nào để xây dựng cho truy vấn này, cách tốt nhất đó là xây dựng chỉ mục hợp.

Chúng ta gọi chỉ mục trên các khoá tìm kiếm của nhiều thuộc tính là chỉ mục hợp. Ngoài việc hỗ trợ cho các truy vấn sử dụng điều kiện tìm kiếm bằng, chỉ mục hợp cũng có thể được sử dụng trong các truy vấn tìm kiếm giá trị khoảng.

Chúng ta hãy xem xét truy vấn sau:

```
Select    E.Eid
From          EMPLOYEE E
Where     E.Age between 20 and 30 and
          E.Sal between 3000 and 5000
```

Với một chỉ mục hợp trên các trường <Age,Sal> sẽ cho kết quả tìm kiếm khá tốt. Và đặc biệt tốt nếu chỉ mục có dạng B-tree và được phân cụm. Tuy nhiên, thứ tự của các thuộc tính tìm kiếm đôi khi cũng tạo ra sự khác biệt khá lớn, hãy xem tiếp truy vấn sau:

```
Select    E.Eid
From          EMPLOYEE E
Where     E.Age = 25 and E.
          E.Sal between 3000 and 5000
```

Trong câu lệnh truy vấn này, chỉ mục hợp theo thứ tự <Age,Sal> kiểu B-tree phân cụm sẽ có thời gian thực hiện khá ít bởi các bộ đầu tiên được sắp xếp theo các giá trị

trong trường Age, sau đó (nếu có nhiều hơn 1 bộ có cùng giá trị trong trường Age) các bộ sẽ được sắp xếp tiếp theo các giá trị trong trường Sal. Do vậy, tất cả các bộ với Age=25 được phân vào cùng một nhóm.

Ngược lại, nếu chúng ta sử dụng chỉ mục hợp theo thứ tự <Sal,Age> thì thời gian thực hiện truy vấn trên lại khác hẳn so với trường hợp đầu. Bởi trong trường hợp này, các bộ đầu tiên được sắp xếp theo Sal, và với 2 bản ghi có cùng một giá trị trong trường Age (như trong trường hợp này ta có Age=25) thì chúng cách nhau khá xa, do Sal của chúng ta là được lựa chọn theo giá trị khoảng.

Tuy nhiên việc sử dụng các chỉ mục hợp cũng cần phải được cân nhắc kỹ, bởi trong quá trình sử dụng cơ sở dữ liệu, khi dữ liệu ở các quan hệ được thay đổi thì các chỉ mục tương ứng cũng được thay đổi theo, và việc cập nhật, sửa đổi các chỉ mục hợp sẽ phức tạp hơn cũng như tốn kém hợp so với các chỉ mục đơn.

6.6. RAID

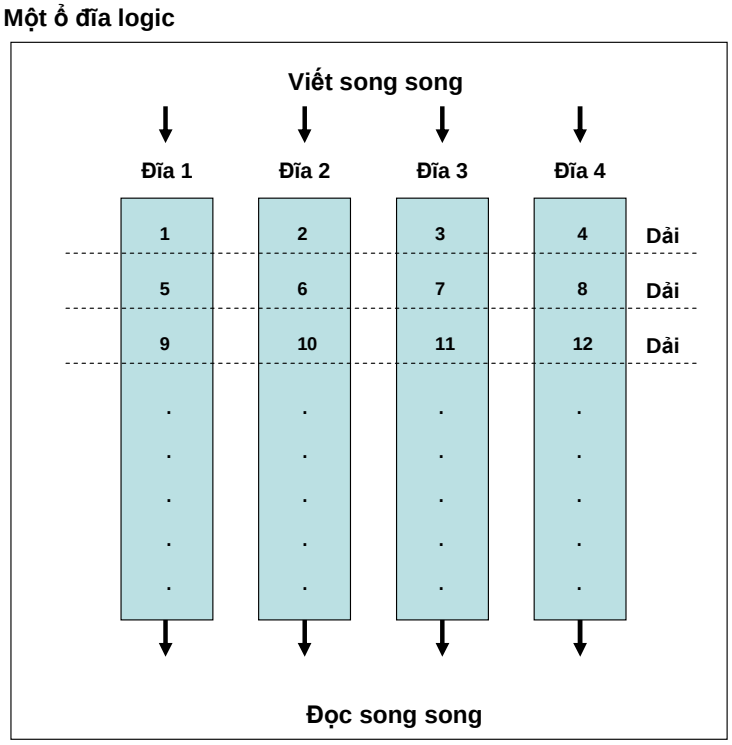
Như các phần trên chúng ta đã thấy việc phi chuẩn, tạo chỉ mục và phân cụm dữ liệu đã làm giảm thời gian truy xuất dữ liệu nếu dữ liệu được sắp xếp hợp lý. Với những bản ghi nhỏ, chúng ta có thể thiết kế để sắp xếp chúng liền nhau trong cùng một trang vật lý, tuy nhiên trên thực tế có rất nhiều những bản ghi lớn mà chúng ta phải sắp xếp chúng trên nhiều trang, do vậy các thao tác đọc đĩa tuần tự vào/ra chiếm khá nhiều thời gian để xử lý loại dữ liệu này. Do vậy các giải pháp phi chuẩn và phân cụm không đủ để giải quyết trong những trường hợp như thế này.

Hai xu hướng vẫn thường thấy trong công nghệ máy tính đó là giảm về giá và và giảm về kích thước. Chính hai xu hướng này đã cho phép người ta kết hợp nhiều thành phần của máy tính để có thể xử lý dữ liệu song song đặc biệt là với các thao tác vào ra của dữ liệu.

RAID, viết tắt của cụm từ Redundant Arrays of Inexpensive Disks, chúng ta có thể hiểu rằng đó là một tập ổ cứng được sử dụng đồng thời nhằm làm tăng tốc độ xử lý vào ra của dữ liệu, và cũng để đơn giản, chúng ta sử dụng cụm từ viết tắt này trong các phần tiếp theo. Điểm ưu việt của RAID đó là nó không làm thay đổi cấu trúc logic, vật lý của các chương trình ứng dụng cũng như các truy vấn trên cơ sở dữ liệu.

Đặc điểm quan trọng nhất của RAID đó là nâng cao hiệu năng truy xuất các tệp dữ liệu bằng cách xử lý song song.

Để có thể làm tăng tối đa tốc độ xử lý vào/ra của RAID, tất cả các ổ đĩa cứng cần phải liên tục được hoạt động. Ở đây chúng ta cần làm quen với một khái niệm mới, đó là dải (strip), một dải như trong hình sau là một tập các trang trên tất cả các ổ đĩa cứng tham gia vào RAID có cùng khoảng cách tương đối từ vùng đầu của đĩa. Như trong hình chúng ta thấy, tại một thời điểm máy tính có thể đọc 4 trang logic liên tiếp của dữ liệu theo cơ chế song song.

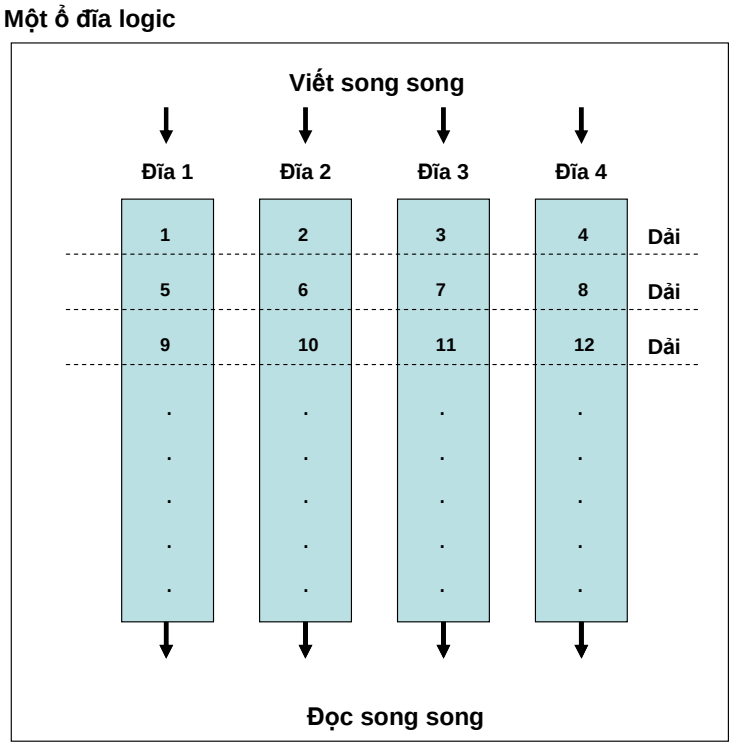


Hình 2.6.6.1 RAID với 4 ổ đĩa và các dải dữ liệu

Tuy nhiên, chúng ta cũng cần phải lưu ý một điểm quan trọng là khi sử dụng RAID sẽ làm tăng khả năng gặp lỗi đối với ổ cứng. Nếu trung bình, một ổ đĩa cứng có thể hỏng sau 1,2 triệu lần thao tác, thì đối với một RAID sử dụng 4 ổ đĩa cứng như vậy thì khả năng gặp lỗi sẽ tăng lên là 300.000 lần thao tác theo cơ chế song song. Để có thể khắc phục vấn đề này, nhiều loại công nghệ RAID được sử dụng để lưu trữ dữ liệu đã được đưa ra trong đó dữ liệu sẽ được lưu thêm một bản copy hoặc là sử dụng các mã sửa lỗi để có thể kịp thời khôi phục lại được dữ liệu trong trường hợp một ổ cứng bị hỏng. Dưới đây là một số mức RAID phổ biến:

- RAID-0: Đây là cách lưu trữ hiệu quả nhất cũng như có thời gian đọc/viết dữ liệu nhanh nhất đối với bất

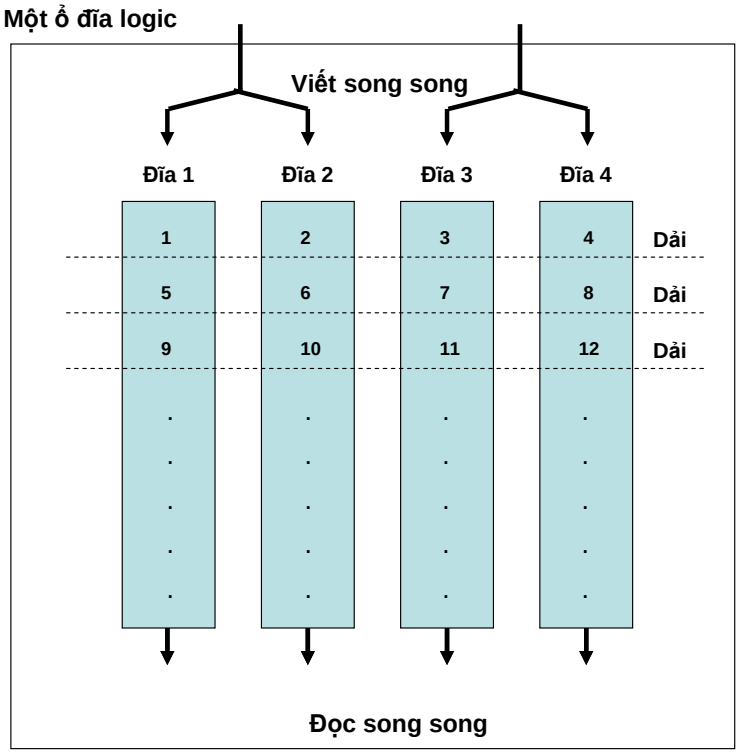
kỳ tập ổ đĩa cứng nào (hình 2.6.6.2). Trong RAID-0, mọi đĩa trong dãy đều được sử dụng để lưu trữ dữ liệu. RAID-0 thường được sử dụng khi tần suất đọc/viết dữ liệu cao cũng như nhu cầu truy suất các bản ghi dữ liệu lớn. Tuy nhiên, nhược điểm lớn nhất khiến RAID-0 ít được sử dụng đó là nó không có khả năng sửa lỗi cũng như khôi phục lại được dữ liệu khi một ổ đĩa cứng bị hỏng bởi tất cả ổ cứng đều được sử dụng tối đa, do vậy khả năng tin cậy không cao.



Hình 2.6.6.2 RAID - 0

- RAID-1: Mức RAID-1 này cho phép nâng cao khả năng phòng chống lỗi đó là lưu trữ dữ liệu dư thừa. Trong mức này, mỗi trang dữ liệu sẽ được lưu trữ thêm một bản trên một ổ cứng khác (hình 2.6.6.3). Số lượng đĩa cứng tham gia vào RAID sẽ được chia

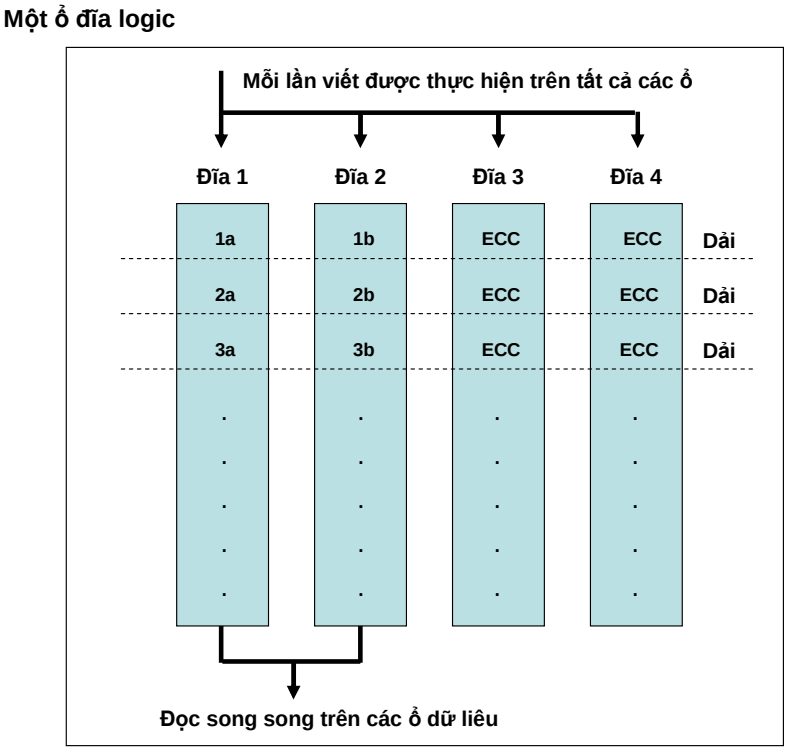
thành những cặp cùng nhau, một thao tác ghi một trang dữ liệu sẽ được tiến hành đồng thời trên mỗi cặp ổ cứng. Do vậy, với RAID-1 này, khả năng khắc phục lỗi rất cao, bởi các trường hợp để một trang dữ liệu được ghi trên 2 ổ cứng cùng xảy ra lỗi là rất ít. Trong RAID-1, các truy suất đọc sẽ hiệu quả hơn các truy suất viết bởi các truy suất viết phải được tiến hành song song vào từng cặp ổ đĩa, còn khi đọc thì có thể được thực hiện ở tất cả các ổ đĩa. RAID-1 này là một lựa chọn khá tốt cho các môi trường một cũng như đa người dùng.



Hình 2.6.6.3 RAID - 1

- RAID-2: trong mức này, người ta dành ra một số đĩa để lưu trữ các mã sửa lỗi. Chúng sẽ được sử dụng để tìm ra lỗi cũng như khôi phục lại trong các dải dữ liệu

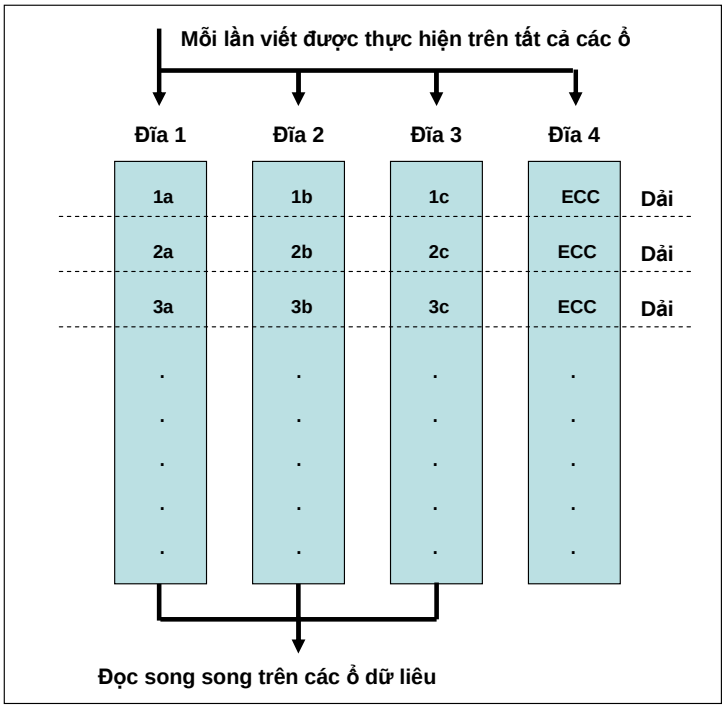
được lưu trong các ổ còn lại. Như vậy, nếu như với RAID-1 người ta sử dụng việc lưu trữ dư thừa dữ liệu để nhằm mục đích đảm bảo an toàn cho dữ liệu thì với RAID-2 (và các mức sau) này, người ta sử dụng những mã sửa lỗi hiệu quả hơn. Tuy nhiên, hầu hết các ổ đĩa cứng ngày nay đều đã được thiết kế sẵn các mã sửa lỗi trong mỗi trang dữ liệu vật lý nên RAID-2 ít được sử dụng hơn so với mức RAID-3 dưới đây.



Hình 2.6.6.4 RAID - 2

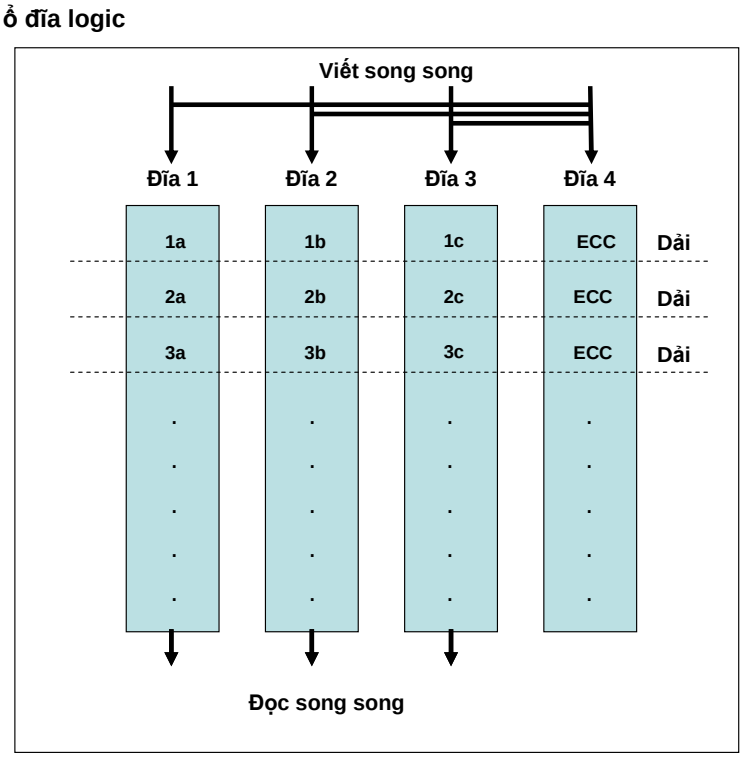
- RAID-3: Như với RAID-2, RAID-3 lưu dữ liệu trên tất cả các đĩa và chỉ dành duy nhất một đĩa để lưu trữ mã sửa lỗi (hình 2.6.6.5). Các mã sửa lỗi được tích hợp vào mỗi trang dữ liệu và được sử dụng để tìm ra các lỗi, nhưng mã sửa lỗi của trang không thể được sử dụng để sửa lỗi của chính trang đó. Giả sử rằng đĩa

số 3 bị lỗi, khi đó trang 3 trong mỗi dải dữ liệu có thể được khôi phục lại từ các trang dữ liệu còn lại kết hợp với trang sửa lỗi trong dải đó thông qua các phép toán logic OR. RAID-3 thường được sử dụng trong các cơ sở dữ liệu mà kích thước các bản ghi dữ liệu khá dài, tức là mỗi một bản ghi có thể được lưu trên một hoặc nhiều dải dữ liệu. Khi đó, mỗi một phép toán vào/ra logic sử dụng tất cả các ổ đĩa trong dãy RAID. Kết quả là nó không thể xử lý song song các yêu cầu từ nhiều người sử dụng vì mỗi một phép vào/ra logic sẽ sử dụng tất cả các ổ đĩa. Do vậy, RAID-3 thường được lựa chọn trong môi trường đơn người dùng với một chương trình duy nhất.



Hình 2.6.6.5 RAID - 3

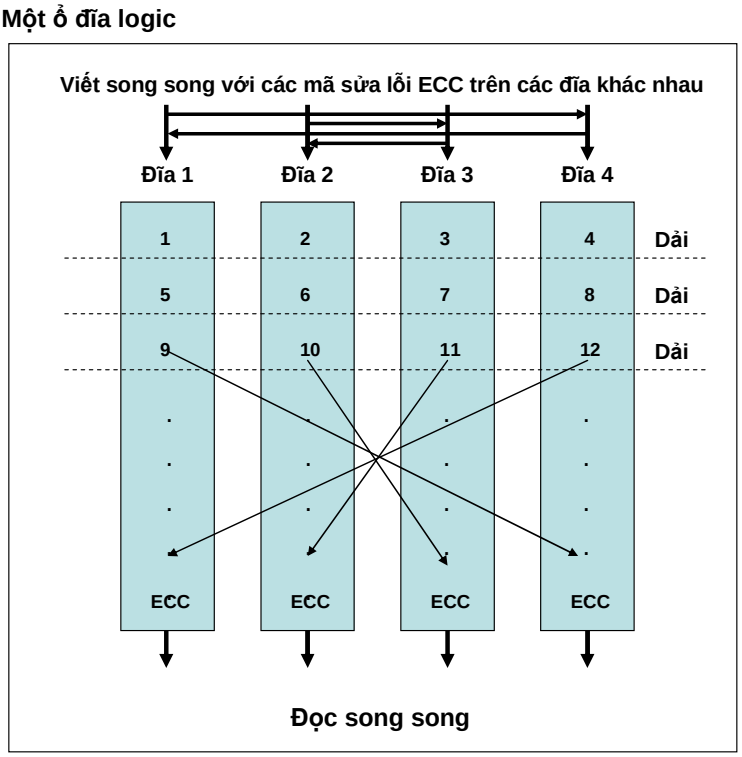
- RAID-4: Trong RAID-4, các mã sửa lỗi của các dải dữ liệu cũng được lưu duy nhất trên một ổ đĩa cứng như trong RAID-3, nhưng nó khác ở chỗ là nhiều bản ghi cùng được lưu trữ trên cùng một dải dữ liệu (hình 2.6.6.6). Do vậy các phép đọc dữ liệu trên các bản ghi khác nhau của cùng một chương trình hoặc nhiều chương trình khác nhau có thể được thực hiện song song qua các ổ đĩa. Tuy nhiên với RAID-4 này lại xảy ra trường hợp thắt cổ chai đối với các trang dữ liệu trên ổ lưu mã sửa lỗi, do khi mỗi bản ghi được cập nhật hệ thống cũng phải cập nhật các trang dữ liệu trên ổ đĩa chứa mã sửa lỗi này. Với những cơ sở dữ liệu liên tục có các phép cập nhật dữ liệu như các cơ sở dữ liệu xử lý giao dịch trực tuyến thì RAID-4 tỏ ra không hiệu quả do các phép cập nhật phải trờ đợi nhau trên ổ đĩa chứa mã sửa lỗi, và làm giảm quá trình xử lý song song. Vấn đề này được khắc phục bằng cách sử dụng RAID-5 như dưới đây.



Hình 2.6.6.6 RAID - 4

- RAID-5: Còn được gọi là mảng sửa lỗi quay vòng (rotating parity array). RAID-5 không sử dụng duy nhất một ổ đĩa làm nhiệm vụ lưu mã sửa lỗi, thay vào đó, mỗi một ổ đĩa đều chứa cả các trang dữ liệu cùng các trang chứa mã sửa lỗi. Do vậy các phép đọc có thể được thực hiện song song trên tất cả các ổ đĩa. Với phép viết, nó sẽ truy nhập đến một hoặc nhiều ổ đĩa để ghi dữ liệu cùng với một ổ đĩa để ghi mã sửa lỗi tương ứng cho trang dữ liệu đó. Các trang khác nhau sẽ lưu trữ các mã sửa lỗi của nó trên những ổ khác nhau. Do vậy, phép ghi dữ liệu cũng có thể được thực hiện song song nhưng khả năng bị trễ lại của nó đã được hạn chế hơn so với RAID-4 bởi việc cập nhật ngẫu nhiên các bản ghi cũng sẽ tạo ra các

phép cập nhật các trang sửa lỗi ngẫu nhiên trên các ổ cứng khác nhau.



Hình 2.6.6.7 RAID - 5

Phân tích hiệu quả của các mức RAID: Như chúng ta thấy, RAID-1,3 và 5 tỏ ra trội hơn so với các mức RAID khác về tính sửa lỗi. Trong đó, RAID-3 được xây dựng chủ yếu phục vụ cho các môi trường đơn người dùng hay cho các ứng dụng đặc biệt. Với hầu hết các ứng dụng cơ sở dữ liệu khác, người ta thường phải lựa chọn giữa RAID-1 và RAID-5. Đã có một số thử nghiệm được tiến hành để so sánh hai mức RAID này, kết quả cho thấy RAID-5 tỏ ra hiệu quả hơn trong việc lưu trữ dữ liệu do nó không lưu trữ dư thừa dữ liệu. Với các phép đọc dữ liệu, nó cũng tỏ ra hiệu quả hơn do sử dụng cơ chế song song trên tất cả các ổ đĩa cứng. Tuy nhiên, với phép ghi thì nó kém hơn từ 1.6

đến 3 lần so với RAID-1. Do vậy, so với RAID-1, RAID-5 ít được lựa chọn hơn trong những hệ thống sử dụng các tệp luôn luôn bị thay đổi. Ngoài ra các thử nghiệm cũng cho thấy rằng, để tăng tốc độ ghi dữ liệu lên gấp đôi so với phương pháp sử dụng một ổ đĩa, trong khi RAID-1 chỉ cần 4 ổ đĩa thì RAID-5 cần tới từ 7 cho tới 12 ổ đĩa.

Như vậy chúng ta có thể thấy rằng, RAID-1 là lựa chọn tốt nhất cho các cơ sở dữ liệu, đặc biệt với hệ thống chỉ có 2 ổ đĩa. Trong khi RAID-5 lại là lựa chọn tốt nhất cho các cơ sở dữ liệu đòi hỏi thời gian đọc dữ liệu nhanh và đỏi hỏi ít nhất từ 3 ổ đĩa trở lên (thông thường là 5).

6.7. Tối ưu hoá hiệu năng truy vấn

Tối ưu hoá hiệu năng xử lý của cơ sở dữ liệu cũng được xem là một mục đích quan trọng trong việc thiết kế cơ sở dữ liệu mức vật lý, đặc biệt đối với các cơ sở dữ liệu đòi hỏi phải đáp ứng một lượng lớn các dữ liệu tìm kiếm và trả về cho người sử dụng. Việc xử lý của cơ sở dữ liệu bao gồm các thao tác thêm, xoá và sửa đổi dữ liệu cũng như các phép tìm kiếm trả về dữ liệu.

Một trong những thay đổi đáng kể nhất trong những năm gần đây đó là việc tăng số lượng chip xử lý trong các máy chủ cơ sở dữ liệu. Các máy chủ cơ sở dữ liệu thường xuyên sử dụng công nghệ các chíp xử lý đối xứng (symmetric) hay các chip xử lý song song. Để có thể tận dụng được ưu điểm này, một số hệ quản trị cơ sở dữ liệu đã cho phép can thiệp xâu hơn vào hệ thống khi tiến hành thiết kế cơ sở dữ liệu như: các chiến lược chia nhỏ một câu lệnh truy vấn thành các phần khác nhau và mỗi phần được thực hiện song song bởi các chip khác nhau.

Một trong những cách tiếp cận thông dụng nhất đó là thực hiện câu lệnh truy vấn trên những phần khác nhau của cơ sở dữ liệu. Ví dụ với một bảng đã được chia thành phân vùng thì có thể sẽ được xử lý bởi các chip khác nhau trong cùng một câu lệnh truy vấn. Kết quả trả về sẽ được kết hợp lại để cung cấp cho người sử dụng, nhờ đó, thời gian thực hiện truy vấn giảm đi đáng kể.

Oracle là một trong những hệ quản trị cơ sở dữ liệu tiêu biểu nhất đã cho phép sử dụng cơ chế này. Ví dụ nếu bảng dữ liệu ORDER của chúng ta chứa khoảng vài triệu bản ghi, nếu được thiết kế bình thường thì khi có một truy vấn trên bảng này sẽ tỏ ra khá chậm. Tuy nhiên nếu máy chủ cơ sở dữ liệu có nhiều chip xử lý thì khi chúng ta thiết kế bảng đó, chúng ta có thể yêu cầu hệ thống sử dụng số lượng chip song song để thực hiện chẳng hạn như câu lệnh sau:

```
ALTER TABLE ORDER PARALLEL 4
```

Với các bảng khác nhau, chúng ta có thể điều chỉnh việc xử lý song song khác nhau.

Một lựa chọn khác đối với việc tận dụng được khả năng xử lý của các chip đó là khi viết câu lệnh truy vấn. Khi hệ quản trị cơ sở dữ liệu phân tích câu lệnh, nó sẽ quyết định lựa chọn kế hoạch thực hiện nào sẽ là tốt nhất dựa trên các dữ liệu thống kê về cơ sở dữ liệu, cấu trúc các bảng, số lượng các giá trị khác nhau của các trường,... để có thể lựa chọn sẽ sử dụng tệp chỉ mục nào cũng như bảng nào được thực hiện đầu tiên. Ngoài ra chúng ta có thể viết chú thích trong các câu lệnh đó để dẫn hướng cho bộ lựa chọn kế hoạch thực hiện câu lệnh đó một cách hợp lý nhất.

7. Thiết kế an toàn bảo mật

Các hệ quản trị cơ sở dữ liệu ngày càng được sử dụng rộng rãi để lưu trữ thông tin về tất cả các khía cạnh của một doanh nghiệp. Dữ liệu được lưu trữ trong một hệ quản trị cơ sở dữ liệu thường là tất yếu đối với các nghiệp vụ của doanh nghiệp và nó được xem là tài sản chung của doanh nghiệp. Thêm vào đó, để bảo vệ giá trị thực sự của dữ liệu, doanh nghiệp phải cân nhắc lựa chọn các giải pháp khác nhau nhằm đảm bảo tính bảo mật và kiểm soát được các truy nhập đến dữ liệu của các nhóm người sử dụng khác nhau với các lý do khác nhau.

Trong bước này chúng ta sẽ xem xét các vấn đề liên quan đến việc kiểm soát các truy nhập cũng như vấn đề đảm bảo an toàn trong một hệ quản trị cơ sở dữ liệu.

Khi chúng ta thiết kế an toàn bảo mật cho một cơ sở dữ liệu chúng ta cần phải cân nhắc 3 mục đích chính sau:

- Tính bí mật: Thông tin không thể được truy nhập tuỳ tiện. Một người sử dụng nếu không có thẩm quyền thì không thể truy nhập đến những phần mà không phân quyền cho anh ta. Ví dụ, một sinh viên không thể được phép truy nhập đến dữ liệu như về điểm của các sinh viên khác chẳng hạn.
- Tính toàn vẹn: Chỉ những người sử dụng có thẩm quyền mới được phép cập nhật dữ liệu. Ví dụ, các sinh viên được phép xem điểm của họ nhưng tất nhiên không được phép sửa đổi chúng.
- Tính sẵn có: Đối với những người sử dụng có thẩm quyền thì họ luôn luôn được phép truy nhập đến

những dữ liệu đã cấp quyền cho họ. Ví dụ, với giáo viên họ có thể được phép cập nhật điểm môn học cho các sinh viên mà họ đã dạy môn học đó.

Để có thể đạt được những mục đích này, một chính sách về an toàn bảo mật phải được xây dựng một cách rõ ràng và nhất quán. Cụ thể hơn, chúng ta phải xác định được những phần nào của dữ liệu cần phải được bảo vệ và những người sử dụng nào thì được phép truy nhập đến những phần dữ liệu nào. Tiếp theo, những cơ chế an toàn phải được triển khai theo những chính sách đã được xây dựng.

Chúng ta cũng cần phải nhấn mạnh đến mức độ an toàn ở những cấp độ khác nhau. Sự rò rỉ an toàn ở mức hệ điều hành hay kết nối mạng có thể phá vỡ các cơ chế đảm bảo an toàn cho cơ sở dữ liệu. Ví dụ, những rò rỉ này có thể cho phép những người sử dụng bình thường được phép đăng nhập vào máy chủ với quyền quản trị cao nhất mặc dù vai trò của họ không thể được như vậy. Ngoài ra các nhân tố liên quan đến con người cũng là một nguồn gây mất an toàn. Ví dụ như những người có thẩm quyền đặt mật khẩu quá ngắn hoặc quá dễ đoán. Trên thực tế những lỗi này chiếm một phần không nhỏ trong việc đảm bảo an toàn cho cơ sở dữ liệu, tuy nhiên ở đây chúng tôi sẽ không trình bày sâu, bởi nó đã được đề cập đến trong các giáo trình về hệ điều hành hay quản trị mạng.

Để dễ minh hoạ, trong phần này chúng tôi sẽ sử dụng những quan hệ có lược đồ như sau cho các phần trình bày tiếp theo:

SAILOR(Sid: integer, Sname: string, Rating: integer, Age:real)

BOAT(Bid: integer, Bname: string, Color: string)

RESERVER(Sname: string, Bid: integer, Day:date)

Trong đó:

- SAILOR là một quan hệ lưu danh sách các thuỷ thủ, Sid là thuộc tính mã hiệu, Sname là thuộc tính tên, Rating là thuộc tính cấp bậc và Age là thuộc tính tuổi.
- BOAT là một quan hệ lưu danh sách các tàu thuyền. Bid là thuộc tính mã hiệu, Bname là thuộc tính tên và Color là thuộc tính màu sắc.
- RESERVER: là một quan hệ đặt chỗ, cho biết thuỷ thủ nào (thuộc tính Sname) đã đặt chỗ trên chiếc tàu nào (Bid) vào ngày nào (Day). Chú ý rằng ở đây chúng tôi sử dụng thuộc tính Sname thay vì Sid trong quan hệ RESERVER để dễ dàng minh hoạ cho các câu lệnh tạo khung nhìn sau này.

7.1. Xác định các nhóm sử dụng và các quyền hạn

Như chúng ta đã biết, một cơ sở dữ liệu cho một doanh nghiệp thường chứa đựng một lượng lớn các thông tin cũng như các nhóm người sử dụng trên cơ sở dữ liệu đó. Hầu hết mỗi một người hoặc một nhóm người sử dụng chỉ làm việc trên một phạm vi dữ liệu nhỏ trong cơ sở dữ liệu đó để phục vụ cho công việc của mình. Do vậy nếu cho phép họ được phép truy nhập đến tất cả các dữ liệu trong cơ sở dữ liệu là điều không cần thiết. Chính vì vậy một hệ quản trị cơ sở dữ liệu thường cung cấp những cơ chế để có thể phân quyền và kiểm soát việc truy nhập dữ liệu này.

Với việc sử dụng các quyền do hệ quản trị cơ sở dữ liệu cung cấp, chúng ta có thể cấp phép các quyền truy nhập cũng như hình thức thao tác trên những đối tượng dữ liệu đến từng người hay nhóm người cụ thể.

Một hệ quản trị cơ sở dữ liệu sẽ lưu trữ đầy đủ thông tin về các quyền đã được cấp cho từng người sử dụng cụ thể, và cũng đảm bảo chắc chắn rằng tại bất kỳ thời điểm nào chỉ những người sử dụng được cấp phép thao tác trên những đối tượng dữ liệu đã được phân quyền cho họ.

SQL-92 hỗ trợ việc kiểm soát an toàn dựa trên các quyền thông qua các lệnh GRANT và REVOKE. Trong đó lệnh GRANT là lệnh cấp quyền cho người sử dụng, còn lệnh REVOKE là lệnh thu lại một quyền của người sử dụng. Và chỉ có những người có quyền hạn nhất định mới được phép thực hiện những lệnh này. Dưới đây chúng ta xem xét việc sử dụng các lệnh đó như thế nào đối với các đối tượng dữ liệu để phân quyền cho các đối tượng sử dụng khác nhau.

7.2. Xác định quyền hạn thao tác trên dữ liệu

SQL-92 hỗ trợ việc kiểm soát truy nhập thông qua các câu lệnh GRANT, REVOKE. Cấu trúc của chúng như sau:

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

Trong đó

- object có thể là một bảng hay một khung nhìn, ngoài ra có thể là các đối tượng khác như hàm hay thủ tục.

- privileges là các quyền thao tác trên các đối tượng đó, với các đối tượng là các bảng, khung nhìn chúng có thể là:

  SELECT: là quyền được phép truy nhập (đọc) đến object. Ví dụ nếu object là một bảng, một khung nhìn thì quyền này cho phép truy nhập đến tất cả các cột của bảng, khung nhìn đó, thậm chí cả những cột mà sau này được thêm vào bảng, khung nhìn đó.

  INSERT (column_name): Quyền được phép chèn thêm số liệu vào các cột liệt kê của object, nếu không có các cột thì tương đương với tất cả các cột.

  UPDATE(column_name): Quyền được phép sửa đổi dữ liệu trên các cột liệt kê của object.

  DELETE: Quyền được phép xoá bỏ dữ liệu của object.

  REFERENCES (column_name): quyền được định nghĩa các khoá ngoại lai (trong các bảng khác) tham chiếu đến các column_name của object.

  Với các hàm hay thủ tục chúng có thể là quyền thực hiện EXCUTE

- users: là tên người sử dụng trong cơ sở dữ liệu.

Nếu một user được gán thêm tuỳ chọn WITH GRANT OPTION, thì anh ta có thể tiếp tục gán các quyền đó cho những người sử dụng khác. Do vậy chúng ta cần phải đặc biệt lưu ý khi sử dụng tùy chọn này, bởi có thể sẽ gặp phải

những lỗi không mong muốn do chúng ta đã trao quyền gán tiếp cho một người sử dụng khác.

Nếu một khung nhìn được tạo dựa trên các bảng và các khung nhìn khác, thì nó cũng sẽ thừa hưởng tất cả các quyền trên từng bảng, từng khung nhìn đó. Người tạo khung nhìn ít nhất phải có quyền SELECT trên các đối tượng dữ liệu mà họ muốn dựa trên chúng để tạo nên khung nhìn mới đó. Người tạo khung nhìn nếu muốn được gán khung nhìn vừa tạo cho những người sử dụng khác thì trước đó anh ta phải được gán tuỳ chọn WITH GRANT OPTION trên tất cả các đối tượng dữ liệu tạo lên khung nhìn đó.

Kết hợp với các lệnh GRANT và REVOKE, các khung nhìn được xem là một công cụ hữu hiệu, một thành phần quan trọng trong các cơ chế đảm bảo an toàn trong các hệ cơ sở dữ liệu quan hệ. Cơ chế khung nhìn được sử dụng để tạo ra một "cửa sổ" trên một tập các dữ liệu sao cho phù hợp với một hoặc một nhóm người cụ thể nào đó. Bằng cách định nghĩa các khung nhìn, chúng ta có thể giới hạn phạm vi truy nhập đối với từng người sử dụng.

Ví dụ với câu lệnh tạo khung nhìn sau:

```
CREATE VIEW ACTIVESAILOR(Sname, Age, Day)
AS SELECT S.Sname, S.Age, R.Day
FROM SAILOR S, RESERVER R
WHERE S.Sname = R.Sname AND S.Rating >6
```

chúng ta đã giới hạn việc truy nhập dữ liệu. Thông qua việc gán quyền thao tác trên khung nhìn này, chúng ta đã không cho phép người sử dụng truy nhập dữ liệu trực tiếp trên các

bảng SAILOR hay RESERVER. Với các thông tin trong ACTIVESAILOR, họ chỉ có thể biết được những thông tin như một số thủy thủ đã đặt chỗ trên những con tàu nào và vào những ngày nào mà không hề biết các thông tin như mã số của các con tàu, màu sắc,...

Dưới đây, chúng ta hãy xem xét các lệnh GRANT, REVOKE:

Giả sử Dần là người đã tạo ra các bảng BOAT, RESERVER và SAILOR trên. Anh ta thực hiện các lệnh gán sau cho các người sử dụng khác:

```
GRANT INSERT, DELETE ON RESERVER TO Ất WITH GRANT OPTION

GRANT SELECT ON RESERVER TO Mùi

GRANT SELECT ON SAILOR TO Mùi WITH GRANT OPTION

GRANT UPDATE (Rating) ON SAILOR TO Tân

GRANT REFERENCES(Bid) ON Boat TO Giáp
```

Với câu lệnh GRANT thứ nhất, Ất có thể chèn, xoá các bản ghi trên bảng RESERVER và thậm chí được phép gán tiếp các quyền này cho những người sử dụng khác, bởi anh ta đã được Dần gán thêm cho cả tuỳ chọn WITH GRANT OPTION.

Với câu lệnh thứ hai và ba, Mùi được phép truy vấn trên các bảng SAILOR và RESERVER. Ngoài ra với bảng SAILOR, anh ta còn được phép gán tiếp các quyền này cho những người sử dụng khác, nhưng với bảng RESERVER thì lại không. Với quyền SELECT, Mùi có thể tạo ra một số khung nhìn dựa trên cả 2 bảng SAILOR và RESERVER

(như khung nhìn ACTIVESAILOR trên chẳng hạn) nhưng anh ta không thể gán quyền SELECT trên khung nhìn đó cho các người sử dụng khác.

Tuy nhiên, giả sử Mùi tạo một khung nhìn lựa chọn những thuỷ thủ trẻ có tuổi dưới 18 như sau:

```
CREATE VIEW YOUNGSAILOR (Sname,Age,Rating)
AS SELECT S.Sname, S.Age, S.Rating
FROM SAILOR S
WHERE S.Age < 18
```

Khung nhìn này chỉ dựa trên duy nhất bảng SAILOR, mà Mùi lại có quyền SELECT với tuỳ chọn WITH GRANT OPTION, do vậy anh ta cũng sẽ có quyền gán quyền SELECT trên khung nhìn này, thậm chí với tuỳ chọn WITH GRANT OPTION cho các người sử dụng khác, ví dụ anh ta sẽ gán quyền SELECT trên khung nhìn này cho Hợi và Dần:

```
GRANT SELECT ON YOUNGSAILOR to Hợi, Quý
```

Khi đó Hợi và Quý có thể thực hiện các truy vấn trên khung nhìn YOUNGSAILOR, nhưng rõ ràng họ không thể truy vấn trực tiếp trên bảng SAILOR.

Mùi cũng có thể định nghĩa các ràng buộc dựa trên thông tin trong các bảng SAILOR và RESERVER. Ví dụ, anh ta có thể định nghĩa một bảng với ràng buộc kiểm tra như sau:

```
CREATE TABLE SNEAKY (Maxrating INTEGER,
CHECK (Maxrating >=
(SELECT MAX(S.Rating)
FROM SAILOR S)))
```

Bằng cách chèn dần các giá trị tăng dần vào bảng SNEAKY, Mùi có thể tìm ra được giá trị cấp bậc Rating lớn nhất trong bảng SAILOR. Điều này minh hoạ cho việc tại sao SQL lại đòi hỏi người tạo ràng buộc trên một bảng liên quan đến bảng SAILOR thì cũng phải có quyền SELECT trên bảng SAILOR.

Quay trở lại câu lệnh thứ tư với quyền UPDATE được gán cho Tân, với lệnh này cô ta chỉ có thể cập nhật các giá trị trong cột Rating của bảng SAILOR. Ví dụ, Tân được phép thực hiện lệnh cập nhật dữ liệu sau cho tất cả các thuỷ thủ:

```
UPDATE SAILOR S
SET S.Rating = 8
```

Tuy nhiên, cô ta lại không thể thực hiện được lệnh cập nhật dữ liệu tương tự nếu mệnh đề SET được thay đổi thành S.Age = 25, vì Tân đã không được gán quyền UPDATE trên trường Age. Một lưu ý khác mà cũng là lỗi rất hay gặp phải trong trường hợp này, đó là nếu Tân thực hiện lệnh sau:

```
UPDATE SAILOR S
SET S.Rating = S.Rating - 1
```

Lệnh này rõ ràng không thể được thực hiện thành công bởi nó đòi hỏi phải có quyền SELECT trên cột Rating mà Tân thì lại không được gán cho quyền này! Như vậy chúng ta thấy rằng giá trị dùng để cập nhật cũng phụ thuộc vào quyền được gán.

Với Giáp, anh ta có thể tham chiếu cột Bid của bảng Boat như là một khoá ngoại lai cho một bảng khác. Ví dụ, Giáp

có thể tạo bảng RESERVER của riêng mình thông qua lệnh sau:

```
CREATE TABLE RESERVER (Sname CHAR(10) NOT NULL,
                       Bid  INTEGER,
                       Day  DATE,
                       PRIMARY KEY (Bid, Day),
                       UNIQUE (Sname),
                       FOREIGN KEY (Bid) REFERENCES BOAT)
```

Nếu Giáp không có quyền REFERENCES trên cột Bid của bảng Boat thì anh ta không thể thực hiện được câu lệnh tạo bảng trên bởi mệnh đề FOREIGN KEY đòi hỏi phải có quyền này.

Để phân biệt sự khác nhau giữa lệnh INSERT trên từng trường và trên toàn bảng SAILOR, chúng ta hãy xem tiếp:

Giả sử Dần chạy tiếp câu lệnh sau:

```
GRANT INSERT ON SAILOR TO Mùi
```

Khi đó, nếu có sự thay đổi các trường trên bảng này, như bổ sung thêm trường mới (thông qua câu lệnh ALTER TABLE) thì Mùi vẫn có quyền chèn thêm số liệu vào các trường này. Nhưng ngược lại, thay vì câu lệnh gán trên, Dần chỉ chạy câu lệnh gán như sau:

```
GRANT INSERT ON SAILOR(Sid), SAILOR(Sname), SAILOR (Rating) to Mùi
```

thì khi bảng SAILOR được bổ sung thêm các trường mới thì Mùi cũng chỉ được phép chèn dữ liệu vào 3 cột trên mà thôi.

Trái với lệnh GRANT là lệnh REVOKE, lệnh này rút lại quyền đã được cấp phát, có cấu trúc như sau:

```
REVOKE [GRANT OPTION FOR] privileges
ON object FROM users {RESTRICT | CASCADE}
```

Lệnh này nhằm loại bỏ một quyền hoặc tuỳ chọn WITH GRANT OPTION trên một đối tượng dữ liệu nào đó. Hai tuỳ chọn RESTRICT và CASCADE cũng phải được lựa chọn khi thực hiện câu lệnh này.

Tuỳ chọn CASCADE khác với RESTRICT ở chỗ, nếu khi thực hiện lệnh này thì tất cả những người sử dụng nào đã được gán tiếp bởi người sử dụng vừa bị thu lại quyền cũng sẽ bị thu lại quyền đó trừ khi anh ta được gán độc lập bởi một người sử dụng khác.

Để minh hoạ cho lệnh REVOKE này, chúng ta hãy xét tiếp ví dụ trên. Đầu tiên, hãy xem cái gì sẽ xảy ra nếu các câu lệnh sau lần lượt được thực hiện, trong đó Dần vẫn là người sở hữu (tạo ra) bảng SAILOR:

```
GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION      (Dần chạy)
GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION     (Sửu chạy)
REVOKE SELECT ON Salior FROM Sửu CASCADE
    (Dần chạy)
```

Trong 3 câu lệnh trên, câu lệnh thứ nhất được Dần chạy, sau câu lệnh này Sửu được quyền SELECT trên bảng SAILOR và anh ta cũng có quyền gán tiếp quyền SELECT này cho những người sử dụng khác. Và câu lệnh thứ 2 chính Sửu đã tận dụng quyền hạn này để thực hiện lệnh gán tiếp quyền SELECT trên bảng SAILOR cho Thìn và cũng với tuỳ chọn WITH GRANT OPTION. Như vậy sau câu lệnh thứ 2, Thìn có quyền SELECT trên bảng SAILOR như Sửu đã có quyền. Tuy nhiên anh ta được gán gián tiếp từ Sửu chứ không phải trực tiếp từ Dần là người đã tạo ra bảng đó và có quyền cao nhất. Do vậy, khi Dần thực hiện câu lệnh REVOKE thứ 3, anh ta sẽ loại bỏ quyền SELECT của Thìn, và với từ khoá CASCADE của câu lệnh này, nó cũng sẽ loại bỏ luôn cả quyền SELECT trên bảng SAILOR của Thìn.

Chúng ta hãy xem trường hợp khác với các câu lệnh sau:

```
GRANT SELECT ON SAILOR TO Sửu WITH GRANT
OPTION  (Dần thực hiện)

GRANT SELECT ON SAILOR TO Thìn WITH GRANT
OPTION  (Dần thực hiện)

GRANT SELECT ON SAILOR TO Thìn WITH GRANT
OPTION  (Sửu thực hiện)

REVOKE SELECT ON SAILOR FROM Sửu CASCADE
    (Dần thực hiện)
```

Với hai câu lệnh đầu, cả Sửu và Thìn đều được gán quyền SELECT trên bảng SAILOR trực tiếp từ Dần. Với câu lệnh thứ 3, Thìn nhận được quyền này gián tiếp từ Sửu. Nhưng với câu lệnh thứ 4, Sửu sẽ bị thu lại quyền SELECT do Dần đã gán. Tuy nhiên với Thìn thì sao? Thìn nhận được quyền

này từ cả Sửu, và trực tiếp từ cả Dần, do vậy Thìn vẫn có quyền này ngay cả khi Sửu không có.

Chúng ta xem tiếp các lệnh sau:

```
GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION  (Dần thực hiện)

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION  (Dần thực hiện)

REVOKE SELECT ON SAILOR FROM Sửu CASCADE
    (Dần thực hiện)
```

Như trên chúng ta đã chạy 2 lần lệnh gán quyền SELECT cho Sửu nhưng lại chỉ sử dụng một lệnh thu, khi đó liệu Sửu có còn quyền đó không? Theo SQL-92 chuẩn, câu trả lời là không và ba câu lệnh trên chỉ tương đương với hai câu lệnh sau cùng.

Tiếp tục, nếu Dần thực hiện tiếp:

```
GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION  (Dần thực hiện)

REVOKE GRANT OPTION FOR SELECT ON SAILOR
FROM Sửu CASCADE          (Dần thực hiện)
```

Hai câu lệnh này minh hoạ cho việc gán và loại bỏ tuỳ chọn WITH GRANT OPTION cho Sửu, sau câu lệnh thứ hai, Sửu vẫn có quyền SELECT trên bảng SAILOR, nhưng anh ta không thể gán quyền này tiếp cho những người sử dụng khác.

Thông thường trong những hệ quản trị cơ sở dữ liệu, khi một lệnh GRANT được thực hiện, các thông tin về nó sẽ được lưu trong từ điển dữ liệu như các thông tin về người gán, người được gán, quyền đã gán (bao gồm cả tên của đối

tượng đã gán) và cả tuỳ chọn WITH GRANT OPTION. Còn khi người sử dụng tạo một đối tượng của riêng mình thì coi như người quản trị cơ sở dữ liệu System đã gán cho anh ta đầy đủ các quyền được phép trên đối tượng mới đó.

Khi tiến hành phân quyền, chúng ta nên sử dụng một sơ đồ trong đó mô tả đồ thị gán quyền. Trong đồ thị, các nút tương ứng với người sử dụng, các cung cho biết quyền được gán như thế nào, ở đây cần lưu ý rằng các cung là định hướng và nhãn trên cung thể hiện quyền được gán.

Sẽ tồn tại một cung từ người sử dụng 1 đến người sử dụng 2 nếu người sử dụng 1 thực hiện một lệnh GRANT gán quyền cho người sử dụng 2. Ví dụ với các lệnh gán sau, chúng ta sẽ xây dựng được một đồ thị như trong hình 2.7.2.1:

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Dần thực hiện)

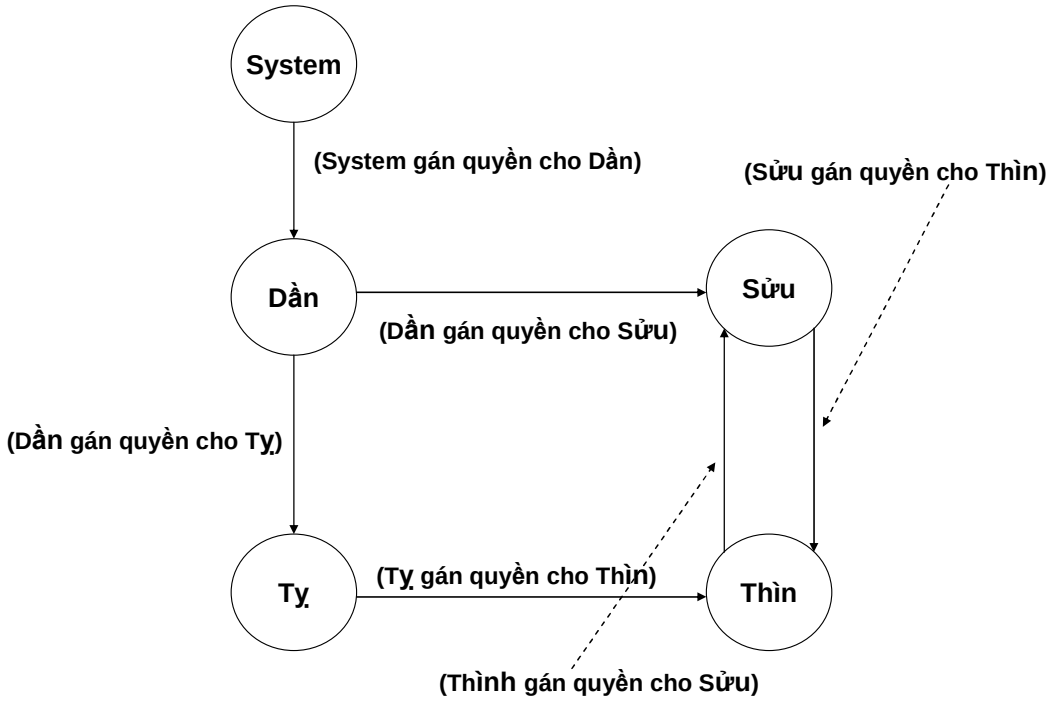
GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION (Sửu thực hiện)

GRANT SELECT ON SAILOR TO Sửu WITH GRANT OPTION (Thìn thực hiện)

GRANT SELECT ON SAILOR TO Tỵ WITH GRANT OPTION (Dần thực hiện)

GRANT SELECT ON SAILOR TO Thìn WITH GRANT OPTION (Tỵ thực hiện)

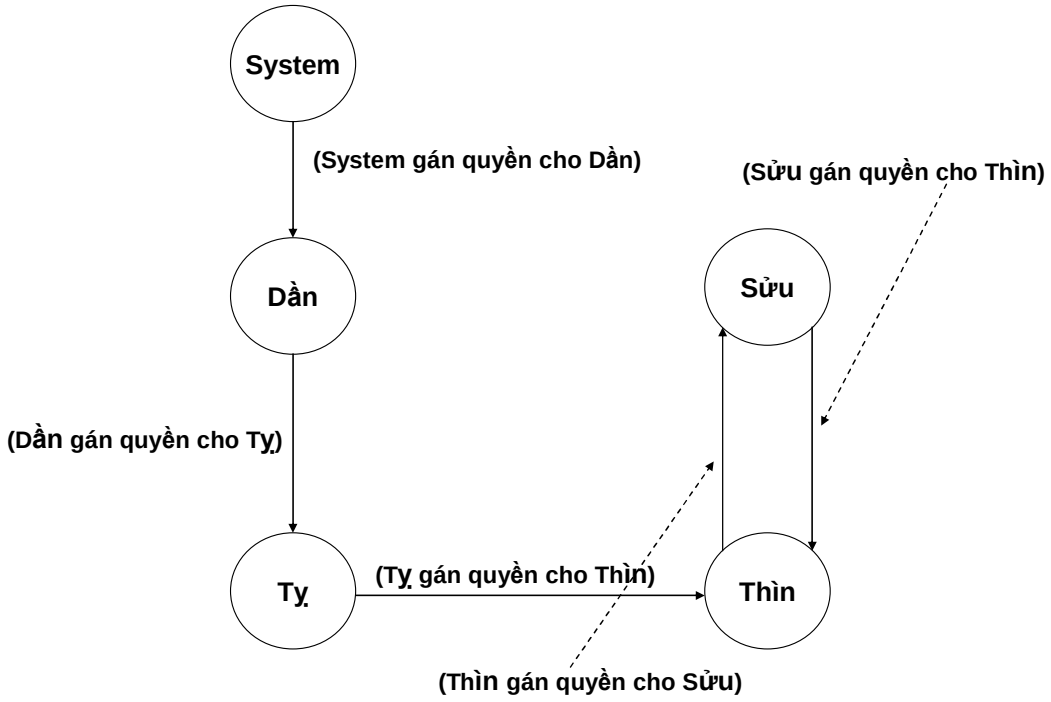
REVOKE SELECT ON SAILOR FROM Sửu CASCADE (Dần thực hiện)



Hình 2.7.2.1 Một đồ thị gán quyền

Giả thiết quyền được gán ở đây là quyền SELECT, như trong hình 2.7.2.1, do Dần sở hữu bảng SAILOR nên coi như anh ta được gán quyền bởi người quản trị cơ sở dữ liệu System. Từ đồ thị chúng ta thấy rằng, Sửu gán quyền cho Thìn, Thìn lại gán quyền lại cho Sửu tạo thành một chu trình. Ngoài ra Thìn cũng nhận được quyền từ Tỵ, mà Tỵ lại là người nhận quyền trực tiếp từ Dần.

Khi Dần quyết định loại bỏ quyền mà anh ta đã gán cho Sửu thì chúng ta sẽ thấy như sau: trên đồ thị cung nối từ Dần đến Sửu sẽ bị loại bỏ tương ứng với lệnh REVOKE. Tất cả các cung còn lại đều vẫn được giữ nguyên, lưu ý rằng ở đây mặc dù Sửu đã bị loại bỏ khỏi quyền SELECT từ Dần, nhưng Sửu vẫn nhận được quyền đó từ Thìn, và ngược lại nó lại gán quyền này cho Thìn, do đó cung từ Sửu đến Thìn vẫn tồn tại.



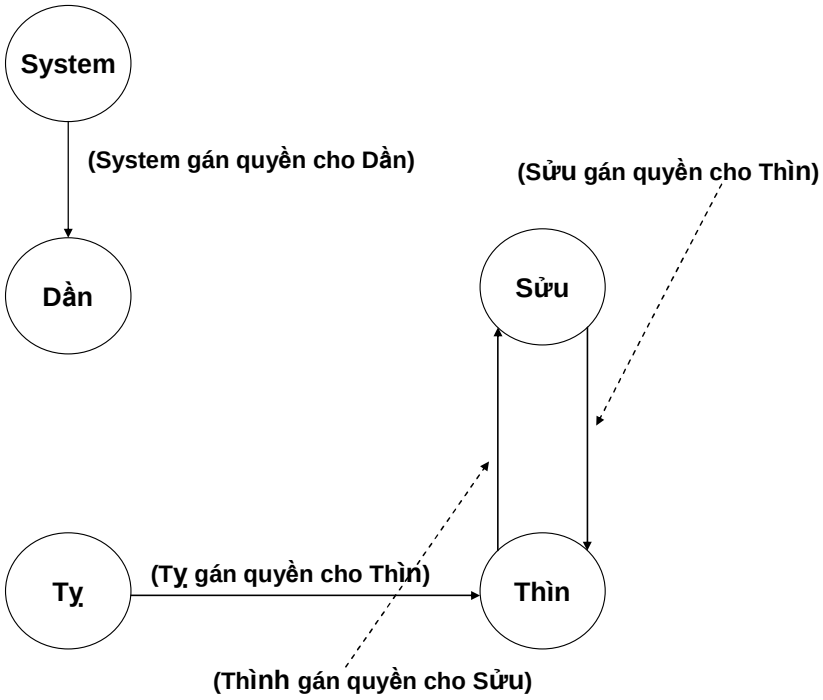
Hình 2.7.2.2 Dần loại bỏ quyền gán cho Sửu

Nói một cách tổng quát, khi loại bỏ một cung khỏi đồ thị, các cung còn lại sẽ được điều chỉnh dựa trên các đường từ nút System đến các nút.

Quay trở lại ví dụ trên, chúng ta thấy vấn đề này hơi phức tạp, do đã loại bỏ quyền SELECT của Sửu mà anh ta vẫn có quyền này!

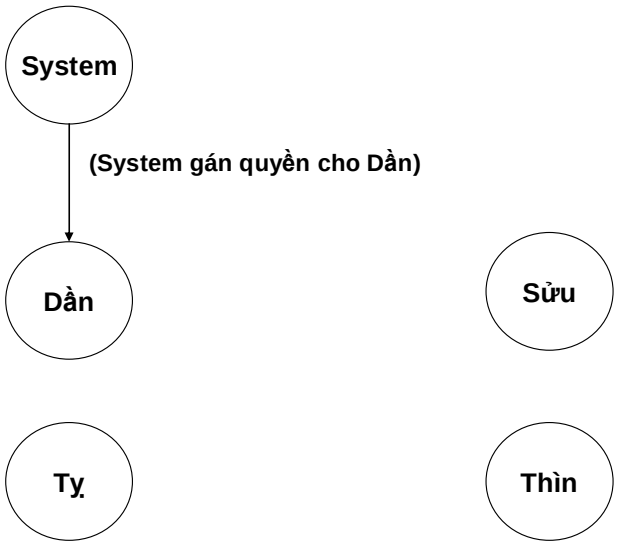
Do vậy khi thiết kế an toàn bảo mật thông qua việc gán các quyền trong cơ sở dữ liệu chúng ta phải đặc biệt lưu tâm đến vấn đề rơi vào vòng lặp loại như thế này.

Chúng ta hãy tiếp tục, nếu Dần thực hiện tiếp lệnh REVOKE để loại bỏ quyền SELECT của Tỵ, rõ ràng khi đó cũng nối từ Dần đến Tỵ sẽ bị loại bỏ và cung nối từ Tỵ đến Thìn cũng bị loại bỏ theo. Đến đây đồ thị chỉ còn hai nút nối với nhau là Sửu và Thìn



Hình 2.7.2.3 Dần loại bỏ tiếp quyền gán cho Tỵ

Tuy nhiên chúng ta nhận thấy rằng các cung này không được nối từ nút System, do đó theo qui định chúng cũng sẽ bị loại bỏ. Cuối cùng Dần đã thu lại toàn bộ các quyền đã gán cho mọi người.



Hình 2.7.2.4 Sơ đồ phân quyền cuối cùng

Dưới đây chúng ta hãy xem tiếp các lệnh GRANT và REVOKE trên các khung nhìn cùng với các ràng buộc toàn vẹn:

Như trên chúng ta đã thấy, việc tạo các khung nhìn phụ thuộc vào các bảng mà nó được xây dựng lên. Trên thực tế các lệnh gán trên các bảng cũng có thể bị thay đổi do đó các khung nhìn cũng sẽ bị thay đổi theo các lệnh gán mới. Dưới đây là một số lưu ý đối với những trường hợp này:

- Một khung nhìn sẽ bị loại bỏ nếu người tạo ra khung nhìn đó không có quyền SELECT trên những bảng hoặc khung nhìn đã được sử dụng để tạo nên khung nhìn đó.
- Nếu người tạo ra khung nhìn có thêm các quyền trên các bảng được sử dụng để tạo nên khung nhìn thì nó cũng tự động có thêm các quyền đó.

- Cần phải phân biệt rõ ràng giữa các quyền SELECT và REFERENCES

Dưới đây chúng ta xét tiếp các ví dụ để minh hoạ cho những điểm lưu ý trên.

Giả sử Dần là người đã tạo ra bảng SAILOR, anh ta gán quyền SELECT trên bảng này cho Mùi với tuỳ chọn WITH GRANT OPTION. Sau khi nhân được quyền này, Mùi tạo ra một khung nhìn mới có tên là YOUNGSAILOR dựa trên bảng SAILOR này và gán quyền SELECT trên khung nhìn này cho Hợi. Với Hợi, anh ta lại tạo một khung nhìn tiếp theo được đặt tên là FINEYOUNGSAILOR dựa trên khung nhìn YOUNGSAILOR do Mùi đã gán cho, các câu lệnh lần lượt như sau:

```
GRANT SELECT ON SAILOR TO Mùi WITH GRANT OPTION  (Dần thực hiện)

CREATE VIEW YOUNGSAILOR (Sname,Age,Rating)
AS SELECT S.Sname, S.Age, S.Rating
FROM SAILOR S
WHERE S.Age < 18                                  (Mùi thực hiện)
GRANT SELECT ON YOUNGSAILOR TO Hợi
      (Mùi thực hiện)

CREATE VIEW FineYOUNGSAILOR (Sname,Age,Rating)
AS SELECT S.name, S.Age, S.Rating
FROM YOUNGSAILOR S
```

WHERE S.Rating > 6 (Hợi thực hiện)

Bây giờ chúng ta hãy xem điều gì sẽ xảy ra nếu Dần loại bỏ quyền SELECT trên bảng SAILOR từ Mùi. Khi đó Mùi sẽ không còn quyền SELECT trên bảng SAILOR, do đó khung nhìn YOUNGSAILOR do anh ta tạo ra sẽ không còn tác dụng, và theo hệ quả, khung nhìn FINEYOUNGSAILOR của Hợi cũng không còn tác dụng do không thể SELECT trên khung nhìn YOUNGSAILOR của Mùi. Các định nghĩa về hai khung nhìn này sẽ bị loại bỏ khỏi từ điển dữ liệu. Ngay cả khi sau đó Dần thực hiện lại quyền gán SELECT cho Mùi thì chúng ta vẫn cứ phải tiến hành lại các lệnh tạo khung nhìn trên.

Hãy xét một trường hợp khác, giả sử các câu lệnh gán và tạo khung nhìn trên vẫn được thực hiện, nếu Dần gán thêm quyền INSERT trên bảng SAILOR cho Mùi, khi đó, nếu khung nhìn do Mùi tạo ra có quyền cập nhật thì tự động quyền INSERT này cũng có tác dụng trên khung nhìn đó. Tuy nhiên nếu Mùi không gán tiếp quyền INSERT này cho Hợi thì khung nhìn do Hợi tạo ra vẫn chỉ có quyền SELECT mà không có quyền INSERT.

Tiếp theo chúng ta hãy xét đến ví dụ minh hoạ cho sự khác nhau giữa quyền SELECT và REFERENCES:

Trong ví dụ này chúng ta giả sử Dần vẫn là người tạo ra bảng BOAT, nhưng Hợi lại là người tạo ra bảng RESERVER, và anh ta muốn tham chiếu đến khoá chính Bid trong bảng BOAT của Dần. Để có thể thực hiện được điều này, Dần nhất thiết phải cấp quyền REFERENCES trên cột Bid của bảng BOAT cho Hợi.

Ngược lại, giả sử nếu Dần chỉ gán quyền SELECT trên bảng BOAT cho Hợi thì Hợi cũng không thể thực hiện được việc tham chiếu này cho dù anh ta có thể SELECT toàn bộ dữ liệu trên bảng BOAT của Dần.

Trong trường hợp Dần gán quyền REFERENCES trên cột Bid cho Hợi và giả thiết Hợi đã tạo xong tham chiếu của khoá ngoại lai trong bảng RESERVER của mình đến cột Bid của bảng BOAT. Sau một thời gian Dần lại muốn loại bỏ quyền REFERENCES này đi, khi đó bảng Reserve của Hợi vẫn tồn tại nhưng ràng buộc khoá ngoại lai đến trường Bid của bảng BOAT sẽ bị loại bỏ.

Để có thể hiểu được tại sao SQL-92 chuẩn lại phải đưa ra hai quyền REFERENCES và SELECT thay vì chỉ sử dụng mỗi quyền SELECT. Chúng ta hãy xem các trường hợp sau:

Nếu Dần chỉ gán quyền SELECT mà không gán quyền REFERENCES cho Hợi, và khi tạo bảng RESERVER Hợi không xác lập tuỳ chọn REFERENCEScho khoá ngoại lai Bid, khi đó Dần dễ dàng xoá bỏ các bản ghi trong bảng BOAT, và chính điều này sẽ gây ra sự dị thường trong bảng RESERVER của Hợi.

Trong trường hợp ngược lại, nếu Hợi được gán đúng quyền REFERENCES và anh ta sử dụng quyền này trong tuỳ chọn tham chiếu của khoá ngoại lai Bid trong bảng RESERVER của mình thì khi Dần xoá các bản ghi trong bảng Boat, nếu giá trị trong cột Bid đã được sử dụng trong bảng RESERVER của Hợi thì Dần không thể xoá được các bản ghi đó. Như vậy chúng ta thấy chỉ khi đó dữ liệu trong cơ sở dữ liệu mới được đảm bảo tính toàn vẹn và nhất quán.

7.3. Mã hoá dữ liệu

Một hệ quản trị cơ sở dữ liệu có thể sử dụng các cơ chế mã hoá để bảo vệ thông tin của mình trong những trường hợp mà các cơ chế an toàn thông thường không đáp ứng đủ. Ví dụ như các tin tặc có thể lấy cắp dữ liệu thông qua các thiết bị gắn vào đường truyền (tap), hay qua các thiết bị lưu trữ dữ liệu. Tuy nhiên nếu dữ liệu được mã hoá trong các công đoạn lưu trữ và truyền dẫn thì hệ quản trị cơ sở dữ liệu có thể đảm bảo rằng những dữ liệu đã bị đánh cắp trên đường truyền hay trong khi lưu trữ sẽ khó có thể sử dụng được hoặc người ta phải mất rất nhiều thời gian và công sức để giải mã nó.

Ý tưởng cơ bản đằng sau vấn đề mã hoá đó là việc áp dụng các thuật toán mã hoá. Việc áp dụng một thuật toán mã hoá phải đi kèm với một thuật toán giải mã cùng với khoá được giải. Dữ liệu sau khi qua thuật toán mã hoá sẽ trở thành dữ liệu được mã hoá và truyền tới người nhận, người nhận muốn hiển thị được dữ liệu thật phải áp dụng một thuật toán giải mã tương ứng thông qua khoá giải mã mà người đã mã hoá cung cấp cho. Cách tiếp cận này đã trở thành một chuẩn mã hoá dữ liệu (DES) xuất hiện từ năm 1977, với một thuật toán mã hoá chứa các ký tự thay thế và hoán vị. Điểm yếu nhất của cách tiếp cận này đó là khoá được sử dụng mã hoá có thể bị đánh cắp do người ta phải cung cấp nó cho những người muốn nhận được dữ liệu đã bị mã hóa.

Một cách tiếp cận khác đối với vấn đề mã hoá đó là sử dụng phương pháp mã hoá với khoá công khai, phương pháp này đã trở nên khá phổ biến trong những năm gần đây. Sơ đồ mã hoá đã được đề xuất bởi Rivest, Shamir và

Adleman, được gọi là RSA. Nguyên lý của cách tiếp cận này như sau, mỗi một người sử dụng đều có một khoá mã hoá, khoá này được công bố công khai đến tất cả mọi người và một khoá giải mã (được sử dụng trong thuật toán giải mã), khoá này thì chỉ mình người sử dụng đó được biết và anh ta sẽ bị lộ dữ liệu nếu để mất khóa giải mã này. Các thuật toán mã hoá và giải mã cũng có thể được công khai cho mọi người biết. Để dễ hình dung hơn đối với cách tiếp cận này, chúng ta hãy xem ví dụ sau:

Ví dụ: Giả sử có một người tên là Sam, bất kể người nào muốn gửi dữ liệu cho anh ta mà không muốn bị đánh cắp dữ liệu đều phải sử dụng khoá công khai cùng với thuật toán do Sam đã công bố. Dữ liệu sẽ được mã hoá và gửi đến Sam. Khi dữ liệu đến Sam, anh ta phải sử dụng khoá riêng của mình cùng với thuật toán giải mã để giải mã dữ liệu, do vậy ngay cả khi dữ liệu bị đánh cắp trên đường truyền thì cũng không thể giải mã được bởi khoá giải mã chỉ có mình Sam biết được. Cách tiếp này đã khắc phục được hạn chế của cách tiếp cận theo chuẩn mã hoá dữ liệu DES thông thường nên nó được sử dụng khá rộng rãi.

Vấn đề chính đối với cách tiếp cận mã hoá với khoá công khai ở chỗ đó là làm sao chọn được khoá mã hoá và khoá giải mã. Về kỹ thuật, các thuật toán mã hoá với khoá công khai sử dụng các hàm một chiều là các hàm mà việc xác định chiều ngược lại rất khó tìm hoặc hầu như không thể tính nổi. Ví dụ, một thuật toán RSA rất nhiều người biết đến đó là dựa trên các số nguyên tố, như chúng ta đã biết việc xác định xem một số có phải là số nguyên tố không thì rất dễ, tuy nhiên để tìm ra các thừa số nguyên tố của một số không phải là số nguyên tố thì lại là vô cùng khó. Người ta

đã chứng minh được rằng để tìm ra các thừa số nguyên tố của một số có trên 100 chữ số thì với một máy tính chạy nhanh nhất hiện nay cũng phải mất hàng năm.

Chúng ta hãy đi xâu tìm hiểu xem thuật toán RSA như thế nào. Giả sử dữ liệu cần được mã hoá là một số nguyên I, để lựa chọn khoá mã hoá và khoá giải mã, Sam đầu tiên lựa chọn một số nguyên rất lớn và đặt tên là limit, chúng ta giả thiết rằng số limit này lớn hơn số nguyên lớn nhất I muốn mã hoá. Sam đã lựa chọn số limit này là tích của 2 số nguyên tố p và q (p*q). Sau đó Sam lựa chọn một số nguyên tố khác e lớn hơn cả p và q làm khoá mã hoá. Cả limit và e đều được đưa ra công khai với mọi người và sẽ được sử dụng trong thuật toán mã hoá.

Bây giờ Sam phải lựa chọn cho mình khoá giải mã d theo một phương pháp đặc biệt dựa trên cả p,q và e (ví dụ, d được lựa chọn sao cho d*e = 1 mode((p-1)*(q-1))). Điểm thiết yếu ở đây đó là d rất dễ tính ra nếu biết cả p,q và e, nhưng lại rất khó tính nếu chỉ biết e và limit (bằng tích của p*q). Bởi như trên chúng ta thấy rằng rất khó để xác định các thừa số nguyên tố của limit.

Ví dụ một công ty buôn bán đã quý cho phép nhận các đơn đặt hàng trên mạng internet và lưu trữ chúng trong một cơ sở dữ liệu, các yêu cầu đặt ra sẽ là:

- Chỉ công ty mới có khả năng đọc được một đơn đặt hàng. Ngoài ra khách hàng (ví dụ Sam sẽ là khách hàng của công ty) đặt đơn hàng đó của công ty cũng muốn các đơn hàng của anh ta được giữ bí mật.

- Công ty có thể kiểm tra chính xác một đơn hàng có thực sự được đặt bởi khách hàng Sam hay không, hay bởi một người nào khác tự nhận là Sam.

Để đảm bảo các yêu cầu đó, công ty sẽ yêu cầu mỗi một khách hàng phải lựa chọn cho mình một khoá mã hoá công khai (giả sử Sam chọn khoá là eSam) và một khoá giải mã bí mật (dSam). Công ty cũng lựa chọn cho mình một khoá mã hoá công khai (eEm). Khoá giải mã của công ty dEm sẽ được giữ bí mật. Các khách hàng cũng được yêu cầu là họ phải giữ bí mật khoá giải mã của mình (như dSam).

Bây giờ chúng ta hãy xem hai yêu cầu trên được thoả mãn như thế nào. Để đặt một đơn hàng, Sam sử dụng khoá công khai eEm của công ty để mã hoá đơn hàng của mình và gửi cho công ty. Công ty khi nhận được đơn hàng này sẽ sử dụng khoá giải mã dEm để giải mã. Như vậy chúng ta thấy rằng giai đoạn này thoả mãn yêu cầu thứ nhất trên, bởi dEm chỉ duy nhất công ty kinh doanh có thể biết được. Tuy nhiên, ở đây chúng ta sẽ đặt câu hỏi là khoá eEm là công khai với tất cả mọi người, vậy thì một kẻ nào đó có thể mạo danh là Sam và gửi đến công ty một đơn hàng cũng được mã hoá bởi khoá eEm thì công ty làm sao biết được? Rõ ràng theo cách này thì công ty không thể biết được đơn hàng đó có đúng do Sam đặt không, họ có thể cung cấp cho mỗi một khách hàng một account để đăng nhập hệ thống và đặt đơn hàng, tuy nhiên khi số lượng khách hàng lớn (hàng nghìn chẳng hạn) thì cách cung cấp tài khoản đăng nhập cho khách hàng cũng tỏ ra không hiệu quả.

Để có thể khắc phục vấn đề trên, và để xác định chính xác đơn đặt hàng là do đúng khách hàng của công ty đặt, công ty đã yêu cầu các khách hàng của mình thực hiện các bước

sau khi đặt một đơn hàng. Thay vì sử dụng ngay lập tức khóa công khai eEm của công ty để mã hoá đơn hàng, khách hàng Sam đầu tiên áp dụng thuật toán giải mã, sử dụng mã riêng của mình dSam (công ty có thể không biết khoá này) để mã hoá. Rõ ràng, sau bước này do Sam không sử dụng thuật toán mã hoá nên đơn hàng sẽ ở một dạng đặc biệt, tiếp theo Sam mới sử dụng khoá công khai eEm của công ty để mã hoá tiếp và gửi cho công ty.

Khi công ty nhận được đơn hàng này, đầu tiên họ sẽ giải mã nó sử dụng khoá riêng dEm của mình. Bước này đơn hàng được đưa về dạng đặc biệt do Sam đã mã hoá ở bước đầu. Tiếp theo công ty áp dụng tiếp thuật toán mã hoá đối với bản này và sử dụng khoá công khai eSam do Sam cung cấp và họ sẽ nhận được đơn hàng cuối cùng. Như vậy theo cách này công ty sẽ nhận được đơn hàng đảm bảo do chính khách hàng Sam gửi đến bởi cả hai khoá bí mật dSam và dEm đều được sử dụng. Như vậy nếu đơn hàng là giả mạo thì nó không thể biết khoá giải mã dSam của Sam, và kết quả mã hoá với khoá công khai eSam của Sam sẽ không thể đưa về đơn hàng nguyên thuỷ ban đầu. Ngoài ra, một ưu điểm nữa đó là công ty cũng không hề biết khoá dSam của khách hàng Sam do đó Sam có thể hoàn toàn tin tưởng rằng không ai có thể biết được khoá bí mật của mình trừ khi anh ta để lộ nó. Chúng ta có sơ đồ cho quá trình này như sau:

> Sam → khoá riêng dSam → khoá công khai eEm → khoá riêng dEm → khóa công khai eSam → công ty.

Việc sử dụng mã hoá với khoá công khai không giới hạn trong các hệ quản trị cơ sở dữ liệu, nó được sử dụng rộng rãi ở khắp các ứng dụng khác, và nhờ có nó mà thương mại điện tử trên mạng Internet đã phát triển đáng kể.

7.4. Vai trò của người quản trị

Người quản trị cơ sở dữ liệu đóng một vai trò quan trọng trong việc đảm bảo các tính năng an toàn liên quan của cơ sở dữ liệu. Anh ta phải là người thực sự có kinh nghiệm và hiểu biết sâu sắc hệ quản trị cơ sở dữ liệu mà mình phải quản lý

Người quản trị cơ sở dữ liệu luôn là người sử dụng có quyền cao nhất mà chúng ta thường gọi là System account (như trong các ví dụ trên đã trình bày) và chịu trách nhiệm về toàn bộ các vấn đề an toàn của hệ thống cơ sở dữ liệu. Dưới đây là một số vai trò và trách nhiệm của người quản trị:

Các vai trò và trách nhiệm cơ bản:

- Cài đặt các phần mềm cho hệ thống và đặc biệt là hệ quản trị cơ sở dữ liệu.
- Tiến hành cấu hình cả phần cứng và phần mềm để cố gắng đảm bảo cho hệ quản trị cơ sở dữ liệu hoạt động hiệu quả nhất.
- Quản trị các vấn đề an toàn cho các đối tượng trong cơ sở dữ liệu.
- Kiểm soát các vấn đề an toàn cho cơ sở dữ liệu.
- Tham gia đóng góp vào việc thiết kế các chính sách an toàn bảo mật cho các đối tượng dữ liệu.
- Kiểm tra các vấn đề liên quan đến hiệu năng của hệ cơ sở dữ liệu và nếu cần thiết điều chỉnh lại các tham hệ thống.

- Thực hiện công việc sao lưu và khôi phục cho hệ cơ sở dữ liệu định kỳ.
- Tiến hành định kỳ các công việc bảo trì cho hệ cơ sở dữ liệu.

Ngoài ra người quản trị cơ sở dữ liệu cũng có thể thực hiện một số công việc sau:

- Tham gia phân tích dữ liệu..
- Tham gia thiết kế cơ sở dữ liệu.
- Hỗ trợ người phát triển chương trình.
- Tham gia vào quá trình đề xuất ra các chuẩn trong doanh nghiệp.
- Tham gia vào quá trình đánh giá và kiểm tra các phần mềm trong hệ thống.
- Lưu trữ lại các thông số về môi trường cho hệ quản trị cơ sở dữ liệu.
- Tham gia đánh giá các vấn đề về mua sắm nâng cấp các phần mềm cũng như phần cứng mới.
- Lên kế hoạch và đề xuất khi thấy hệ thống cần thiết phải nâng cấp để đáp ứng các nhu cầu tiếp theo của doanh nghiệp.

Chúng ta sẽ có dịp tìm hiểu cụ thể những vai trò và trách nhiệm trên của người quản trị hệ cơ sở dữ liệu trong chủ đề tiếp theo: Quản trị hệ thống cơ sở dữ liệu.

8. Kết luận

Việc chia quá trình xây dựng cơ sở dữ liệu thành các bước nên được xem như là việc phân nhóm các bước khác nhau trong thiết kế. Trên thực tế, mặc dù chúng ta có thể bắt đầu với sáu bước trình bày trên, việc thiết kế một cơ sở dữ liệu hoàn chỉnh có lẽ vấn đòi hỏi những khâu điều chỉnh thêm mà trong đó cả sáu bước thiết kế trên đều được xen vào và được lặp lại cho tới khi thoả mãn các yêu cầu thiết kế. Ngoài ra, ở đây chúng ta cũng không trình bày một số bước quan trọng khác như bước cài đặt thiết kế cơ sở dữ liệu, các bước xây dựng và cài đặt các lớp ứng dụng chạy trên hệ quản trị cơ sở dữ liệu. Tất nhiên, trên thực tế những bước này có thể dẫn tới việc xem xét lại quá trình thiết kế cơ sở dữ liệu.

Những khái niệm và công nghệ của hệ quản trị cơ sở dữ liệu quan hệ là rất hữu ích cho những người muốn cài đặt và bảo trì một hệ thống cơ sở dữ liệu. Đối với những người làm công tác quản trị, điều quan trọng là họ phải biết một hệ quản trị cơ sở dữ liệu hoạt động như thế nào. Nắm vững những thành phần bên trong của một hệ cơ sở dữ liệu sẽ giúp ích rất nhiều cho công việc thiết kế thành công một cơ sở dữ liệu, đặc biệt là trong các bước thiết kế vật lý và điều chỉnh cơ sở dữ liệu.

9. Bài tập

1. Vẽ sơ đồ cho mỗi liên kết sau:

   - Giữa kiểu thực thể sinh viên STUDENT và khoá học COURSE (sinh viên đăng ký học các khoá học khác nhau)

- Giữa kiểu thực thể sách BOOK và các bản photo BOOK_COPY.
- Giữa kiểu thực thể giảng viên TEACHER và khoá học COURSE
- Giữa kiểu thực thể ngân hàng BANK và chi nhánh BRANCH (mỗi một chi nhánh luôn thuộc vào một ngân hàng nhất định).

2. Lấy một số ví dụ về kiểu thực thể mạnh và một số kiểu thực thể yếu đi cùng với nó.

3. Thế nào là thuộc tính khoá chính, thuộc tính khoá ngoại lai? Lấy các ví dụ minh hoạ

4. Thế nào là liên kết một-một, liên kết một-nhiều, liên kết nhiều-nhiều.? Lấy ví dụ minh hoạ trên thực tế cho từng loại liên kết đó.

5. Vẽ sơ đồ thực thể - liên kết cho những trường hợp sau (bạn có thể bổ sung thêm một số giả thiết nếu thấy cần):

- Một công ty có một số lượng nhân viên. Các thuộc tính của kiểu thực thể nhân viên EMPLOYEE bao gồm Emp_id (đồng thời là thuộc tính định danh), Name, Address, Birth_date. Đồng thời công ty cũng quản lý một số dự án, các thuộc tính của kiểu thực thể dự án PROJECT này bao gồm Proj_id (thuộc tính định danh), Proj_name, Start_date. Mỗi một nhân viên có thể tham gia vào một hoặc một số dự án, hoặc có thể không tham gia vào bất cứ dự án nào. Tuy nhiên, mỗi một dự án phải bao gồm ít nhất một nhân viên. Công ty có nhu cầu quản lý việc cấp kinh phí cho từng dự án đến từng nhân viên. Bạn hãy tạo thêm những kiểu thực thể mới để giải quyết nhu cầu đó, và

xác định các liên kết với hai kiểu thực thể EMPLOYEE và PROJECT đã có.

- Một phòng nghiên cứu bao gồm một số nhân viên EMPLOYEE cùng một số dự án, mỗi một dự án sẽ sử dụng một số thiết bị nào đó, chúng ta đặt tên cho kiểu thực thể thiết bị là EQUIPMENT bao gồm hai thuộc tính là Serial_no và Cost. Người ta muốn quản lý việc phân phối sử dụng các thiết bị cho từng nhân viên trong các dự án khác nhau thông qua một thuộc tính là ngày giao thiết bị Assign_date. Bạn hãy xây dựng một sơ đồ thực thể - liên kết cho nhu cầu đó.

6. Tìm hiểu bộ công cụ Oracle Designer2000 của Oracle. Thể hiện sơ đồ thực thể - liên kết trong hình 2.3.7.6 bằng tiện ích Entity Relationship Diagrammer của bộ công cụ này.

7. Vẽ sơ đồ thực thể - liên kết cho bài tập 6 trong chủ đề 1.

Chủ đề III: Quản trị hệ thống cơ sở dữ liệu

Mục tiêu:

Trong chủ đề này cần phải nắm được những vấn đề sau:

- Các mô hình tổ chức cơ sở dữ liệu, đặc biệt là mô hình tổ chức hệ cơ sở dữ liệu tập trung.
- Vai trò và nhiệm vụ chính của người quản trị cơ sở dữ liệu.
- Các vấn đề quản trị hệ thống cơ sở dữ liệu về mặt nội dung
- Các vấn đề quản trị hệ thống cơ sở dữ liệu về mặt kỹ thuật
- Các vấn đề quản trị hệ thống cơ sở dữ liệu về mặt an ninh, phân quyền
- Cách khắc phục một số lỗi thường gặp đối với hệ quản trị cơ sở dữ liệu.

1. Khái quát chung

Sau khi đã xây dựng xong một cơ sở dữ liệu và đưa vào hoạt động, công việc tiếp theo sẽ là vận hành và quản lý cơ sở dữ liệu đó hay nói một cách khác là quản trị hệ thống cơ sở dữ liệu đó.

Việc quản trị một hệ thống cơ sở dữ liệu bao gồm rất nhiều vấn đề và thông thường, với một hệ thống cơ sở dữ liệu lớn thì cần phải có một đội ngũ chuyên gia tin học không những chỉ giỏi về chuyên môn và có nhiều kinh nghiệm mà cần cả sự hiểu biết về các nghiệp vụ của ứng dụng vận hành trên cơ sở dữ liệu đã được xây dựng. Họ được gọi là những nhà quản trị hệ thống cơ sở dữ liệu.

Trong chủ đề này chúng ta sẽ đi sâu tìm hiểu vai trò và trách nhiệm đối với một người quản trị hệ thống cơ sở dữ liệu. Các vấn đề cần phải thực hiện khi quản trị một hệ thống cơ sở dữ liệu như quản trị hệ thống cơ sở dữ liệu về mặt nội dung, về mặt kỹ thuật, về các vấn đề liên quan đến đảm bảo an ninh, phân quyền dữ liệu. Ngoài ra, trong phần cuối của chủ đề này chúng tôi sẽ dành một mục để đề cập đến một số lỗi thường gặp trong quá trình quản trị một hệ thống cơ sở dữ liệu và hướng dẫn một số cách khắc phục chúng.

Trước tiên, chúng ta cùng xem xét một số mô hình tổ chức cơ sở dữ liệu.

2. Các mô hình tổ chức cơ sở dữ liệu

Các mô hình tổ chức cơ sở dữ liệu thường được phân chia thành hai loại, đó là các mô hình tổ chức cơ sở dữ liệu tập trung và các mô hình tổ chức cơ sở dữ liệu phân tán. Mặc dù mô hình tổ chức cơ sở dữ liệu tập trung có các ưu điểm vượt trội của nó, song trong một số trường hợp, người ta vẫn phải xây dựng những hệ thống cơ sở dữ liệu phân tán đặc biệt là khi hệ thống cần phải tiến hành thực hiện song song ở nhiều vị trí vật lý khác nhau và đặc biệt là khi có các yêu cầu về tốc độ cũng như đặc trưng của dữ liệu. Thường

những hệ thống phải sử dụng mô hình cơ sở dữ liệu phân tán là những hệ thống lớn mà vị trí địa lý của nó trải dài trên một vùng lãnh thổ như trong một quốc gia, một châu lục hay trên toàn thế giới.

Việc thiết kế cơ sở dữ liệu theo mô hình phân tán và giải quyết các vấn đề liên quan đến quản trị những hệ thống này là rất phức tạp, đòi hỏi khá nhiều thời gian và công sức. Bởi vậy, trong chủ đề này chúng tôi chỉ trình bày các vấn đề liên quan đến quản trị các hệ thống cơ sở dữ liệu theo mô hình tập trung.

2.1. Các hệ cơ sở dữ liệu tập trung

Với một hệ cơ sở dữ liệu tập trung, tất cả các dữ liệu được định vị tại một trạm đơn lẻ, những người sử dụng tại các trạm từ xa có thể truy nhập vào cơ sở dữ liệu thông qua các công cụ truy thông dữ liệu. Các hệ cơ sở dữ liệu tập trung đảm bảo sự kiểm soát tốt hơn trong việc truy nhập và cập nhật dữ liệu so với các hệ cơ sở dữ liệu phân tán, song lại có thể bị mắc lỗi nhiều hơn do chúng phụ thuộc vào tính sẵn sàng của các tài nguyên.

a. Hệ cơ sở dữ liệu cá nhân

Hệ cơ sở dữ liệu cá nhân là một kiểu hệ cơ sở dữ liệu rất quen thuộc đối với hầu hết mọi người. Các hệ cơ sở dữ liệu cá nhân thường do một người sử dụng đơn lẻ vừa thiết kế, tạo lập cơ sở dữ liệu, trực tiếp viết chương trình ứng dụng, đồng thời cũng là người sử dụng cuối của hệ. Thông thường, hệ cơ sở dữ liệu này hỗ trợ một ứng dụng hay một số các ứng dụng đơn lẻ.

Các ứng dụng phổ biến nhất của các hệ cơ sở dữ liệu cá nhân là các ứng dụng về các công việc kinh doanh và quản lý có quy mô nhỏ. Các ứng dụng điển hình bao gồm: quản lý thanh toán, kiểm kê hàng hoá và quản lý khách hàng. Việc phát triển và sử dụng các hệ cơ sở dữ liệu cá nhân là khá đơn giản và dễ dàng.

Các hệ cơ sở dữ liệu cá nhân được sử dụng trong các tổ chức có quy mô vừa và nhỏ, tuy nhiên, trong khi phát triển các ứng dụng như vậy, những người sử dụng có nguy cơ phải đối mặt với sự rủi ro khi tạo ra các ứng dụng mang tính đặc thù lớn. Mặt khác, dữ liệu khó có thể được chia sẻ cho nhiều ứng dụng khác nhau. Nếu có nhu cầu chia sẻ dữ liệu, cơ sở dữ liệu của hệ này phải được lưu trữ trên một máy chủ cơ sở dữ liệu hay trong một mạng cục bộ.

b. Hệ cơ sở dữ liệu trung tâm

Trong các tổ chức lớn, dữ liệu mà hầu hết các ứng dụng đều có thể truy nhập tới được lưu trữ trên một máy tính trung tâm. Trong nhiều hệ thống, những người sử dụng từ xa có thể truy nhập tới cơ sở dữ liệu này thông qua các thiết bị đầu cuối và các móc nối truyền thông dữ liệu. Tuỳ thuộc vào quy mô của tổ chức, máy tính trung tâm này thường lưu trữ các cơ sở dữ liệu tích hợp rất lớn và được nhiều người sử dụng truy nhập. Việc sử dụng thường có cường độ lớn, hàng trăm giao dịch trong một giây đang cần được xử lý (một số hệ thống có thể hỗ trợ hàng nghìn giao dịch trong một giây). Một trong các ứng dụng điển hình của hệ cơ sở dữ liệu trung tâm là hệ thống đăng ký giữ chỗ trên máy bay, các hệ thống thông tin của các cơ quan tài chính….

c. Hệ cơ sở dữ liệu khách/chủ

Các máy tính trung tâm lớn thường rất đắt so với các máy tính nhỏ và các máy trạm. Do vậy, nhiều tổ chức đã giảm kích thước các ứng dụng để có thể cài đặt trên các máy tính nhỏ với giá thành thấp và hiệu quả hơn.

Các máy tính được nối với nhau trong một mạng cục bộ sao cho chúng có thể chia sẻ các tài nguyên như các máy in, máy quét, các thiết bị lưu trữ,...

Mục đích chính của một 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 sử dụng và các yêu cầu đặt ra đối với ứng dụng được xử lý trên máy khách, còn việc xử lý dữ liệu được thực hiện trên máy chủ cơ sở dữ liệu.

Các hệ thống khách/chủ (và các mạng cục bộ kết hợp) thường được sử dụng để hỗ trợ các tính toán theo một nhóm công việc, có nghĩa là việc sử dụng các tài nguyên tính toán để hỗ trợ ra quyết định và các ứng dụng khác bởi một nhóm người sử dụng. Một vài mạng như vậy có thể được móc nối với nhau sao cho các nhóm công việc khác nhau có thể chia sẻ công việc. Các hệ thống khách/chủ đang nhanh chóng phát triển thành các modun chuẩn cho các hệ thống tính toán quy mô toàn xí nghiệp tại nhiều tổ chức.

2.2. Các hệ cơ sở dữ liệu phân tán

Ngày nay, nhiều tổ chức phân bố trên nhiều vị trí khác nhua như các thành phố hay thậm chí các quốc gia khác nhau. Trong những trường hợp như vậy, việc xây dựng các

hệ cơ sở dữ liệu tập trung đối với các tổ chức này thường là không thực tế và không kinh tế.

Một cơ sở dữ liệu phân tán là một cơ sở dữ liệu logic đơn lẻ được phân bố về mặt vật lý trên nhiều máy tính ở nhiều vị trí địa lý khác nhau. Có hai kiểu chung nhất của các hệ cơ sở dữ liệu phân tán, đó là: hệ cơ sở dữ liệu phân tán thuần nhất và hệ cơ sở dữ liệu phân tán không thuần nhất.

a. Hệ cơ sở dữ liệu phân tán thuần nhất

Khi áp dụng đối với các hệ cơ sở dữ liệu, thuật ngữ thuần nhất có nghĩa là công nghệ về cơ sở dữ liệu là như nhau (hay ít nhất là có thể tương thích) tại mỗi một vị trí địa lý và dữ liệu tại các vị trí địa lý khác nhau cũng có thể tương thích.

Ví dụ, một cơ sở dữ liệu của một hệ thống được phân tán trên 3 nút sẽ tương ứng với 3 vị trí địa lý khác nhau. Đối với các hệ cơ sở dữ liệu phân tán thuần nhất này, các vấn đề sau đây có thể xảy ra:

- Các hệ điều hành máy tính tại mỗi một vị trí địa lý là như nhau hay ít nhất chúng có khả năng tương thích cao.
- Các mô hình dữ liệu được sử dụng tại mỗi một vị trí địa lý là như nhau (mô hình quan hệ được sử dụng chung nhất đối với các hệ cơ sở dữ liệu phân tán ngày nay).
- Các hệ quản trị cơ sở dữ liệu được sử dụng tại mỗi một vị trí địa lý là như nhau hay ít nhất chúng có khả năng tương tích cao.

- Dữ liệu tại các vị trí khác nhau có các định nghĩa và khuôn dạng chung

Các hệ cơ sở dữ liệu phân tán thuần nhất đơn giản hoá việc chia sẻ dữ liệu giữa những người sử dụng khác nhau. Ví dụ, với mô hình 3 nút trên, một người sử dụng tại vị trí A có thể dễ dàng tìm kiếm hay cập nhật dữ liệu được lưu tại các vị trí B và C.

Các cơ sở dữ liệu phân tán thuần nhất thể hiện một mục đích thiết kế đối với các hệ cơ sở dữ liệu phân tán. Cụ thể, các cơ sở dữ liệu phân tán thuần nhất được thiết kế bằng cách chia nhỏ một cơ sở dữ liệu thành nhiều cơ sở dữ liệu địa phương, các cơ sở dữ liệu địa phương định vị trên các trạm làm việc khác nhau nhưng chúng được biểu diễn bởi cùng một mô hình dữ liệu và được quản trị bởi cùng một hệ quản trị dữ liệu địa phương.

b. Hệ cơ sở dữ liệu phân tán không thuần nhất

Trong hầu hết các doanh nghiệp, các hệ cơ sở dữ liệu liên quan đến một chu kỳ dài không được chỉ đạo và lập kế hoạch cẩn thận. Các máy tính khác nhau và các hệ điều hành khác nhau có thể được sử dụng tại mỗi một vị trí địa lý. Các mô hình dữ liệu khác nhau và các hệ quản trị cơ sở dữ liệu khác nhau cũng có thể được lựa chọn sử dụng. Ví dụ, một ví trí có thể sử dụng công nghệ cơ sở dữ liệu quan hệ mới nhất, trong khi một vị trí khác có thể lưu trữ dữ liệu sử dụng các tệp truyền thống hay các cơ sở dữ liệu mạng, phân cấp cũ hơn.

Phức tạp hơn nữa, dữ liệu trên các vị trí thường không tương tích. Các mâu thuẫn điển hình bao gồm các khác biệt về cú pháp (sự biểu diễn khác nhau của các khoản mục dữ

liệu tại hai vị trí) và các khác biệt về ngữ nghĩa (các ngữ nghĩa khác nhau đối với cùng một khoản mục dữ liệu tại các vị trí khác nhau).

Sớm hay muộn thì những người sử dụng tại các vị trí khác nhau sẽ phát hiện ra rằng họ cần chia sẻ dữ liệu cho dù có sự không tương thích. Để giải quyết vấn đề này, giải pháp tỏ ra tương đối hữu hiệu là phát triển một cơ sở dữ liệu mới hoàn toàn hợp nhất được tất cả các hệ đang tồn tại. Tuy nhiên, giải pháp này không dễ thực hiện về mặt kỹ thuật cũng như kinh tế. Thay vào đó, đôi khi các cơ sở dữ liệu được móc nối với nhau và kết quả là tạo ra một tập các cơ sở dữ liệu không thuần nhất. Một hệ thống như vậy sẽ hạn chế các kiểu xử lý mà những người sử dụng có thể thực hiện. Ví dụ, một người sử dụng tại một vị trí có thể đọc nhưng không thể cập nhật dữ liệu tại một vị trí khác.

3. Quản trị hệ thống cơ sở dữ liệu

Để có thể hiểu rõ nội dung này, trước hết người đọc cần hiểu được các công việc cơ bản cần làm khi thực hiện công việc quản trị các hệ thống cơ sở dữ liệu. Các khái niệm cơ bản về các hệ thống cơ sở dữ liệu đã được trình bày ở các mục trên, ở mục này, chúng ta sẽ chỉ tập trung vào việc xác định vai trò và trách nhiệm của người quản trị hệ thống cơ sở dữ liệu.

Nhìn chung, về cơ bản công việc quản trị hệ thống của một hệ cơ sở dữ liệu ở các doanh nghiệp là giống nhau. Nhưng điều này sẽ không còn đúng nếu đem so sánh vai trò, trách nhiệm của người quản trị trên các doanh nghiệp có quy mô khác nhau. Ví dụ, việc quản trị chỉ một hệ thống bán hàng tại một cửa hàng sách sẽ nhẹ nhàng hơn nhiều so với việc

quản trị toàn bộ hệ thống cơ sở dữ liệu bao gồm các hệ thống tại các cửa hàng bán sách nhỏ. Có thể coi yếu tố kinh nghiệm và những hiểu biết về các hệ quản trị cơ sở dữ liệu của một doanh nghiệp như là những người thực hiện công việc quản trị các hệ thống cơ sở dữ liệu này.

Trước hết, phải xác định được vai trò và trách nhiệm của người quản trị hệ thống để từ đó sẽ quyết định các kỹ năng cần thiết để thực hiện các vai trò này. Ở đây, vai trò và trách nhiệm sẽ được chia ra làm 2 phạm trù chính: cơ bản và các phần khác. Tuy vậy, người đọc cần lưu ý rằng, trên thực tế, ranh giới phân chia giữa các phạm trù này là không rõ ràng mà chỉ mang tính chất tương đối.

Dưới đây sẽ chỉ nêu ra các vai trò và trách nhiệm xét ở mức tổng quát nhất, bởi vì trên thực tế, người quản trị có thể không cần thực hiện tất cả các nội dung này hoặc có thể sẽ phải làm nhiều hơn:.

- Cài đặt các phần mềm:Cài đặt các phiên bản mới của các phần mềm ứng dụng, các phần mềm liên quan đến việc quản trị cơ sở dữ liệu.Các phần mềm này cần được kiểm tra cẩn thận trước khi tiến hành cài đặt và đưa vào môi trường sử dụng chính thức.
- Cấu hình phần cứng, phần mềm cùng với người quản trị hệ thống:Trong rất nhiều trường hợp, các hệ thống phần mềm chỉ có thể truy cập, cài đặt với vai trò của người quản trị hệ thống. Khi đó, người quản trị hệ thống cơ sở dữ liệu cần làm việc cùng với người quản trị hệ thống trong suốt quá trình thực hiện cài đặt phần mềm và thực hiện cấu hình phần cứng, phần mềm sao cho hệ thống mới sẽ là môi trường hoạt động tối ưu cho hệ quản trị cơ sở dữ liệu.

- Quản trị về an ninh:Nội dung của phần này sẽ được nêu rõ ràng hơn trong các mục sau, ở đây chỉ nêu ra vấn đề ở mức tổng quát nhất. Một trong những trách nhiệm chính của người quản trị hệ thống cơ sở dữ liệu là theo dõi và quản lý an ninh các hệ quản trị cơ sở dữ liệu. Công việc này bao gồm: thêm, loại bỏ người dùng, kiểm soát dung lượng đĩa cứng dùng cho hệ quản trị cơ sở dữ liệu, kiếm soát truy nhập và khi cần, thực hiện các thao tác xử lý sự cố về an ninh hệ thống.
- Điều chỉnh và kiểm soát hiệu năng hoạt động của hệ quản trị cơ sở dữ liệu: Người quản trị hệ thống cơ sở dữ liệu cần luôn thực hiện theo dõi hiệu năng hoạt động của hệ cơ sở dữ liệu và sẵn sàng điều chỉnh lại hệ thống khi cần. Công việc này cần được thực hiện ngay cả đối với các hệ thống đã được xây dựng và hiệu chỉnh tốt. Ví dụ như thay đổi các thông số hay thiết lập lại các tệp chỉ mục, cấu trúc lại các bảng dữ liệu.
- Sao lưu và phục hồi dữ liệu: Đây có lẽ là công việc quan trọng bậc nhất của người quản trị hệ thống, bởi lẽ họ có trách nhiệm duy trì và đảm bảo hệ thống cơ sở dữ liệu hoạt động một cách thông suốt tại mọi thời điểm, đảm bảo tính sẵn sàng của dữ liệu. Để có thể thực hiện được điều này, người quản trị cơ sở dữ liệu cần xây dựng một chiến lược sao lưu và khôi phục dữ liệu đủ tin cậy, tiến hành thực hiện công việc này một cách thường xuyên. Trách nhiệm chính của người quản trị hệ thống cơ sở dữ liệu là đảm bảo sự sẵn sàng và toàn vẹn của cơ sở dữ liệu, vì vậy, quá trình

sao lưu và phục hồi dữ liệu cần phải được thực hiện và kiểm tra thường xuyên.

- Bảo trì định kỳ: Trách nhiệm của người quản trị hệ thống cơ sở dữ liệu cũng là lập lịch bảo trì các hệ thống cơ sở dữ liệu và thực hiện theo lịch đó. Thông thường, việc bảo trì định kỳ được thực hiện vào sáng sớm hoặc vào các ngày cuối tuần. Khi đó sẽ tránh được sự ảnh hưởng đến việc sử dụng, truy cập các thông tin lưu tại cơ sở dữ liệu của người sử dụng.
- Xử lý sự cố: Trong trường hợp hệ thống hay hệ quản trị cơ sở dữ liệu không hoạt động do hỏng hóc hay do các nguyên nhân khác, người quản trị cần thực hiện xử lý sự cố hoặc hỗ trợ xử lý sự cố. Họ cũng cần tham gia hoặc chịu trách nhiệm chính để tìm ra và loại bỏ tận gốc các vấn đề gặp phải.
- Phục hồi sau sự cố: Trong trường hợp hệ thống ngừng hoạt động do hỏng thì người chịu trách nhiệm chính trong việc đưa hệ thống trở lại hoạt động bình thường là người quản trị hệ thống cơ sở dữ liệu. Giải pháp tốt nhất là người quản trị cần có kế hoạch xử lý cho các sự kiện ngừng hoạt động bất thường của hệ thống. Khi đó, hệ thống sẽ rất nhanh chóng sẵn sàng hoạt động trở lại sau khi đã được xử lý sự cố.

Ngoài ra, người quản trị hệ thống cơ sở dữ liệu có thể sẽ phải thực hiện hoặc tham gia thực hiện các công việc sau trong doanh nghiệp ngoài các trách nhiệm chính nêu trên:

- Tham gia phân tích dữ liệu: Người quản trị hệ thống cơ sở dữ liệu có thể sẽ được yêu cầu thực hiện phân tích các dữ liệu lưu trữ trong cơ sở dữ liệu và từ đó,

đưa ra các khuyến nghị liên quan đến hiệu năng của việc lưu trữ đó. Điều này có thể liên quan đến những việc như: nâng cao hiệu quả sử dụng các chỉ mục, hay là việc sử dụng các câu lệnh (tùy chọn) làm tăng tốc độ thực hiện truy vấn.

- Tham gia thiết kế cơ sở dữ liệu: Người quản trị cũng có thể tham gia vào giai đoạn đầu khi tiến hành thiết kế cơ sở dữ liệu, với kinh nghiệm và chức năng của mình, người quản trị có thể đóng góp rất nhiều ý kiến hữu ích cho người thiết kế cơ sở dữ liệu trong các vấn đề như tạo lập các đối tượng dữ liệu, lưu trữ đối tượng dữ liệu, tối ưu hoá hiệu năng truy vấn,...
- Tham gia vào việc mô hình hóa dữ liệu và tối ưu hóa: Thông qua mô hình hóa dữ liệu, người quản trị cơ sở dữ liệu có thể tối ưu hóa được cách thức tổ chức hệ thống sao cho tận dụng được những ưu điểm của hệ thống vào/ra dữ liệu, từ đó có thể nâng hiệu suất phục vụ của cơ sở dữ liệu đối với các chương trình ứng dụng.
- Trợ giúp người phát triển: Người quản trị hệ thống cơ sở dữ liệu cũng được xem như là một nguồn lực có thể trợ giúp những người phát triển hệ thống và người sử dụng trong việc hỗ trợ xây dựng các câu lệnh SQL, các thủ tục được cài đặt trong cơ sở dữ liệu.
- Tham gia vào quá trình đề xuất ra các chuẩn trong doanh nghiệp: Với một hệ thống được phát triển và triển khai bởi nhiều nhóm phần mềm khác nhau, vai trò của người quản trị trong vấn đề đưa ra các chuẩn trong quá trình phân tích, thiết kế cơ sở dữ liệu cũng

như xây dựng phần mềm ứng dụng phải được quan tâm. Các chuẩn này có thể là quy ước đặt tên, quy ước sử dụng dữ liệu, quy ước sử dụng tài nguyên hệ thống,...

- Tham gia vào quá trình đánh giá và kiểm tra các phần mềm trong hệ thống: với vai trò là người phải đảm bảo được tính sẵn sàng và tin cậy của hệ quản trị cơ sở dữ liệu, người quản trị hệ thống có thể tham gia vào các quá trình như kiểm tra các phần mềm sẽ được cài đặt trên hệ quản trị cơ sở dữ liệu, các phiên bản nâng cấp tiếp theo,...
- Lập các tài liệu về môi trường hệ thống: Người quản trị hệ thống cơ sở dữ liệu cần lập các tài liệu về tất cả các khía cạnh của hệ thống, bao gồm: cấu hình phần cứng, các thời điểm bảo trì, các lần cập nhật, nâng cấp phần mềm, các thay đổi về ứng dụng và hệ quản trị cơ sở dữ liệu, và tất cả các mục liên quan đến những thay đổi của hệ thống. Các tài liệu này cần có khả năng sẵn sàng để có thể cung cấp cho người quản trị hệ thống cơ sở dữ liệu, những nhà phát triển hệ thống, người sử dụng, và những người có liên quan khi có yêu cầu.
- Đánh giá phần mềm cần mua mới: Công vệc này bao gồm việc tư vấn cho lãnh đạo về các phần mềm mới, bao gồm các phần mềm xây dựng mới hoặc các phiên bản tiếp theo sắp được đưa ra thị trường. Việc đánh giá này dựa trên cơ sở đảm bảo tính ổn định và tin cậy của toàn hệ thống khi đưa vào sử dụng các phần mềm này.

- Tham gia đánh giá các vấn đề về mua sắm nâng cấp các phần mềm cũng như phần cứng mới, đặc biệt là trong vấn đề đánh giá các chức năng và tính tương thích của của chúng với những thiết bị và phần mềm đã có.
- Lên kế hoạch và đề xuất khi thấy hệ thống cần thiết phải nâng cấp để đáp ứng các nhu cầu tiếp theo của doanh nghiệp. Đưa ra các quyết định liệu có cần mua thêm các hệ thống phần cứng, phần mềm mới nhằm đáp ứng được các nhu cầu ngày càng gia tăng về dữ liệu là công việc của người quản trị hệ thống cơ sở dữ liệu nhằm nâng cao khả năng đáp ứng, chất lượng dịch vụ của hệ thống cho người sử dụng.

3.1. Quản trị nội dung cơ sở dữ liệu

Trên đây, chúng ta đã làm quen với các khái niệm về công việc của những người quản trị hệ thống cơ sở dữ liệu. Thực chất, các công việc có liên quan đến nội dung cơ sở dữ liệu nêu trên có thể được phân chia thành các nhóm công việc chính như sau:

- Theo dõi cơ sở dữ liệu
- Hiệu chỉnh cơ sở dữ liệu
- Sao lưu và phục hồi dữ liệu

a. Theo dõi cơ sở dữ liệu

Theo dõi cơ sở dữ liệu *(Database Monitoring)* là một trong các công việc hết sức quan trọng khi thực hiện quản trị hệ thống cơ sở dữ liệu. Một hệ thống cơ sở dữ liệu cần đảm bảo khả năng phục vụ ổn định, thời gian trả lời các truy vấn cũng như các thao tác cập nhật dữ liệu tối thiểu. Tuy vậy,

sau một thời gian dài hoạt động, nếu không có sự hiệu chỉnh thì khả năng đáp ứng dịch vụ của hệ cơ sở dữ liệu có thể giảm đi, và do đó, chất lượng phục vụ cũng giảm theo. Việc theo dõi cơ sở dữ liệu giúp chúng ta có thể giảm thiểu các rủi ro do việc giảm chất lượng phục vụ của các cơ sở dữ liệu tới người sử dụng.

Trong thực tế, theo dõi cơ sở dữ liệu liên quan trực tiếp đến việc trả lời các câu hỏi:

- Máy chủ cơ sở dữ liệu có hoạt động hay không?
- Cơ sở dữ liệu có thể truy cập được không?
- Có thể thực hiện phục hồi dữ liệu và đảm bảo hệ thống hoạt động thông suốt khi xảy ra các sự cố làm hệ thống ngừng hoạt động hay không?

Đó là những câu hỏi cơ bản nhất mà một người quản trị hệ thống cơ sở dữ liệu cần phải trả lời. Nói cách khác là họ phải đảm bảo rằng hệ thống cơ sở dữ liệu đang hoạt động tốt. Việc theo dõi các cơ sở dữ liệu thực chất là việc theo dõi về trạng thái, các thông số kỹ thuật của cơ sở dữ liệu và các thông số liên quan đến các thông số kỹ thuật này.

- Theo dõi trạng thái

Trong quá trình hoạt động, việc đáp ứng đầy đủ các nhu cầu về sử dụng dữ liệu là rất quan trọng của các doanh nghiệp hiện nay. Các hệ thống có thể luôn phải sẵn sàng 24 giờ một ngày, 7 ngày trong tuần. Nếu hệ thống không thể đáp ứng nhu cầu, người sử dụng có thể quay sang sử dụng một hệ thống khác, của một công ty cạnh tranh khác chẳng hạn. Chính vì vậy, theo dõi trạng thái của cơ sở dữ liệu là rất cần thiết và thiết thực đối với các tổ chức.

Trạng thái của cơ sở dữ liệu bao gồm: tình trạng của máy chủ cơ sở dữ liệu có hoạt động hay không, cơ sở dữ liệu có truy cập được không, …

- Theo dõi các thông số kỹ thuật

Các thông số kỹ thuật cần thực hiện theo dõi bao gồm: hiện trạng sử dụng bộ nhớ của cơ sở dữ liệu, hiệu suất sử dụng đường truyền, các thủ tục chạy ngầm (*job*) theo định kỳ, dung lượng của cơ sở dữ liệu, dung lượng của các tệp chỉ mục, lượng bộ nhớ sử dụng,…

- Các thông số khác.

b. Hiệu chỉnh cơ sở dữ liệu

Việc theo dõi các thông tin trạng thái của cơ sở dữ liệu là rất cần thiết bởi lý do đảm bảo tính sẵn sàng cao phục vụ cho các nhu cầu về dữ liệu của người sử dụng. Trong thực tế, có thể xảy ra trường hợp sự cố gây ra bởi các tác nhân khách quan như: dung lượng đường truyền, số lượng người sử dụng, dung lượng bộ nhớ vật lý, dung lượng lưu trữ có thể, tính đúng đắn của dữ liệu ... làm cho hệ thống chậm phục vụ hay thậm chí người sử dụng bị từ chối dịch vụ. Trong các trường hợp này, người quản trị hệ thống cơ sở dữ liệu cần tìm ra được nguyên nhân gây ra sự cố và từ đó có thể đưa ra các hiệu chỉnh cần thiết nhằm đưa hệ thống trở lại hoạt động bình thường.

Các hiệu chỉnh đối với cơ sở dữ liệu có thể là:

Điều chỉnh các tham số khởi tạo của các cơ sở dữ liệu, bởi vì các tham số khởi tạo này có thể liên quan đến dung lượng lưu trữ có thể sử dụng cho cơ sở dữ liệu, lượng bộ nhớ mà cơ sở dữ liệu có thể sử dụng, ... ảnh hưởng đến khả

năng phục vụ của cơ sở dữ liệu; tăng cường số máy chủ phục vụ, giúp cho hệ thống được phân tải đều cho các yêu cầu từ phía người sử dụng để từ đó giảm thời gian trả lời tới người sử dụng; tăng lượng bộ nhớ cho máy chủ phục vụ, giúp nâng cao khả năng phục vụ cho cơ sở dữ liệu; thiết lập hay tạo lại các tệp chỉ mục cho các bảng dữ liệu, giúp hệ thống tìm kiếm các dữ liệu mà người sử dụng yêu cầu với thời gian trả lời nhanh nhất; kiểm tra và đảm bảo các ràng buộc dữ liệu, giúp cho dữ liệu được đảm bảo đúng đắn ...

Trong các điều kiện cụ thể, theo các nguyên nhân do người quản trị hệ thống cơ sở dữ liệu phát hiện được thông qua theo dõi cơ sở dữ liệu, các phản hồi từ người sử dụng, các sự cố có thể được phát hiện sớm và có các giải pháp cụ thể xử lý triệt để cũng như nâng cao hiệu năng phục vụ cho toàn hệ thống. Tuy vậy, chiến lược đúng đắn nhất vẫn là lập kế hoạch chi tiết, cụ thể các giải pháp để có thể xử lý các sự cố thường gặp, theo dõi và phát hiện các nguy cơ tiềm tàng có thể dẫn đến các sai sót hay sự cố về dữ liệu. Điều này yêu cầu người quản trị hệ thống cơ sở dữ liệu phải thường xuyên cập nhật các thông tin về kỹ thuật và luôn theo dõi sát sao các thay đổi của hệ thống.

c. Sao lưu và phục hồi dữ liệu

Như đã nêu trên, người quản trị hệ thống cơ sở dữ liệu cần đảm bảo hoạt động của hệ thống cơ sở dữ liệu thông suốt, nói một cách khác là cơ sở dữ liệu luôn hoạt động nhằm đáp ứng các nhu cầu về dữ liệu trong quá trình tác nghiệp của cơ quan, tổ chức. Trong thực tế, các sự cố như: mất điện, quá tải, virus, hỏng đĩa cứng lưu trữ hay thậm chí là các xâm nhập hệ thống bất thường với mục tiêu phá hoại có thể làm hệ thống cơ sở dữ liệu bị đình trệ hay không thể

phục vụ nhu cầu của người sử dụng. Nguyên nhân của việc không thể phục vụ được tại một thời điểm nào đó có thể là do quá tải đường truyền hoặc dữ liệu bị hư hại do các tác nhân ngoài ý muốn nêu trên.

Để có thể khắc phục các sự cố gây ra tình trạng từ chối dịch vụ, người quản trị có thể thực hiện tăng cường các biện pháp an ninh, bảo mật hệ thống. Nhưng đối với trường hợp dữ liệu bị mất mát, hư hỏng thì trách nhiệm của người quản trị hệ thống cơ sở dữ liệu là cần có biện pháp phục hồi lại trạng thái hoạt động của hệ thống như tại thời điểm ngay trước khi hệ thống bị ngừng hoạt động. Để làm được điều này cần lập các bản sao đầy đủ của các cơ sở dữ liệu, để khi cần có thể nhanh chóng phục hồi lại hệ thống cơ sở dữ liệu, đảm bảo tính sẵn sàng cho hệ thống.

Các hệ quản trị cơ sở dữ liệu tiên tiến thường cung cấp các công cụ rất trực quan, tiện dụng để người quản trị hệ thống cơ sở dữ liệu có thể dễ dàng thực hiện các thao tác này. Vì vậy, một người quản trị hệ thống cơ sở dữ liệu cần lập cho doanh nghiệp của mình một chiến lược cụ thể và thực hiện kế hoạch này thường xuyên theo một định kỳ rõ ràng tạo lập các bản sao đầy đủ của các cơ sở dữ liệu để khi có sự cố có thể nhanh chóng đưa hệ thống cơ sở dữ liệu vào hoạt động trở lại.

3.2. Quản trị kỹ thuật

Vấn đề về quản trị kỹ thuật cũng là một chủ đề quan trọng khi nói đến các vấn đề về quản trị cơ sở dữ liệu vì một hệ thống là tổng thể hài hòa của các thành phần tạo nên chúng. Nhưng do đặc thù và tính quan trọng của dữ liệu luôn được đặt lên hàng đầu, đôi khi vấn đề về quản trị kỹ thuật có thể

bị xem nhẹ. Nói một cách đơn giản là hệ thống cơ sở dữ liệu cần hoạt động trên một môi trường ổn định, và có khả năng đáp ứng chính xác mọi nhu cầu xuất phát từ người sử dụng.

Các yếu tố kỹ thuật chính mà chúng ta cần quan tâm, đó là:

- Cài đặt, cấu hình hệ thống
- Đảm bảo về đường truyền, mạng
- Quản trị tài nguyên cơ sở dữ liệu

Chúng ta sẽ lần lượt đi vào từng nội dung nêu trên để hiểu được các công việc này thực chất là phải làm những gì.

a. Cài đặt, cấu hình hệ thống

Như đã nêu trong các mục trước đây, quản trị hệ thống cơ sở dữ liệu liên quan đến việc phải theo dõi sát sao quá trình cài đặt các phần cứng, phần mềm cùng với người quản trị hệ thống. Việc cấu hình hệ thống cơ sở dữ liệu cần có một hiểu biết sâu sắc về các hệ quản trị cơ sở dữ liệu, các cơ sở dữ liệu và các nhân tố ảnh hưởng đến khả năng hoạt động của các cơ sở dữ liệu.

Việc thiết lập các thông số cấu hình của các cơ sở dữ liệu cần tuân thủ sát sao các khuyến cáo từ tài liệu kỹ thuật do những người phát triển hệ thống cung cấp. Các thông số này có thể là các khuyến cáo về dung lượng bộ nhớ sử dụng, dung lượng lưu trữ tối thiểu, các thông số khởi tạo hệ thống cơ sở dữ liệu,... Trong quá trình đưa hệ thống cơ sở dữ liệu vào phục vụ tác nghiệp, người quản trị hệ thống cơ sở dữ liệu có thể cần thực hiện cài đặt hệ thống từ đầu dựa trên các tài liệu do người phát triển hệ thống cung cấp đảm bảo môi trường tối ưu cho khai thác, cập nhật dữ liệu.

b. Đảm bảo về đường truyền, mạng

Trong thực tế, công việc của người quản trị hệ thống cơ sở dữ liệu có thể không liên quan trực tiếp đến vấn đề về đường truyền, mạng. Vì vậy, người quản trị hệ thống cơ sở dữ liệu cần làm việc sát sao cùng với người quản trị mạng để có thể đảm bảo được dung lượng đương truyền có thể đáp ứng tối đa các nhu cầu dữ liệu của người sử dụng.

Nguyên nhân là do môi trường tính toán hiện tại thường được tổ chức theo mô hình từ hai cho đến ba lớp: client / server hay client / middle tier / server nên các hệ thống chắc chắn phải hoạt động trên môi trường mạng. Khi đó, việc đảm bảo các truy cập của người sử dụng đến dữ liệu của mình thông qua các hệ thống mạng là hết sức cần thiết. Đôi khi, người quản trị hệ thống cơ sở dữ liệu cần phải trang bị cho mình các kiến thức cơ bản về đường truyền và mạng nhằm phát hiện, giải quyết các sự cố có thể xảy ra trong quá trình sử dụng cơ sở dữ liệu trong môi trường tác nghiệp.

c. Quản trị tài nguyên cơ sở dữ liệu

Các hệ thống cơ sở dữ liệu trong thực tế là một trong những tài sản sở hữu bởi các cơ quan, tổ chức. Nó không chỉ bao gồm các thông tin về khách hàng mà chủ yếu còn về các dữ liệu tác nghiệp trong cơ quan hoặc tổ chức đó. Bảo đảm tài nguyên này luôn sẵn sàng cho người sử dụng hệ thống là một công việc khá nặng nề đối với những người quản trị hệ thống cơ sở dữ liệu.

3.3. Đảm bảo an ninh, phân quyền truy nhập

Như đã nêu, nếu hệ thống bị ngừng trệ hoặc tốc độ phục vụ quá chậm sẽ dẫn đến hậu quả không tốt cho hoạt động tác nghiệp của các cơ quan, tổ chức. Trong thực tế, có thể có các truy cập không hợp lệ nhằm các ý đồ xấu, phá hoại dẫn đến các hậu quả nghiêm trọng như: mất cắp thông tin, mất, hỏng dữ liệu... Để có thể tránh được tình trạng nêu trên, người quản trị hệ thống cơ sở dữ liệu cần có các biện pháp bảo vệ dữ liệu, loại bỏ và từ chối các truy cập trái phép từ đó đảm bảo tính sẵn sàng và tính an toàn cho hệ thống cơ sở dữ liệu.

Các sự cố có thể xảy ra bao gồm việc mất mát dữ liệu, dữ liệu bị làm sai lệch so với nguyên bản, người sử dụng truy cập các chức năng, các dữ liệu không được phép có thể gây hậu quả nghiêm trọng. Biện pháp để giải quyết các vấn đề này là:

- Thiết lập cơ chế kiểm soát các truy nhập vào cơ sở dữ liệu của người sử dụng trong nội bộ cơ quan, tổ chức
- Phân quyền sử dụng các đối tượng dữ liệu cho từng người dùng khác nhau
- Có thể hạn chế quyền truy cập cơ sở dữ liệu của người sử dụng đến tận từng nội dung dữ liệu cụ thể
- Kết hợp với quản trị mạng để có cơ chế kiểm soát người sử dụng đăng nhập hệ thống của các nhân viên trong nội bộ cơ quan
- Cùng người quản trị hệ thống thiết lập các biện pháp bảo vệ mạng bằng cách sử dụng các hệ thống kiểm soát các truy nhập trái phép từ bên ngoài hệ thống

như: tường lửa (firewall), định kỳ quét và kiểm tra virus, cập nhật các bản vá lỗi an ninh, bảo mật cho các hệ điều hành, ...

3.4. Một số lỗi thường gặp và cách khắc phục

Trong mục này, chúng tôi sẽ trình bày một số lỗi thường gặp trong quá trình vận hành và khai thác cơ sở dữ liệu, các nguyên nhân chính có thể dẫn đến gây lỗi và cách khắc phục:

a. Không thể truy cập được cơ sở dữ liệu

Lỗi này thông thường là do các nguyên nhân: hệ thống mạng không được kích hoạt tại máy khách, hệ thống mạng không hoạt động, hệ quản trị cơ sở dữ liệu không hoạt động, hệ thống bị quá tải, lỗi do chương trình ứng dụng, … Các nguyên nhân nêu trên có độ ưu tiên từ trái qua phải, người quản trị hệ thống cần thực hiện kiểm tra tất cả các lý do nêu trên có thể gây ra tình trạng cơ sở dữ liệu không thể truy cập được.

Nếu hệ thống mạng không được kích hoạt tại máy khách có nghĩa là tất cả các giao dịch khác của máy khách với các máy tính còn lại trong mạng sẽ không thể thực hiện được. Thông thường, để kiểm tra lỗi này, người quản trị có thể sử dụng lệnh ‘ping.exe’, một tiện ích có sẵn của Windows để kiểm tra kết nối mạng giữa máy khách và máy chủ cơ sở dữ liệu.

Nếu hệ thống mạng của toàn bộ cơ quan không hoạt động cũng có nghĩa là chúng ta không thể thực hiện một giao tác mạng nào. Trong trường hợp nghi ngờ, hãy cùng với quản trị mạng kiểm tra sự hoạt động của toàn bộ hệ thống. Đừng

bỏ qua những chi tiết nhỏ, chẳng hạn như nguồn cấp điện cho thiết bị kết nối tập trung: hub, switch.

Nếu hệ thống mạng hoạt động tốt và chúng ta có thể thực hiện các giao dịch mạng với máy chủ cơ sở dữ liệu thì lỗi có thể là do các dịch vụ (service) của các hệ quản trị cơ sở dữ liệu này không được cho phép hoạt động. Các hệ quản trị cơ sở dữ liệu chạy trên các máy chủ thường có các dịch vụ chạy ngầm đóng vai trò cầu nối giữa các máy trạm và cơ sở dữ liệu. Ví dụ điển hình là các hệ quản trị cơ sở dữ liệu Oracle, MS SQL Server. Nguyên nhân này có thể xuất hiện sau trường hợp bị mất điện bất ngờ làm cho hệ thống máy chủ không thể kích hoạt, khởi động các dịch vụ cho hệ quản trị cơ sở dữ liệu. Để khắc phục, dựa vào các tài liệu hướng dẫn sử dụng thực hiện các bước khởi động các dịch vụ cho hệ quản trị cơ sở dữ liệu và đừng quên kiểm tra lại sự hoạt động của hệ thống.

Trong trường hợp tất cả các dịch vụ của hệ quản trị cơ sở dữ liệu đều đã hoạt động, lý do không truy cập được cơ sở dữ liệu đã được loại bỏ, vấn đề còn lại có thể nằm ở các chương trình ứng dụng, các thông số cấu hình khác của cơ sở dữ liệu. Hãy yêu cầu đối tác hoặc nhóm phát triển các ứng dụng kiểm tra hệ thống của họ.

b. Hệ thống báo lỗi timeout (chậm quá thời gian cho phép)

Khi hệ thống báo lỗi ‘timeout’, có nghĩa là nó có dấu hiệu của sự quá tải, hoặc do lỗi của đường truyền. Thông thường, các giao tác mạng được thực hiện theo kiểu: Gửi – Chờ – Nhận. Có nghĩa là các tiến trình thường phải chờ một khoảng thời gian trước khi nhận dữ liệu. Nếu thời gian chờ

dài hơn giới hạn cho phép, hệ thống tự động báo lỗi quá hạn (timeout). Giải pháp cho trường hợp này là kiểm tra tình trạng cơ sở dữ liệu, kiểm tra thời gian thực hiện các giao tác mạng.

Kiểm tra tình trạng cơ sở dữ liệu cho phép người quản trị hệ thống cơ sở dữ liệu có thể nắm được các thông tin về bộ nhớ sử dụng, dung lượng đĩa cứng, số lượng các giao dịch và lượng bộ nhớ mà các giao dịch có thể đang chiếm giữ. Các thông tin này rất có ích trong việc kiểm tra khả năng đáp ứng các giao dịch đến cơ sở dữ liệu từ người sử dụng. Từ đó có thể đưa ra các giải pháp khắc phục bằng cách hiệu chỉnh các thông số cơ sở dữ liệu, giải phóng bớt bộ nhớ từ các chương trình không cần thiết, dọn dẹp đĩa cứng, tăng dung lượng vùng swap cho hệ điều hành,… Đồng thời, xem xét khả năng tối ưu tài nguyên sử dụng, tăng tốc độ truy vấn của các câu lệnh SQL mà người sử dụng thực hiện trên cơ sở dữ liệu.

c. Hệ thống báo lỗi Out of Memory (tràn bộ nhớ)

Trong các hệ thống máy chủ hiện đại, hiện tượng tràn bộ nhớ hiếm khi xảy ra. Tuy nhiên, đối với các hệ thống cũ có thể xảy ra tình trạng này do các hệ quản trị cơ sở dữ liệu đôi khi chiếm lượng bộ nhớ rất lớn khiến cho hệ thống không thể đáp ứng được. Giải pháp là nên tăng cường thêm RAM, xem xét mua thêm ổ đĩa cứng mới hoặc nghiên cứu khả năng nâng cấp toàn bộ hệ thống của máy chủ. Tuy nhiên việc tăng RAM, tăng dung lượng ổ đĩa cứng chỉ có thể là giải pháp tạm thời. Với lượng dữ liệu ngày càng lớn theo thời gian thì giải pháp triệt để nhất vẫn là nghiên cứu chuyển cơ sở dữ liệu sang một hệ thống mới ổn định và mạnh hơn.

d. Không thể cung cấp được dữ liệu

Nguyên nhân không thể cung cấp được dữ liệu phần lớn là do một trong các trường hợp nêu trên. Tuy nhiên, cũng có trường hợp các bảng dữ liệu, các khung nhìn (view) hoặc các thủ tục định nghĩa sẵn bởi người sử dụng bị mất do cài đặt lại hoặc do các bảng dữ liệu chính của cơ sở dữ liệu có dung lượng quá lớn gây ra hiện tượng chậm cung cấp dữ liệu. Khi đó, người quản trị hệ thống cơ sở dữ liệu cần thực hiện các biện pháp kiểm tra tổng hợp như đã nêu trong các phần trên, đồng thời thực hiện công việc bảo trì hệ thống cơ sở dữ liệu. Cần nhớ một nguyên tắc bất di bất dịch là trước khi thực hiện bất kỳ thao tác bảo trì nào cần thực hiện sao lưu đầy đủ hệ thống cơ sở dữ liệu hiện tại như đã nêu trong phần "Sao lưu và phục hồi dữ liệu".

e. Virus

Các virus và biến thể của chúng ngày nay đang là những mối nguy hiểm đe dọa sự an toàn và ổn định của hệ thống mạng cũng như các hệ thống cơ sở dữ liệu. Điều này đặc biệt nguy hiểm khi các hệ cơ sở dữ liệu ngày nay không chỉ phục vụ cho riêng một doanh nghiệp nào. Virus có thể gây ra các hiện tượng tắc nghẽn đường truyền, mất dữ liệu, dữ liệu bị phá hủy, … và do đó hệ thống cơ sở dữ liệu sẽ bị ảnh hưởng đến khả năng cung cấp dữ liệu của mình.

Do sự nguy hiểm ngày càng tăng của virus, người quản trị hệ thống cơ sở dữ liệu cần thực hiện thường xuyên các thao tác sao lưu toàn bộ cơ sở dữ liệu theo định kỳ một cách nghiêm ngặt, đồng thời thường xuyên cập nhật các bản vá lỗi của hệ điều hành, hệ quản trị cơ sở dữ liệu mỗi khi các hãng cung cấp các hệ quản trị cơ sở dữ liệu thông báo, cập

nhật các phiên bản phần mềm chống virus mới nhất. Điều này nêu ra để nhấn mạnh đến tầm quan trọng của việc sử dụng các phần mềm có bản quyền và do đó có thể nhận được các hỗ trợ về kỹ thuật từ các nhà cung cấp sản phẩm, cũng như luôn đảm bảo hệ thống hiện tại là mới nhất và có khả năng ứng phó kịp thời đối với các nguy cơ khiến cho hệ thống mất ổn định.

4. Kết luận

Trong chủ đề này, chúng tôi đã giới thiệu các vấn đề liên quan đến việc quản trị đối với một hệ thống cơ sở dữ liệu. Các vấn đề chính được đề cập đến trong chủ đề này, đó là: hiểu và nắm vững các vai trò và trách nhiệm đối với một người quản trị hệ thống cơ sở dữ liệu; hiểu và nắm được các vấn đề liên quan đến quản trị cơ sở dữ liệu về mặt nội dung, kỹ thuật, an toàn, bảo mật dữ liệu. Một số lỗi thường gặp đối với cơ sở dữ liệu chúng tôi cũng đã trình bày, giúp người quản trị đỡ bỡ ngỡ và có thêm kinh nghiệm khi phải đảm nhận vai trò quản trị hệ thống cơ sở dữ liệu.

Đối với những hệ quản trị cơ sở dữ liệu lớn, ngoài việc cung cấp các công cụ quản trị qua giao diện người sử dụng thì đối với một người quản trị nhiều kinh nghiệm, việc sử dụng các câu lệnh sẽ cho phép họ có thể can thiệp sâu hơn vào hệ thống.

Trong chủ đề tiếp theo, chúng tôi sẽ tập trung vào việc trình bày một số hệ quản trị cơ sở dữ liệu phổ biến ở Việt Nam hiện nay.

5. Bài tập

1. Liệt kê các vai trò và trách nhiệm đối với một người quản trị hệ thống cơ sở dữ liệu.

2. Tìm và đọc thêm tài liệu sau: System Administration for Microsoft SQL Server

3. Tìm và đọc thêm tài liệu sau: Oracle Administrator's Guide, Oracle Administrator's Guide for Windows.

4. Tìm hiểu công cụ Enterprise Manager của Microsoft SQL Server

5. Tìm hiểu công cụ Oracle Enterprise Manager của Oracle

Chủ đề IV: Một số hệ quản trị cơ sở dữ liệu phổ biến

Mục tiêu:

Trong chủ đề này, cần phải nắm được các vấn đề sau:

- Kiến trúc và các thành phần của một hệ quản trị cơ sở dữ liệu.
- Các hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ: Foxpro, Access
- Các hệ quản trị cơ sở dữ liệu cỡ lớn SQL Server, Oracle
- Lựa chọn hệ quản trị cơ sở dữ liệu cho phù hợp đối với những bài toán thực tế.
- Thực hành thiết kế sơ bộ một cơ sở dữ liệu đã được trình bày trong các chủ đề I và II trên hệ quản trị cơ sở dữ liệu SQL Server.

1. Giới thiệu chung

Như chúng ta đã biết, cơ sở dữ liệu là một tập các dữ liệu được tổ chức. Để thực hiện các thao tác như chèn, sửa, xoá và tìm kiếm dữ liệu, cơ sở dữ liệu cần phải được quản lý bởi một phần mềm quan trọng, phần mềm này thường được gọi là hệ quản trị cơ sở dữ liệu (Database Management System - DBMS). Mục đích chính của một hệ quản trị cơ sở dữ liệu đó là cho phép người sử dụng lưu trữ, cập nhật và tìm kiếm dữ liệu thông qua các thuật ngữ trừu tượng, do

đó có thể dễ dàng bảo trì và quản lý thông tin bằng cơ sở dữ liệu. Nhờ hệ quản trị cơ sở dữ liệu, người sử dụng không phải quan tâm đến các vấn đề liên quan đến việc biểu diễn dữ liệu ở mức vật lý cũng như các thuật toán lưu trữ, cập nhật và tìm kiếm và trả về dữ liệu.

Một hệ quản trị cơ sở dữ liệu thường là một gói phần mềm rất lớn thực hiện rất nhiều các công việc khác nhau như cung cấp các công cụ cho người sử dụng có thể truy nhập và sửa đổi thông tin trong cơ sở dữ liệu chẳng hạn. Một cơ sở dữ liệu là một kết nối trung gian giữa cơ sở dữ liệu vật lý, phần cứng, hệ điều hành, và với người sử dụng. Để đáp ứng các nhu cầu sử dụng của các loại người dùng khác nhau, ngoài việc tạo ra các công cụ sử dụng khác nhau, một hệ quản trị cơ sở dữ liệu thường cung cấp thêm một hoặc một vài ngôn ngữ lập trình chuyên dụng thường được gọi là ngôn ngữ cơ sở dữ liệu. Các hệ quản trị cơ sở dữ liệu khác nhau cung cấp các ngôn ngữ cơ sở dữ liệu khác nhau, mặc dù chúng hầu như đều dựa trên ngôn ngữ chuẩn SQL.

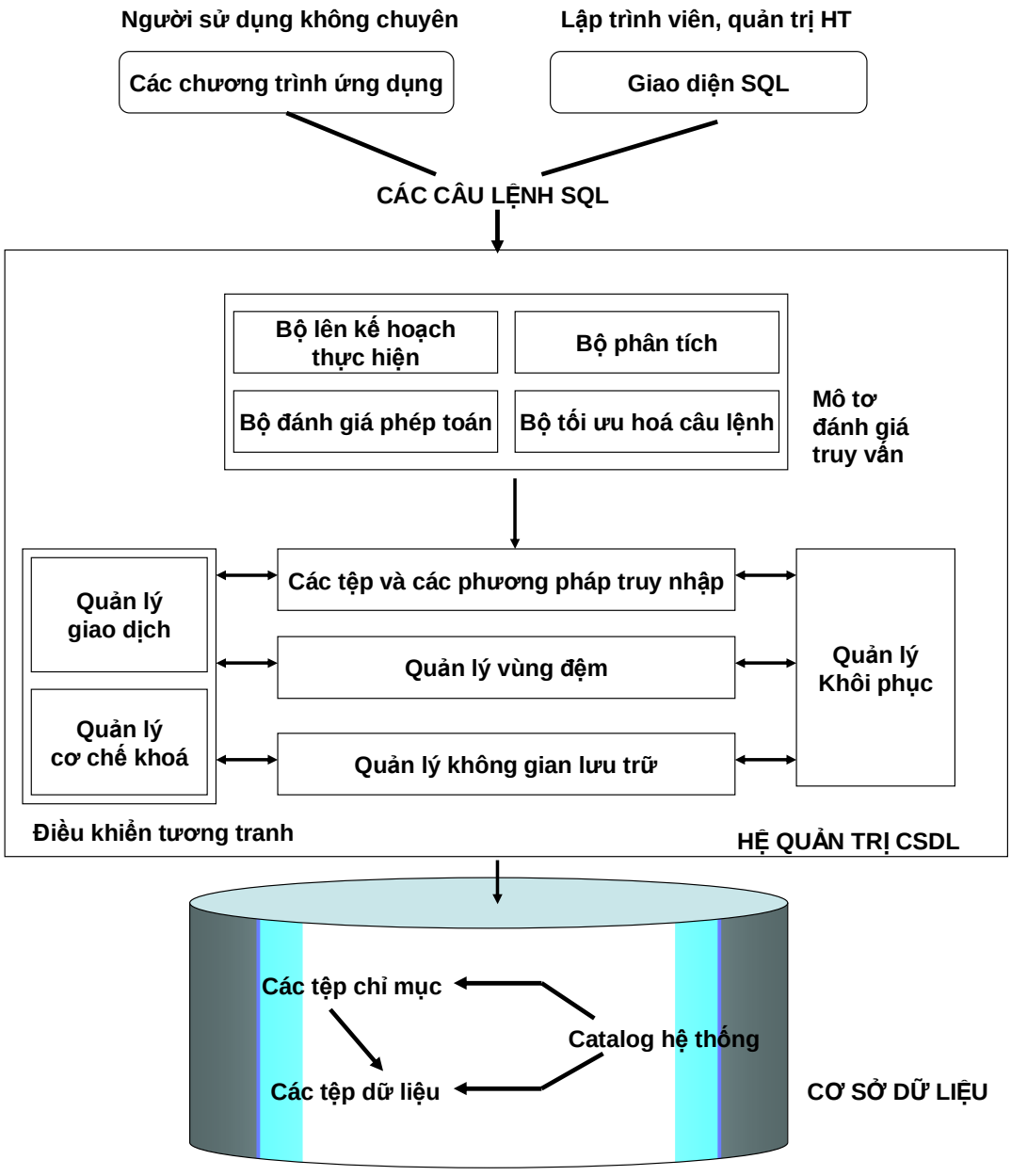
Ngày nay, trên thị trường các hệ quản trị cơ sở dữ liệu thương mại hầu hết là các hệ quản trị cơ sở dữ liệu quan hệ được xây dựng dựa trên mô hình dữ liệu quan hệ. Phiên bản đầu tiên của một hệ quản trị cơ sở dữ liệu quan hệ đó là hệ phần mềm lưu trữ dữ liệu quan hệ Multics, nó được tung ra thị trường lần đầu tiên vào năm 1978. Các hệ quản trị cơ sở dữ liệu quan hệ tiếp theo lần lượt được đưa ra, đó là Berkeley Ingres QUEL và IBM BS12.

Trong chủ đề này, chúng tôi sẽ trình bày những hệ quản trị cơ sở dữ liệu sau: Access, Visual Foxpro, SQL Server, Oracle, chúng đều là các hệ quản trị cơ sở dữ liệu quan hệ được thương mại hoá cao và rất phổ biến trên thị trường

Việt Nam cũng như trên thế giới. Tuy nhiên, trước tiên chúng ta hãy tìm hiểu kiến trúc đối với một hệ quản trị cơ sở dữ liệu.

2. Hệ quản trị cơ sở dữ liệu

2.1. Kiến trúc của một hệ quản trị cơ sở dữ liệu



Hình 4.2.1.1 Kiến trúc của một hệ quản trị CSDL

Hình 4.2.1.1 chỉ ra các thành phần chủ yếu của một hệ quản trị cơ sở dữ liệu. Tại đáy của kiến trúc, chúng ta thấy một

sự biểu diễn của các thiết bị lưu trữ cơ sở dữ liệu. Chú ý rằng thành phần này không chỉ chứa các dữ liệu được lưu trữ trong cơ sở dữ liệu mà còn bao gồm cả các siêu dữ liệu, tức là thông tin về cấu trúc của cơ sở dữ liệu cũng như các tệp chỉ mục dữ liệu.

Trong hình 4.2.1.1 chúng ta cũng thấy một bộ quản lý không gian lưu trữ mà nhiệm vụ của nó là lấy ra các thông tin được yêu cầu từ những thiết bị lưu trữ dữ liệu và thay đổi những thông tin này khi được yêu cầu bởi các mức trên nó trong hệ thống (chúng ta sẽ xem xét ký hơn thành phần này trong mục lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu dưới đây).

Thành phần tiếp theo là mô tơ đánh giá truy vấn hay bộ xử lý câu hỏi. Nhiệm vụ của nó là tìm ra phương án tốt nhất cho một thao tác được yêu cầu và phát ra các lệnh cho bộ quản lý lưu trữ để thực thi thao tác đó (chúng ta sẽ xem xét kỹ hơn thành phần này trong mục truy vấn trong một hệ quản trị cơ sở dữ liệu dưới đây).

Bộ quản trị giao dịch có trách nhiệm đảm bảo tính toàn vẹn của hệ thống. Kết hợp với bộ quản lý cơ chế khoá và quản lý khôi phục, đảm bảo các thao tác được thực hiện theo đúng thứ tự (quản lý tương tranh) và hệ thống sẽ không bị mất dữ liệu thậm chí khi có lỗi xảy ra. Bộ quản trị giao dịch cũng tương tác với bộ quản lý lưu trữ bởi việc xử lý giao dịch thường kéo theo việc lưu trữ nhật ký các thay đổi đối với dữ liệu để khi gặp lỗi, các thay đổi chưa được ghi vào đĩa có thể được thực hiện lại hoặc khôi phục lại.

Tại đỉnh của kiến trúc, chúng ta thấy các kiểu thao tác đối với hệ quản trị cơ sở dữ liệu:

- Thông qua một giao diện truy vấn chung: mỗi một hệ quản trị cơ sở dữ liệu thường cho phép người sử dụng gõ trực tiếp các truy vấn SQL thông qua một giao diện có sẵn. Thông thường giao diện này thường được thể hiện dưới dạng một số công cụ tích hợp trực tiếp vào hệ quản trị cơ sở dữ liệu và thường được dành cho những người sử dụng chuyên sâu, hiểu biết hệ thống.
- Thông qua các chương trình ứng dụng: Một hệ quản trị cơ sở dữ liệu điển hình cho phép những người lập trình viết các chương trình ứng dụng gọi đến hệ quản trị cơ sở dữ liệu này và truy vấn dữ liệu trong cơ sở dữ liệu. Các truy vấn được đưa ra thông qua một giao diện đặc biệt có thể chứa các hộp thoại. Một người sử dụng không thể đưa ra các câu truy vấn tuỳ ý thông qua giao diện này, nhưng nói chung sẽ dễ dàng hơn đối với họ khi sử dụng giao diện này so với việc phải viết một truy vấn trực tiếp bởi SQL.

Trên đây chúng ta đã có một cái nhìn tổng quát về kiến trúc của một hệ quản trị cơ sở dữ liệu chung. Phần tiếp theo chúng ta sẽ tìm hiểu xem một hệ quản trị cơ sở dữ liệu lưu trữ dữ liệu như thế nào cũng như các truy vấn được thực hiện ra sao.

2.2. Lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu

Các cơ sở dữ liệu thường đòi hỏi một khối lượng lớn không gian lưu trữ. Các cơ sở dữ liệu hợp nhất thường được đo theo các đơn vị gigabyte hay thậm chí terabyte. Do bộ nhớ

chính của máy tính không thể lưu trữ nhiều dữ liệu như vậy, nên các dữ liệu này thường được lưu trên các thiết bị nhớ ngoài như đĩa từ, băng từ, đĩa quang,…. Trong quá trình làm việc với cơ sở dữ liệu, dữ liệu được chuyển giữa thiết bị lưu trữ ngoài và bộ nhớ chính. Sự di chuyển dữ liệu này thường là khá chậm so với tốc độ của bộ xử lý trung tâm, do vậy các hệ quản trị cơ sở dữ liệu thường phải tổ chức lưu trữ dữ liệu vật lý sao cho tối thiểu hoá số yêu cầu chuyển đổi dữ liệu giữa thiết bị lưu trữ ngoài và bộ nhớ chính.

Như vậy, việc lưu trữ dữ liệu trong một hệ quản trị cơ sở dữ liệu được thực hiện thông qua bộ quản lý lưu trữ. Đây thực chất là một môđun chương trình cung cấp giao diện giữa các dữ liệu được lưu trữ trong cơ sở dữ liệu mức thấp và các chương trình ứng dụng hay các truy vấn được thực hiện bởi hệ thống. Bộ quản lý lưu trữ có trách nhiệm tương tác với các bộ quản lý tệp và các phương pháp truy nhập. Dữ liệu gốc được lưu trữ trên thiết bị lưu trữ ngoài sử dụng hệ thống xử lý tệp thường được cung cấp bởi một hệ điều hành nào đó. Bộ quản lý lưu trữ dịch các câu lệnh của ngôn ngữ SQL thành các lệnh hệ thống xử lý tệp ở mức thấp. Do vậy, bộ quản lý lưu trữ có trách nhiệm lưu trữ, tìm kiếm và cập nhật dữ liệu trong cơ sở dữ liệu.

2.3. Truy vấn trong một hệ quản trị cơ sở dữ liệu

Hệ quản trị cơ sở dữ liệu nhận các câu lệnh SQL thông qua các giao diện người dùng, lên kế hoạch đánh giá câu lệnh, thực hiện các câu lệnh này trên cơ sở dữ liệu và trả lại kết quả cho người sử dụng.

Khi một người sử dụng đưa ra một câu lệnh truy vấn, bộ tối ưu hoá truy vấn sẽ tiến hành phân tích câu lệnh bằng cách sử dụng các thông tin như dữ liệu được lưu như thế nào trong cơ sở dữ liệu, các chỉ mục nào nên được sử dụng, phần nào sẽ được thực hiện trước phần nào sẽ được thực hiện sau.,… để có thể đưa ra được một phương án thực hiện hiệu quả nhất trong khả năng có thể cho câu lệnh đó. Một phương án thực hiện là một kế hoạch cụ thể để đánh giá một câu hỏi, và nó thường được biểu diễn dưới dạng một cây các phép toán quan hệ. Thông qua phương án này, câu lệnh sẽ được chuyển tiếp xuống lớp tiếp theo để thực hiện. Sau khi được thực hiện xong, nếu dữ liệu được trả về, chúng sẽ được đưa trở lại cho người sử dụng đã đưa ra câu lệnh truy vấn đó.

3. Một số hệ thống quản trị cơ sở dữ liệu thông dụng

Trong phần này chúng tôi sẽ trình bày bốn hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay, xuất phát từ những hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ như Access, Foxpro đến những hệ quản trị cơ sở dữ liệu cớ lớn như SQL Server, Oracle. Trong mỗi hệ quản trị cơ sở dữ liệu, chúng tôi cố gắng trình bày các bước khi tiến hành thiết kế các đối tượng dữ liệu cho một cơ sở dữ liệu.

3.1. Các hệ quản trị cơ sở dữ liệu cỡ vừa và nhỏ

a. Hệ quản trị cơ sở dữ liệu Access

Microsoft Access là một hệ quản trị cơ sở dữ liệu quan hệ của hãng Microsoft chạy trên nền của hệ điều hành

Microsoft Windows, được đóng gói trong cùng bộ sản phẩm Microsoft Office. Nó được xây dựng chủ yếu nhằm phục vụ cho những người sử dụng không chuyên, không có nhiều kinh nghiệm lập trình nhưng vẫn có thể xây dựng được cho mình những ứng dụng client/server dựa trên cơ sở dữ liệu.

Một trong những lợi điểm chính của Access từ khía cạnh của những người thiết kế, lập trình đó là khả năng tương thích với các truy vấn SQL, câu lệnh SQL thân thiện, dễ dùng. Mặt khác, người sử dụng có thể sử dụng VBA để tạo lập các form màn hình nhập liệu cũng như khai thác cơ sở dữ liệu. Access cũng cho phép chúng ta tạo lập các báo cáo nghiệp vụ không quá phức tạp.

Access thường được sử dụng cho các doanh nghiệp nhỏ và những người lập trình nghiệp dư để tạo lập các hệ thống dễ thay đổi nhằm đáp ứng các công việc nhỏ, đơn giản. Ngoài ra nó cũng có khả năng truy nhập đến các cơ sở dữ liệu lớn hơn như Oracle, SQL Server.

Tuy nhiên, Access không phải là một hệ quản trị cơ sở dữ liệu đầy đủ bởi nó thiếu các phần như các kích hoạt (trigger), các thủ tục trong cơ sở dữ liệu (stored procedure). Cơ chế an toàn, bảo mật kém bởi nó không chia ra các user khác nhau trong cơ sở dữ liệu.

Dưới đây là một số hướng dẫn chính về hệ quản trị cơ sở dữ liệu Access:

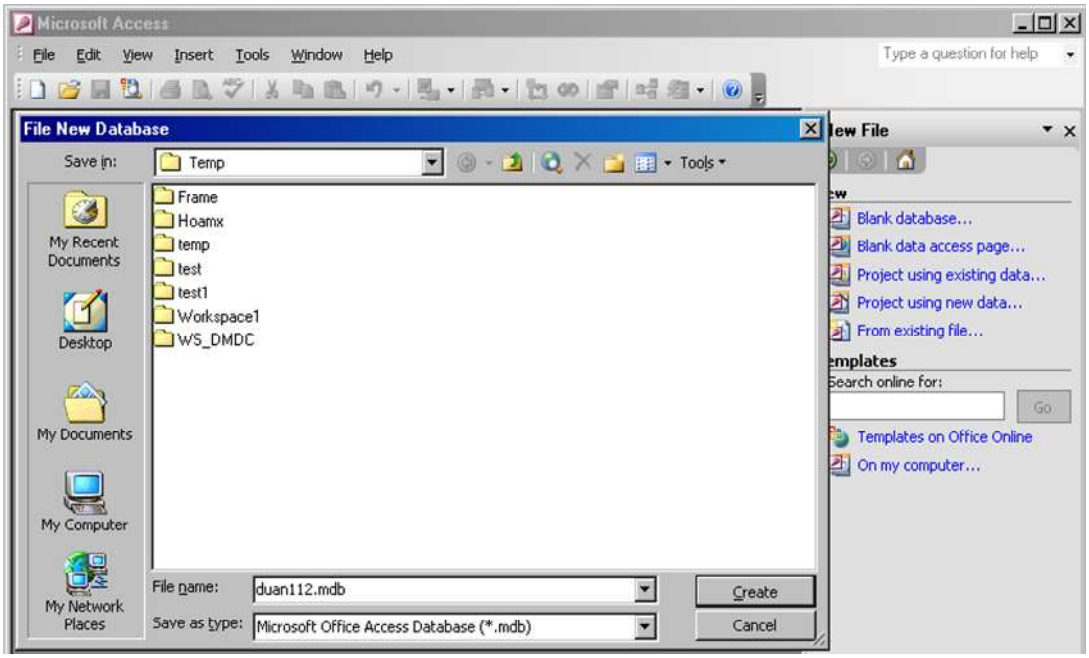
Access lưu trữ toàn bộ ứng dụng cơ sở dữ liệu trong một tệp đơn. Một tệp có đuôi .mdb của Access có thể chứa các đối tượng dữ liệu như các bảng, các chỉ mục, các truy vấn cũng như các đối tượng của ứng dụng như các màn hình

nhập liệu (forms), các báo cáo (reports), các lệnh macro và các mã visual basic.

Bước một: Kích hoạt Access

Để bắt đầu chúng ta hãy chọn START—>PROGRAMS—>Microsoft Access trên thanh công cụ của Windows. Access sẽ được kích hoạt. Chúng ta hãy lựa chọn để tạo một cơ sở dữ liệu mới.

Một cửa sổ khác sẽ xuất hiện yêu cầu chúng ta đặt tên tệp cho cơ sở dữ liệu mới đó. Chúng ta hay đưa vào tên mà chúng ta muốn đặt cho cơ sở dữ liệu đó cùng với vị trí thư mục và nhấn nút Create như màn hình 4.3.1.1 sau:



Hình 4.3.1.1 Tạo dựng một tệp cơ sở dữ liệu .mdb trong Access

Sau khi chúng ta nhấn nút create, Access sẽ đưa ra một cửa sổ với các lựa chọn sau:

- Tables: Dữ liệu trong cơ sở dữ liệu Access sẽ được lưu trong các bảng. Access cho phép tạo ra các bảng

theo các hình thức sau: tự khai báo các định dạng cho một bảng; tạo bảng từ các bảng khác đã có (wizard); và tạo bảng bằng cách đưa ngay dữ liệu vào bảng.

- Queries: các truy vấn, cho phép tạo ra các truy vấn lọc dữ liệu từ các bảng.
- Forms: Dùng để tạo ra các màn hình để nhập dữ liệu vào cơ sở dữ liệu cũng như các thao tác trình bày ứng dụng.
- Báo cáo: Dùng để tạo và định dạng dữ liệu đưa ra dưới dạng các biểu mẫu báo cáo.
- Pages: Dùng để tạo các trang web sử dụng ngôn ngữ đánh dấu siêu văn bản HTML nhằm đưa được dữ liệu lên trang web.
- Macros: được sử dụng để tự động thực hiện một loạt các thao tác tự động nào đó
- Modules: Được dùng chủ yếu trong khâu lập trình tạo ứng dụng, là nơi chứa các thủ tục, hàm của ứng dụng trên cơ sở dữ liệu Access.

Phần tiếp theo dưới đây sẽ chỉ tập trung vào việc sử dụng các bảng và truy vấn trong Access để có thể tạo dựng thành công một cơ sở dữ liệu quan hệ Access.

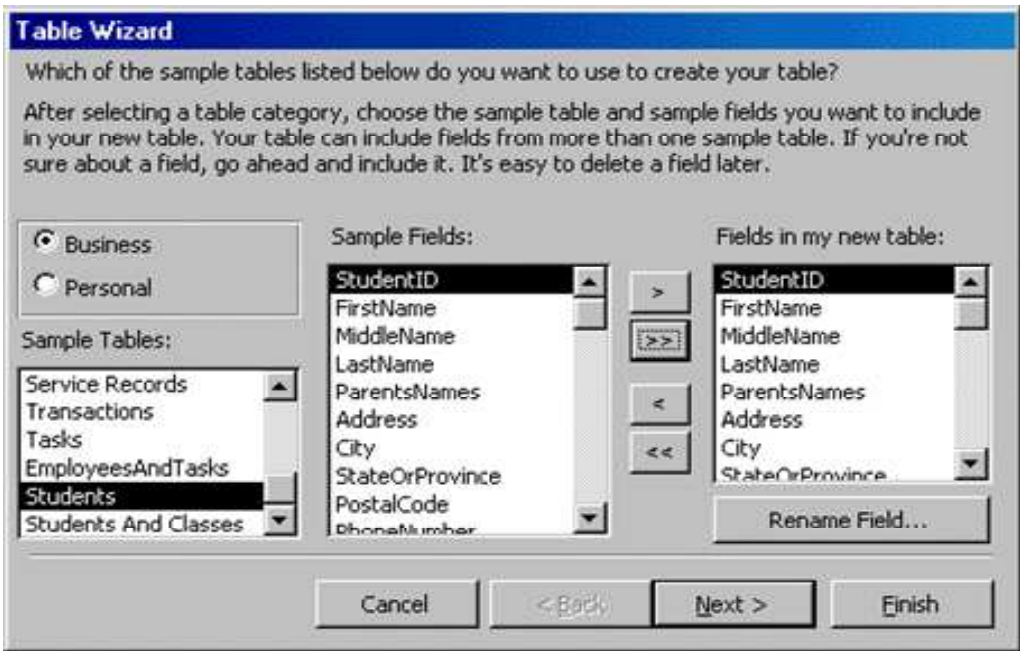
Bước hai: Thiết kế các bảng

Do trong Access không có công cụ cho phép thiết kế các đối tượng dữ liệu như kiểu thực thể, thuộc tính, liên kết nên chúng ta phải tự thực hiện bước thiết kế cơ sở dữ liệu mức logic này trên giấy hoặc bằng một số công cụ của các nhà cung cấp khác. Sau đó, dựa trên sơ đồ thực thể - liên kết

này, chúng ta tiến hành tạo dựng các bảng thông qua công cụ do Access cung cấp.

Bước ba: Tạo bảng trong Access

Trong cửa sổ chính, chọn chức năng tạo bảng “Create table”. Nếu muốn sử dụng cách tạo bảng từ các bảng đã có hãy chọn chức năng "Create table by using Wizard" màn hình sẽ xuất hiện như sau:



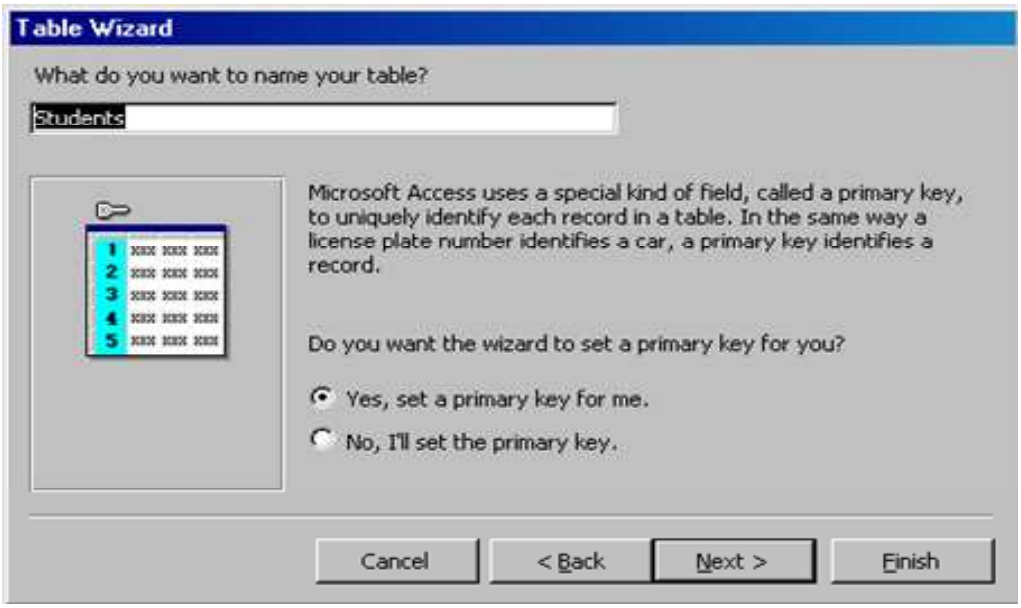
Hình 4.3.1.2 Tạo bảng bằng công cụ Wizard

Hãy lựa chọn bảng trong phần bên trái, các trường tương ứng của bảng đó sẽ xuất hiện ở giữa. Với những trường này, chúng ta có thể lựa chọn các trường cho bảng muốn xây dựng bằng cách chọn và nhấn nút mũi tên trỏ sang phải như hình trên. Sau khi lựa chọn xong các trường cho bảng mới nhấn nút Next.

Hãy đưa vào tên cho bảng vừa tạo ở ô text trên cùng. Tiếp theo Access sẽ hỏi xem có muốn tạo khoá chính (primary key) cho bảng này không vì tùy chọn mặc định là các khoá

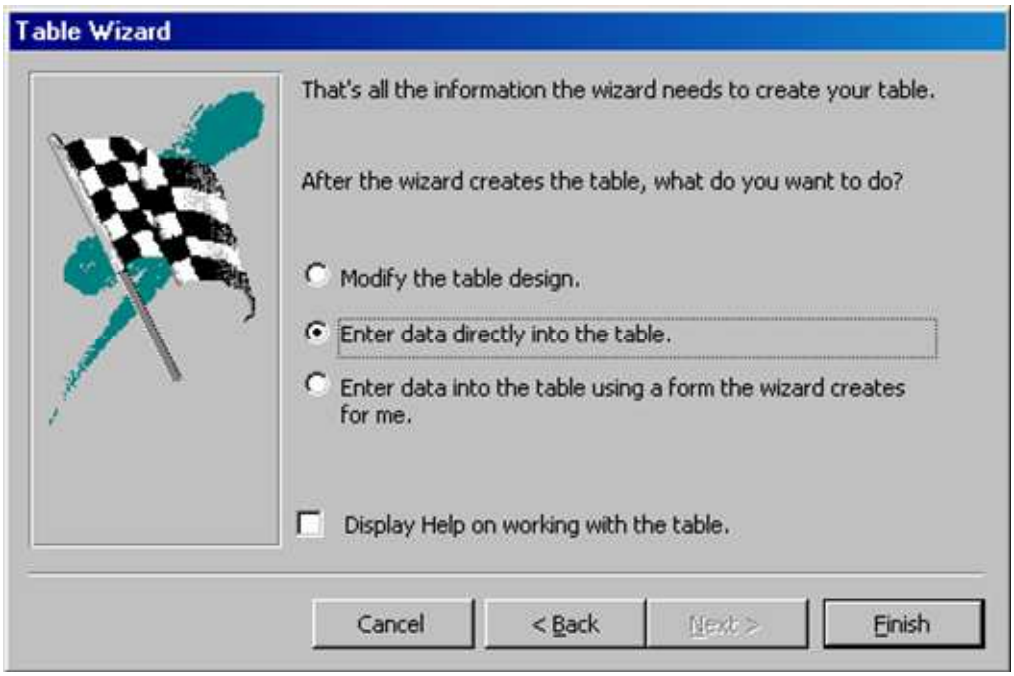
chính sẽ do Access tự tạo (thường là những trường có giá trị tự tăng).

Nếu muốn tạo khoá chính cần thực hiện theo hướng dẫn sau:



Hình 4.3.1.3 Tạo khoá chính cho bảng

Màn hình cuối cùng sẽ hỏi chúng ta có muốn sửa bảng tiếp, đưa dữ liệu vào bảng hay là tiếp tục sử dụng chức năng tạo màn hình nhập liệu tự động? Ở đây chúng ta hãy chọn tuỳ chọn thứ 2 là đưa dữ liệu vào bảng và nhấn phím Finish.

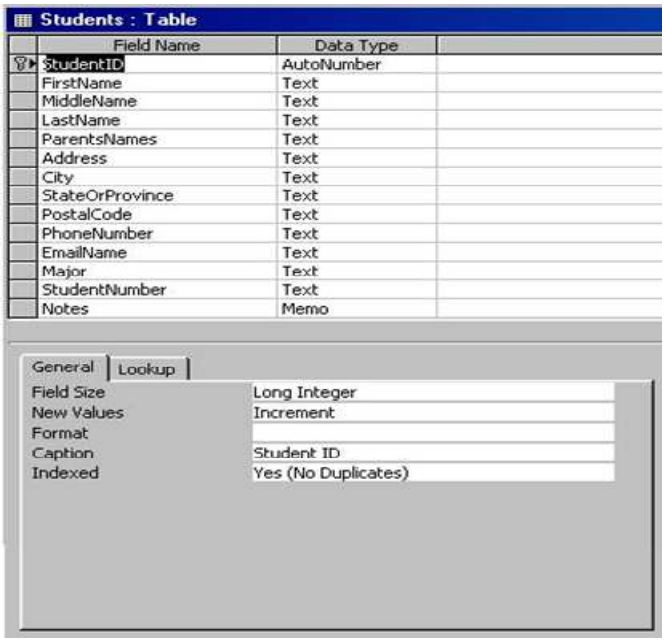


Hình 4.3.1.4 Kết thúc quá trình tạo một bảng dữ liệu

Bảng mà chúng ta vừa tạo sẽ xuất hiện trong khung datasheet. Khi thao tác với các bảng này, chúng ta sẽ thấy nó rất giống với các spreadsheet của Excel. Việc đưa các dữ liệu vào bảng sẽ được thực hiện bằng cách nháy kép vào bảng này và điền dữ liệu vào các trường tương ứng (như trong spreadsheet) hoặc có thể lấy (import) dữ liệu từ một số nguồn khác.

Dưới đây là màn hình của một bảng trong "Design View". Trong "Design View" chúng ta có thể tạo các trường, lựa chọn kiểu dữ liệu cho chúng cũng như các định dạng và khoá chính. Để lựa chọn trường nào làm khoá chính, hãy giữ Ctrl + nhấp chuột vào trường đó, sau đó di chuột đến cột bên trái ngoài cùng của bảng và nhấp chuột phải, lựa chọn chức năng Primary key, các trường vừa lựa chọn sẽ xuất hiện một biểu tượng chiếc chìa khoá bên trái trường đó. Khi chúng ta lựa chọn một trường trong lúc "Design View",

các tính chất đi kèm với thuộc tính đó sẽ xuất hiện trong màn hình "Field property", tại màn hình này chúng ta có thể đặt các chức năng như giá trị mặc định, các luật ràng buộc khi đưa dữ liệu vào, các yêu cầu là bắt buộc phải nhập (required) hay trường sẽ được sử dụng làm chỉ mục,... Ví dụ như màn hình dưới đây:



Hình 4.3.1.5 Khung thiết kế cho một bảng dữ liệu

Màn hình trên là khung thiết kế cho bảng "Students" với thuộc tính đang chọn là StudentID, đây là thuộc tính khoá với kiểu giá trị lựa chọn là tự tăng "AutoNumber".

Bước bốn: Tạo liên kết giữa các bảng

Liên kết giữa các bảng luôn luôn đóng vai trò quan trọng trong các cơ sở dữ liệu quan hệ, nó cho phép các bản ghi (thực thể) kết nối thông qua các trường.

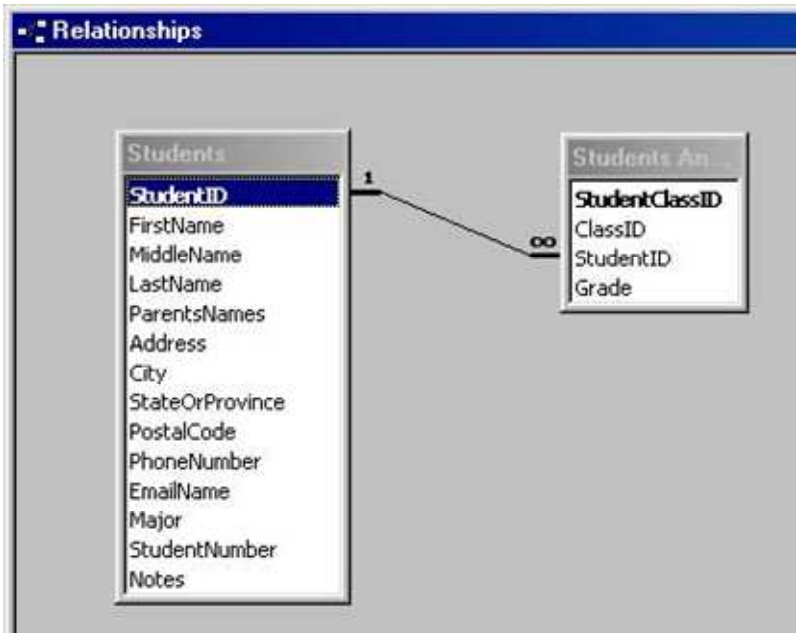
Hãy xem tiếp ví dụ sau:

Chúng ta tạo một bảng khác bằng chức năng “Create table by using wizard” và đặt tên cho bảng mới này là "Students and Classes". Tiếp theo, lặp lại các khâu như trong bước 3 cho đến khi chức năng wizard hỏi có muốn bảng mới liên kết với các bảng đã có hay không? Khi đó, hãy nhấn nút "Relationships" và chọn lựa chọn "The tables aren't related" rồi nhấn OK.

Đóng bảng vừa tạo lại và lựa chọn chức năng “Relationship” trên thực đơn công cụ “Tools”. Một màn hình sẽ hiện cho phép tạo dựng và chỉnh sửa các liên kết giữa các bảng. Nếu cả hai bảng cần kết nối không xuất hiện, hãy nhấn chuột phải vào màn hình hiện thời và lựa chọn chức năng “Show Table”. Hãy đánh dấu hai bảng cần tạo liên kết rồi nhất nút “Add”. Ở đây hai bảng sẽ là “Student” và “Students and Classes”. Access sẽ chèn những bảng này vào màn hình đang làm việc hiện thời. Nhấn nút Close để đóng cửa sổ lựa chọn bảng.

Đến đây trong màn hình đang làm việc xuất hiện hai bảng, hãy nhấn và kéo trường StudentID trong bảng “Students” sang trường StudentID trong bảng thứ hai “Students and Classes”. Một cửa sổ sẽ xuất hiện cho phép thay đổi kiểu của liên kết. Hãy đánh dấu vào ô có nhãn “"Enforce Referential Integrity" và nhấn nút "Create" để kết thúc việc tạo liên kết giữa hai bảng. Một đường kết nối sẽ cho biết liên kết đã được tạo. Các ký tự “1" and "∞" cho biết liên kết chúng ta vừa tạo là liên kết loại một - nhiều. Tức là một bản ghi ở phía bảng “1” sẽ có nhiều bản ghi kết nối với nó ở phía bảng “∞", nhưng ngược lại, một bản ghi ở phía bảng

“∞" chỉ có một bản ghi ở phía bảng “1” kết nối với nó mà thôi.



Hình 4.3.1.6 Một kết nối một nhiều trong Access

Như trong hình trên, liên kết một-nhiều đã tạo ra một toàn vẹn tham chiếu, các giá trị mà chúng ta có thể đưa vào thuộc tính StudentClassId bên bảng StudentsAndClasses không thể nằm ngoài các giá trị đã xuất hiện trong thuộc tính StudentId bên bảng Students. Và ngược lại, khi loại bỏ một bản ghi bên bảng Students thì trước đó chúng ta phải loại bỏ tất cả các bản ghi có giá trị trong thuộc tính StudentClassId bằng với giá trị trong thuộc tính StudentId của bản ghi chúng ta muốn xoá bên bảng Students.

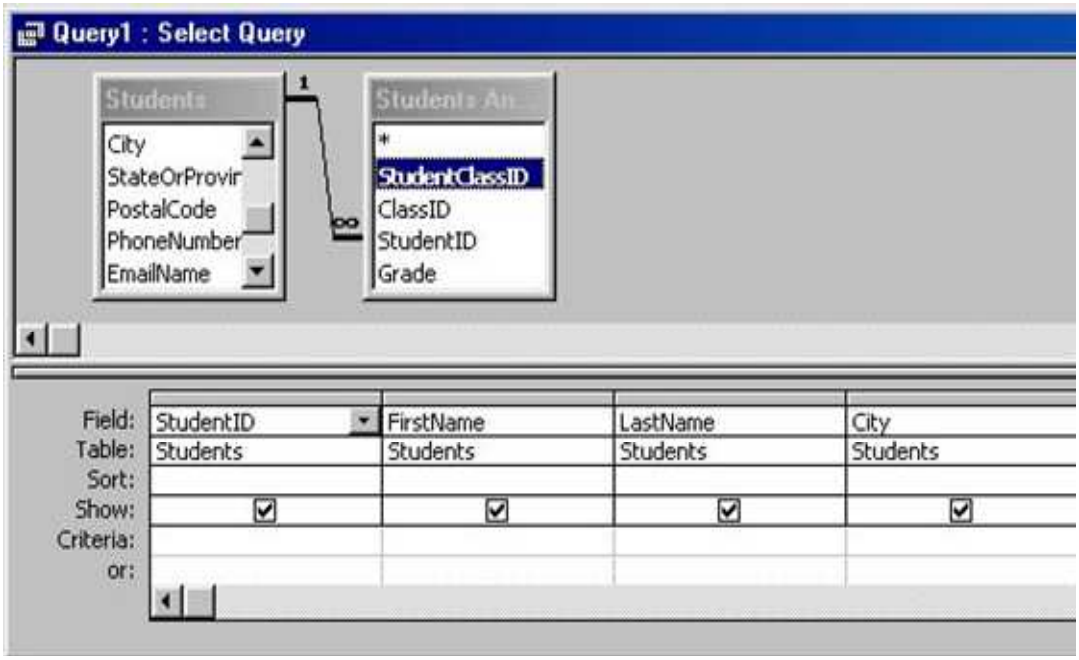
Hãy đóng cửa sổ tạo liên kết lại và nhấn “Yes” để lưu lại những thay đổi vừa tạo.

Bước năm: Truy vấn cơ sở dữ liệu

Sau khi tạo xong cơ sở dữ liệu gồm các bảng và các liên kết giữa chúng, chúng ta có thể đưa dữ liệu vào và tiến hành các truy vấn trên dữ liệu đó.

Microsoft Access cung cấp một công cụ gọi là Query-By-Example (QBE) để hỗ trợ cho việc xây dựng các câu lệnh SQL, giao diện của công cụ này rất thân thiện và chúng ta có thể dễ dàng sử dụng chúng. Khi một truy vấn được xây dựng xong thông qua QBE, chúng ta có thể xem được câu lệnh SQL mà Access đã tạo ra.

Để tạo các truy vấn, hãy nhấn kép vào chức năng "Create Query in Design View." Một cửa sổ sẽ xuất hiện và hỏi các bảng dữ liệu mà chúng ta muốn truy vấn. Hãy chọn các bảng bằng cách nhấn nút “Add”, sau khi chọn xong các bảng, hãy nhấn nút “Close”. Các bảng vừa chọn sẽ xuất hiện trên cửa sổ truy vấn phía trên. Access sẽ tự động hiển thỉ các liên kết giữa các bảng kết nối với nhau như hình sau:



Hình 4.3.1.7 Xây dựng một truy vấn trong Access

Để lựa chọn các trường cần truy vấn, hãy chọn trường đó và kéo xuống cửa sổ QBE phía dưới. Trong màn hình QBE, mỗi một cột thể hiện cho trường mà chúng ta đã chọn, hàng thứ nhất là tên trường, hàng thứ hai là bảng chứa trường đó, hàng thứ ba cho phép lựa chọn kiểu sắp xếp, hàng thứ tư cho phép lựa chọn trường đó có được hiển thị ra trong kết quả tìm kiếm hay không.

Vùng Criteria phía dưới cùng cho phép chúng ta định nghĩa các điều kiện tìm kiếm. Nếu trong cùng một hàng thì đó là điều kiện And, còn giữa các hàng khác nhau là điều kiện Or.

Ví dụ, để tạo một truy vấn tìm kiếm tất cả các sinh viên sống ở Hà nội hay Hải phòng và có chữ cái đầu tiên trong tên bắt đầu bằng "B" chúng ta đưa điều kiện vào vùng Criteria như sau:

Gõ xâu "B*" và ô tên riêng Lastname tại vùng Criteria (chú ý bao gồm cả ký tự ")

Gõ xâu "Hà nội" tại ô CityField tại vùng Criteria

Gõ xâu "Hải phòng" tại ô CityField tại vùng Criteria nhưng ở hàng Or.

Sau đó, hãy chọn chức năng "SQL View" trong thực đơn View. Một cửa sổ hiển thị câu lệnh SQL tương ứng với truy vấn đó sẽ được đưa ra

b. Hệ quản trị cơ sở dữ liệu Visual Foxpro

FoxPro là hệ quản trị cơ sở dữ liệu khá thông dụng trong những năm 90 của thế kỷ 20. Mặc dù sự lớn mạnh của các hệ quản trị cơ sở dữ liệu như Oracle, SQL Server, DB2, ... FoxPro vẫn tiếp tục được hãng Microsoft phát triển

và tiếp tục hoàn thiện cho các bài toán cỡ nhỏ. Các công cụ mà hãng Microsoft cung cấp cho người sử dụng được thực hiện thông qua các Wizard: Application Wizard, Form Wizard, Report Wizard,... Ngoài ra, các công cụ SQL Upsizing Wizard và Oracle Upsizing Wizard cho phép người sử dụng có thể thực hiện chuyển cơ sở dữ liệu của mình sang các hệ quản trị cơ sở dữ liệu nói trên đem lại khả năng phát triển tiềm tàng cho FoxPro.

Sau đây, chúng ta sẽ tìm hiểu công cụ chính của hệ quản trị cơ sở dữ liệu Visual FoxPro, Project Manager, để từ đó có thể có được cái nhìn toàn diện về cách thức xây dựng và phát triển các hệ thống cơ sở dữ liệu sử dụng công cụ này. Để thông tin đưa ra là cập nhật, chúng tôi sử dụng bộ công cụ Microsoft Visual FoxPro 6.0 để minh họa.

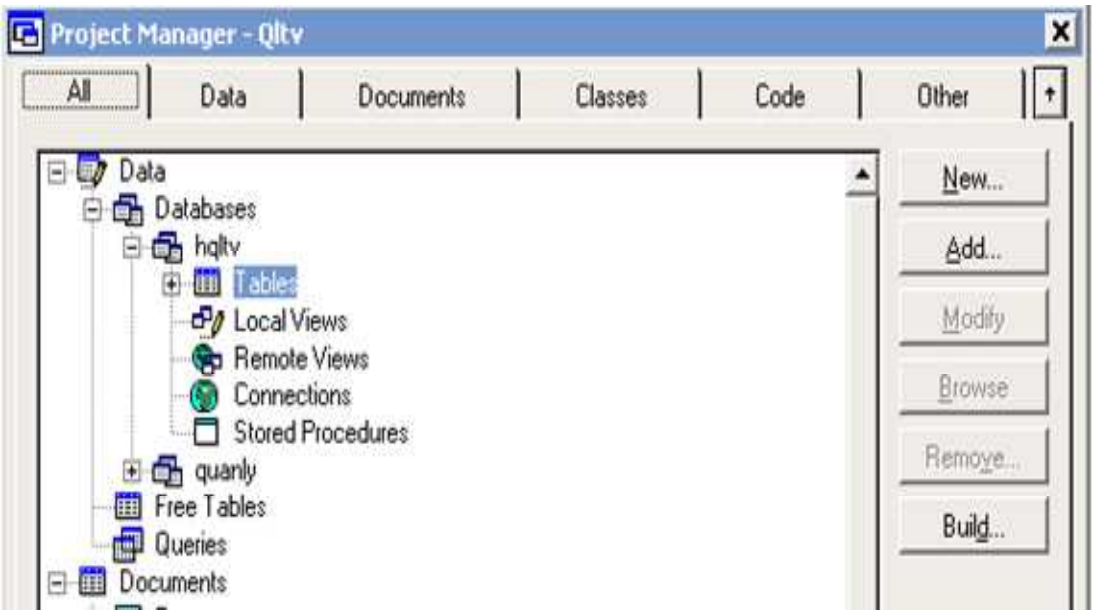
Công cụ quản lý các dự án (Project Manager)

Công cụ này có 2 mục đích chính:

1. Giúp việc tổ chức các file dựa trên các kiểu file
2. Cung cấp khung quản lý các thành phần của một ứng dụng để có thể dịch chúng sang dạng APP hay EXE.

Hình 4.3.1.8 Cửa sổ Project Manager

Project Manager cho phép người sử dụng quản lý tất cả các loại file trong cơ sở dữ liệu, các mục chính bao gồm: dữ liệu (data), tài liệu (documents), thư viện (class libraries), mã (code), và các mục khác. Mục dữ liệu chứa các mục nhỏ hơn là: các cơ sở dữ liệu (Databases), các bảng tự do (Free Tables), và các truy vấn (Queries). Như trong hình trên, mục dữ liệu chứa 2 cơ sở dữ liệu có tên tên lần lượt là: HQLTV và QUANLY.



Hình 4.3.1.9 Thành phần của một cơ sở dữ liệu

Thành phần chính của một cơ sở dữ liệu bao gồm: Các bảng (Tables), các khung nhìn (Local Views),... Các bảng tự do là các bảng không thuộc cơ sở dữ liệu nào, thường được sử dụng như các bảng tạm trong quá trình xử lý dữ liệu hay các bảng chứa các tham số của hệ thống. Các thủ tục lưu trong cơ sở dữ liệu và các khung nhìn từ xa thông qua một kết nối có thể liên kết đến một cơ sở dữ liệu khác (có thể không phải là Visual Foxpro).

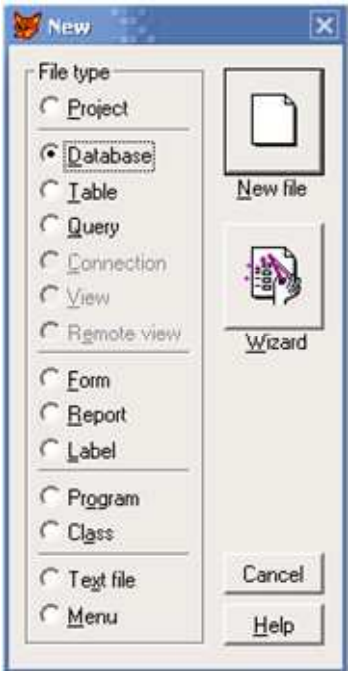
Mục Documents chứa các biểu mẫu: Forms và Reports: Reports sử dụng cho một chương trình ứng dụng chạy trên các cơ sở dữ liệu. Class Libraries chứa các lớp do người sử dụng tự định nghĩa. Mục Code chứa các file mã chương trình dạng .PRG, các hàm thư viện API, các file thực thi dưới dạng .EXE. Mục Others chứa các file khác mà Dự án sử dụng chẳng hạn như: các file biểu tượng, ảnh tĩnh, các file tham số, ...

Chúng ta sẽ thực hiện tạo cơ sở dữ liệu, và các bảng dữ liệu theo quy trình của Visual FoxPro. Lưu ý rằng, các bảng dữ liệu được lưu dưới dạng các file riêng rẽ có phần mở rộng .DBF, trong khi cơ sở dữ liệu được lưu dưới dạng .DBC (Database Container). Để có thêm thông tin về các dạng file của Visual FoxPro, người sử dụng có thể tra cứu trong phần Hướng dẫn sử dụng của chương trình hay các cuốn sách về nó.

Để tạo một cơ sở dữ liệu mới,cần thực hiện theo các bước sau:

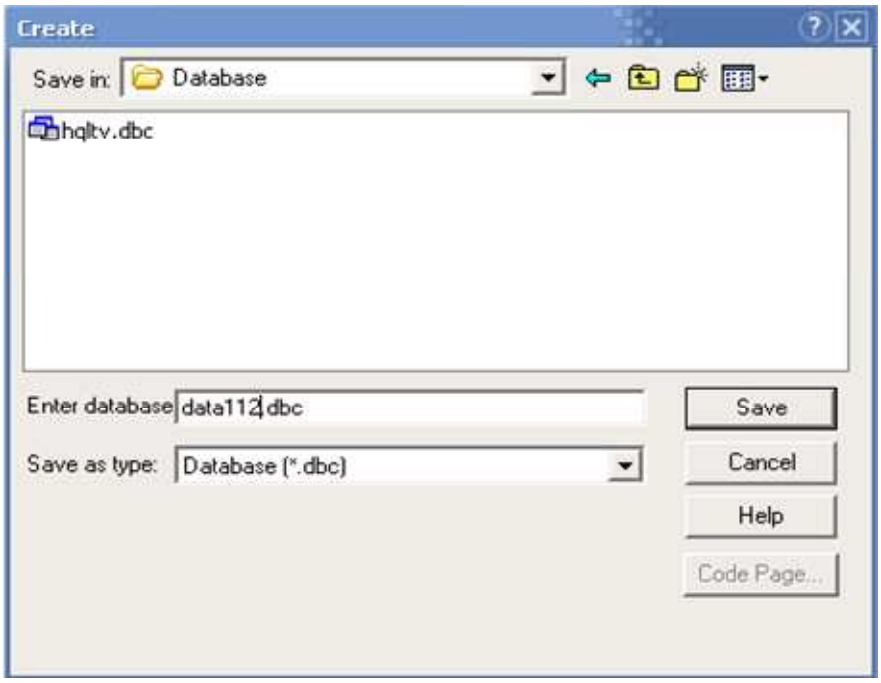
Bước 1: Tạo một Database mới bằng cách chọn trên thực đơn chương trình: ***File\New...***

Visual FoxPro sẽ hỏi xem muốn tạo file loại gì trong hộp thoại sau đây:



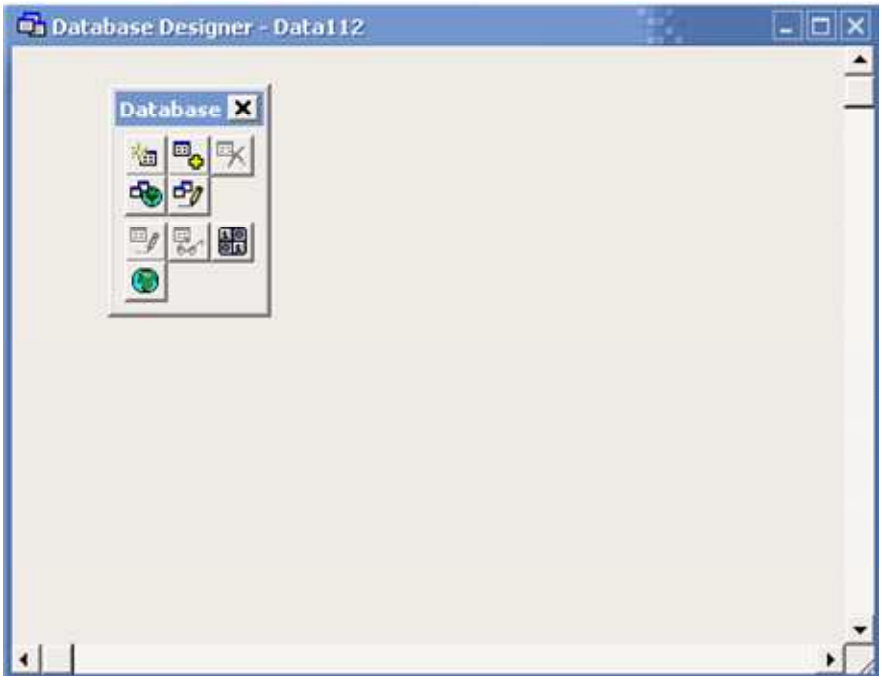
Hình 4.3.1.10 thoại chọn tạo mới một cơ sở dữ liệu

Hãy chọn Database như trong hình trên và nhấn nút lệnh "New file". Sau đó nhập đường dẫn và tên của cơ sở dữ liệu vào hộp thoại kế tiếp như sau:



Hình 4.3.1.11 Nhập đường dẫn và tên của cơ sở dữ liệu cần tạo

Cần lưu ý rằng các file cơ sở dữ liệu của Visual FoxPro có phần mở rộng là DBC. Đây chính là điểm khác biệt so với các phiên bản FoxPro for DOS, đó là sự phân biệt rõ ràng giữa các bảng dữ liệu và cơ sở dữ liệu. Sau khi nhập đường dẫn và tên file cơ sở dữ liệu, nhấn nút lệnh Save để lưu lại. Visual FoxPro sẽ mở cửa sổ thiết kế cơ sở dữ liệu để người sử dụng có thể thực hiện bổ sung, cập nhật các thành phần khác của cơ sở dữ liệu.



Hình 4.3.1.12 Cửa sổ Database Designer

Bước 2: Tạo các bảng dữ liệu

Để tạo mới một bảng dữ liệu, chúng ta sử dụng thanh công cụ Database, chọn nút lệnh **New Table**.

Hình 4.3.1.13 Nút tạo dựng bảng dữ liệu

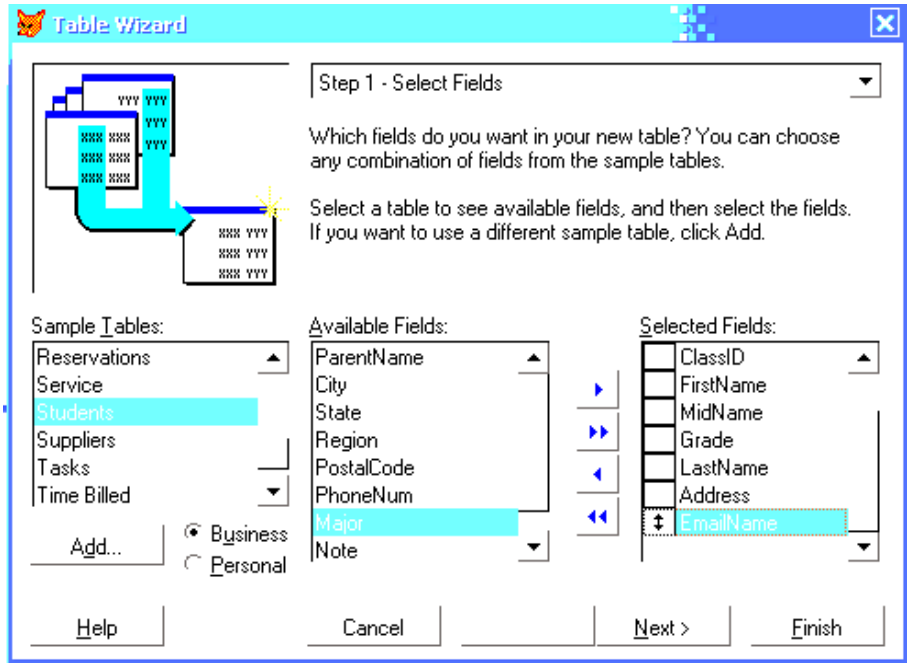
Người sử dụng có thể chọn một trong hai cách tạo bảng:

- Sử dụng Wizard

- Tạo bảng thủ công

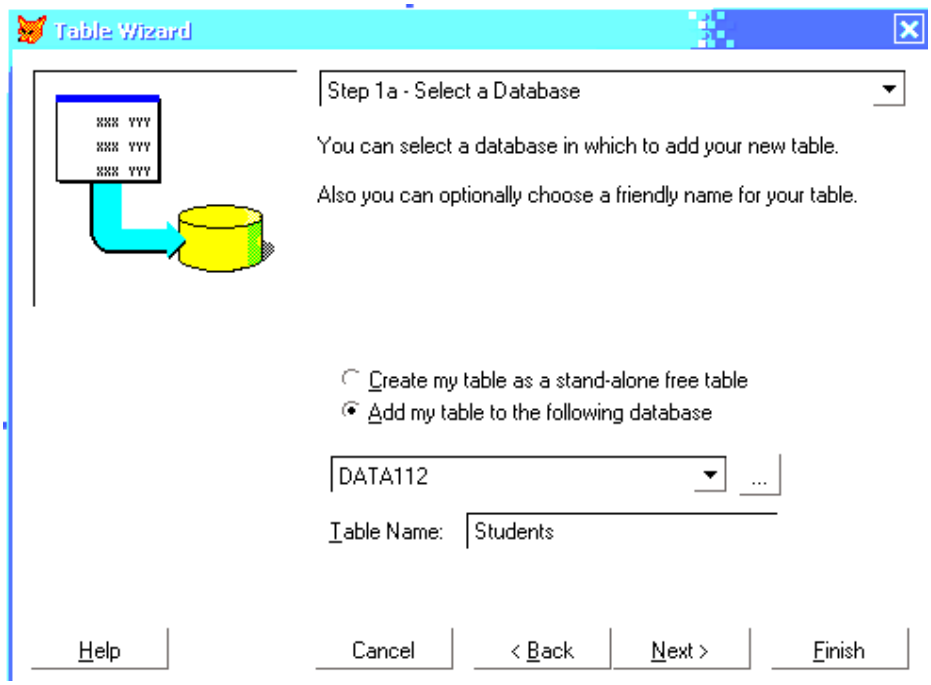
Hình 4.3.1.14 Hộp thoại lựa chọn cách tạo bảng

Nếu chọn Table Wizard, chương trình đưa ra các bước để người sử dụng có thể chọn:



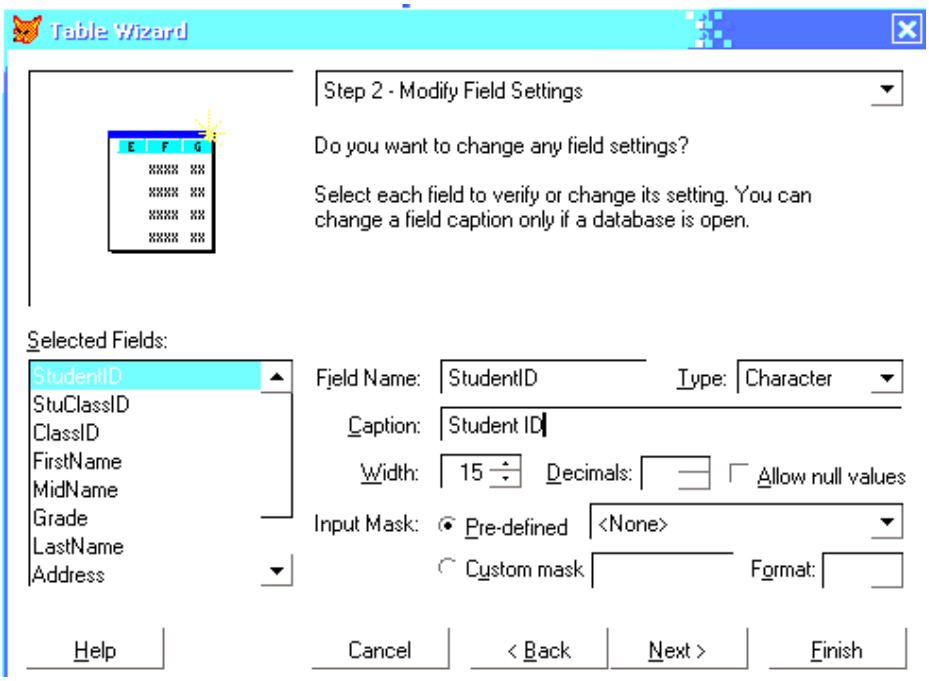
Hình 4.3.1.15 Chọn bảng mẫu, các trường dữ liệu cần thiết để tạo bảng

Nhấn **Next** để sang bước tiếp theo, chọn lựa cơ sở dữ liệu mà bảng sẽ tạo thuộc về:



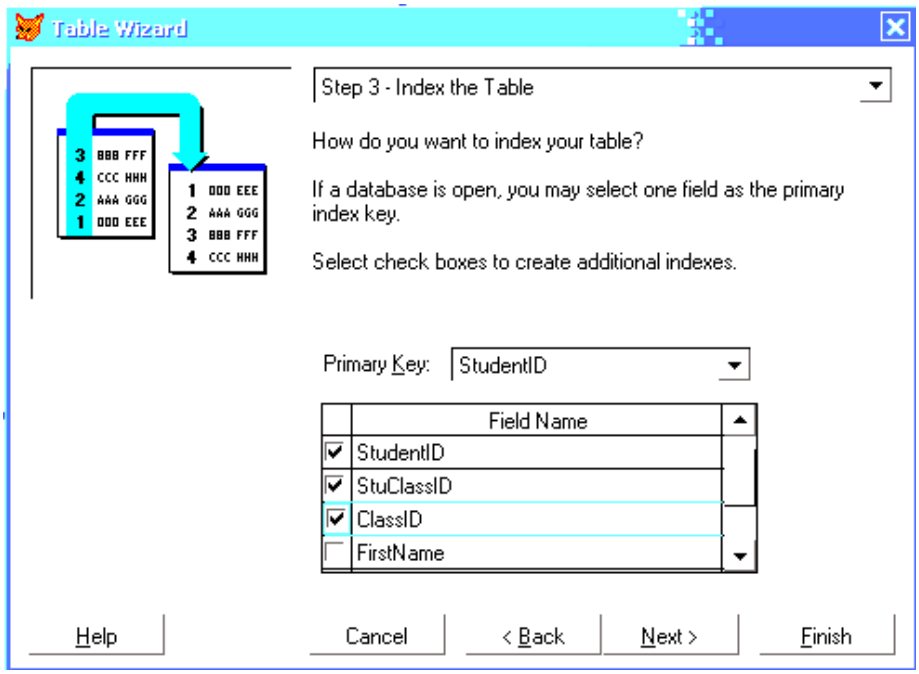
Hình 4.3.1.16 Chọn cơ sở dữ liệu

Nếu muốn tạo bảng mới như một bảng tự do, chọn “Create my table as a stand-alone free table”, ngược lại, chọn cơ sở dữ liệu mà chúng ta vừa tạo ra. Nhấn **Next** để sang bước kế tiếp tạo lập cấu trúc cụ thể cho bảng:



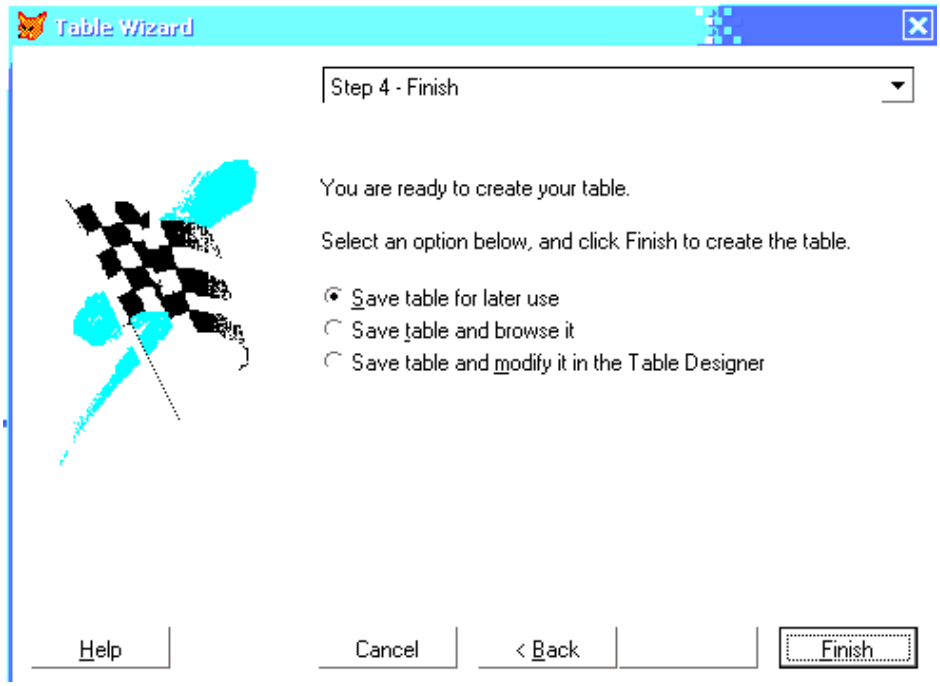
Hình 4.3.1.17 Sửa đổi cấu trúc các trường dữ liệu cho bảng

Người sử dụng cần xác định kiểu dữ liệu, mô tả độ rộng,... cho các trường dữ liệu. Khi hoàn tất, nhấn **Next** để chuyển sang bước kế tiếp tạo lập chỉ mục cho bảng



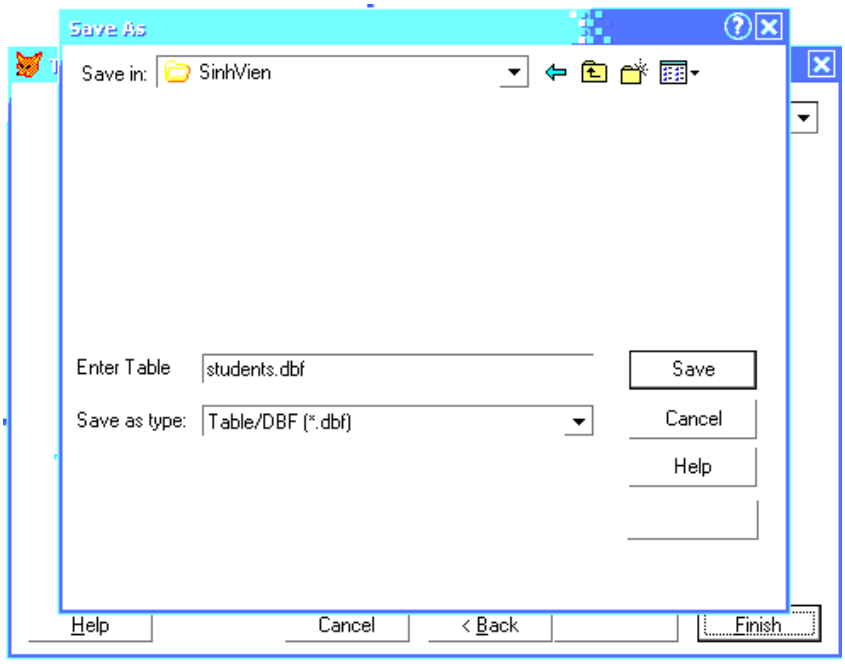
Hình 4.3.1.18 Tạo lập chỉ mục

Mục Primary Key cho phép người sử dụng xác định trường đóng vai trò khóa chính cho bảng dữ liệu. Nếu muốn tạo thêm các chỉ mục cho bảng, chọn các ô check phía bên dưới. Nhấn **Next** để sang bước kế tiếp tạo lập và lưu lại bảng đó



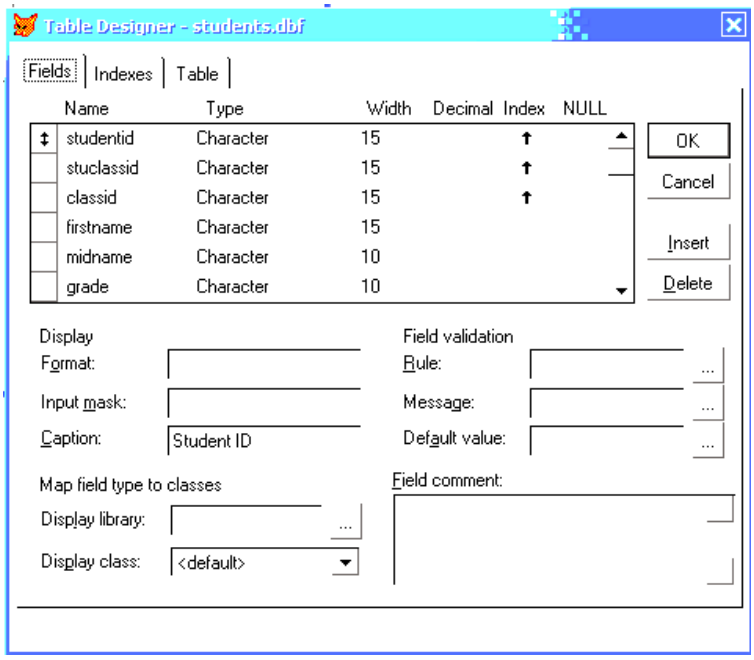
Hình 4.3.1.19 Kết thúc và lưu giữ bảng

Nhấn **Finish** để tạo và lưu lại hoặc xem lại cấu trúc của bảng (Chọn Save table and modify ...). Sau đó xác định đường dẫn và tên file chứa bảng dữ liệu này.



Hình 4.3.1.20 Xác định thư mục lưu bảng

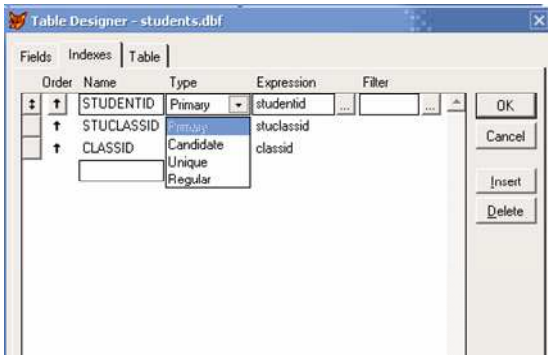
Và nhấn **Save** để lưu lại.



Hình 4.3.1.21 Sửa đổi, xem lại cấu trúc bảng

Nếu chọn New Table, để tạo lập bảng mới bằng tay, cửa sổ để người sử dụng tạo bảng tương tự như trong hình trên. Để tạo các chỉ mục, chọn Tab **Indexes** như trong hình trên, và

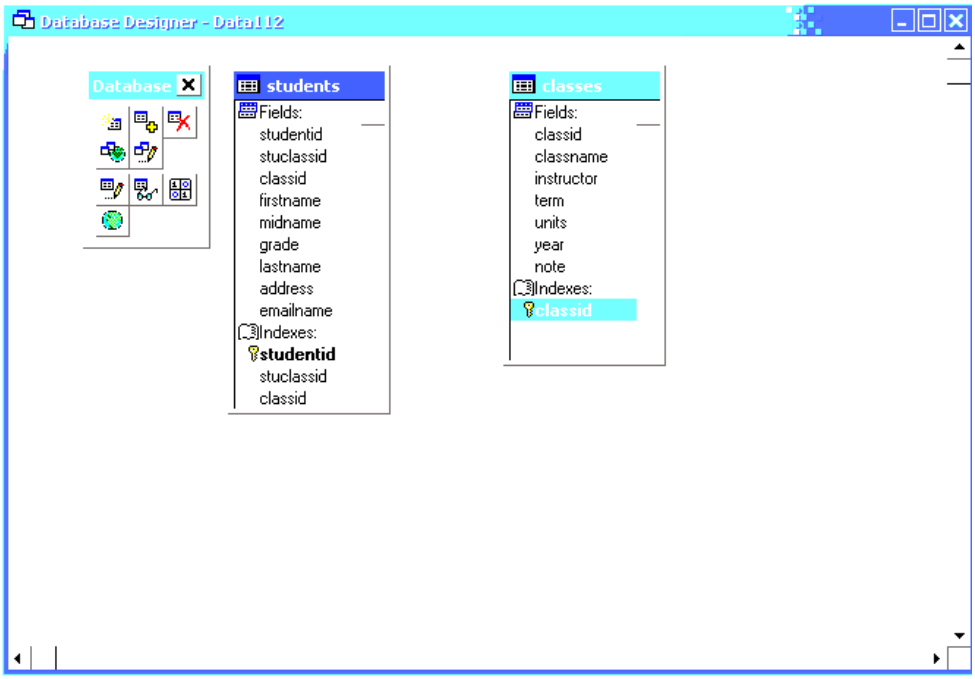
thiết lập các chỉ mục bằng cách chọn trường dữ liệu và cách sắp xếp của chỉ mục.



Hình 4.3.1.22 Tạo chỉ mục

Có thể chọn 1 trong 4 kiểu chỉ mục: Khóa chính (Primary), Ứng cử (candidate), Duy nhất (Unique), hay thông thường (Regular). Cột Expression cho phép tạo các chỉ mục bằng cách ghép các trường hoặc sử dụng các hàm do người sử dụng tự tạo.

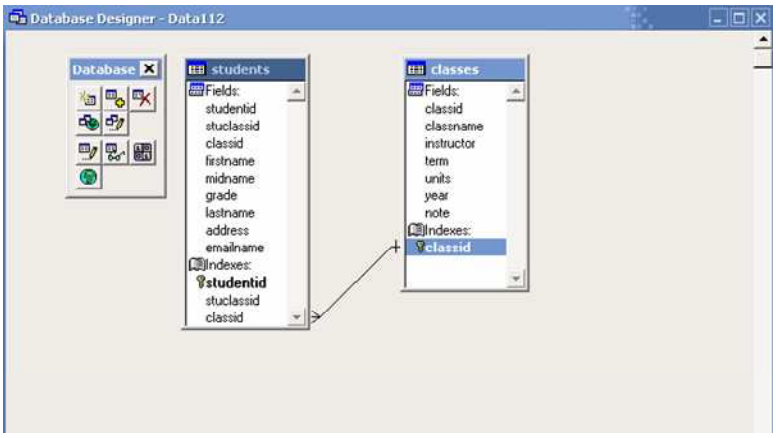
Lặp lại các thao tác trên, người sử dụng có thể tạo ra các bảng dữ liệu khác cho mình.



Hình 4.3.1.23 Các bảng trong cơ sở dữ liệu

Bước 3: Thiết lập các liên kết trên các bảng đã tạo

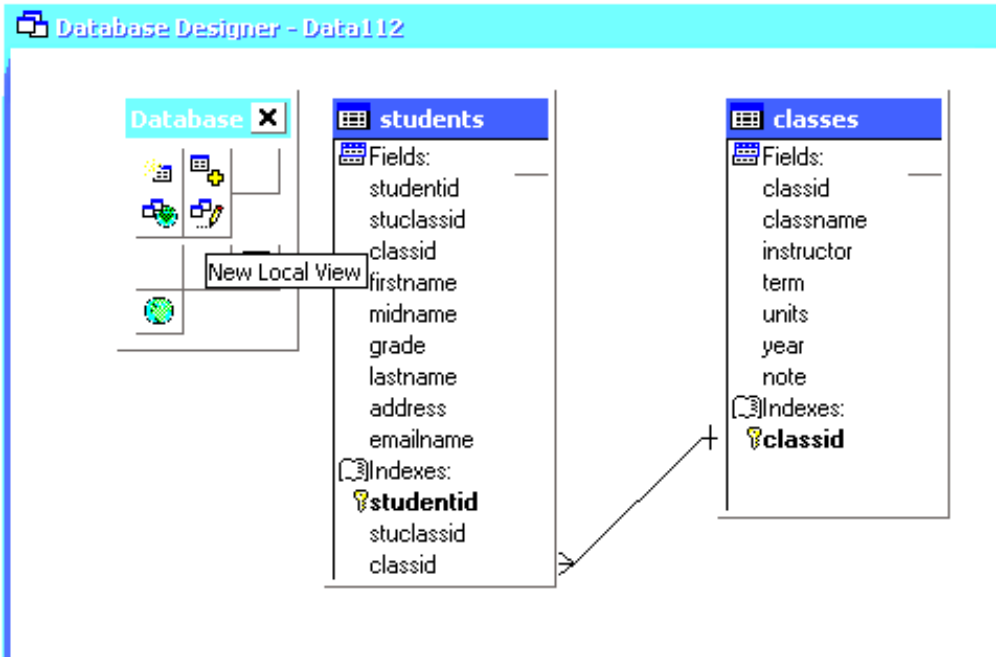
Cần chú ý rằng, để có thể tạo ra các liên kết giữa các bảng, cần thực hiện định nghĩa và tạo các chỉ mục trên các trường mà thông qua đó, chúng liên kết được với nhau. Chẳng hạn, trong hình vẽ trên, chúng ta có 2 bảng: Students và Classes. Liên kết được xác định giữa chúng là: Mỗi lớp (class) sẽ có 0 hoặc nhiều sinh viên (student). Chúng ta đã thực hiện tạo chỉ mục cho bảng Students trên trường *classid*, việc tạo liên kết giữa 2 bảng này được thực hiện bằng cách thông qua cửa sổ Database Designer. Giữ chuột vào trường **classid** đã được định nghĩa là khóa chính của bảng Classes, kéo và thả tại chỉ mục *classid* bên bảng Students. Khi đó Visual FoxPro sẽ tạo liên kết giữa 2 bảng này dựa trên trường dữ liệu chung: *classid*.



Hình 4.3.1.24 Tạo liên kết giữa 2 bảng dữ liệu

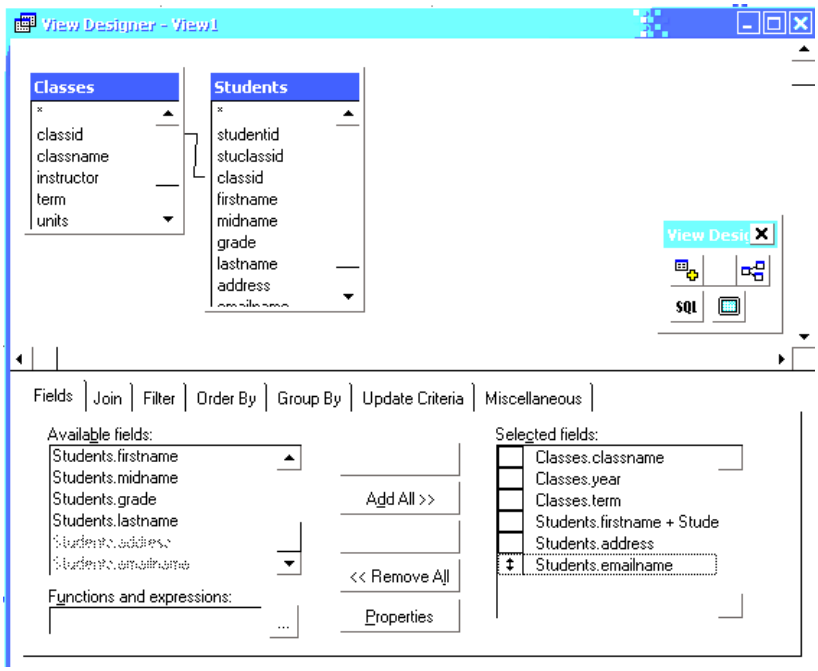
Bước 4: Tạo lập các khung nhìn trên các bảng đã có

Nhấn nút lệnh **New Local View** để bắt đầu thực hiện chức năng tạo khung nhìn.



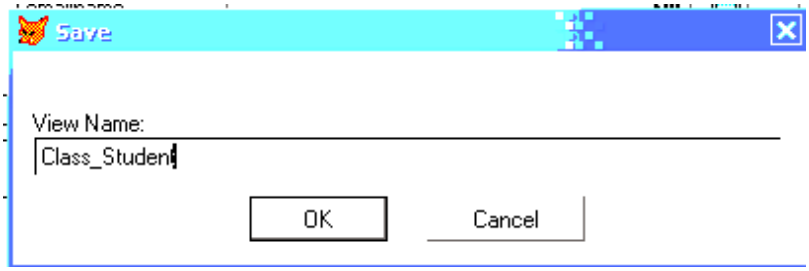
Hình 4.3.1.25 Tạo một khung nhìn nội bộ

Màn hình tiếp theo đó trợ giúp việc tạo các khung nhìn một cách trực quan hơn. Các bảng được chọn sẽ tự động hiển thị các liên kết giữa chúng như trong hình sau đây.



Hình 4.3.1.26 Xác định các bảng và trường dữ liệu

Hãy xác định các trường dữ liệu trong mục **Fields**, xác nhận và thêm vào các điều kiện kết nối nếu cần,... sau đó nhấn **Save** trên thanh công cụ chuẩn. Khi đó, khung nhìn mới định nghĩa sẽ được tạo với tên mà chúng ta nhập vào trong màn hình tiếp sau.



Hình 4.3.1.27 Nhập tên khung nhìn mới

3.2. Các hệ quản trị cơ sở dữ liệu cỡ lớn

a. Hệ quản trị cơ sở dữ liệu SQL Server

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu do công ty Microsoft xây dựng. Là một trong những hệ quản trị cơ sở dữ liệu phục vụ cho các hệ thống nhỏ lúc ban đầu, ngày nay, Microsoft SQL Server không ngừng được đổi mới,hoàn thiện và ngày càng đáp ứng được những yêu cầu khắt khe về tính tiện dụng, khả năng quản lý cơ sở dữ liệu,...

Phiên bản rút gọn của Microsoft SQL Server là MSDE (Microsoft SQL Server Desktop Engine) được đưa ra phục vụ cho các sản phẩm khác của Microsoft như: Visual FoxPro, MS Access, MS Web Matrix, và các sản phẩm khác. Hiện tại, MSDE bị giới hạn ở mức 2 GB bộ nhớ, 8 kết nối đồng thời, và không có công cụ để quản trị.

Microsoft SQL Server sử dụng một biến thể của ngôn ngữ SQL, Transact-SQL, một tập bao của ngôn ngữ chuẩn SQL-92 (Tiêu chuẩn ISO cho SQL, được chứng nhận vào năm 1992). T-SQL được gọi là tập bao của SQL-92 vì nó có hỗ trợ thêm các cú pháp sử dụng trong việc viết các thủ tục lưu trực tiếp trong cơ sở dữ liệu (stored procedures), và thực hiện các giao dịch (transactions support). Theo số liệu thống kê của hãng Gartner, vào tháng 09 năm 2001, Microsoft SQL Server chiếm khoảng 14% thị trường cơ sở dữ liệu thương mại.

Một điểm khác biệt của Microsoft SQL Server so với các hệ quản trị cơ sở dữ liệu khác đó là không thực hiện khóa dữ liệu ở mức bản ghi một cách chính tắc. Thay vào đó, Microsoft SQL Server sử dụng một ứng dụng khác để đảm bảo rằng không có 2 người sử dụng cùng thực hiện truy vấn

tại một thời điểm. Nếu tình trạng này xảy ra có thể gây ra hệ thống bị treo.

Các phiên bản của Microsoft SQL Server:

Việc hiểu rõ về khả năng và hạn chế của các phiên bản sẽ giúp chúng ta có được sự lựa chọn đúng đắn nhất khi mua sản phẩm này. Một điều cần lưu ý là tất cả các phiên bản này đều sử dụng cùng cấu trúc tệp và tập các lệnh của Transact-SQL (T-SQL). Điều đó có nghĩa là các file cơ sở dữ liệu tạo bởi một phiên bản có thể được sử dụng bởi các phiên bản khác của Microsoft SQL Server.

Standard Edition

Đây là phiên bản có lẽ được sử dụng rộng rãi nhất của Microsoft SQL Server. Phiên bản này hỗ trợ từ 1-4 bộ xử lý (CPU) và 2GB RAM, có thể đáp ứng phần lớn các nhu cầu của các cơ quan, xí nghiệp cỡ nhỏ. Phiên bản này hỗ trợ tất cả các tính năng mà Microsoft SQL Server đã nổi tiếng:

- Data Transformation Services (DTS), Các dịch vụ hỗ trợ chuyển đổi dữ liệu, cung cấp tính năng import, export, và chuyển đổi dữ liệu giữa nhiều nguồn khác nhau.
- Cho phép thực hiện tạo bản sao (Replication) giữa Microsoft SQL Server và các sản phẩm của các nhà cung cấp khác
- Có khả năng thực hiện các truy vấn gần ngôn ngữ Tiếng Anh (English Query)
- Hỗ trợ XML (eXtensive Mark-up Language)
- Có khả năng tìm kiếm toàn văn (Full-text searches)

Mặc dù vậy, phiên bản này không hỗ trợ khả năng phân cụm (clustering) và một vài chức năng nâng cao cho việc phân tích dữ liệu trực tuyến (OLAP). Phiên bản này chỉ có thể cài đặt trên hệ thống Windows 2000 Server hay cao hơn.

Personal Edition

Phiên bản này cũng có các chức năng giống như phiên bản Standard. Ràng buộc chính của nó là về bản quyền và các giới hạn về hệ điều hành.

Phiên bản dùng cho cá nhân này không thể sử dụng nhiều hơn 2 bộ vi xử lý, và nó có một bộ điều chỉnh tốc độ truy vấn. Bộ điều chỉnh truy vấn này sẽ làm ảnh hưởng đến hiệu năng nếu như bị quá tải. Đồng thời, phiên bản này cũng không thực hiện được việc tạo bản sao theo mô hình giao dịch. Mặc dù vậy nó có thể là một subscriber trong mô hình tạo bản sao nêu trên.

Phiên bản này có thể cài đặt trên Windows 98 hoặc cao hơn. Phiên bản này có kèm theo chức năng tìm kiếm toàn văn (full-text search) và dịch vụ phân tích dữ liệu (SQL Server 2000 Analysis Services), nhưng các tính năng này không thể cài đặt trên Windows 98 hay Windows ME. Một giới hạn nữa là các hệ điều hành như Windows 98 không hỗ trợ các dịch vụ như Windows 2000, XP, do vậy, một số các chức năng như SQL Agent sẽ không chạy khi chúng ta chưa đăng nhập vào Windows..

Microsoft Data Engine (MSDE)

Microsoft Data Engine (MSDE) cũng là một phiên bản được sử dụng rộng rãi của SQL Server. Bộ cài đặt của phần mềm này có thể tìm thấy trong bộ công cụ Visual Studio Development. Cũng có thể mua nó riêng lẽ hoặc trong bộ

Office Professional. MSDE thay thế JET với vai trò như cơ chế cung cấp dữ liệu trong các ứng dụng phân tán.

MSDE có cùng các đặc trưng cơ bản như phiên bản Standard, và cũng là một công cụ tốt cho việc phát triển các ứng dụng vì có thể chuyển đổi việc sử dụng nó sang các phiên bản Microsoft SQL Server khác do có cùng cấu trúc file như đã nêu trong các phần trên.

Rất nhiều các máy chủ web sử dụng phiên bản Microsoft SQL Server này bởi lẽ việc phân phối nó trong các bộ cài đặt là miễn phí. Nói một cách khác, nếu sử dụng các công cụ như: SQL Server 2000 (Developer, Standard, and Enterprise Editions), Visual Studio .NET (Architect, Developer, and Professional Editions), ASP.NET Web Matrix Tool, Office XP Developer Edition, MSDN Universal and Enterprise subscriptions thì chúng ta có thể cung cấp MSDE cài đặt cùng với ứng dụng của mình. Tất nhiên là chúng ta phải mua bản quyền của công cụ mà chúng ta sử dụng để phát triển.

Có 2 hạn chế chủ yếu của MSDE, đó là:

- Nó có một bộ điều chỉnh tốc độ truy vấn, giới hạn hiệu năng của các ứng dụng có lớn hơn 5 user, và tất nhiên, điều đó phụ thuộc vào cách viết chương trình. Không có các công cụ trực quan đi kèm để quản trị và các sách hướng dẫn trực tuyến.
- Các cơ sở dữ liệu bị giới hạn ở mức dưới 2GB, và MSDE không thể dùng SQL Mail.

Enterprise Edition

Phiên bản Enterprise cung cấp tất cả các chức năng mà các phiên bản khác có và hơn thế nó có thể sử dụng đến 32 bộ

vi xử lý và 64GB RAM có khả năng đem lại hiệu năng tốt nhất cho người sử dụng. Khả năng này đặc biệt rõ khi thực hiện các truy vấn song song do máy chủ cơ sở dữ liệu sử dụng nhiều bộ vi xử lý để thực hiện các truy vấn trong cùng một lúc.

Một lợi thế khác của phiên bản này là khả năng mở rộng. Nếu cài đặt trên hệ điều hành Windows 2000 Advanced Server, chúng ta có thể thực hiện chuyển hướng các yêu cầu dữ liệu sang một máy chủ cơ sở dữ liệu khác nhằm đảm bảo hệ thống hoạt động thông suốt.

Khi sử dụng phiên bản này, chúng ta có thể định nghĩa các phân đoạn OLAP (quá trình phân tích trực tuyến) và tạo các khối (cubes) chứa các dữ liệu đa chiều. Phiên bản này cũng hỗ trợ thực hiện các phân tích trực tuyến gần như trong thời gian thực.

Một điểm mạnh nữa là người phát triển ứng dụng có thể tạo các khung nhìn được đánh chỉ số, góp phần làm tăng tốc độ các báo cáo. Tuy nhiên, để đạt được điều này thì yêu cầu về cấu hình của máy chủ chắc chắn sẽ cao hơn so với các phiên bản khác.

Developer Edition

Với phiên bản này, chúng ta có thể sử dụng đế phát triển các ứng dụng chạy trên nền Hệ quản trị cơ sở dữ liệu SQL Server. Phiên bản này có tất cả các đặc trưng của phiên bản Enterprise ngoại trừ bản quyền phát triển không cho phép triển khai phiên bản này. Nó có thể cài trên Windows 2000 Professional và cao hơn.

Với phiên bản này, chúng ta có quyền cài đặt SQL Server 2000 phiên bản cho Windows CE và triển khai các ứng

dụng đó miễn phí, mặc dù các thiết bị CE cần có bản quyền cho phiên bản SQL Server chạy trên CE.

Một đặc điểm khác là nó có thể nâng cấp lên phiên bản Standard hoặc Enterprise.

Các phiên bản dùng thử (Evaluation)

Các phiên bản này thường có đầy đủ các chức năng nhưng người sử dụng chỉ có thể xem và test các chức năng đó trong vòng 120 ngày. Các phiên bản này không thể dùng để phát triển hay triển khai mà chỉ để phục vụ cho mục đích có đi đến quyết định mua sản phẩm hay không.

CE Edition

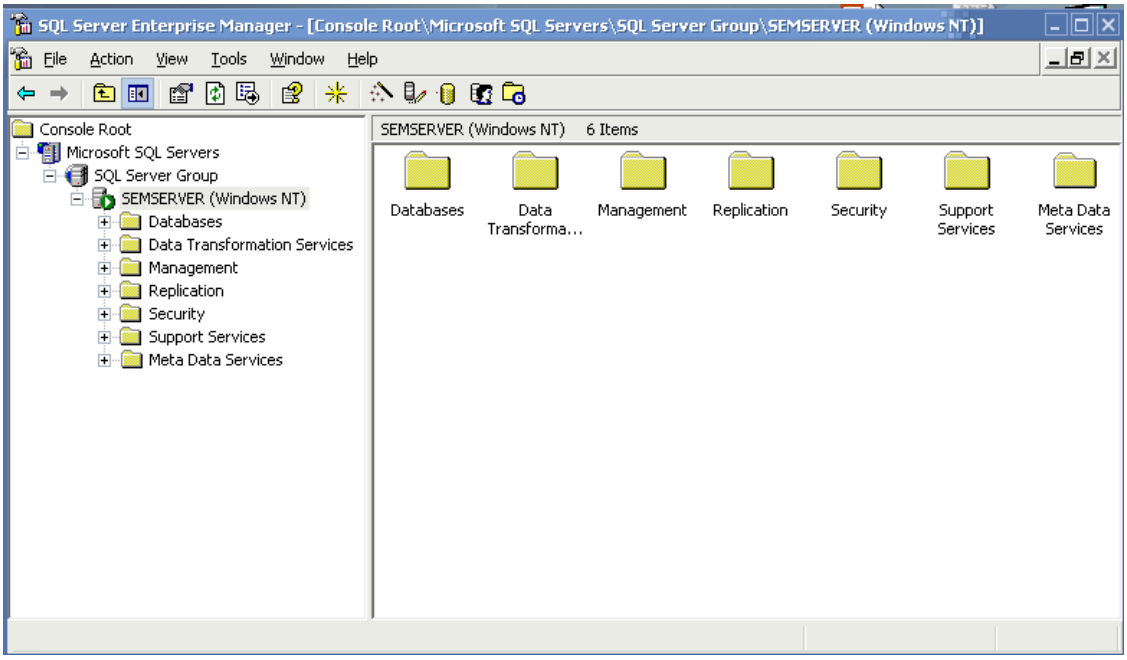
Phiên bản CE của Microsoft SQL Server có cùng các công cụ và cú pháp SQL mà người phát triển có thể dùng trong các ứng dụng dựa trên nền cơ sở dữ liệu SQL Server. Cơ sở dữ liệu này ở dạng quan hệ và có thể chạy trên các thiết bị bỏ túi. Mục đích chính của phiên bản này là dùng cho các ứng dụng phân tán. Điều này được thực hiện thông qua giao thức mạng HTTP, có hỗ trợ mã hóa. Hệ điều hành là Windows CE 2.11 hoặc cao hơn, và có thể sử dụng với cơ sở dữ liệu đến 2 GB tuy rằng, hiện tại chưa có thiết bị CE nào có đến 2 GB RAM.

Sau đây, chúng ta sẽ tìm hiểu cách thức tạo các cơ sở dữ liệu, các bảng, các quan hệ, ...

Công cụ Enterprise Manager

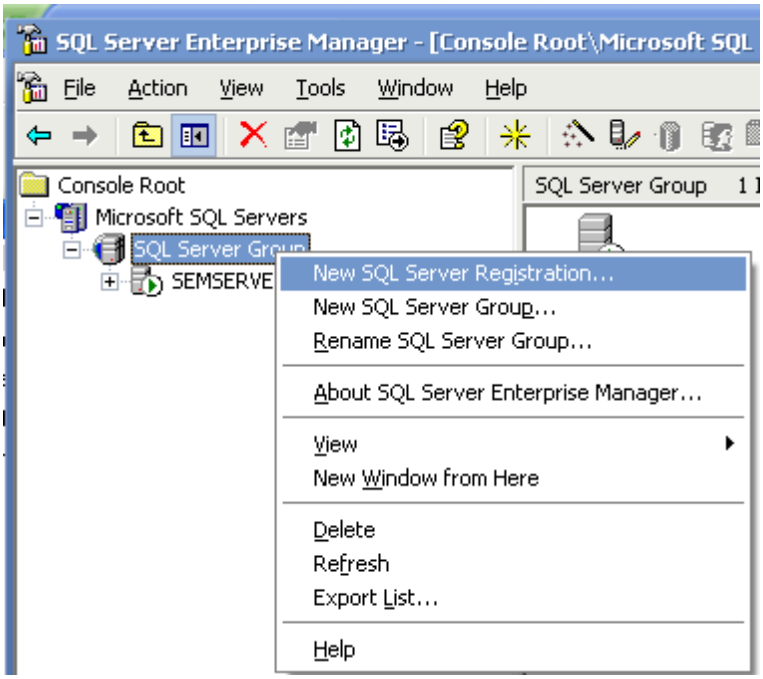
Như đã nêu trên, công cụ quản trị trực quan cơ sở dữ liệu được bán kèm trong các phiên bản như Standard, Enterprise, Developer. Công cụ này có tên là Enterprise Manager, giúp người sử dụng có thể thực hiện các thao tác

quản trị cơ sở dữ liệu một cách trực quan và dễ dàng. Màn hình chính của công cụ này như sau:



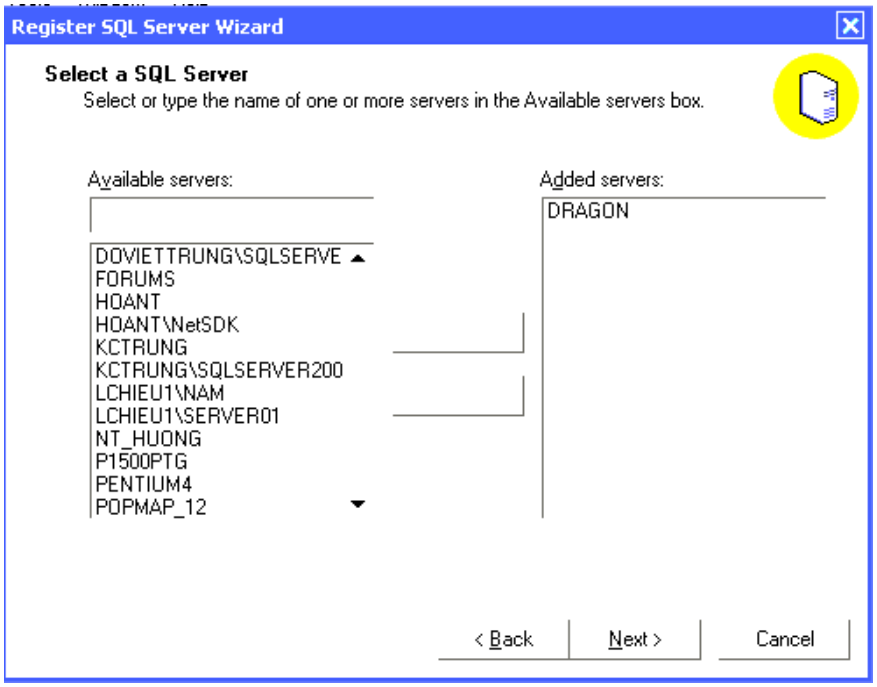
Hình 4.3.2.1 Màn hình chính công cụ Enterprise Manager

Công cụ này giúp chúng ta thực hiện các công việc liên quan đến cơ sở dữ liệu một cách dễ dàng. Điều đầu tiên là thực hiện đăng ký một mục cho máy chủ SQL Server mà chúng ta cần thực hiện quản trị. Để làm được điều này, nhấn chuột phải vào mục SQL Server Group, chọn **New SQL Server Registration** để đăng ký một mục mới như trong hình sau.



Hình 4.3.2.2 Đăng ký một máy chủ SQL Server mới

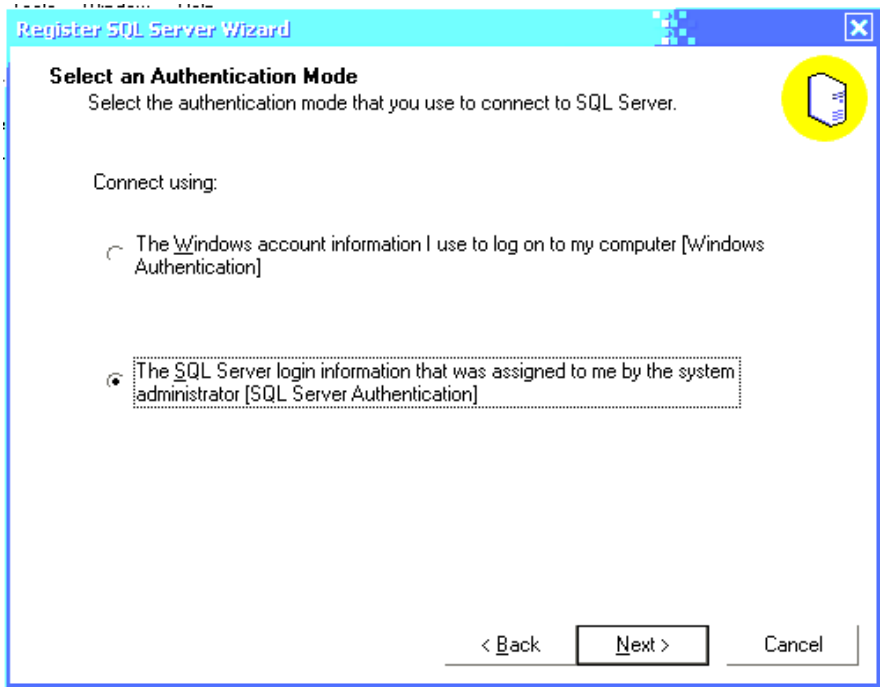
Nhấn **Next** ở màn hình Welcome đầu tiên, chương trình sẽ tự động tìm kiếm các máy chủ SQL Server đang hoạt động trên mạng nội bộ. Chọn đúng tên máy chủ SQL Server mà chúng ta cần thực hiện quản trị và nhấn **Next**:



Hình 4.3.2.3 Chọn máy chủ SQL Server cần quản trị

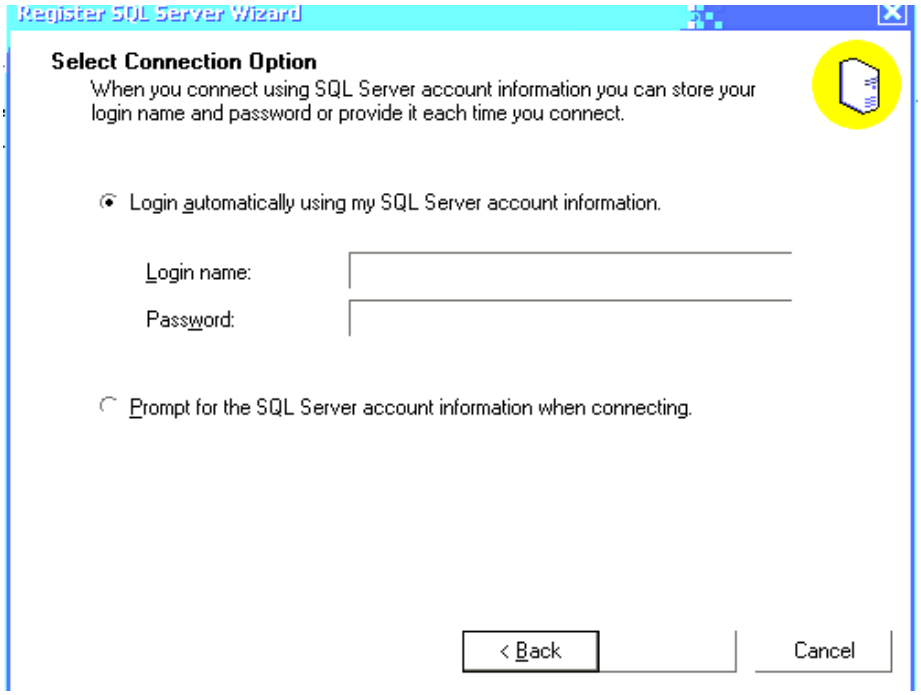
Màn hình tiếp theo cho phép chúng ta lựa chọn cách thức đăng nhập SQL Server thông qua tài khoản truy nhập mạng hoặc thông qua người sử dụng (được gán quyền)của SQL Server đó. Đối với trường hợp thứ nhất, chúng ta không cần phải nhập thông tin đăng nhập của SQL Server vì chúng ta đã thực hiện đăng nhập hệ thống thông qua tài khoản truy nhập mạng (do máy chủ SQL Server cung cấp). Chúng ta sẽ đi vào trường hợp thứ hai.

Chọn SQL Server Authentication như trong hình sau và nhấn **Next**:



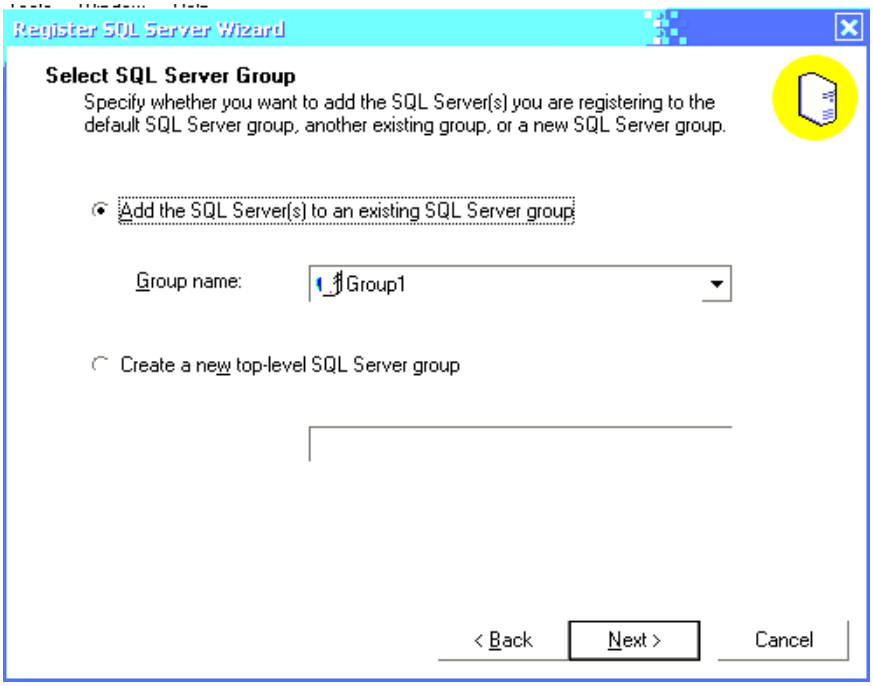
Hình 4.3.2.4 Chọn SQL Server Authentication

Trong màn hình tiếp theo, nhập các thông tin đăng nhập vào SQL Server, thông thường, các thông tin này được người quản trị hệ thống cơ sở dữ liệu cung cấp.



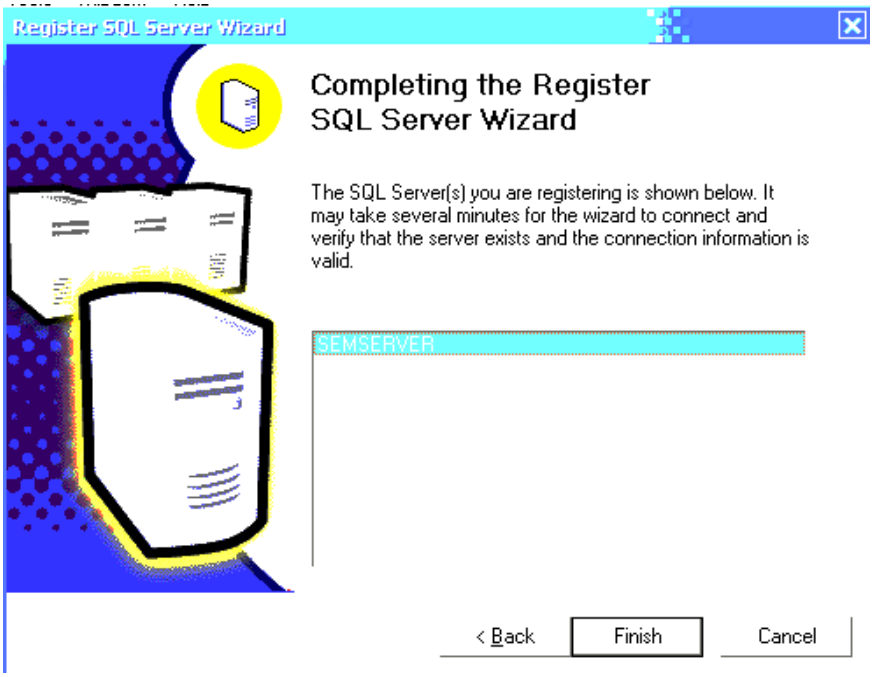
Hình 4.3.2.5 Nhập các thông tin đăng nhập

Sau khi nhập các thông tin đầy đủ, nhấn **Next** để tiếp tục và chọn nhóm mà chúng ta muốn đưa máy chủ đó vào:



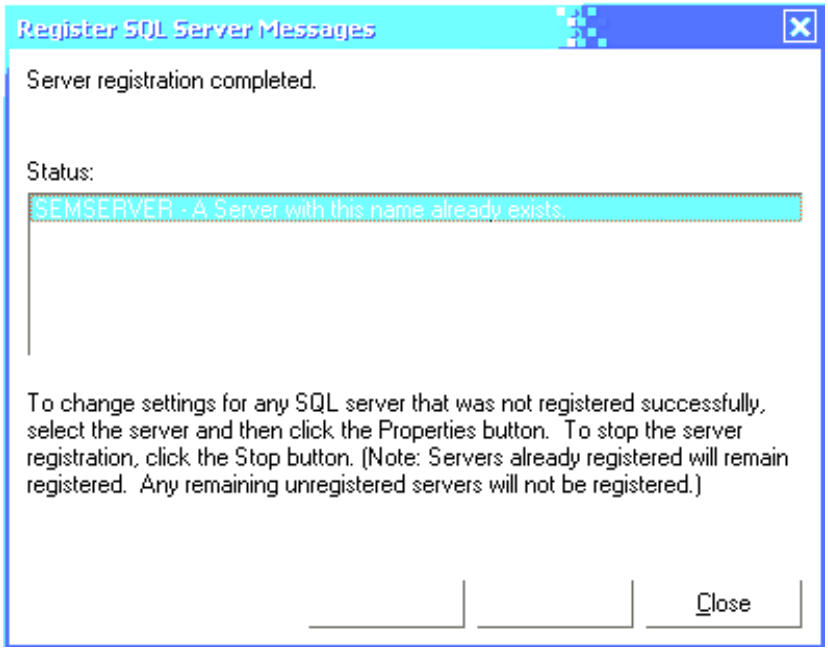
Hình 4.3.2.6 Chọn nhóm cho máy chủ SQL Server

Nhấn **Next**, màn hình tiếp theo cho phép kiểm tra việc đăng ký có thành công hay không. Nhấn **Finish** để thực hiện kiểm tra.



Hình 4.3.2.7 Kết thúc bước đăng ký

Trong trường hợp gặp lỗi, ví dụ như trong hình sau:



Hình 4.3.2.8 Lỗi máy chủ đã được đăng ký

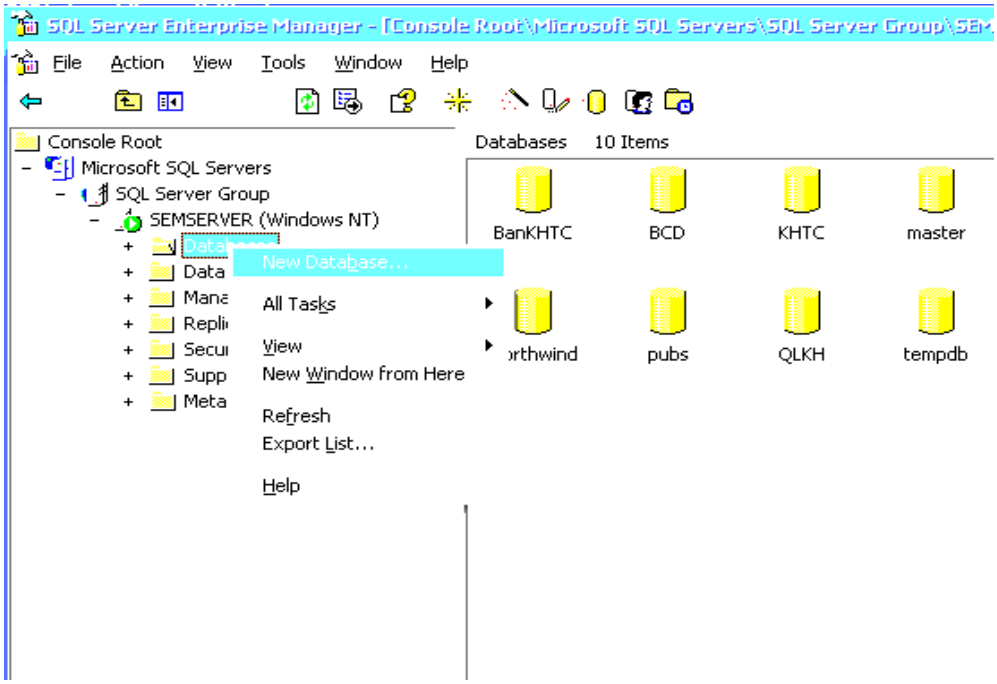
Khi gặp lỗi trên, nghĩa là chúng ta đã đăng ký máy chủ SQL Server đó từ trước rồi, vì vậy có thể bắt đầu các thao

tác khác. Nếu các thông số đều chính xác, chương trình sẽ thông báo thành công như sau:

Hình 4.3.2.9 Đăng ký thành công

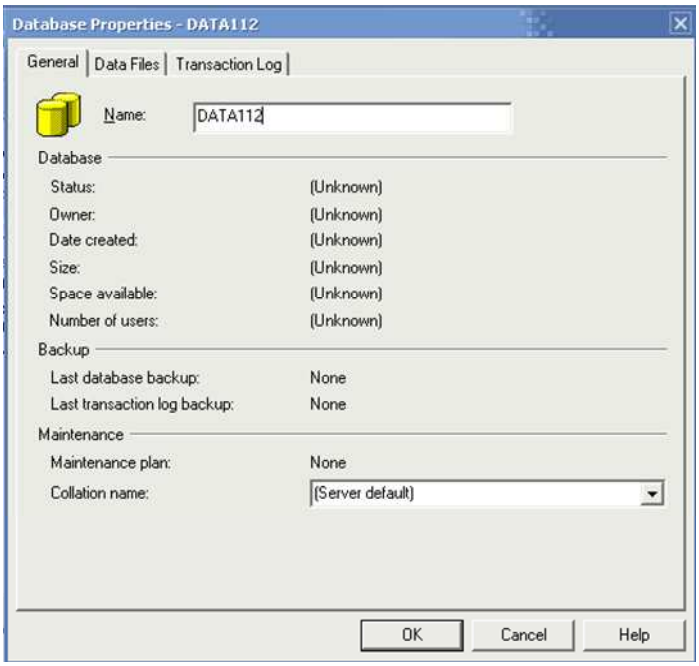
Tạo cơ sở dữ liệu

Sau khi đăng ký, chúng ta có thể thực hiện tạo lập cơ sở dữ liệu và các nội dung cho cơ sở dữ liệu. Nhấn chuột phải vào mục **Databases**, chọn mục **New Database**.



Hình 4.3.2.10 Lựa chọn tạo mới một cơ sở dữ liệu

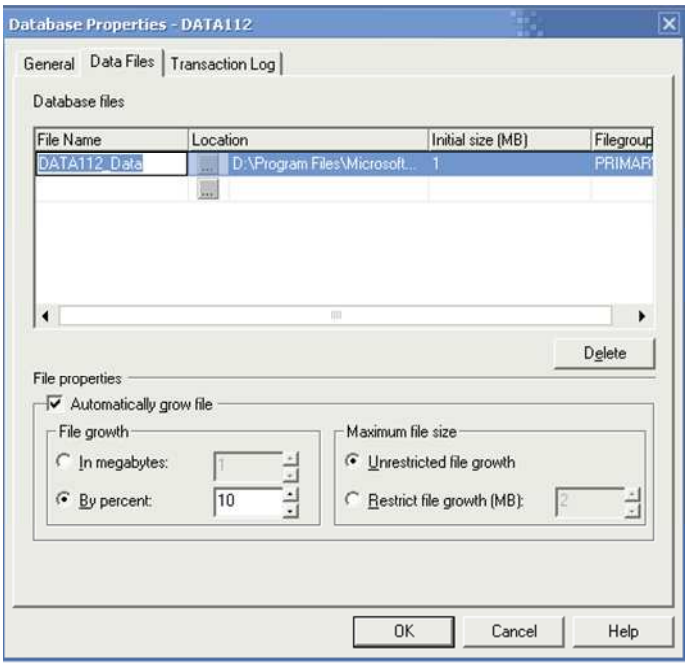
Enterprise Manager sẽ đưa ra màn hình mới để có thể tạo cơ sở dữ liệu.



Hình 4.3.2.11 Đặt tên cho một CSDL

Nhập tên cơ sở dữ liệu vào mục **Name**, đây sẽ là tên chính thức sử dụng để truy nhập cơ sở dữ liệu này. Mục **Collation name** cho phép chúng ta đặt thuộc tính của cơ sở dữ liệu, quy định cơ sở dữ liệu sẽ sử dụng cách thức sắp xếp theo bảng ký tự nào.

Ở tab **Data Files**, người sử dụng có thể thiết lập các thông số khác của cơ sở dữ liệu: tên file chứa dữ liệu cùng đường dẫn đến file đó như trong hình sau:



Hình 4.3.2.12 Xác định đường dẫn và tên tệp của cơ sở dữ liệu

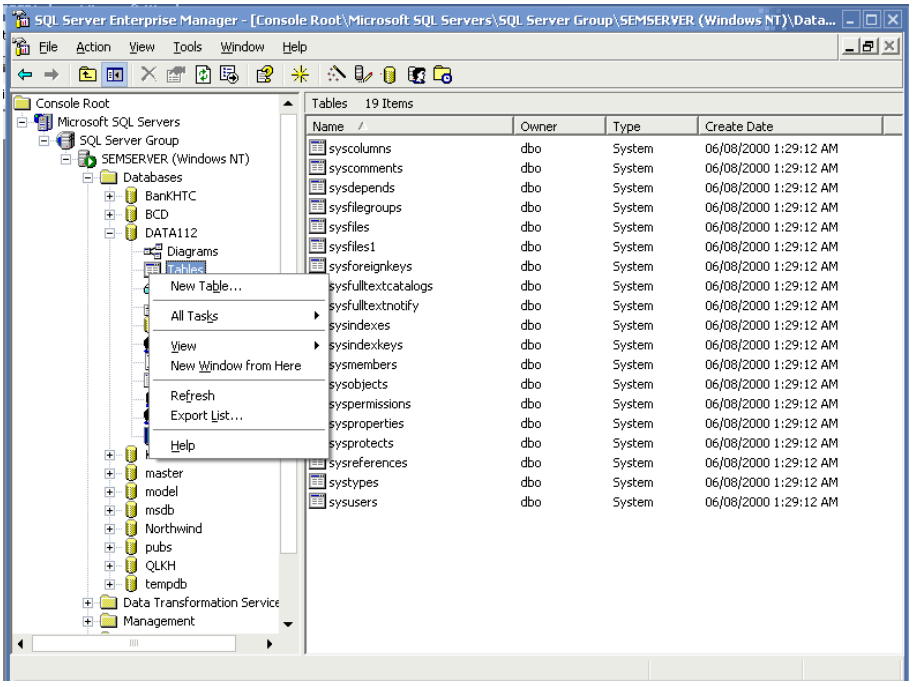
Tại đây, chúng ta có thể chấp nhận các thông số mặc định của SQL Server khuyến nghị sử dụng. Trong trường hợp chúng ta muốn thay đổi, do lượng đĩa cứng chỉ có hạn, do kích thước dữ liệu nhỏ, có thể sửa đổi các thông số này cho phù hợp với nhu cầu.

Nhấn **OK** để tạo cơ sở dữ liệu và kết thúc thao tác này. Kiểm tra lại trong cửa sổ Enterprise Manager cơ sở dữ liệu vừa tạo. Nếu chúng ta thấy cơ sở dữ liệu này hiển thị trong mục **Databases** có nghĩa là thao tác tạo cơ sở dữ liệu đã thành công.

Tạo các bảng dữ liệu và các chỉ số, khóa chính:

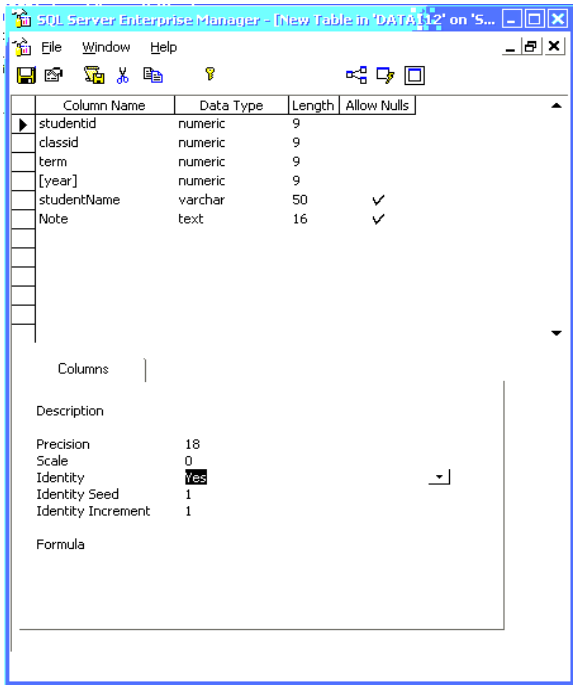
Tạo bảng dữ liệu

Sau khi tạo xong cơ sở dữ liệu, đặt tên là DATA112, chúng ta có thể thực hiện tạo các bảng dữ liệu cho cơ sở dữ liệu này. Trên khung bên trái, nhấn vào dấu + bên cạnh cơ sở dữ liệu chúng ta vừa tạo. Chọn mục **Tables** bằng cách nhấn chuột phải vào mục đó, sau đó chọn **New Table**… để tạo một bảng dữ liệu mới.



Hình 4.3.2.13 Các bảng dữ liệu ngầm định

Chương trình sẽ đưa ra cửa sổ để chúng ta có thể thiết kế bảng. Giả sử chúng ta tạo một bảng sinh viên như sau:

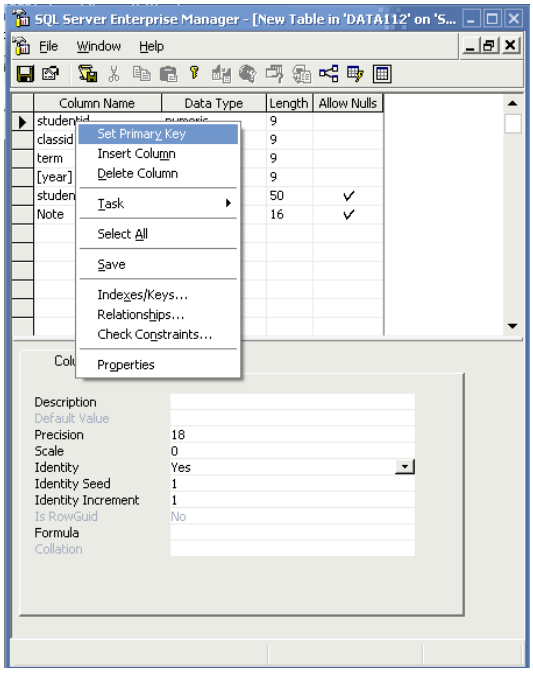


Hình 4.3.2.14 Thiết kế bảng dữ liệu

Nhập tên trường dữ liệu vào cột **Column Name**, kiểu dữ liệu vào cột **Data Type**, và các thông số khác như độ dài vào cột **Length**, cột **Allow Nulls** cho phép xác định cột dữ liệu có được phép để trống hay không khi nhập dữ liệu.

Tạo khóa chính

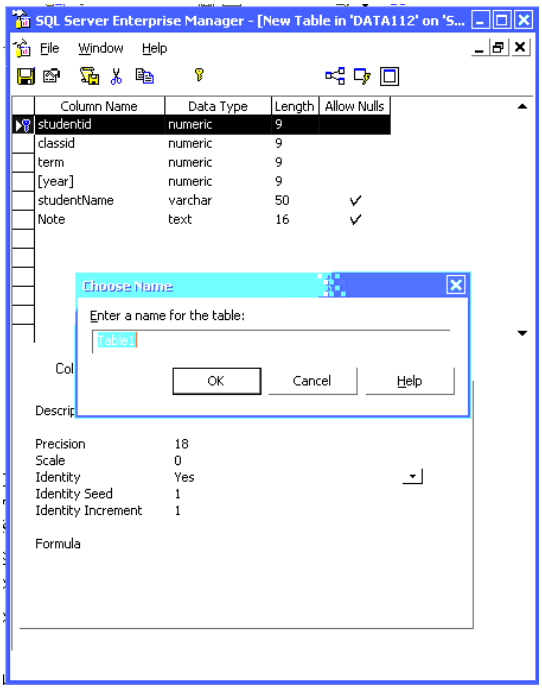
Sau khi đã nhập đủ các trường cho bảng sinh viên, có thể thiết lập khóa chính cho bảng luôn. Chọn trường cần thiết lập làm khóa chính bằng cách nhấn chuột phải vào trường đó và chọn **Set Primary Key**, như trong hình sau:



Hình 4.3.2.15 Thiết lập khóa chính cho bảng

Trong trường hợp muốn tạo khóa chính là nhiều trường, giữ phím **Control** và nhấn chuột vào vùng phía bên trái tên các trường. Sau đó cũng nhấn chuột phải và chọn **Set Primary Key**.

Sau khi chọn xong trường khóa, nhấn nút lệnh **Save** trên thanh công cụ chuẩn của SQL Server để lưu lại.

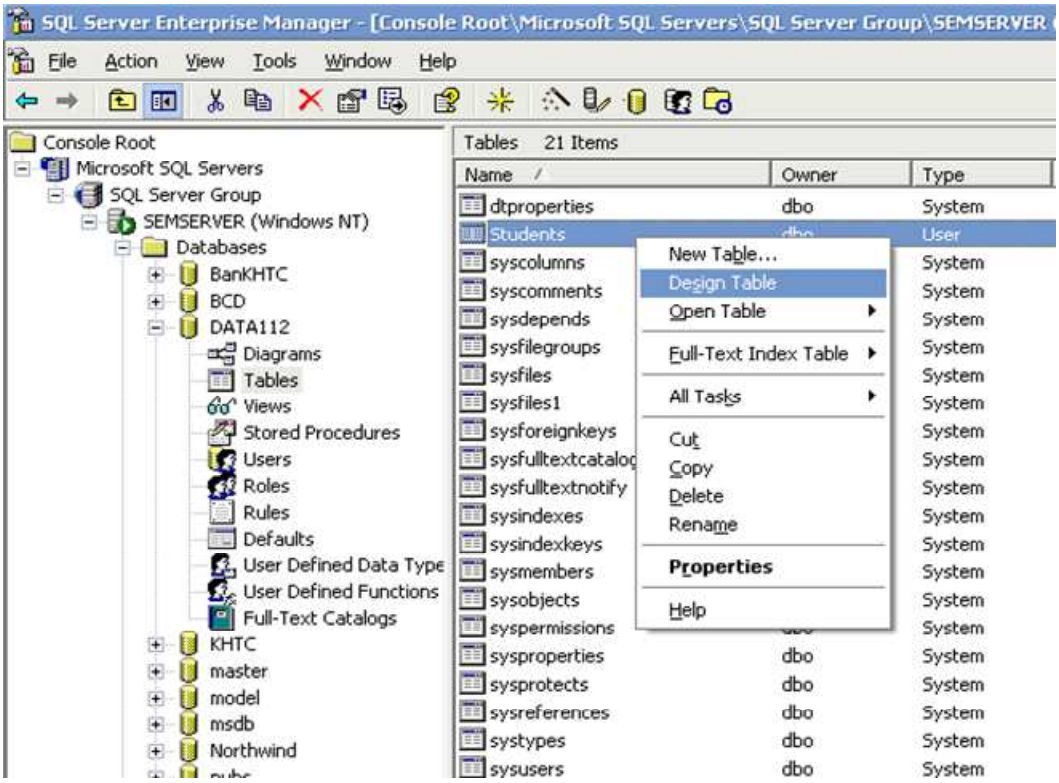


Hình 4.3.2.16 **Nhập tên cho bảng dữ liệu**

Sau đó, nhập tên cho bảng dữ liệu mới là Student, nhấn OK và đóng cửa sổ thiết kế bảng lại và quay trở lại Enterprise Manager.

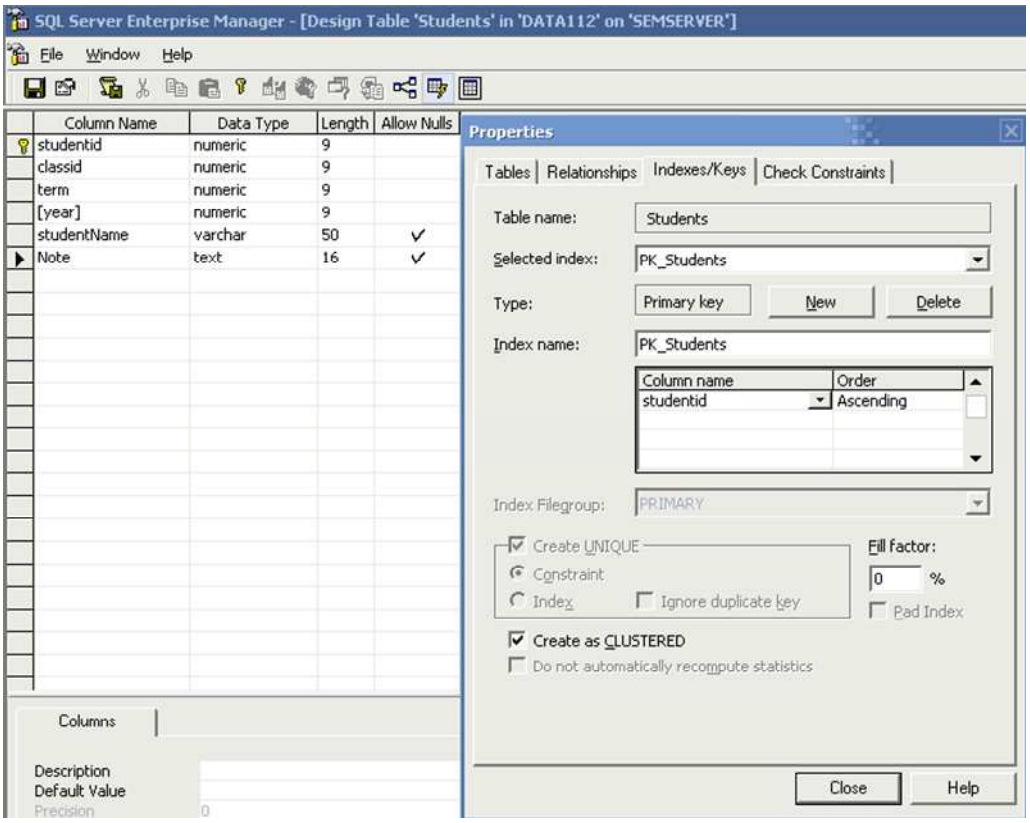
Tạo các chỉ mục

Sau khi tạo bảng và khóa chính, chúng ta tiếp tục thực hiện tạo các chỉ mục cho bảng dữ liệu này. Trong cửa sổ bên trái, tìm cơ sở dữ liệu chúng ta vừa tạo và vào mục **Tables**. Bảng **Students** mà chúng ta vừa tạo sẽ được hiển thị. Nhấn chuột phải vào bảng và chọn: **Design Table**



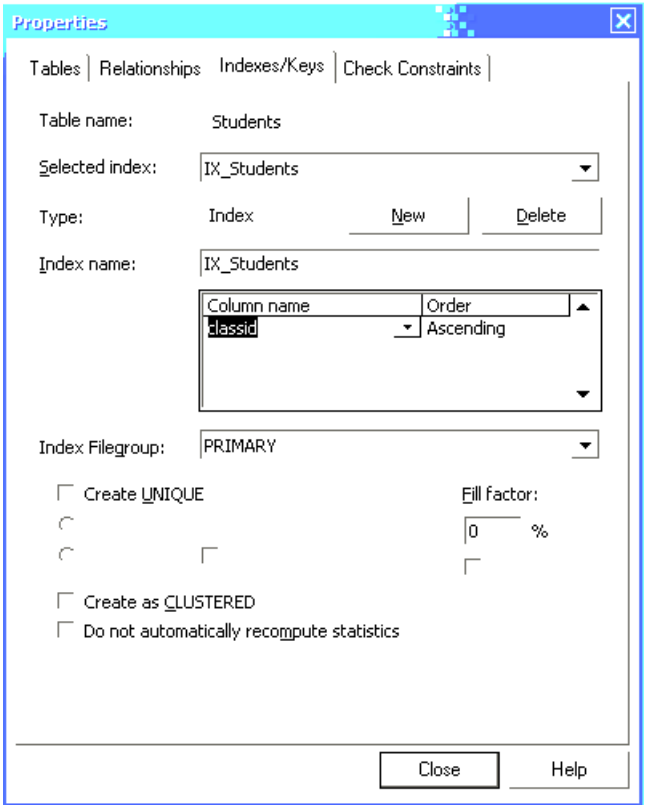
Hình 4.3.2.17 Chọn Design Table

SQL Server sẽ mở màn hình thiết kế bảng một lần nữa. Tại đây, chọn nút lệnh **Manage Indexes / Keys** trên thanh công cụ chuẩn của Enterprise Manager.



Hình 4.3.2.18 Chọn nút Manage Indexes / Keys

SQL Server sẽ mở cửa sổ mới để chúng ta thực hiện tạo, cập nhật các chỉ mục và khóa chính. Chọn nút lệnh **New** trên màn hình này:

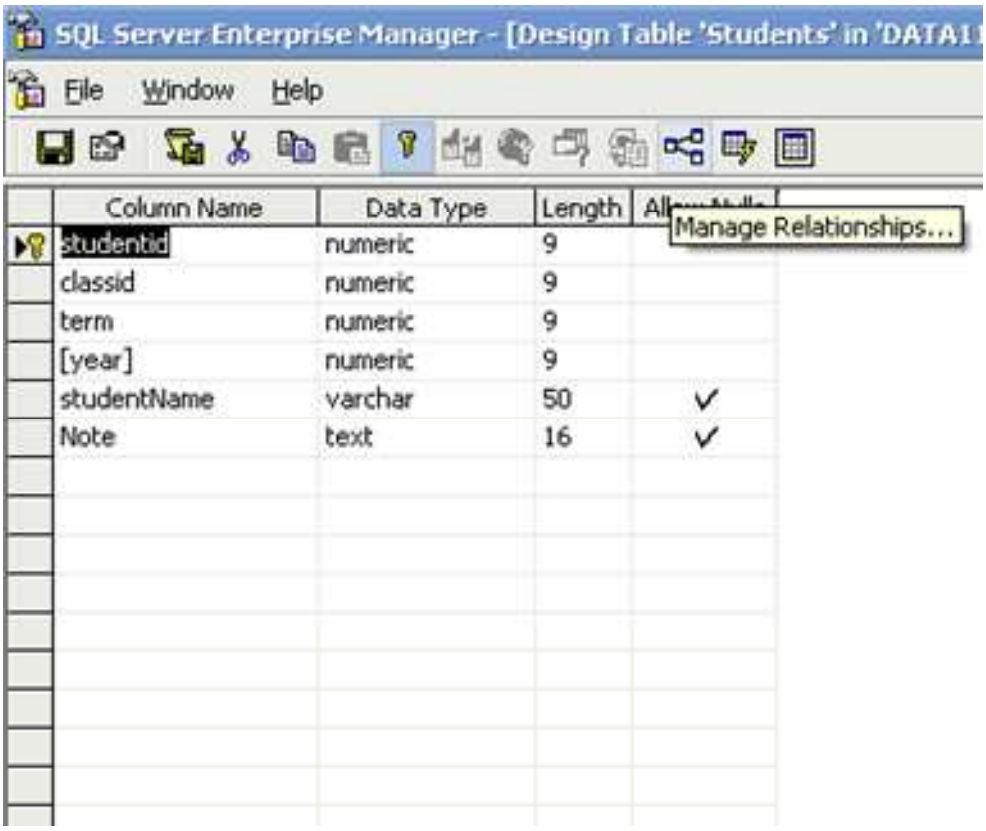


Hình 4.3.2.19 Tạo mới chỉ mục

Trong mục **Index name,** gõ tên của chỉ mục mới, sau đó, chọn cột dữ liệu mà chúng ta muốn thực hiện tạo chỉ mục trong cột **Column name**. Chọn cách thức sắp xếp của chỉ mục này bằng cách chọn trong danh sách thả xuống tại cột **Order**. Sau khi tạo xong, nhấn **Close** để đóng cửa sổ này lại.

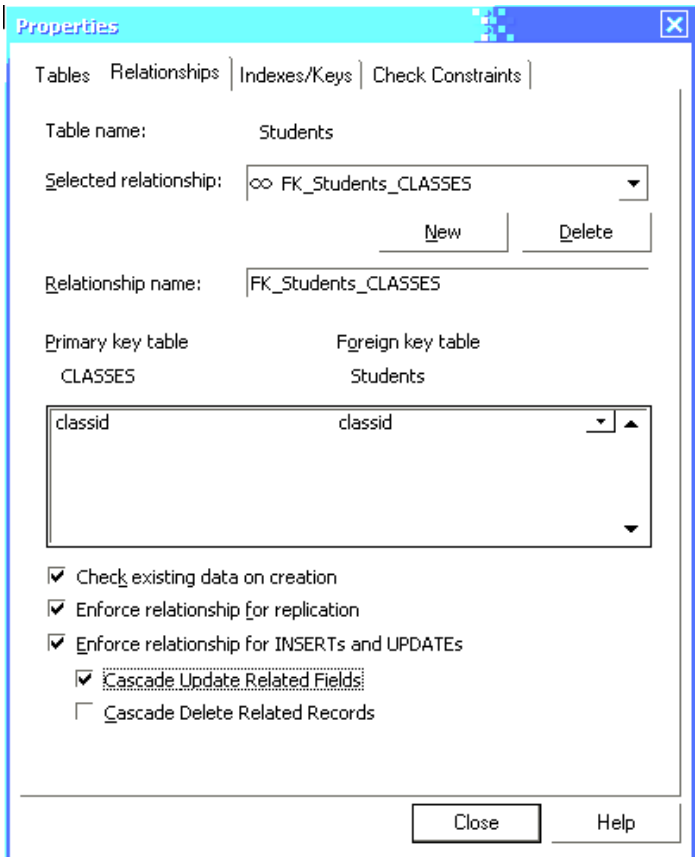
Tạo các liên kết giữa các bảng dữ liệu

Giả sử chúng ta muốn tạo liên kết giữa bảng STUDENTS với bảng có sẵn CLASSES, chúng ta chọn bảng **Students**, nhấn chuột phải và chọn Design Table.



Hình 4.3.2.20 **Nút lệnh Manage Relationships**

Sau đó, chọn nút lệnh **Manage Relationships** trên thanh công cụ chuẩn. Enterprise Manager sẽ đưa ra màn hình để quản lý, cập nhật các liên kết trên bảng mà chúng ta vừa chọn như sau.



Hình 4.3.2.21 Tạo quan hệ giữa các bảng

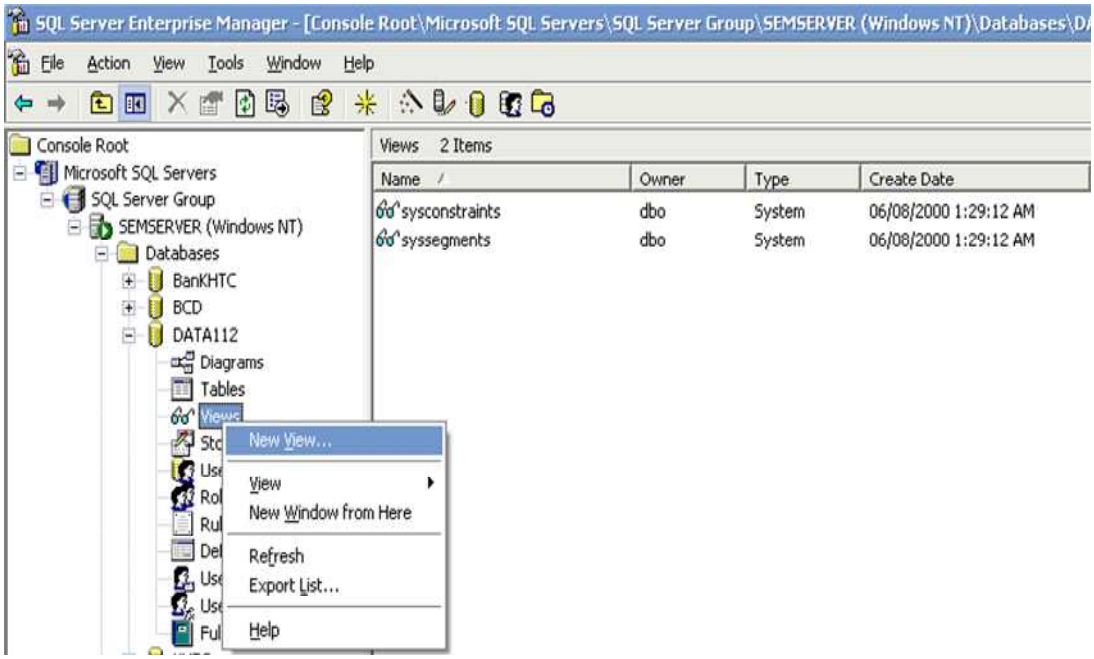
Nhấn nút lệnh **New** để tạo liên kết mới. Ở mục **Relationship Name** nhập tên của liên kết, sau đó, xác định bảng, trường ở bảng chính và xác định trường ở bảng phụ tạo ra liên kết. Các mục phía dưới cho phép chúng ta thực hiện các kiểm tra các dữ liệu ở hai bảng liên quan đến liên kết.

Tạo các khung nhìn (View)

Khung nhìn (View) thực chất là kết quả của việc lấy dữ liệu của một bảng hoặc kết nối các bảng dữ liệu với nhau. Thông thường, để tăng tốc xử lý, chúng ta có thể tạo ra các khung nhìn là kết quả của việc lấy dữ liệu trên một hoặc

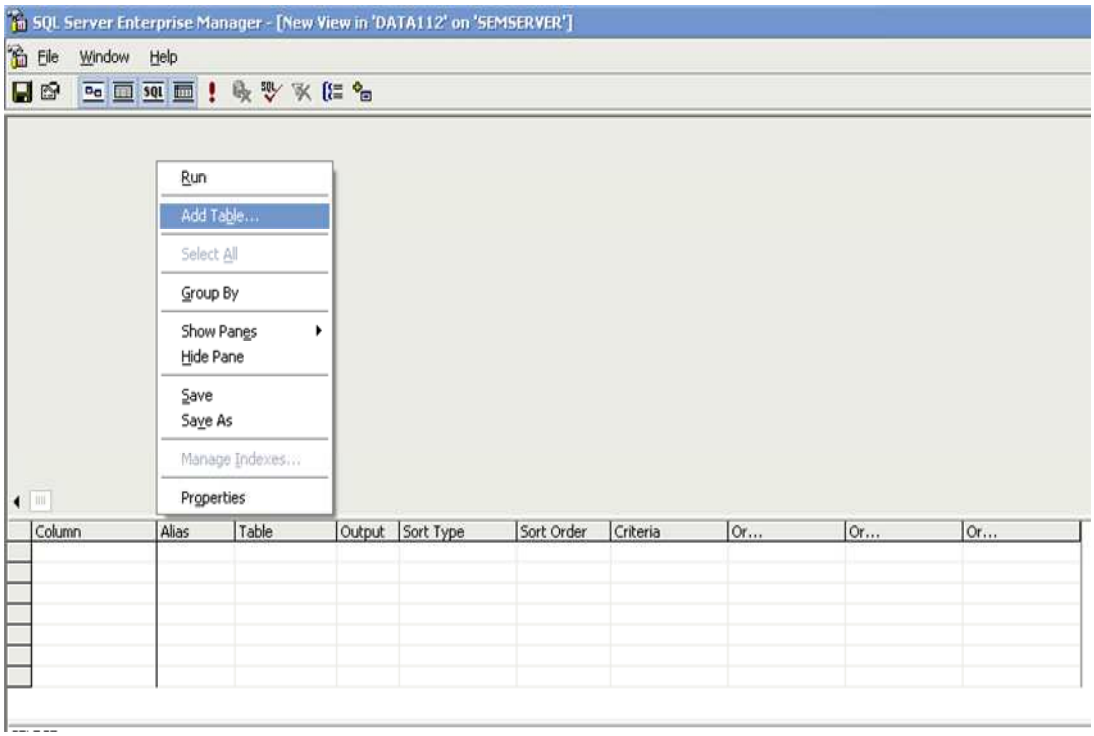
nhiều bảng trước khi thực hiện các thao tác như: hiển thị báo cáo, thiết kế các báo cáo, …

Để tạo một khung nhìn mới, trong danh sách các mục con của cơ sở dữ liệu, nhấn chuột phải vào **Views** và chọn **New View…**



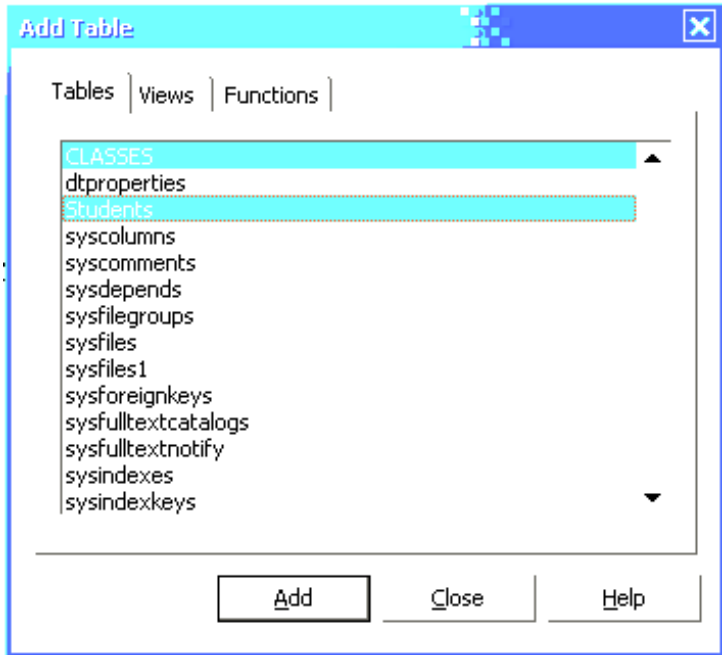
Hình 4.3.2.22 Tạo một khung nhìn mới

Bộ thiết kế khung nhìn sẽ được kích hoạt trợ giúp việc tạo mới. Để thực hiện tạo một khung nhìn, nhấn chuột phải vào vùng trống phía trên và chọn **Add Table…**



Hình 4.3.2.23 Thao tác tạo mới một khung nhìn

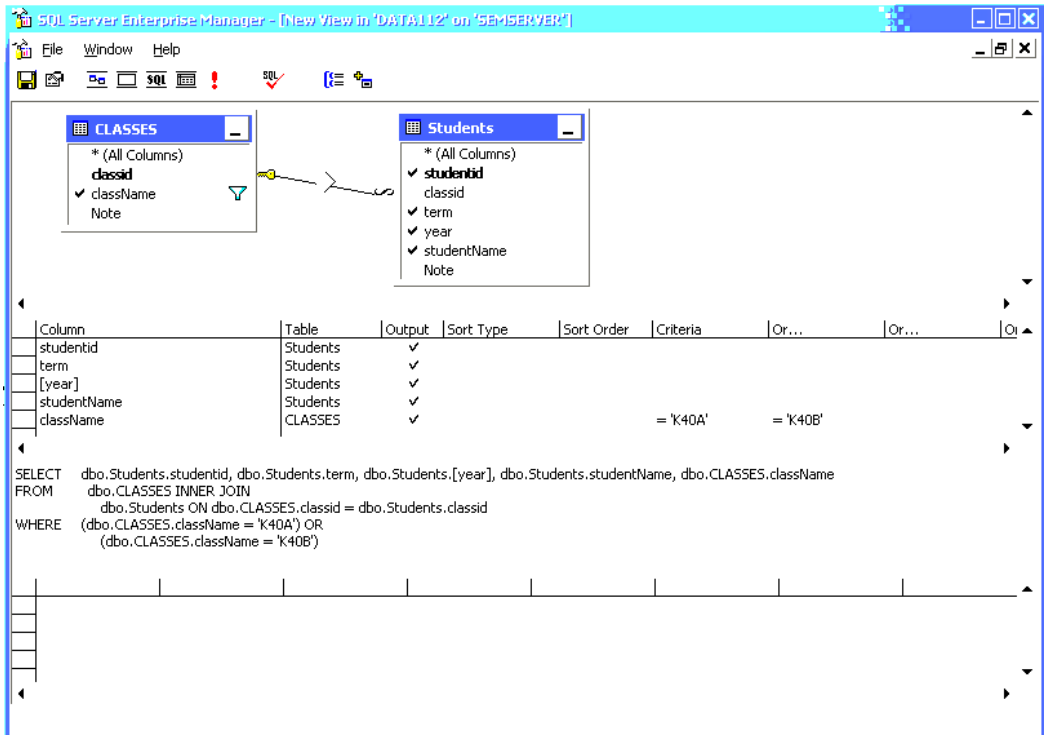
Chương trình sẽ hiển thị danh sách các bảng dữ liệu để chúng ta có thể chọn theo mục tiêu của chúng ta. Kích chuột vào tên bảng cần chọn, nếu muốn chọn nhiều bảng một lúc có thể giữ phím **Ctrl** và kích chuột vào các tên bảng.



Hình 4.3.2.24 Chọn các bảng thành phần tạo khung nhìn

Sau đó nhấn **Add** để chọn các bảng này. Sau khi chọn xong, nhấn **Close** để tiếp tục. Các bảng đã chọn sẽ hiển thị trong phần phía trên cửa sổ thiết kế khung nhìn. ở đây liên kết của các bảng cũng sẽ được hiển thị.

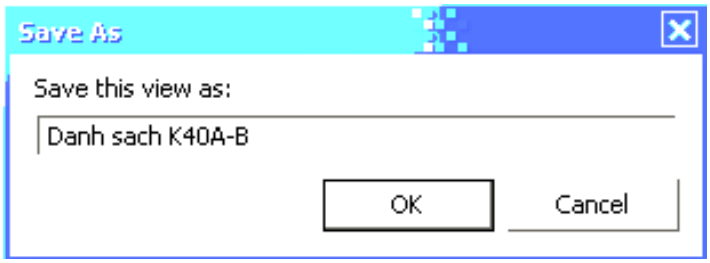
Ví dụ, chúng ta muốn lấy danh sách các sinh viên của lớp K40A và K40B, các bước cần thực hiện như trong hình sau:



Hình 4.3.2.25 Thiết kế khung nhìn

Như trong hình 4.3.2.25 trên, chúng ta cần thực hiện việc xác định các trường dữ liệu cần đưa ra khung nhìn bằng cách check vào các ô bên trái của các cột hoặc chọn trong danh sách (mục **Column**) trong phần cửa sổ thứ 2. Phần cửa sổ thứ3 hiển thị câu lệnh SQL thực tế SQL Server sẽ phải thực hiện để lấy dữ liệu. Nhấn nút lệnh ! trên thanh công cụ để xem trước kết quả.

Sau khi đã đồng ý với kết quả, chúng ta nhấn **Save** và nhập tên của khung nhìn mới.

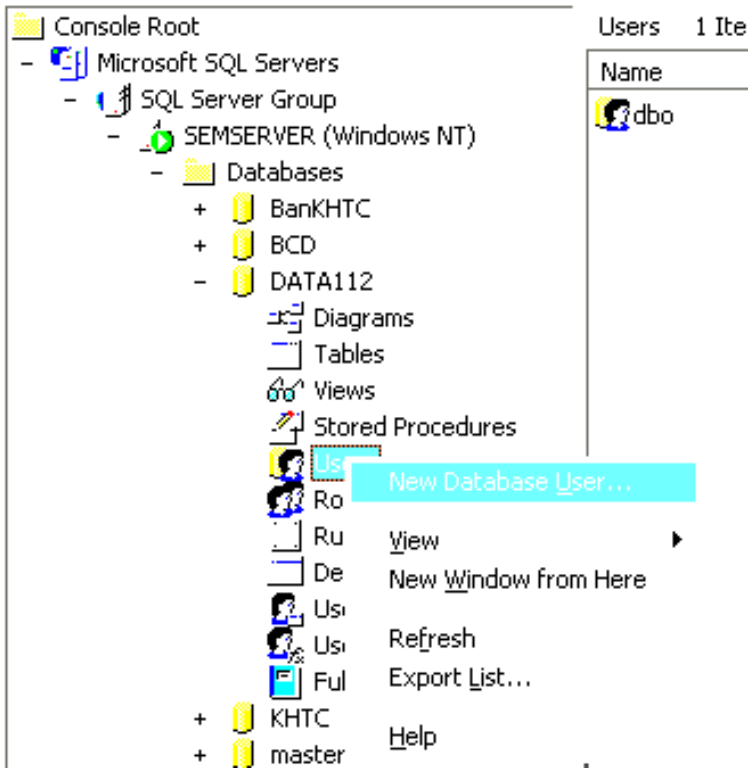


Hình 4.3.2.26 Nhập tên khung nhìn mới

Chúng ta có thể xem lại kết quả hoặc thiết kế lại các khung nhìn này vào bất kỳ thời điểm nào khi cần trong mục **Views** của cơ sở dữ liệu.

Phân quyền sử dụng các đối tượng dữ liệu cho người sử dụng

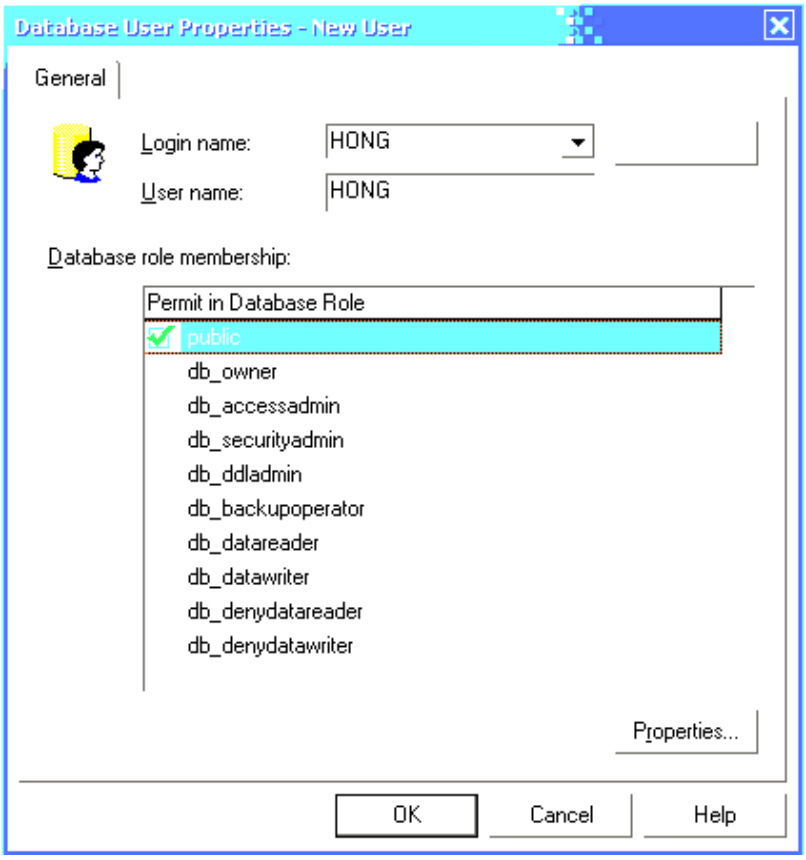
Sau khi tạo xong cơ sở dữ liệu, có thể thực hiện phân quyền sử dụng cho các người sử dụng khác trong hệ thống. Nhấn chuột phải vào mục **Users** và chọn **New User**.



Hình 4.3.2.27 Phân quyền mới cho người sử dụng khác

Tại màn hình kế tiếp, thực hiện chọn người sử dụng và các quyền hạn mà người sử dụng đó có thể thực hiện trên cơ sở dữ liệu. Cần chú ý rằng, danh sách người sử dụng là do người quản trị hệ thống tạo ra, và chúng ta không có quyền

thì không tạo được người sử dụng mới. Nếu có nhu cầu, hãy liên hệ với quản trị hệ thống cơ sở dữ liệu để tạo người sử dụng này. Màn hình để thực hiện việc này như hình 4.3.2.28 sau:



Hình 4.3.2.28 Gán quyền sử dụng cơ sở dữ liệu cho người sử dụng khác

Chọn người sử dụng trong mục **Login Name** và gán các quyền cho người sử dụng này tại mục **Database role membership**. Sau khi chọn xong, nhấn **OK** để kết thúc. Người sử dụng được gán quyền sẽ có thể truy cập và thực hiện các thao tác mà người tạo cơ sở dữ liệu (db_owner: chủ cơ sở dữ liệu) đã gán cho.

b. Hệ quản trị cơ sở dữ liệu Oracle

Một hệ quản trị cơ sở dữ liệu có thể nói là lớn nhất, chặt chẽ nhất và mạnh nhất trên thị trường phần mềm hệ quản

trị cơ sở dữ liệu hiện nay chính là hệ quản trị cơ sở dữ liệu ORACLE, do hãng Oracle cung cấp và phát triển. Hệ này hỗ trợ phát triển các hệ thống cơ sở dữ liệu cũng như các chương trình ứng dụng chạy trên hầu hết các hệ điều hành máy tính hiện nay như Windows, Unix, Linux, …

Trước tiên, chúng ta sẽ đi vào tìm hiểu về cấu trúc một cơ sở dữ liệu Oracle. Về tổng thể, một cơ sở dữ liệu Oracle có một tầng logic và một tầng vật lý. Tầng vật lý bao gồm các file nằm trên đĩa, còn tầng logic ánh xạ dữ liệu tới các thành phần vật lý nêu trên.

Tầng vật lý

Tầng vật lý của một cơ sở dữ liệu bao gồm 3 loại file:

Một hoặc nhiều file chứa dữ liệu (datafiles) chứa các thông tin được lưu trữ trong cơ sở dữ liệu. Chúng ta có thể có từ một đến hàng trăm file dữ liệu. Thông tin trong một bảng dữ liệu có thể được mở rộng và chứa trong nhiều file dữ liệu và tương tự, nhiều bảng dữ liệu có thể chứa trong cùng một file. Việc thực hiện trải các không gian bảng (tablespaces) trên nhiều file dữ liệu có thể mang lại các ảnh hưởng tích cực đến hiệu năng của toàn bộ hệ thống. Số file dữ liệu có thể thực hiện cấu hình được quy định bằng tham số MAXDATAFILES.

Hai hay nhiều các file redo (redo log files) chứa các thông tin sử dụng cho việc phục hồi trong trường hợp hệ thống hỏng hóc. Các file này lưu vết của tất cả các thay đổi được thực hiện trên cơ sở dữ liệu. Các vết này được sử dụng để phục hồi lại các thông tin thay đổi và lưu lại mà chưa được ghi vào các file dữ liệu. Vì vậy, nếu mất các file này, hệ

thống sẽ không thể được phục hồi nếu như hệ thống cơ sở dữ liệu bị hỏng.

Một hay nhiều các tệp điều khiển chứa các thông tin sử dụng cho việc khởi động một thể hiện (instance), ví dụ như: vị trí lưu các file dữ liệu và các file redo; Oracle cần các thông tin này để khởi động thể hiện của cơ sở dữ liệu. Các file điều khiển (Control files) cần phải được bảo vệ, do vậy Oracle cung cấp giải pháp là lưu nhiều bản sao của các file điều khiển này.

Tầng Logic

Tầng logic bao gồm các thành phần sau:

Một hoặc nhiều các không gian bảng (tablespaces).

Các lược đồ cơ sở dữ liệu (database schema), chứa các mục như: các bảng (tables), các phân đoạn (clusters), các chỉ mục (indexes), các khung nhìn (views), các thủ tục (stored procedures), các thủ tục kích hoạt cho cơ sở dữ liệu (database triggers), các dãy tự tăng (sequences), ...

Sau đây, chúng ta sẽ đi sâu nghiên cứu về các khái niệm, tác dụng và chức năng của các thành phần trong tầng logic, từ đó có được cái nhìn tổng thể và bao quát hơn về cơ sở dữ liệu Oracle.

Tablespaces và Datafiles

Cơ sở dữ liệu được chia thành một hoặc nhiều các phần logic, các phần này chính là các không gian bảng (tablespaces). Một tablespace được sử dụng để nhóm các dữ liệu vào với nhau một cách logic. Ví dụ: chúng ta có thể tạo một tablespace riêng cho tác nghiệp và một tablespace cho điều hành. Việc phân chia này sẽ đơn giản hóa việc

quản trị các nhóm này. Mỗi tablespace chứa một hoặc nhiều file dữ liệu (datafiles). Cách sử dụng nhiều file dữ liệu cho phép trải dữ liệu lưu trữ trên nhiều đĩa khác nhau để phân tải các thao tác vào ra dữ liệu có thể giúp tăng hiệu năng của hệ thống.

Trong quá trình tạo mới cơ sở dữ liệu, Oracle tự động tạo tablespace tên là SYSTEM. Mặc dù, đối với một hệ cơ sở dữ liệu nhỏ chúng ta có thể sử dụng ngay tablespace này, tuy nhiên, vẫn nên tạo thêm một tablespace riêng cho các dữ liệu người dùng bởi lẽ, tablespace SYSTEM được sử dụng cho các từ điển dữ liệu bao gồm: các bảng, các chỉ mục, …

Các file dữ liệu (datafiles) có thể là các file thuộc hệ điều hành hoặc trong một số hệ điều hành, nó là các RAW device, thiết bị giả. Vì vậy, việc truy nhập vào các file dữ liệu cũng phụ thuộc vào các hệ điều hành cụ thể.

Lược đồ cơ sở dữ liệu (Database Schema)

Lược đồ cơ sở dữ liệu là một tập hợp các đối tượng, hay còn gọi là các đối tượng lược đồ, thuộc tầng logic cho chúng ta hình dung một cơ sở dữ liệu bao gồm những dữ liệu kiểu gì. Các đối tượng này gồm các cấu trúc như: các bảng, các phân đoạn, các chỉ mục, các khung nhìn, các thủ tục, các kích hoạt cơ sở dữ liệu, và các dãy tự tăng.

Bảng dữ liệu

Một bảng được định nghĩa bao gồm tên, các hàng và các cột dữ liệu. Bảng là đơn vị lưu trữ dữ liệu logic cơ bản trong cơ sở dữ liệu Oracle. Các cột được định nghĩa với tên và kiểu dữ liệu. Thường thì một bảng được lưu trong một tablespace và có nhiều bảng cùng sử dụng một tablespace.

Một phân cụm – cluster – là một tập các bảng được lưu trữ trên thiết bị vật lý cùng nhau mà chia sẻ một cột dữ liệu chung. Nếu dữ liệu ở 2 hoặc nhiều bảng thường được lấy ra cùng nhau dựa trên cột dùng chung, khi đó, sử dụng các phân cụm là khá hiệu quả. Các bảng có thể được truy nhập riêng rẽ mặc dù chúng thuộc một bảng đã được phân cụm. Do cấu trúc như vậy nên các dữ liệu có liên quan sẽ yêu cầu ít hơn thời gian thực hiện các thao tác vào/ra nếu được truy cập đồng thời.

Chỉ mục (Index) – Một chỉ mục được tạo lập nhằm tăng cường khả năng lấy dữ liệu trên các bảng. Chỉ mục được định nghĩa dựa trên một hoặc một tập các cột. Khi chúng ta thực hiện các truy vấn dựa trên giá trị của cột đã được đánh chỉ mục (dùng trong mệnh đề WHERE của SQL) thì chỉ mục sẽ được sử dụng để xác định dữ liệu trong bảng.

Khung nhìn (View) – Trong Oracle thì khung nhìn là cửa sổ nhìn vào trong dữ liệu của một hoặc nhiều bảng. Một khung nhìn có thể truy cập và thực hiện truy vấn, cập nhật và xóa dữ liệu như một bảng mà không có sự ràng buộc nào. Thực chất, các khung nhìn được sử dụng như một cách làm đơn giản hóa nhận thức của người sử dụng về việc truy cập dữ liệu bằng cách cung cấp các thông tin giới hạn từ một bảng hay một tập các thông tin từ vài bảng. Khung nhìn cũng có thể được sử dụng để ngăn chặn các dữ liệu mà chúng ta không muốn người sử dụng truy cập hay tạo các liên kết giữa nhiều bảng dữ liệu.

Các thủ tục (Stored procedure)—Có thể hiểu các stored procedure như một câu truy vấn được định nghĩa trước lưu trong từ điển dữ liệu. Chúng được thiết kế nhằm tăng cường khả năng truy vấn dữ liệu do giảm được lượng thông

tin phải trao đổi giữa các máy khách và hệ quản trị cơ sở dữ liệu.

Kích hoạt cơ sở dữ liệu (Database trigger)—Đây là một dạng thủ tục được kích hoạt tự động khi một sự kiện nào đó xảy ra, chẳng hạn như: thêm mới, cập nhật hay thậm chí là thực hiện truy vấn dữ liệu trên một bảng nào đó.

Dãy tự tăng (Sequence)—Bộ tạo dãy tự tăng, Oracle sequence generator, được sử dụng để tự động tạo một dãy số tự tăng trong bộ đệm, giúp tránh việc tạo bằng tay rất tốn thời gian: khóa bảng – lấy giá trị cuối dãy – tạo giá trị mới – bỏ khóa bảng.

Phân đoạn, phần mở rộng, và các khối dữ liệu (Segments, Extents, and Data Blocks)

Bên trong Oracle, không gian dùng lưu trữ dữ liệu được kiểm soát bằng cách sử dụng các cấu trúc logic bao gồm:

Các khối dữ liệu (Data blocks) - Một khối là đơn vị lưu trữ nhỏ nhất trong một cơ sở dữ liệu Oracle. Các khối này chứa các thông tin trong phần đầu liên quan đến chính nó và dữ liệu chứa trong đó. Nói đúng nhất, các khối này được lưu trực tiếp trên các đĩa cứng nhưng chúng ta hoàn toàn có thể điều chỉnh kích thước các khối dữ liệu nhằm tăng hiệu quả truy cập tùy thuộc vào ứng dụng và hệ điều hành. (Mặc định là 2KB trong phần lớn các hệ thống)

Phần mở rộng (Extents) - chứa các khối dữ liệu. Một phần mở rộng được sử dụng để giảm thiểu các lãng phí lưu trữ vật lý (các vùng dữ liệu rỗng). Khi dữ liệu được nhập ngày càng nhiều vào cơ sở dữ liệu, các phần mở rộng này có thể được tăng trưởng (hoặc thu gọn lại khi cần thiết). Theo cách như vậy, nhiều tablespace có thể chia sẻ cùng một

không gian lưu trữ mà không cần phải xác định trước các phân khu giữa các tablespace này.

Trong quá trình tạo tablespace, chúng ta có thể xác định số tối thiểu các phần mở rộng cần thiết lập cũng như số các phần mở rộng này cần thêm khi trật tự này được thiết lập. Cách sắp xếp này cho phép quản lý hiệu quả không gian sử dụng trong cơ sở dữ liệu.

Phân đoạn (Segments)- Một phân đoạn là một tập của các phần mở rộng sử dụng để lưu một kiểu dữ liệu đặc thù.

Một cơ sở dữ liệu Oracle có thể sử dụng 4 kiểu phân đoạn:

Phân đoạn dữ liệu – Lưu dữ liệu người dùng trong cơ sở dữ liệu.

Phân đoạn chỉ mục - Lưu các chỉ mục.

Phân đoạn quay lui (Rollback segment) – Lưu thông tin sử dụng khi cần phục hồi lại thông tin trước khi thay đổi.

Phân đoạn tạm thời (Temporary segment) – Được tạo ra khi một câu lệnh SQL cần một vùng làm việc tạm thời. Các phân đoạn tạm thời sẽ được hủy bỏ khi câu lệnh SQL đã kết thúc. Các phân đoạn này thường dùng cho các hoạt động của cơ sở dữ liệu, chẳng hạn như: sắp xếp, làm các phép hợp, giao các tập dữ liệu,...

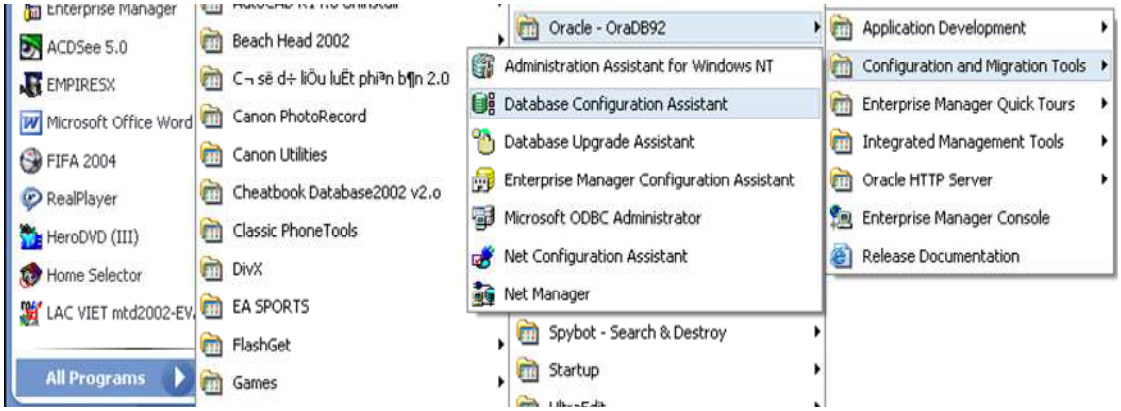
Các thao tác cơ bản

Chúng ta vừa xem xét các khái niệm cơ bản của một cơ sở dữ liệu Oracle. Cũng cần lưu ý rằng, riêng thao tác tạo lập cơ sở dữ liệu mới thường được người quản trị hệ thống thực hiện nên người sử dụng không nên tùy tiện thực hiện mặc dù các thao tác này cũng khá dễ thực hiện. Các thao

tác cơ bản như tạo bảng, tạo chỉ mục…sẽ được bàn chi tiết về cách thức thực hiện dưới đây:

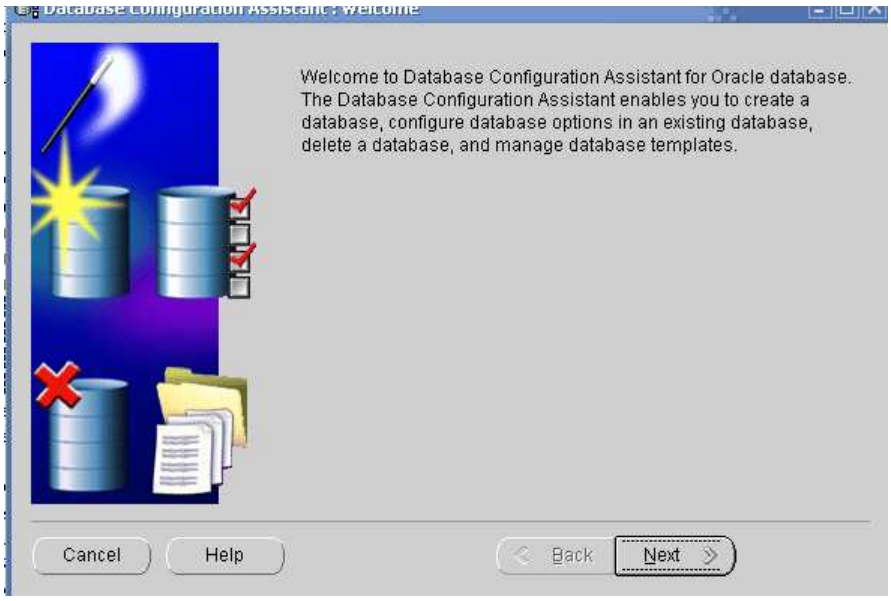
Tạo mới cơ sở dữ liệu

Để bắt đầu tạo mới một cơ sở dữ liệu, chúng ta sử dụng công cụ **Database Configuration Assistant** bán kèm theo bộ **Oracle Database**. Công cụ này cho phép thiết lập, tạo mới cơ sở dữ liệu Oracle. Cần lưu ý rằng, trước khi sử dụng công cụ này, cần đặt Oracle Home về đúng Oracle Home của Database Oracle.



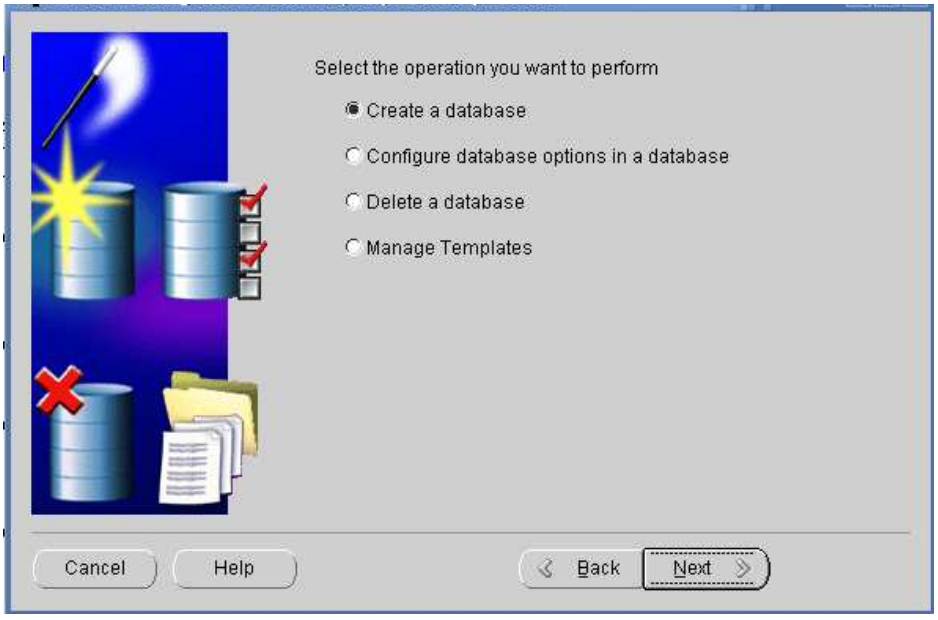
Hình 4.3.2.29 Kích hoạt Database Configuration Assistant

Màn hình bắt đầu của công cụ này như sau:



Hình 4.3.2.30 Màn hình khởi đầu của Database Configuration Assistant

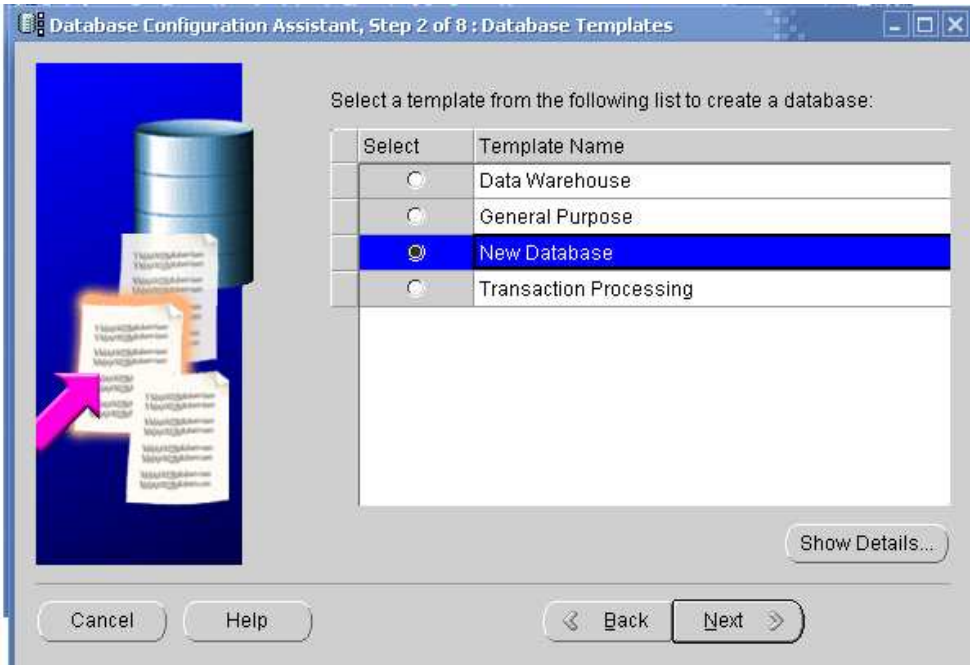
Nhấn **Next** để bắt đầu thực hiện thao tác tạo một cơ sở dữ liệu mới.



Hình 4.3.2.31 Các lựa chọn của công cụ Database Configuration Assistant

Chọn mục **Create Database** và nhấn **Next** để tiếp tục. Màn hình tiếp theo cho phép chọn một mẫu sẵn có để tạo cơ sở

dữ liệu. Chúng ta có thể chọn một trong các mục như trong hình sau:

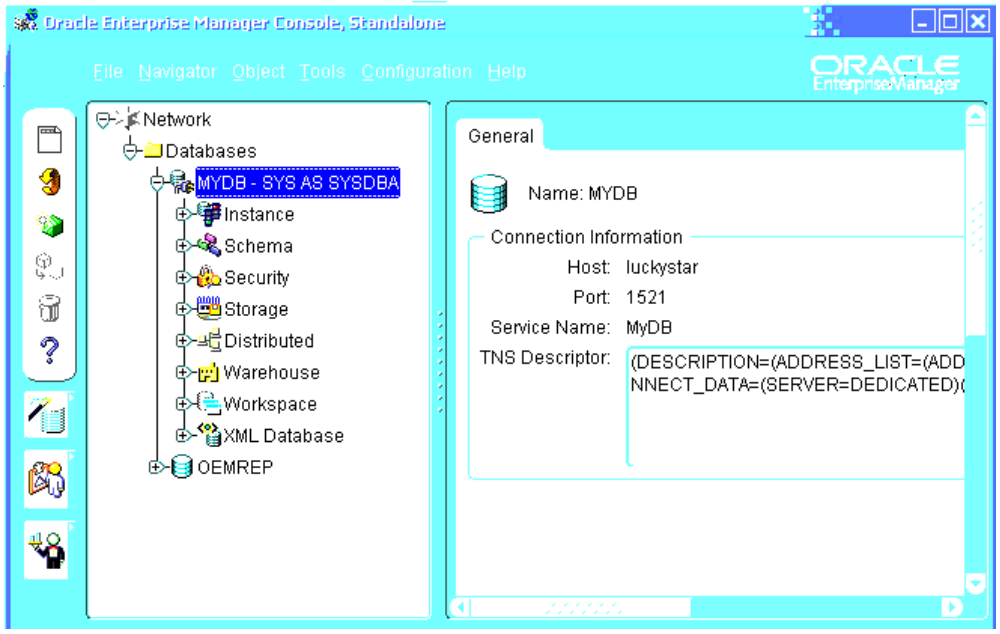


Hình 4.3.2.32 Các mẫu chọn để tạo cơ sở dữ liệu mới

Các bước tiếp theo, làm theo chỉ dẫn của công cụ để tạo cơ sở dữ liệu.

Tạo các bảng

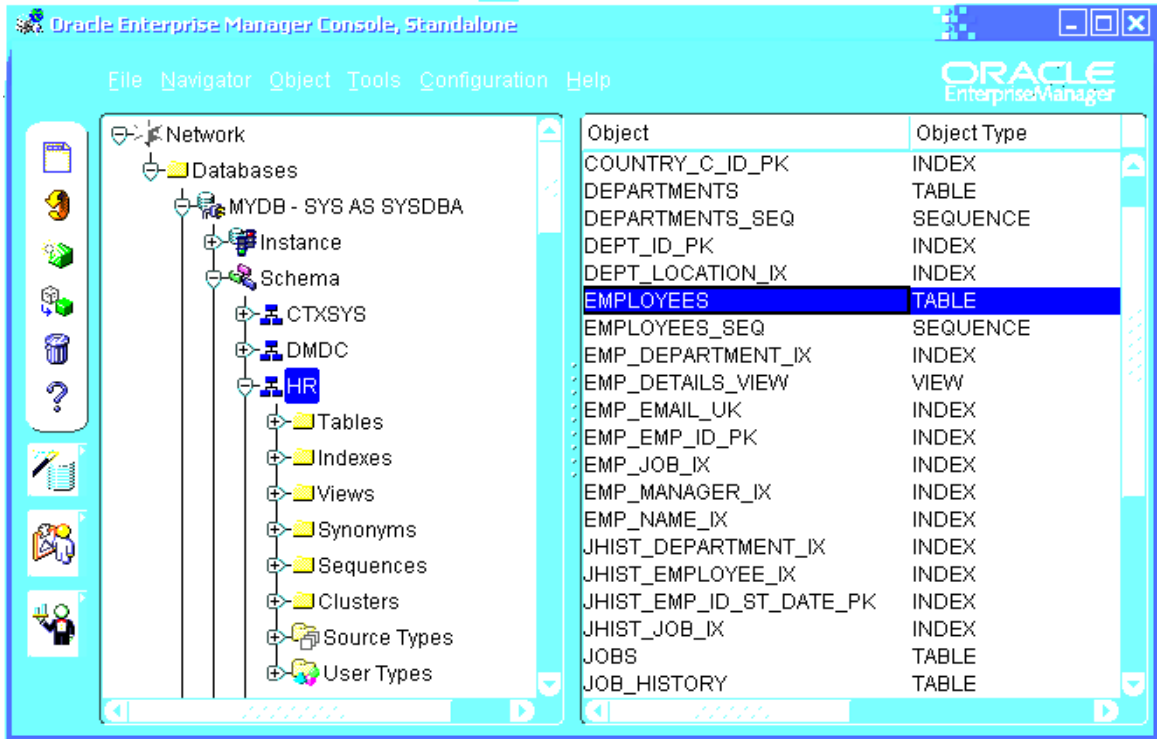
Chúng ta đã làm quen với các khái niệm như: Lược đồ cơ sở dữ liệu, bảng, chỉ mục, ... trong phần trên. Trong mục này chúng ta sẽ tìm hiểu cách thức tạo một bảng dữ liệu trong cơ sở dữ liệu Oracle. Để thực hiện điều này, chúng ta sẽ làm quen với bộ công cụ **Oracle Enterprise Manager**, một công cụ tiện ích giúp chúng ta thực hiện phần lớn các thao tác trên cơ sở dữ liệu.



Hình 4.3.2.33 Công cụ Oracle Enterprise Manager

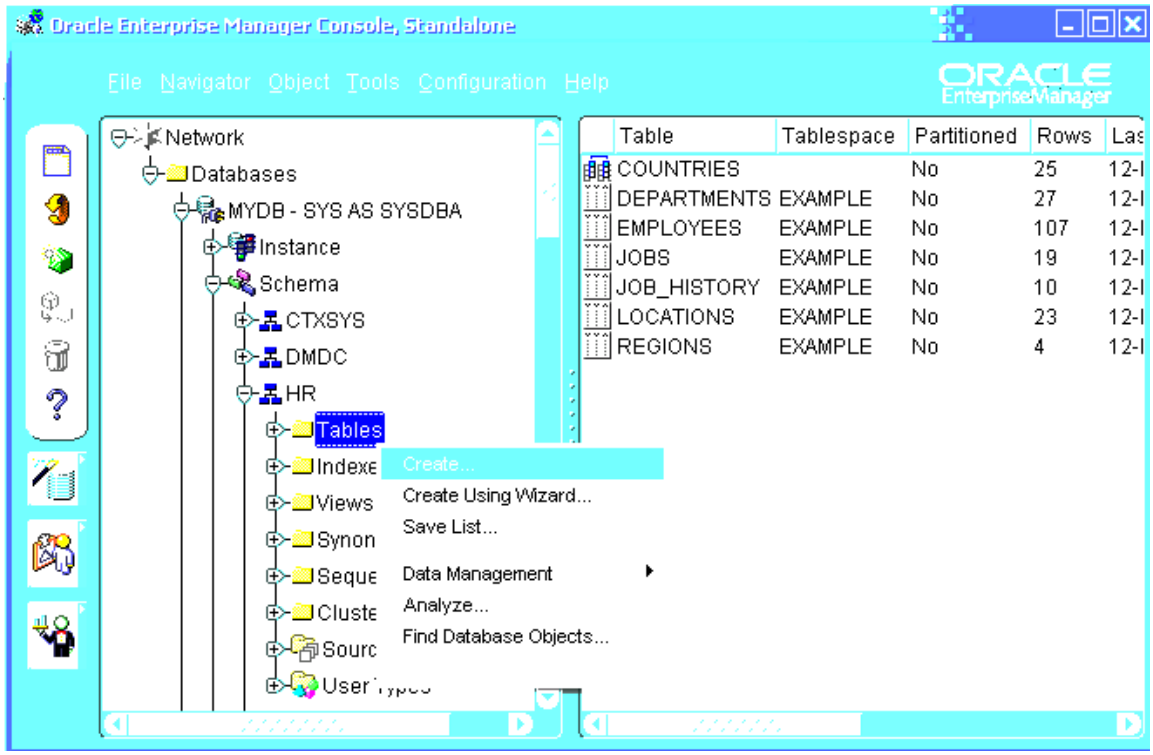
Để có thể thực hiện các thao tác với cơ sở dữ liệu, chúng ta cần thiết lập kết nối đến cơ sở dữ liệu. Hãy đọc thêm các tài liệu của Oracle để biết rõ hơn cách thực hiện thao tác này. Trong màn hình trên, **MYDB** là tên cơ sở dữ liệu (SID), phần sau dấu – là tên người sử dụng và vai trò của người sử dụng này trên cơ sở dữ liệu đã chọn.

Nhấn chuột vào mục **Schema** để xem thông tin của các lược đồ hiện có của cơ sở dữ liệu. Danh sách hiện ra là các lược đồ của cơ sở dữ liệu. Thực chất, các lược đồ này thuộc về từng người sử dụng.



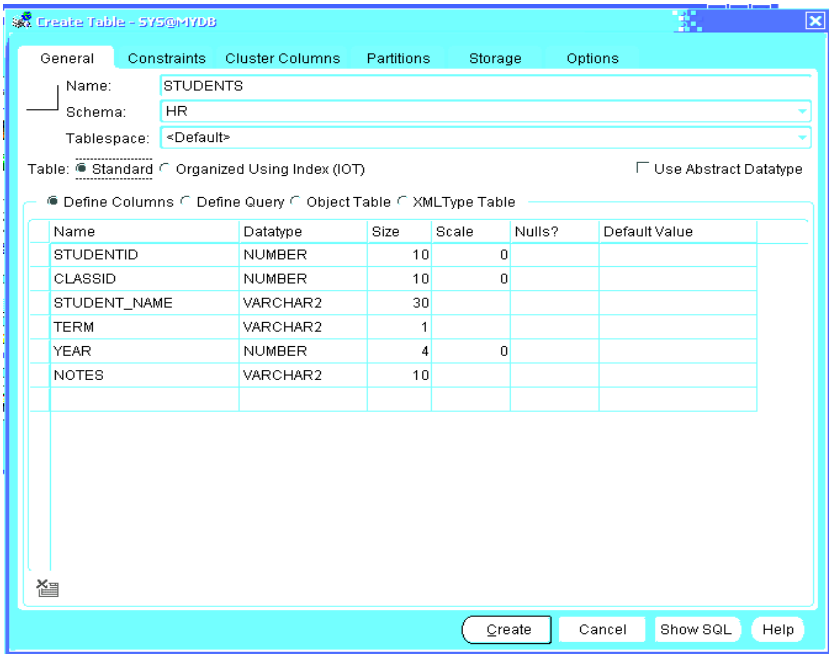
Hình 4.3.2.34 Xem các thành phần của một lược đồ

Để thực hiện tạo bảng, nhấn chuột phải vào mục **Tables** và chọn **Create**



Hình 4.3.2.35 Tạo bảng dữ liệu – bước 1

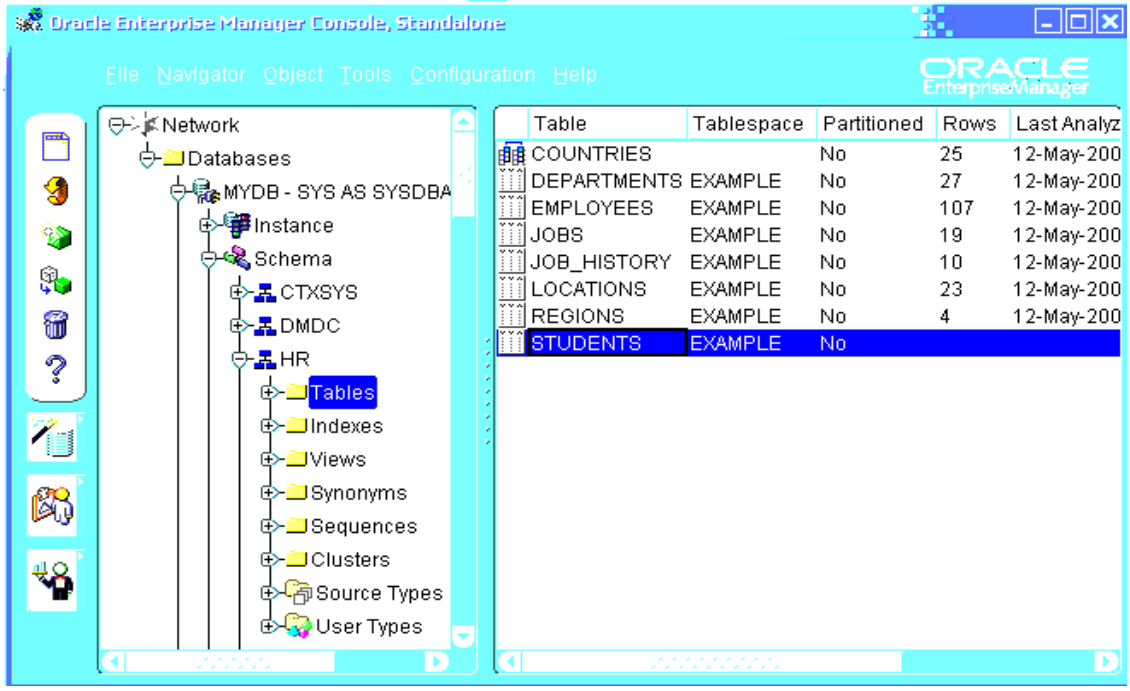
Chúng ta cũng có thể thực hiện tạo các bảng dữ liệu bằng cách sử dụng Wizard như trong cơ sở dữ liệu Access, Visual FoxPro hay SQL Server. Màn hình tiếp theo cho phép thực hiện định nghĩa cấu trúc của bảng.



Hình 4.3.2.36 Tạo bảng dữ liệu – Bước 2

Cũng tương tự như các cơ sở dữ liệu khác, chúng ta cần thực hiện nhập cấu trúc của bảng cần tạo, sau đó nhấn **Create**.

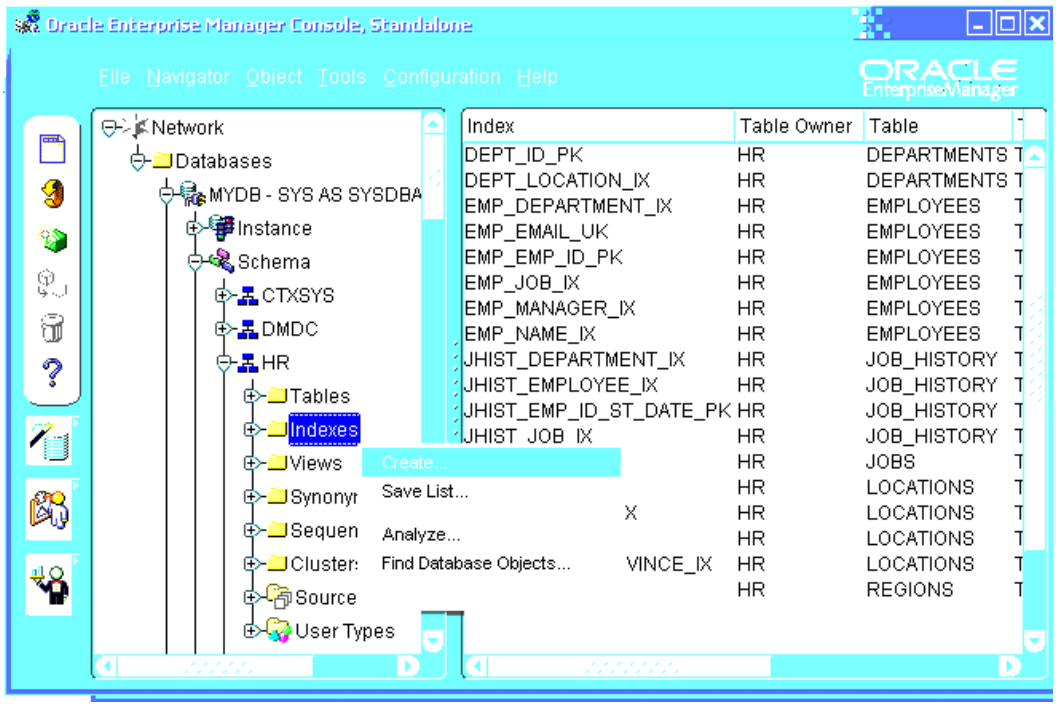
Sau khi tạo xong, chúng ta có thể kiểm tra trong màn hình Oracle Enterprise Manager



Hình 4.3.2.37 Danh sách các bảng dữ liệu

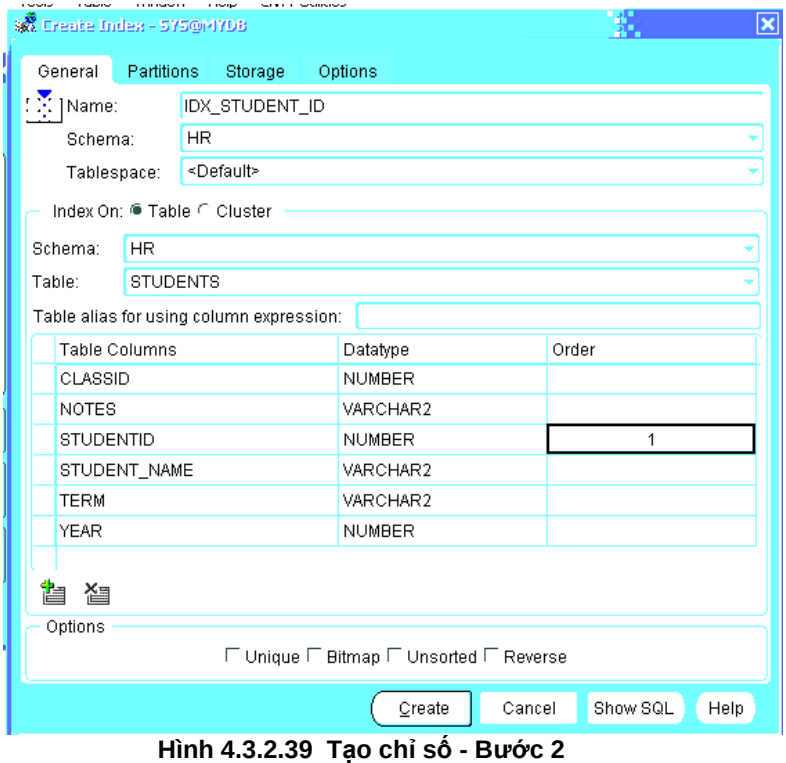
Tạo các chỉ mục

Để thực hiện tạo chỉ mục, nhấn chuột phải vào mục **Indexes** trong lược đồ đã chọn và cũng chọn **Create**.



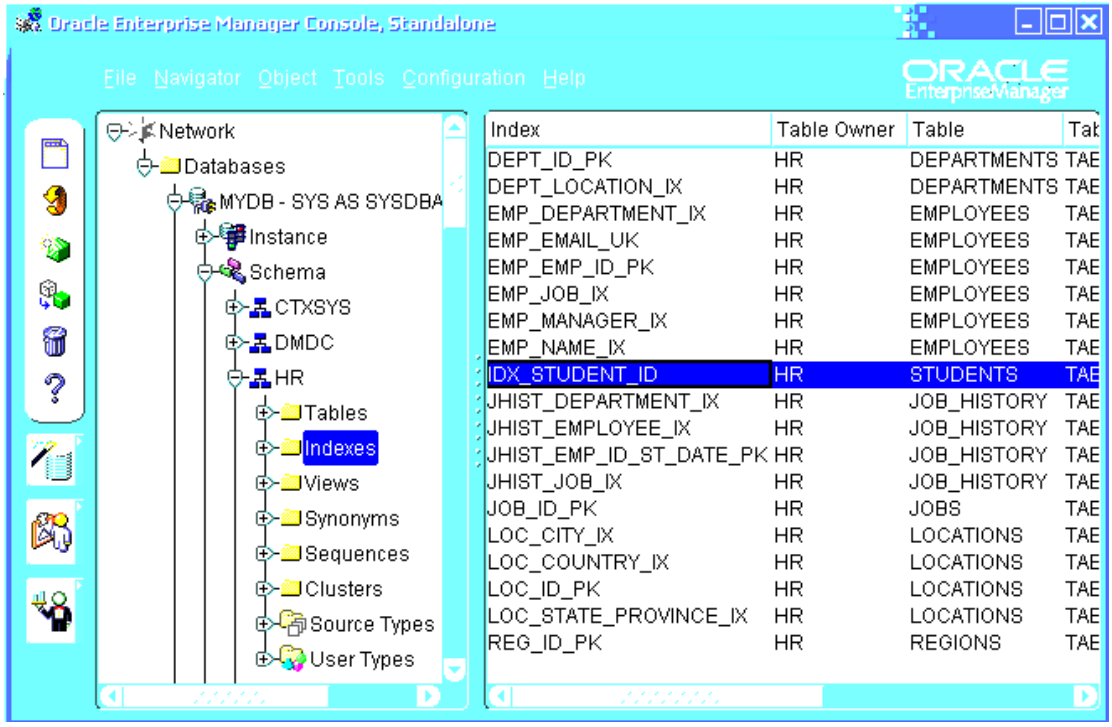
Hình 4.3.2.38 Tạo chỉ mục - Bước 1

Màn hình tiếp theo cho phép thực hiện định nghĩa chỉ mục



Hình 4.3.2.39 Tạo chỉ số - Bước 2

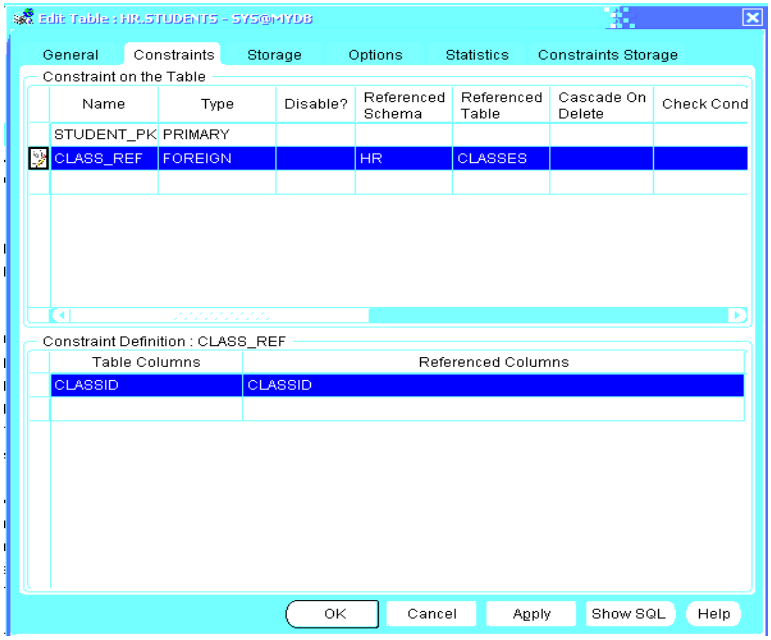
Xác định tên chỉ mục, lược đồ, và tên bảng mà chúng ta muốn tạo chỉ mục sau đó nhấn **Create**. Kiểm tra lại trong danh sách các chỉ mục của lược đồ, chúng ta sẽ thấy chỉ mục mà chúng ta vừa tạo.



Hình 4.3.2.40 Danh sách các chỉ mục

Tạo các liên kết

Để thực hiện tạo liên kết giữa các bảng dữ liệu, chọn bảng ở chiều nhiều, chọn tab **Constraints**.



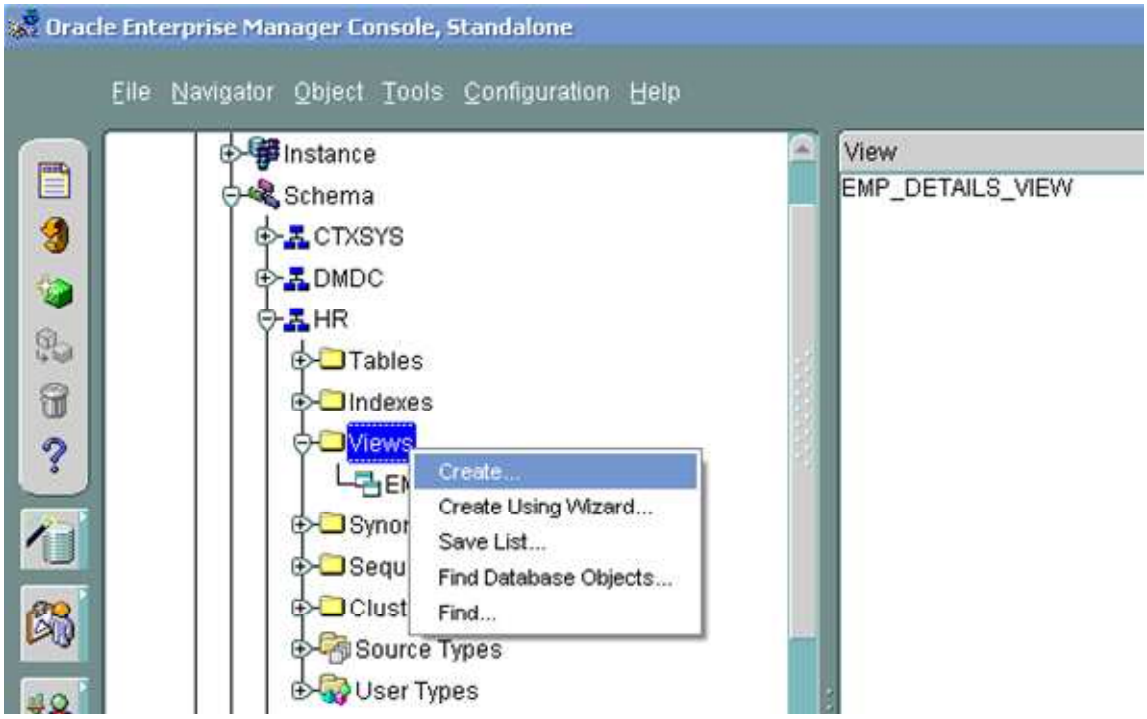
Hình 4.3.2.41 Tạo liên kết

Nhập mới một tên vào mục **Name** trong ô lưới phía trên, đặt kiểu (**Type**) là **FOREIGN** (khóa ngoại lai) vào lược đồ, bảng mà chúng ta sẽ đặt liên kết.

Trong ô lưới phía dưới, chọn bảng tham chiếu và cột trong bảng tham chiếu mà chúng ta sẽ đặt liên kết. Nhấn **OK** để lưu lại liên kết này.

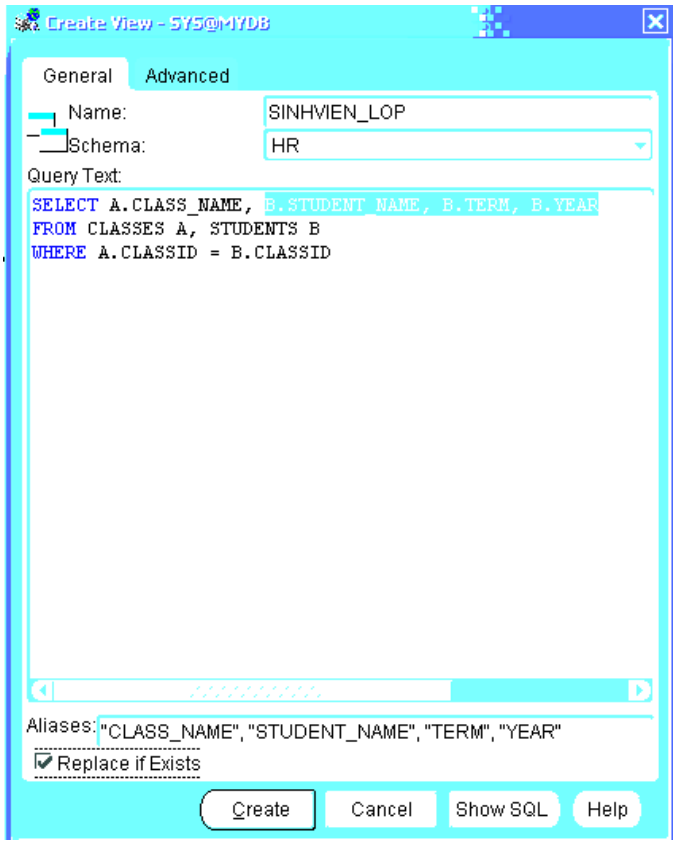
Tạo các khung nhìn

Chọn mục **Views** trong lược đồ hiện tại, nhấn chuột phải và chọn **Create**



Hình 4.3.2.42 Tạo khung nhìn – Bước 1

Màn hình tiếp sau đó cho phép thực hiện tạo khung nhìn mới.



Hình 4.3.2.43 Câu lệnh tạo khung nhìn

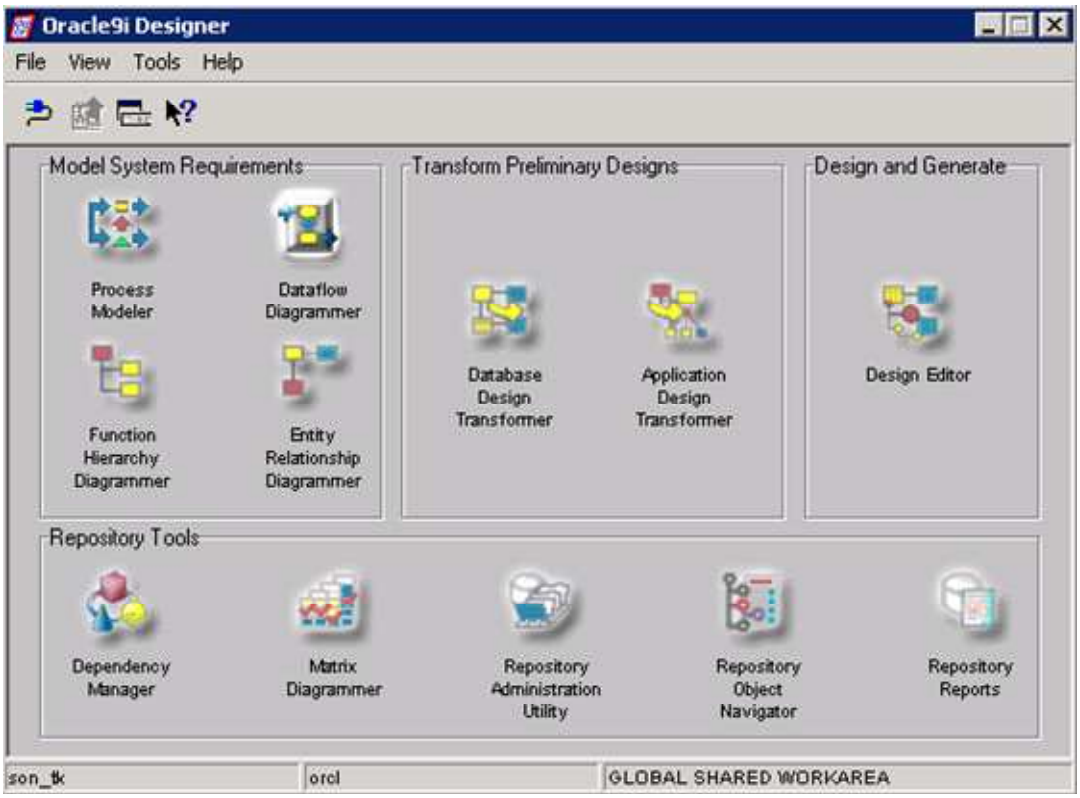
Để có thể sử dụng được các câu lệnh tạo khung nhìn, người sử dụng cần có những hiểu biết nhất định về cú pháp SQL để có thể tạo khung nhìn, bởi lẽ Oracle không cung cấp công cụ trực quan để thực hiện tạo các đối tượng kiểu này. Thay vào đó, việc nhập dòng lệnh SQL là để định nghĩa khung nhìn cho riêng chúng ta. Sau khi nhập đủ thông tin tại các mục, nhấn **Create** để tạo khung nhìn. Chú ý mục **Aliases** cần nhập tên các cột mà chúng ta muốn có trong khung nhìn này.

Sau khi tạo xong khung nhìn, hãy kiểm tra sự tồn tại của khung nhìn trong cơ sở dữ liệu, Đồng thời thực hiện kiểm tra dữ liệu của khung nhìn xem đã đúng ý chưa. Nếu chưa

thỏa mãn, hãy quay lại màn hình thiết kế khung nhìn để bổ sung các điểm cần thiết

Trên đây chúng ta đã xem xét các bước để tạo được các đối tượng dữ liệu trong cơ sở dữ liệu Oracle bằng công cụ quản trị hệ cơ sở dữ liệu Enterprise Manager. Đi kèm với hệ quản trị cơ sở dữ liệu, Oracle cũng đã cung cấp thêm các gói phần mềm hỗ trợ quá trình phân tích thiết kế cơ sở dữ liệu cũng như xây dựng ứng dụng được đánh giá là khá mạnh.

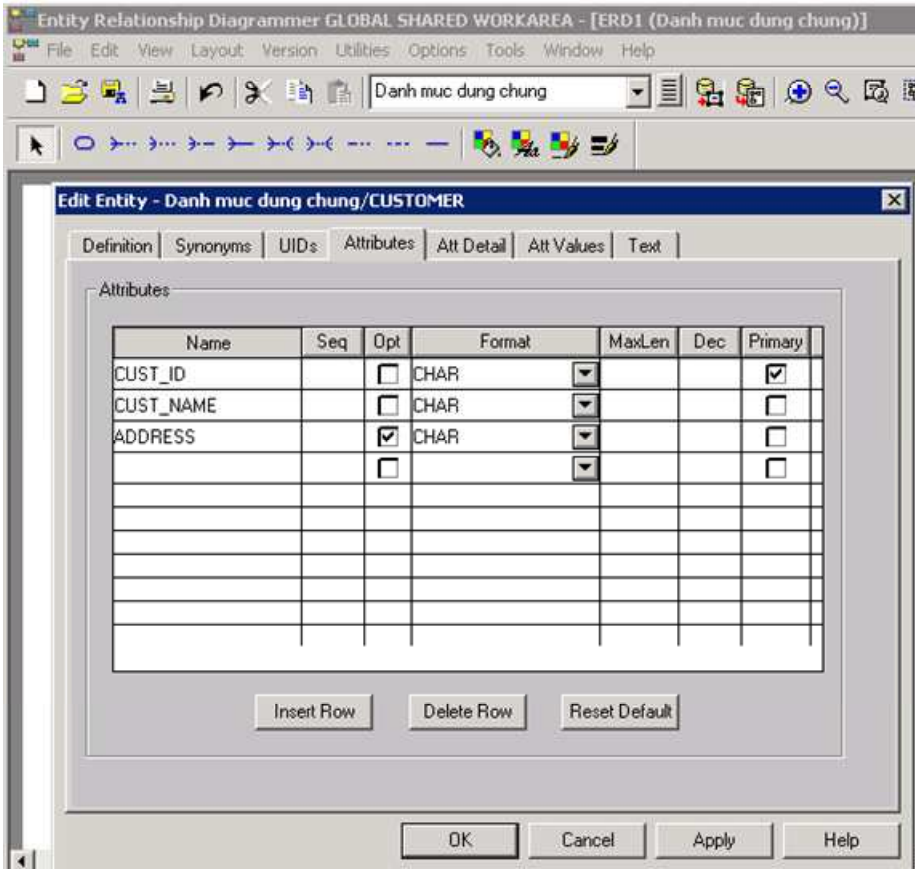
Một công cụ phân tích thiết kế cơ sở dữ liệu được biết nhiều nhất của Oracle đó là Oracle Designer. Ngoài ra, với các cơ sở dữ liệu được xây dựng theo mô hình kho dữ liệu hãng cũng cung cấp thêm một công cụ phân tích thiết kế mới đó là Oracle Warehouse Builder trong các phiên bản từ 9i trở đi. Những bộ công cụ này khá dễ sử dụng và tiện lợi, dưới đây chúng tôi giới thiệu sơ qua về công cụ phân tích, thiết kế cơ sở dữ liệu Oracle9i Designer.



Hình 4.3.2.44 Các công cụ của Oracle9i Designer

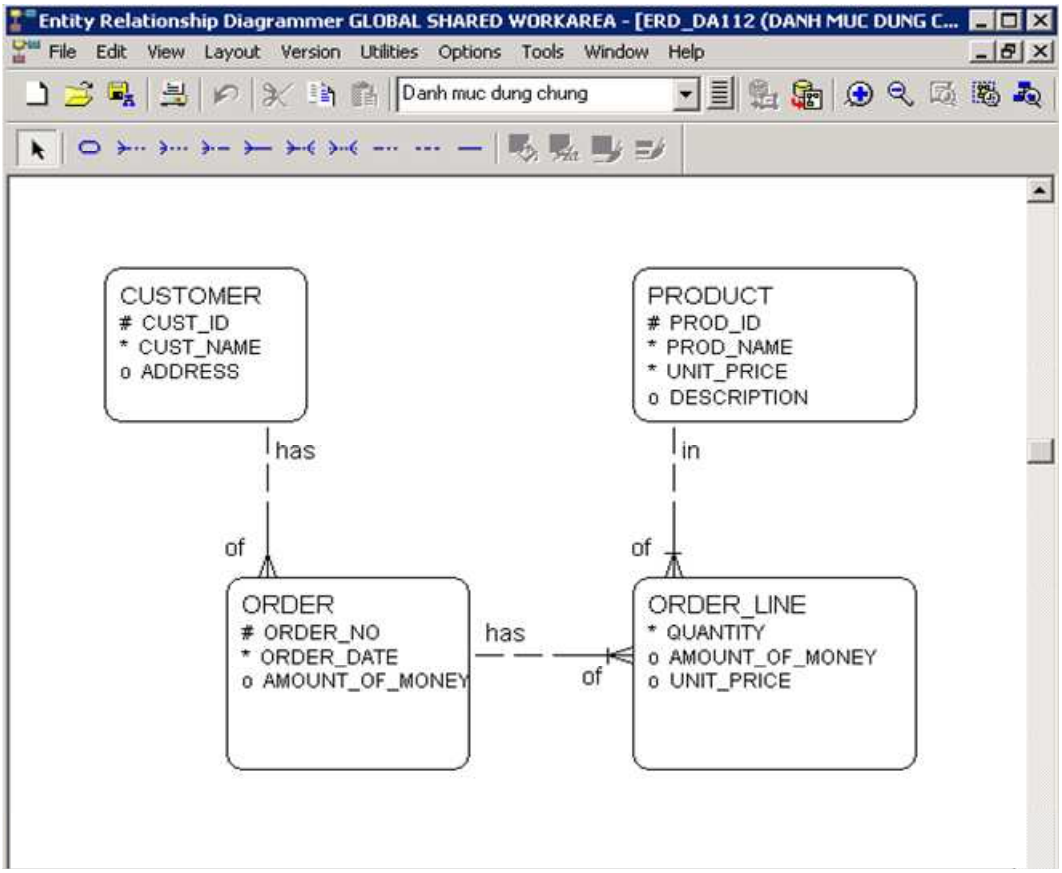
Màn hình trên hiển thị các chức năng trong bộ công cụ Oracle9i Designer. Các công cụ Entity Relationship Diagram, Database Design Transformer và Design Editor thường được sử dụng trong quá trình phân tích thiết kế cơ sở dữ liệu.

Công cụ Entity Relationship Diagram cho phép người sử dụng tạo dựng một sơ đồ thực thể - liên kết theo mô hình dữ liệu quan hệ cho cơ sở dữ liệu. Màn hình tiếp theo dưới đây minh hoạ cho việc tạo dựng một kiểu thực thể và các thuộc tính tương ứng của nó.



Hình 4.3.2.45 Tạo dựng một kiểu thực thể và các thuộc tính của nó

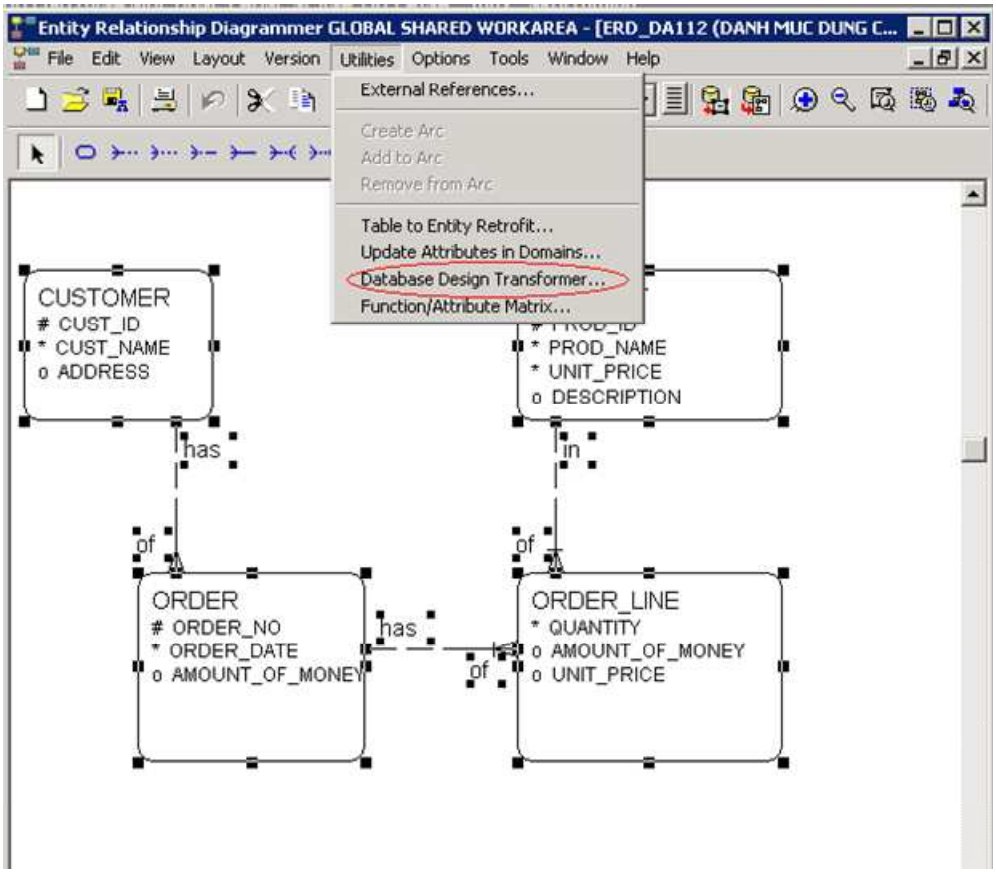
Sau khi tạo xong các kiểu thực thể, chúng ta sẽ có một sơ đồ thực thể - liên kết cho cơ sở dữ liệu, như màn hình sau minh hoạ cho bốn kiểu thực thể đã được tạo trong ví dụ chung trong giáo trình của chúng ta là CUSTOMER, PRODUCT, ORDER, ORDER_LINE:



Hình 4.3.2.46 Một sơ đồ thực thể liên kết trong Oracle

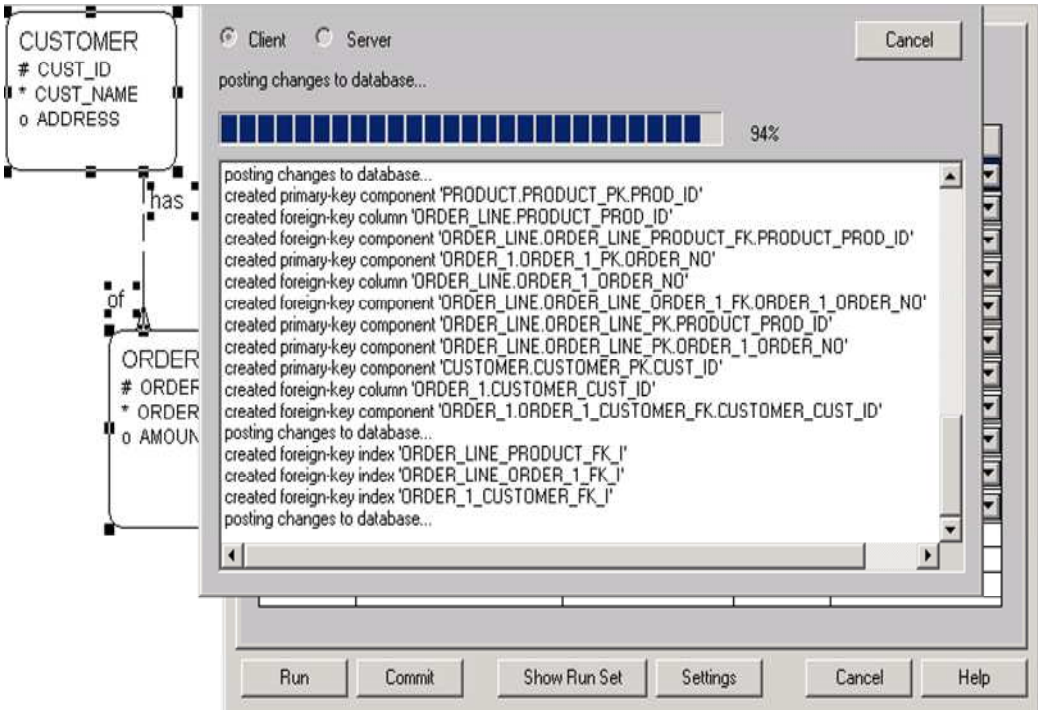
Trong hình trên, các kiểu thực thể được ký hiệu bằng các hình chữ nhật, các thuộc tính mô tả được đặt trong kiểu thực thể. Thuộc tính khoá được ký hiệu bằng một dấu thăng (#) phía trước, thuộc tính có giá trị NOT NULL được ký hiệu bằng một dấu sao (*) phía trước, các thuộc tính thông thường được ký hiệu bằng một dấu tròn (o) phía trước. Ký hiệu các liên kết giữa các kiểu thực thể vẫn được giữ nguyên. Ngoài ra, chúng ta nhận thấy rằng, nếu trong liên kết một nhiều, phía đầu liên kết nhiều có thể có thêm một nét gạch ngang phía trước, ký hiệu này cho biết các thuộc tính khoá chính của kiểu thực thể phía liên kết một sẽ được chuyển sang kiểu thực thể phía liên kết nhiều và tham gia vào làm khoá chính.

Sau khi mô tả xong sơ đồ thực thể - liên kết, chúng ta có thể tiến hành chuyển đổi chúng thành sơ đồ dữ liệu mức logic thông qua chức năng Database Design Transformer như hình dưới đây:



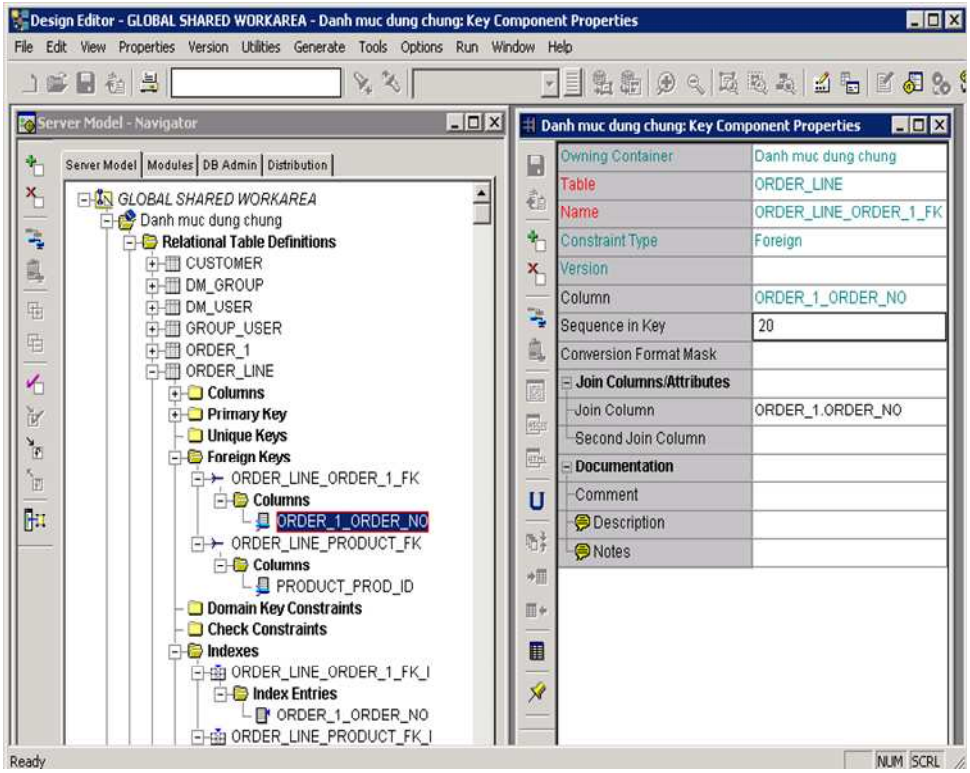
Hình 4.3.2.47 Chuyển đổi sơ đồ thực thể liên kết thành sơ đồ dữ liệu mức logic

Oracle Designer sẽ tự động sản sinh ra các quan hệ (hay bảng) cùng các khoá ngoại lai dựa trên các kiểu thực thể và các liên kết giữa chúng trong sơ đồ thực thể - liên kết trên. Quá trình chuyển đổi được thực hiện như hình sau:



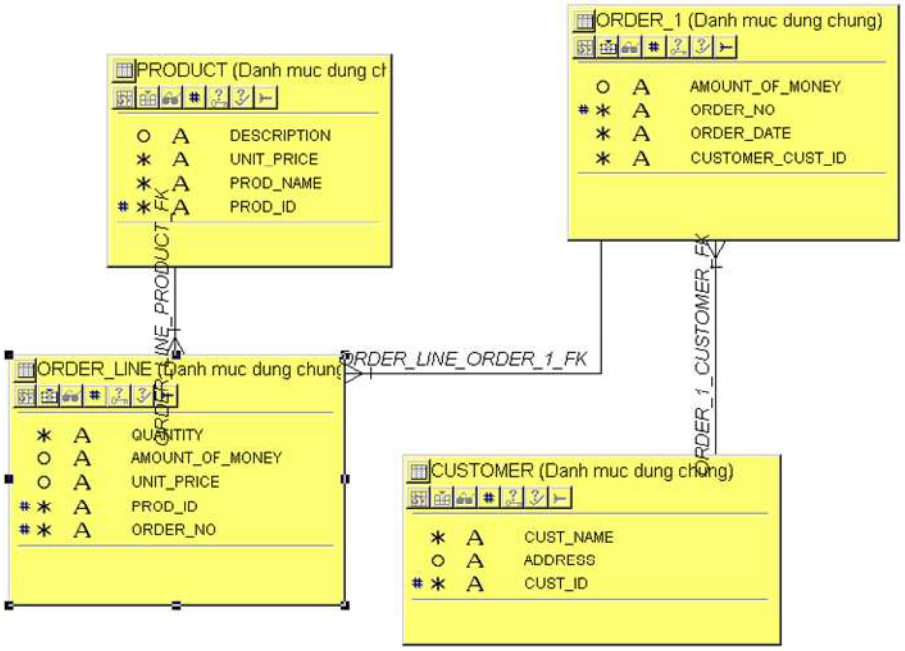
Hình 4.3.2.48 Quá trình chuyển đổi thành các quan hệ, khoá ngoại lai,...

Sau quá trình chuyển đổi trên, chúng ta sẽ có các quan hệ cùng các thuộc tính tương ứng từ các kiểu thực thể và các liên kết, tức là có sơ đồ cơ sở dữ liệu mức logic. Thông qua công cụ Desiger Editor như hình sau, chúng ta có thể tiến hành bổ sung các chỉ mục, các ràng buộc toàn vẹn cho quan hệ. Có một lưu ý là với các khoá ngoại lai được sinh từ liên kết giữa các kiểu thực thể xuất hiện trong quan hệ sẽ có một chỉ mục được tạo ra tương ứng với nó (Oracle Designer đã tự động thực hiện động tác này), như trong hình sau:



Hình 4.3.2.49 Các quan hệ tương ứng được tạo ra

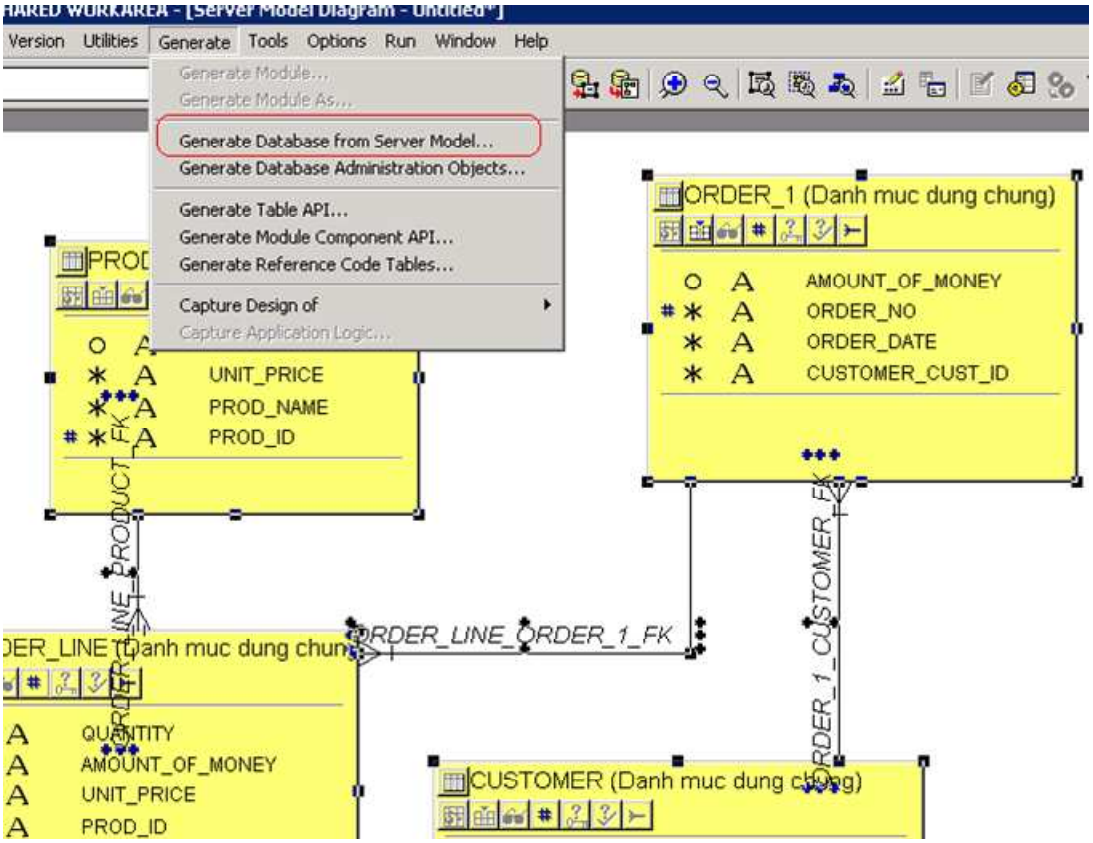
Chúng ta có thể hiển thị các quan hệ này cùng các mối quan hệ giữa chúng thông qua sơ đồ cơ sở dữ liệu mức logic:



Hình 4.3.2.50 Một sơ đồ cơ sở dữ liệu mức logic

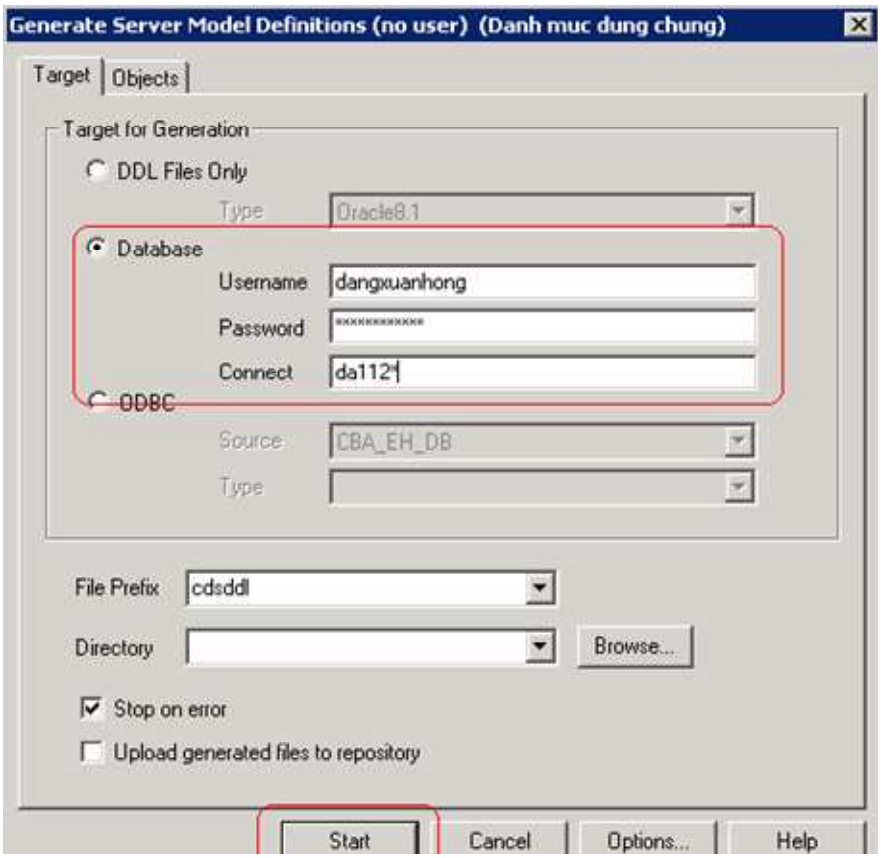
Chúng ta thấy sơ đồ này cũng tương tự như sơ đồ thực thể - liên kết. Tuy nhiên, các thuộc tính trong mỗi quan hệ đã được hình thành đầy đủ theo thiết kế. Và trong sơ đồ này, chúng ta vẫn có thể tiếp tục xây dựng các chỉ mục, phân tách quan hệ, tạo lập các ràng buộc dữ liệu,... Đây chính là một phần của quá trình thiết kế cơ sở dữ liệu mức vật lý.

Sau khi hoàn thành, Oracle sẽ cho phép chúng ta tạo dựng các câu lệnh tạo các đối tượng dữ liệu cho cơ sở dữ liệu thông qua chức năng Generate Database from Server Model như hình sau:



Hình 4.3.2.51 Câu lệnh tạo khung nhìn

Chúng ta lựa chọn cơ sở dữ liệu để cài đặt các đối tượng, cũng cần lưu ý là quá trình thiết kế cơ sở dữ liệu mức vật lý có thể được can thiệp vào bước này với việc chỉnh sửa thêm các câu lệnh sinh các đối tượng cho cơ sở dữ liệu.



Hình 4.3.2.52 Câu lệnh tạo khung nhìn

Sau khi thực hiện xong bước trên, chúng ta đã có một cơ sở dữ liệu hoàn chỉnh.

3.3. Hệ quản trị cơ sở dữ liệu mã nguồn mở

Ngày nay với xu thế sử dụng mã nguồn mở, trên thị trường cũng đã xuất hiện những hệ quản trị cơ sở dữ liệu đi theo xu hướng này, điển hình là hai hệ quản trị cơ sở dữ liệu MySQL và Postgre. Ưu điểm của những hệ quản trị cơ sở dữ liệu này là bạn hầu như không phải quan tâm đến vấn đề bản quyền và quan trọng hơn, bạn có thể thay đổi hay bổ sung thêm các chức năng cho chúng (nếu bạn là người am hiểu hệ thống và lập trình tốt).

Với hệ quản trị cơ sở dữ liệu MySQL, do công ty MySQL AB phát triển, phân phối và hỗ trợ, hiện nay có bốn phiên bản chủ yếu sau:

- MySQL Standard: phiên bản này được xây dựng nhắm tới những người dùng muốn có một cơ sở dữ liệu đủ mạnh, có hiệu suất tính toán cao.
- MySQL Max: phiên bản này nhắm tới những người dùng muốn sử dụng những tính năng mới không có ở các phiên bản khác như bộ xử lý lưu trữ Berkeley database, cho phép các bảng dữ liệu có thể được chia nhỏ thành các tệp, tránh được hạn chế về giới hạn kích thước ở một số hệ điều hành.
- MySQL Pro là một phiên bản thương mại của MySQL Standard.
- MySQL Classic: khác với các phiên bản Standard và Pro, đây là một phiên bản chỉ bao gồm một số chức năng chính của MySQL.

Hệ quản trị cơ sở dữ liệu MySQL cũng có hệ thống bảo đảm an toàn bảo mật cao, bao gồm cả việc hỗ trợ mã hoá tầng vận chuyển theo SSL (Security Socket Layer). Từ các phiên bản 4.0, hệ quản trị cơ sở dữ liệu này còn cho phép giới hạn tài nguyên đến từng người dùng. Ngoài ra với cơ chế query caching đã làm tăng đáng kể hiệu năng thực hiện các câu lệnh truy vấn thông thường.

Hệ quản trị cơ sở dữ liệu này cũng cho phép thực hiện các chức năng tạo bản sao dữ liệu (data replication) cho các máy server phụ (slave), giúp cho hệ thống phân tải tốt hơn cũng như hỗ trợ cơ chế phân tán dữ liệu.

Cũng như MySQL, Postgre cũng là một hệ quản trị cơ sở dữ liệu mã nguồn mở bắt nguồn từ trường đại học Berkely, hiện nay đã có đến phiên bản 7.3 với ngôn ngữ hỏi đáp dữ liệu cũng tuân theo chuẩn SQL99. Một ưu điểm đối với hệ quản trị cơ sở dữ liệu này đó là nó cho phép khôi phục dữ liệu ở mức thời điểm (point-in-time), tức là khi có sự cố, người quản trị không phải khôi phục lại toàn bộ cơ sở dữ liệu.

Các hệ quản trị cơ sở dữ liệu mã nguồn mở trên đều có thể tải về từ mạng internet với cả những phiên bản đã được thương mại hoá và có hỗ trợ kỹ thuật. Chúng ta có thể tham khảo chúng thêm ở những địa chỉ internet sau:

- www.MySQL.com
- www.PostgreSQL.org

3.4. Nhận xét chung

Các hệ quản trị cơ sở dữ liệu như Access hay Visual Fox rất thích hợp cho những cơ sở dữ liệu cỡ vừa và nhỏ (khoảng từ một vài Gigabyte trở lại). Ưu điểm chung của chúng là dễ cài đặt, giao diện thân thiện, khi vận hành không chiếm nhiều không gian nhớ. Nhưng nhược điểm lớn nhất của chúng đó là tính bảo mật không cao, tốc độ giảm đi đáng kể khi phải đáp ứng nhiều kết nối đến cơ sở dữ liệu. Trong các hệ quản trị cơ sở dữ liệu này không phân chia ra các người sử dụng khác nhau nên thường các ứng dụng dựa trên cơ sở dữ liệu phải tự xây dựng các môđun để thực hiện công việc này. Bởi vậy, chúng ta chỉ nên chọn những hệ quản trị cơ sở dữ liệu này khi cơ sở dữ liệu cần xây dựng chỉ phải đáp ứng một lượng người dùng nhỏ

(khoảng vài chục người), yêu cầu bảo mật với dữ liệu không đòi hỏi quá khắt khe.

Đối với các hệ quản trị cơ sở dữ liệu như SQL Server: ưu điểm chính vẫn là tính dễ sử dụng, giao diện thân thiện gần gũi với người sử dụng đã quen với hệ điều hành Windows. Với các phiên bản Enterprise, đã có thể đáp ứng được những cơ sở dữ liệu tầm cỡ doanh nghiệp có thể bao gồm vài trăm thậm chí vài nghìn người kết nối làm việc cùng một lúc. Tính bảo mật trong hệ quản trị cơ sở dữ liệu này cũng đã được chú trọng đáng kể, trong đó các tài khoản người dùng có thể được tích hợp với hệ điều hành WindowsNT hoặc Windows2000. Trong cùng một hệ quản trị cơ sở dữ liệu có thể tạo ra một số cơ sở dữ liệu khác nhau. Mỗi một người sử dụng có thể sở hữu cũng như có thể có quyền truy nhập đến một số cơ sở dữ liệu. Các cơ sở dữ liệu có thể có kích cỡ lớn, trong đó mỗi bảng có thể chứa đến vài Gigabyte dữ liệu hoặc hơn. Một số phiên bản còn hỗ trợ các ứng dụng theo mô hình kho dữ liệu. Ngoài ra, các bước cài đặt cho hệ quản trị này cũng không quá phức tạp. Nhược điểm chính của hệ quản trị cơ sở dữ liệu này chính là nó chỉ có thể chạy trên các hệ điều hành máy chủ Windows mà thôi. Tuy nhiên, với những ưu điểm đã trình bày, SQL Server rất phù hợp với các cơ quan, tổ chức, doanh nghiệp ở Việt Nam hiện nay, do đó nó thường được xem là lựa chọn đầu tiên cho những cơ sở dữ liệu tầm cỡ.

Đối với hệ quản trị cơ sở dữ liệu Oracle, đây được xem là một hệ quản trị cơ sở dữ liệu chiếm thị phần lớn nhất. Các ưu điểm chính là có thể đáp ứng được những cơ sở dữ liệu tầm cỡ khá lớn. Cũng như hệ quản trị cơ sở dữ liệu SQL Server, những phiên bản Enterprise có thể có kích thước

đến hàng Gigabyte thậm chí hàng Tetrabyte dữ liệu. Các bảng dữ liệu lớn có thể được chia ra thành các phân đoạn (partition) khác nhau và cũng có thể được lưu trữ trên những vị trí vật lý (ổ đĩa) khác nhau. Các cơ chế phân quyền và bảo mật cao, tuy nhiên so với hệ quản trị cơ sở dữ liệu SQL Server, việc quản lý giữa người sử dụng và cơ sở dữ liệu trong hệ quản trị cơ sở dữ liệu Oracle được tổ chức theo mô hình khác. Đó là một hệ quản trị cơ sở dữ liệu cũng có thể tạo dựng nhiều cơ sở dữ liệu khác nhau, tuy nhiên mỗi một người sử dụng phải thuộc vào một cơ sở dữ liệu nhất định. Việc gán quyền sử dụng trên các đối tượng dữ liệu được thực hiện giữa các người sử dụng trong cùng một cơ sở dữ liệu. Một ưu điểm nữa của hệ quản trị cơ sở dữ liệu này là có thể được cài đặt trên nhiều hệ điều hành khác nhau như Unix, Windows, Linux,... Chúng ta chỉ nên chọn hệ quản trị cơ sở dữ liệu này cho những hệ thống thực sự lớn, cơ sở dữ liệu chứa hàng Gigabyte trở lên, lượng người truy nhập nhiều, chỉ có như vậy mới có thể tận dụng được các ưu điểm do hệ quản trị cơ sở dữ liệu này mang lại. Tuy nhiên, cũng cần lưu ý rằng, việc quản trị một hệ thống cơ sở dữ liệu Oracle là khá phức tạp, tốn nhiều thời gian để tiếp cận và phải đòi hỏi những người quản trị thật chuyên sâu.

Hai hệ quản trị cơ sở dữ liệu mã nguồn mở MySQL và Postgre cũng còn tương đối mới mẻ đối với những nhà thiết kế cơ sở dữ liệu Việt Nam, trên thị trường hiện nay mới chỉ xuất hiện một số ứng dụng web viết bằng ngôn ngữ PHP kết nối với hệ quản trị cơ sở dữ liệu MySQL. Tuy nhiên có thể trong những năm tới các hệ quản trị cơ sở dữ liệu mã nguồn mở sẽ được để ý đến nhiều hơn và có thể chúng sẽ là một xu hướng mới trong ngành công nghiệp cơ sở dữ liệu.

4. Minh hoạ đối với một hệ quản trị cơ sở dữ liệu cụ thể

Trong các phần trình bày trên, chúng tôi đã trình bày khá kỹ các bước tạo dựng các đối tượng dữ liệu trong mỗi một hệ quản trị cơ sở dữ liệu. Do vậy trong phần này, các bạn có thể lựa chọn một trong bốn hệ quản trị cơ sở dữ liệu trên để có thể thực hành.

5. Kết luận

Trong chủ đề này chúng ta đã được làm quen với kiến trúc của một hệ quản trị cơ sở dữ liệu, bao gồm các thành phần như khối nhận câu lệnh truy vấn SQL từ người sử dụng, khối phân tích xử lý câu lệnh, khối quản lý các giao dịch, điều khiển tương tranh, khối quản lý không gian lưu trữ. Giữa các khối có tính độc lập tương đối nhưng cũng phối hợp chặt chẽ với nhau.

Ngoài ra, cũng trong chủ đề này, chúng ta đã được làm quen với bốn hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay trên thị trường, đó là hệ quản trị cơ sở dữ liệu Access, hệ quản trị cơ sở dữ liệu Visual Foxpro, hệ quản trị cơ sở dữ liệu SQL Server và hệ quản trị cơ sở dữ liệu Oracle. Trong mỗi một hệ quản trị cơ sở dữ liệu chúng ta cũng đã xem xét các đối tượng dữ liệu được tạo dựng như thế nào cũng như các bước để xây dựng lên một cơ sở dữ liệu hoàn chỉnh. Các ưu nhược điểm của mỗi một hệ quản trị cơ sở dữ liệu cũng đã được xem xét đến, giúp chúng ta có được những lựa chọn chính xác trước các hệ thống cụ thể.

6. Bài tập

1. Tiến hành cài đặt các hệ quản trị cơ sở dữ liệu sau:

- Access
- Visual Foxpro
- Microsoft SQL Server 2000
- Oracle Workgroup hoặc Oracle Enterprise Edition

2. Tiến hành thực hiện các bước đã nêu trong mục 3 của chủ đề đối với mỗi hệ quản trị cơ sở dữ liệu đã được cài đặt ở bài tập 1.

3. Với mỗi hệ quản trị cơ sở dữ liệu trong bài tập 1, dựa trên sơ đồ thiết kế cơ sở dữ liệu mức logic 2.4.11.7 trong chủ đề 2, tiến hành xây dựng các bảng (quan hệ), cùng các mối liên kết giữa chúng.

4. Với sơ đồ thực thể - liên kết đã xây dựng được trong bài tập 7 chủ đề 2, tiến hành xây dựng chúng với mỗi hệ quản trị cơ sở dữ liệu đã nêu trong bài tập 1 trên.

5. Tiếp tục tìm hiểu công cụ Oracle Designer2000, sử dụng chức năng Database Design Transformer để chuyển đổi sơ đồ thực thể - liên kết có được trong bài tập 7 chủ đề 2 thành sơ đồ dữ liệu và sử dụng chức năng Generate Database from Server Model để sinh mã cho cơ sở dữ liệu.

6. Lựa chọn hệ quản trị cơ sở dữ liệu SQL Server hoặc Oracle để tiến hành thiết kế, cài đặt cho cơ sở dữ liệu được trình bày trong phụ lục B của giáo trình.

Phụ lục A: Các hệ thống CSDL sẽ được xây dựng trong khuôn khổ dự án 112

Mục tiêu:

Trong phụ lục này chúng tôi muốn giới thiệu với các bạn một số nét chính về hệ thống các cơ sở dữ liệu mang tính chất định hướng sẽ được xây dựng trong khuôn khổ dự án 112.

1. Mô hình tổ chức hệ thống CSDL các cấp

Hệ thống thông tin (HTTT) phục vụ điều hành và quản lý hành chính nhà nước (đề án 112) bao gồm các hệ thống sau:

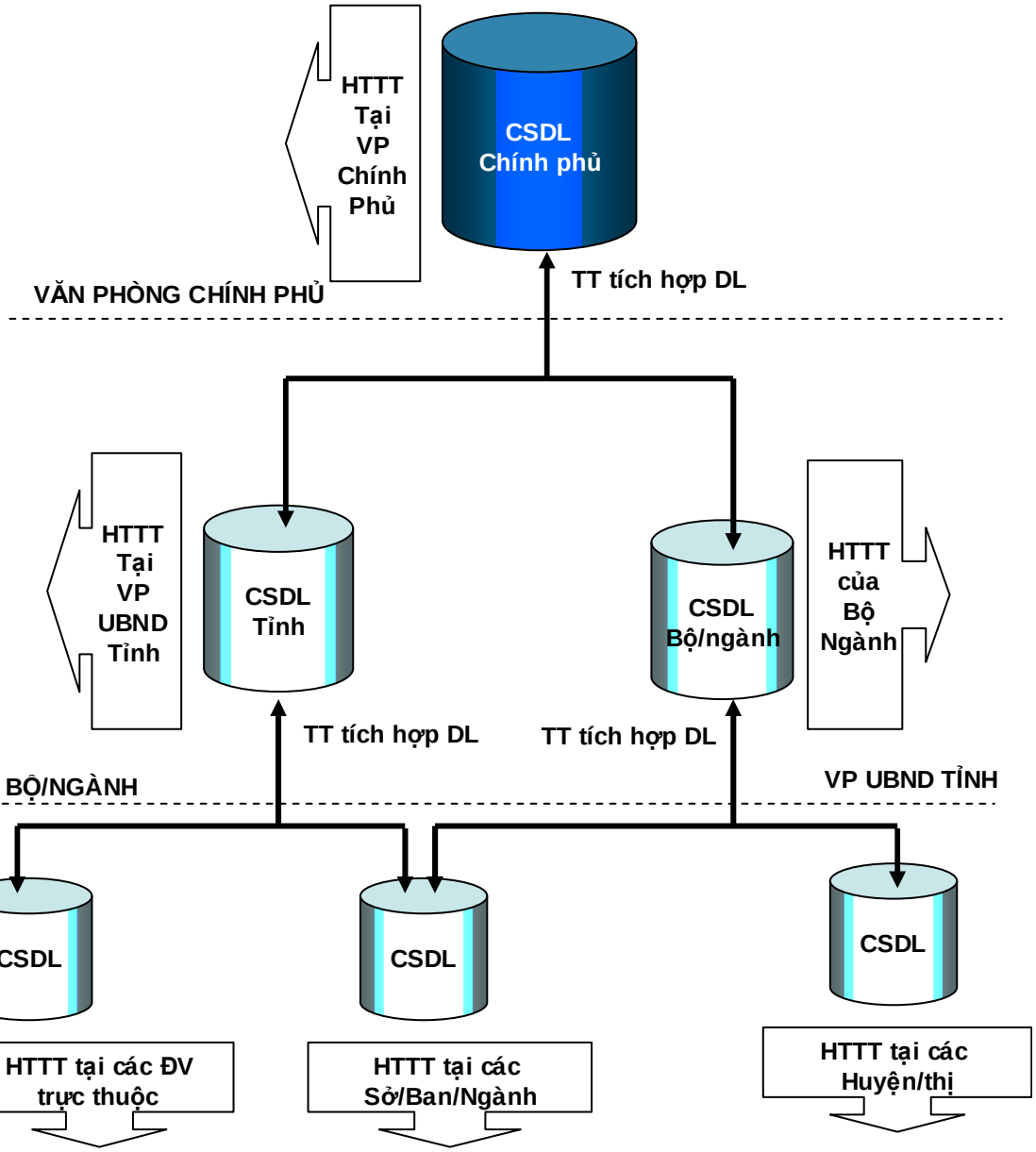
- Hệ thống thông tin Chính phủ
- Hệ thống thông tin tại các Bộ/Ngành
- Hệ thống thông tin tại các Tỉnh thành phố trực thuộc Trung ương
- Hệ thống các cơ sở dữ liệu Quốc gia.

Trong mỗi hệ thống trên bao gồm nhiều hệ thống con và hạt nhân của mỗi hệ thống là các cơ sở dữ liệu. Giữa chúng có các quan hệ liên kết, trao đổi, chia sẻ thông tin theo chiều ngang và chiều dọc. Quan hệ ngang là quan hệ trao đổi, chia sẻ thông tin có tính chất chuyên môn giữa các đơn vị ngang cấp, còn quan hệ theo chiều dọc là quan hệ dựa

trên cấu trúc phân cấp thẩm quyền chức năng quản lý nhà nước trong hệ thống các cơ quan hành chính nhà nước.

Tại mỗi hệ thống, bên cạnh việc xây dựng các HTTT sẽ hình thành các trung tâm dữ liệu, đóng vai trò là nơi liên kết, tích hợp các cơ sở dữ liệu của các đơn vị trong hệ thống và có chức năng cung cấp, chia sẻ thông tin, truyền các báo cáo lên cấp trên, truyền các mệnh lệnh quản lý xuống các cấp dưới.

Chúng ta có mô hình tổ chức các cơ sở dữ liệu đi kèm với HTTT ở các cấp như sau:



Hình 5.1.1 Mô hình tổ chức CSDL và HTTT ở các cấp

2. Các cơ sở dữ liệu phục vụ điều hành, quản lý

a. Cơ sở dữ liệu phục vụ hệ thống thông tin quản lý đơn thư khiếu nại tố cáo

Tại mỗi cơ quan phải giải quyết đơn thư khiếu nại tố cáo sẽ xây dựng một cơ sở dữ liệu quản lý đơn thư khiếu nại tố

cáo. Cơ sở dữ liệu này sẽ kết nối với HTTT cung cấp dịch vụ công để tiếp nhận và trả lời kết quả giải quyết đơn thư khiếu nại tố cáo cho dân.

Các cơ sở dữ liệu khiếu nại tố cáo của các cơ quan phải thoả mãn một số chuẩn chung và được kết nối với nhau nhằm trao đổi thông tin, chuyển đơn thư khiếu nại tố cáo cho nhau trong các trường hợp phải giải quyết liên ngành, liên cấp

Quá trình xây dựng và triển khai hệ cơ sở dữ liệu khiếu nại tố cáo bao gồm:

- Xây dựng, cài đặt và triển khai cơ sở dữ liệu khiếu nại tố cáo tại các cơ quan liên quan trên địa bàn.
- Kết nối với HTTT cung cấp dịch vụ công
- Liên kết các cơ sở dữ liệu khiếu nại tố cáo của các cơ quan nhà nước với nhau.

b. Cơ sở dữ liệu phục vụ HTTT luật và văn bản qui phạm pháp luật

Xây dựng một cơ sở dữ liệu trung tâm về văn bản quy phạm pháp luật tại trung tâm tích hợp dữ liệu, liên kết cơ sở dữ liệu trung tâm với hệ điều hành tác nghiệp tại văn phòng UBND và cơ sở dữ liệu quốc gia về luật để thường xuyên bổ sung nguồn dữ liệu cho cơ sở dữ liệu trung tâm.

Tại mỗi cơ quan xây dựng một cơ sở dữ liệu văn bản quy phạm pháp luật và kết nối với cơ sở dữ liệu trung tâm để cập nhật dữ liệu thuộc các lĩnh vực do cơ quan phụ trách và khai thác nguồn dữ liệu văn bản quy phạm pháp luật từ cơ sở dữ liệu trung tâm.

c. Các cơ sở dữ liệu ứng dụng chuyên ngành

Tuỳ thuộc vào các lĩnh vực chuyên môn nghiệp vụ của cơ quan, tại mỗi cơ quan nhà nước cần tiến hành xây dựng các cơ sở dữ liệu và chương trình ứng dụng chuyên ngành. Mỗi cơ sở dữ liệu chuyên ngành gắn với một lĩnh vực chuyên môn nghiệp vụ và có nhiệm vụ cập nhật, xử lý và lưu trữ các dữ liệu liên quan. Cụ thể, các cơ sở dữ liệu chuyên ngành có nhiệm vụ:

- Phục vụ trực tiếp các hoạt động chuyên môn theo lĩnh vực nghiệp vụ.
- Cung cấp, tiếp nhận và trao đổi thông tin chuyên môn với các phòng ban cấp dưới
- Cung cấp, thu nhận và trao đổi các dữ liệu tổng hợp phục vụ điều hành với các cơ sở dữ liệu tổng hợp kinh tế - xã hội đặt tại cơ quan.
- Một số cơ sở dữ liệu chuyên ngành sẽ kết nối với các HTTT phục vụ dịch vụ công để cung cấp, thu nhận, trao đổi thông tin.
- Cung cấp, thu nhận và trao đổi thông tin với các cơ sở dữ liệu Quốc gia cùng lĩnh vực.

Như vậy, các cơ sở dữ liệu chuyên ngành được liên kết với nhau hình thành nên một hệ thống các cơ sở dữ liệu chuyên ngành, đóng vai trò là kho dữ liệu nền phục vụ đa mục tiêu, có mối liên kết với các HTTT khác trên địa bàn.

Các cơ sở dữ liệu chuyên ngành bao gồm:

- Cơ sở dữ liệu chuyên ngành tại VP UBND Tỉnh

- Cơ sở dữ liệu chuyên ngành tại các Sở, được kết nối với các cơ sở dữ liệu liên quan như: cơ sở dữ liệu tổng hợp kinh tế - xã hội đặt tại Sở; cơ sở dữ liệu quốc gia liên quan; HTTT phục vụ dịch vụ công; cơ sở dữ liệu tại các phòng/ ban của UBND các huyện/thị
- Cơ sở dữ liệu chuyên ngành tại UBND huyện/thị: bao gồm các cơ sở dữ liệu chuyên ngành cấp cơ sở tại các phòng/ban, được kết nối với cơ sở dữ liệu chuyên ngành tại các Sở, với cơ sở dữ liệu tổng hợp kinh tế - xã hội tại UBND huyện/thị và với các HTTT phục vụ dịch vụ công trên địa bàn cấp huyện

Việc xây dựng và triển khai các cơ sở dữ liệu chuyên ngành tại các cơ quan nhà nước gồm các công việc sau:

- Phân tích, thiết kế và xây dựng các cơ sở dữ liệu chuyên ngành
- Kết nối các cơ sở dữ liệu chuyên ngành với cơ sở dữ liệu tổng hợp kinh tế - xã hội phục vụ điều hành tại cơ quan
- Kết nối các cơ sở dữ liệu chuyên ngành với các cơ sở dữ liệu quốc gia liên quan
- Tổ chức việc thu thập, phân loại, xử lý dữ liệu từ nhiều nguồn để cập nhật vào các cơ sở dữ liệu chuyên ngành (tạo nguồn dữ liệu cho cơ sở dữ liệu hoạt động).

3. Các cơ sở dữ liệu phục vụ dịch vụ công

Hệ thống các HTTT phục vụ dịch vụ công nhằm mục tiêu:

- Nâng cao tính hiệu quả, rút ngắn thời gian, tự động hoá, đơn giản hoá và làm gọn nhẹ các qui trình giải quyết, xử lý các dịch vụ công của các cơ quan nhà nước
- Thực hiện việc phục vụ cung cấp các dịch vụ công, nhất là các dịch vụ công mang tính liên ngành theo chế độ một cửa.
- Cung cấp các thông tin công cộng về luật pháp, về các lĩnh vực dịch vụ công cho mọi người dân, nâng cao hiểu biết của nhân dân.

Để xây dựng và triển khai các HTTT phục vụ dịch vụ công cần phải:

- Xây dựng các cơ sở dữ liệu và giao diện xử lý
- Xây dựng các Web site kết nối đến các cơ sở dữ liệu.
- Tạo nguồn dữ liệu cho các cơ sở dữ liệu.

Các cơ sở dữ liệu cần xây dựng có thể là:

- Cơ sở dữ liệu phụ vụ cấp giấy chứng nhận quyền sử dụng đất
- Cơ sở dữ liệu phục vụ cấp phép xây dựng
- Cơ sở dữ liệu phục vụ cấp phép đầu tư
- Cơ sở dữ liệu phụ vụ cấp đăng ký và quản lý kinh doanh
- Cơ sở dữ liệu phục vụ đăng ký hộ khẩu và dân cư
- Cơ sở dữ liệu phụ vụ tiếp nhận và giải quyết đơn thư khiếu nại, tố cáo

4. Các cơ sở dữ liệu quốc gia

Trong quá trình xây dựng các hệ thống thông tin chuyên ngành, có một số cơ sở dữ liệu quan trọng mang tính liên ngành, nhiều đơn vị sử dụng chung, đóng vai trò then chốt cho việc cung cấp thông tin kịp thời, đủ tin cậy phục vụ cho công tác quản lý nhà nước và các đối tượng khác nhau, đó là các cơ sở dữ liệu Quốc gia. Phù hợp với các cơ sở dữ liệu Quốc gia này, tại địa bàn Tỉnh cần xây dựng các cơ sở dữ liệu là thành phần của các cơ sở dữ liệu quốc gia trên địa bàn tỉnh. Việc xây dựng các cơ sở dữ liệu quốc gia trên địa bàn tỉnh phải tuân theo nội dung của các cơ sở dữ liệu quốc gia đang được triển khai xây dựng trong phạm vi toàn quốc.

Đó là các cơ sở dữ liệu sau đây:

- Cơ sở dữ liệu quốc gia về thống kê kinh tế xã hội
- Cơ sở dữ liệu quốc gia về tài chính, ngân sách
- Cơ sở dữ liệu quốc gia về văn bản quy phạm pháp luật
- Cơ sở dữ liệu quốc gia về cán bộ, công chức
- Cơ sở dữ liệu quốc gia về dân cư
- Cơ sở dữ liệu quốc gia về tài nguyên đất
- …..

Phụ lục B: Phân tích, thiết kế cơ sở dữ liệu cho một cửa hàng sách

Mục tiêu:

Trong phụ lục này, chúng ta sẽ xem xét một ví dụ minh hoạ cho các bước phân tích, thiết kế và xây dựng một cơ sở dữ liệu như đã trình bày trong giáo trình. Nội dung bài toán là Xây dựng một cơ sở dữ liệu cho một cửa hàng sách. Các bạn có thể lựa chọn một trong số bốn hệ quản trị cơ sở dữ liệu đã trình bày trong chủ đề IV để thiết kế cho cơ sở dữ liệu này..

1. Phân tích yêu cầu

Người đại diện cho cửa hàng sách đã suy nghĩ về vấn đề này và trình bày tương đối rõ ràng trong tóm tắt sau:

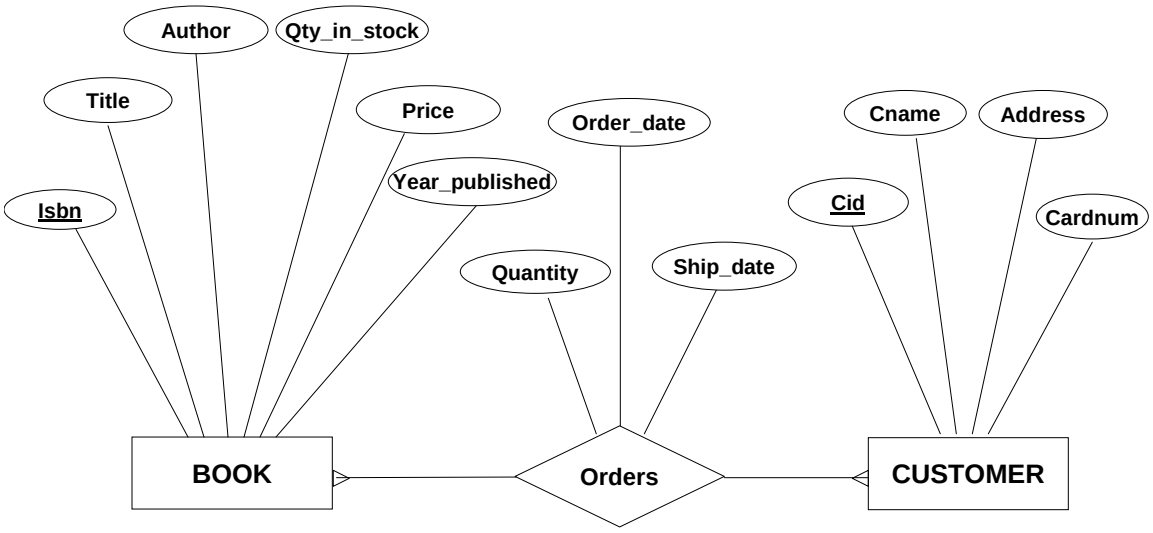
“Tôi muốn khách hàng có thể tìm kiếm các danh mục sách trong cửa hàng và có thể đặt hàng ngay trên mạng internet. Hiện nay, các công việc đặt mua sách này vẫn đang được thực hiện qua điện thoại. Các khách hàng gọi điện đến cửa hàng và nói với tôi số hiệu sách ISBN cùng số lượng đối với mỗi loại mà họ muốn mua. Nếu tôi không có đủ số lượng sách đối với một đầu sách nào đó, tôi sẽ phải đặt hàng thêm và trì hoãn lại ngày giao sách cho tới khi số lượng sách mới tới cửa hàng. Tôi muốn giao toàn bộ sách trong yêu cầu đặt hàng cho khách hàng cùng một lúc với bảng kê bao gồm tất cả các sách mà tôi bán. Với mỗi đầu sách sẽ được lưu trong bảng kê như sau: số hiệu ISBN của

sách, tiêu đề, tác giả, giá mua, giá bán và năm xuất bản. Hầu hết các khách hàng đều là khách quen và tôi đã ghi lại danh sách của họ bao gồm: tên, địa chỉ, số thẻ tín dụng. Đối với một khách mới của cửa hàng thì họ phải liên hệ trước để tôi tạo cho họ một tài khoản trước khi họ có thể sử dụng trang web của tôi để có thể tìm kiếm và đặt hàng mua sách trực tuyến trên trang web".

Với trình bày trên đây, chúng ta sẽ phải tiến hành phân tích chi tiết các thông tin. Bước tiếp theo chúng ta phải làm đó là:

2. Thiết kế cơ sở dữ liệu mức khái niệm

Trong bước này chúng ta sẽ phải mô tả dữ liệu dưới dạng mô hình thực thể liên kết. Bản thiết kế đầu tiên như trong hình sau. Sách BOOK và khách hàng CUSTOMER là những kiểu thực thể chúng ta có thể xác định được đầu tiên, hai kiểu thực thể này kết nối với nhau thông qua liên kết đơn đặt hàng "Orders". Với mỗi đơn đặt hàng, những thông tin sau cần phải được lưu lại: số lượng Quantity, ngày đặt hàng Order_date và ngày giao hàng Ship_date. Khi ngày giao hàng Ship_date mang giá trị rỗng, tương ứng với thông tin cho biết là đơn đặt hàng chưa được giao. Ngay sau khi đơn đặt hàng được giao, giá trị trong ngày giao hàng Ship_date sẽ được đặt lại giá trị và nó cho biết đơn giao hàng đó đã được giao.



Hình 6.2.1 Sơ đồ thực thể - liên kết giữa BOOK và CUSTOMER

Đến đây chúng ta đã có một bản thiết kế đầu tiên và chúng ta cần phải thảo luận thêm trong nhóm thiết kế cơ sở dữ liệu về một số trường hợp có thể xảy ra trên thực tế và cách giải quyết ở bước này như thế nào?

CH (câu hỏi): Nếu một khách hàng đặt 2 đơn đặt hàng cho cùng một quyển sách trong cùng một ngày thì sẽ phải xử lý như thế nào?

TL (trả lời): Đối với lần đặt hàng thứ nhất, nó sẽ được tạo ra một đơn đặt hàng trong liên kết đơn hàng “Orders”, đối với lần đặt hàng thứ 2 nó sẽ được xử lý bằng cách cập nhật lại thuộc tính số lượng Quantity trong liên kết “Orders” đã tạo.

CH: Nếu một khách hàng đặt 2 đơn đặt hàng cho 2 quyển sách khách nhau trong cùng một ngày thì sẽ được xử lý như thế nào:

TL: Khi đó ta coi mỗi lần đặt sách là một đơn đặt hàng khác nhau như trong các trường hợp thông thường. Tức là mỗi một thể hiện của liên kết "Orders" liên quan đến khách hàng cho mỗi quyển sách đó.

CH: Nếu khách hàng đặt hàng cùng một quyển sách nhưng ở 2 ngày khác nhau thì sẽ được xử lý như thế nào?

TL: Cũng tương tự như trong trường hợp trên, ta sẽ phải lưu 2 thể hiện cho 2 đơn đặt hàng đó và chúng sẽ có giá trị tại thuộc tính ngày đặt hàng Order_date khác nhau.

CH: Nếu xử lý như vậy thì không thể được, bởi các giá trị kết nối giữa hai thuộc tính mã hiệu khách hàng Cid và số hiệu sách Isbn khi được liên kết với nhau không thể xác định ra duy nhất một đơn hàng trong "Orders". Như vậy bản thiết kế ban đầu này chưa cho phép khách hàng đặt các đơn đặt hàng khác nhau cho cùng một cuốn sách trong những ngày khác nhau.

TL: Đúng như vậy, có lẽ khách hàng chưa quan tâm đến trường hợp này, chúng ta cần phải tìm hiểu thêm.

Chúng ta hãy chuyển sang bước thiết kế tiếp theo và xem xét vấn đề trên sẽ được xử lý tiếp như thế nào

3. Thiết kế cơ sở dữ liệu mức logic

Ở bước này chúng ta ánh xạ sơ đồ thực thể liên kết trong hình 6.2.1 sang mô hình quan hệ, sinh ra các quan hệ như sau:

```
Create table BOOK  (Isbn  char(10),
                    Title char(80),
                    Author char(80),
```

```
                Qty_in_stock integer,
                Price real,
                Year_published integer,
                primary key (Isbn))
Create table CUSTOMER (Cid integer,
                Cname char(80),
                Address char(200),
                Cardnum char(16),
                primary key (Cid),
                unique (Cardnum))
Create table ORDER (Isbn char(10),
                Cid integer,
                Quantity integer,
                Order_date date,
                Ship_date date,
                primary key (Isbn, Cid),
                foreign key (Isbn) references BOOK,
                foreign key (Cid) references CUSTOMER)
```

Chúng ta nhận thấy rằng liên kết nhiều-nhiều "Orders" đã được chuyển thành một quan hệ (bảng).

Quay trở lại vấn đề phát sinh trong phần trên, đến đây chúng ta nhận thấy rằng, quan hệ ORDER chứa một trường Order_date và khoá của bàng bao gồm 2 thuộc tính Cid và Isbn. Chính vì lý do này mà khách hàng không thể đặt 2 đơn hàng cho cùng một cuốn sách ở các ngày khác nhau,

hạn chế này chúng ta hoàn toàn không muốn và trên thực tế có những trường hợp như vậy xảy ra. Khi đó để loại bỏ hạn chế này, chúng ta chỉ cần chèn thêm thuộc tính Order_date vào khoá của quan hệ ORDER, và có quan hệ mới như sau:

```
Create table ORDER (Isbn char(10),
        Cid integer,
        Quantity integer,
        Order_date date,
        Ship_date date,
        primary key (Isbn, Cid, Order_date),
        foreign key (Isbn) references BOOK,
        foreign key (Cid) references CUSTOMER)
```

Tuy nhiên đây có vẻ như không hợp lý với sơ đồ thực thể liên kết, và chúng ta quyết định phải gặp lại khách hàng.

Khách hàng lại cho chúng ta biết thêm một số yêu cầu mà chưa được đề cập đến ở lần trước đó như sau: “Khách hàng có thể đặt mua một số đầu sách trong cùng một đơn đặt hàng. Ví dụ, nếu khách hàng muốn mua 3 quyển sách Văn học và 2 quyển Luật học thì họ có thể yêu cầu trong cùng một đơn đặt hàng”. Và chúng ta có thể hỏi khi đó nó sẽ ảnh hưởng như thế nào đến quá trình giao hàng nếu cửa hàng không đủ số lượng cho hai đầu sách đó? Cửa hàng sẽ giao đầy đủ tất cả các cuốn sách theo đơn hàng hay có thể giao từng loại? Và khách hàng cho chúng ta biết như sau: “Ngay sau khi cửa hàng có đủ số lượng sách của một đầu sách như yêu cầu trong đơn hàng, cửa hàng sẽ tiến hành giao hàng ngay mặc dù trong đơn đặt hàng có nhiều đầu sách. Tức là có thể xảy ra trường hợp 3 cuốn sách Văn học sẽ được giao

ngày hôm nay bởi trong kho của cửa hàng có 5 cuốn loại này, tuy nhiên sách Luật học có thể ngày mai mới được giao bởi trong kho chỉ có một cuốn, và ngày mai số lượng sách đó mới về tới kho của cửa hàng. Thêm vào đó, khách hàng có thể đặt nhiều đơn hàng trong một ngày và họ cũng muốn biết thông tin về các đơn đặt hàng mà họ đã đặt".

Như vậy, chúng ta nhận thấy rằng đã có 2 yêu cầu mới phát sinh trong lần gặp gỡ khách hàng này: thứ nhất, hệ thống phải đáp ứng được khả năng đặt nhiều đầu sách khác nhau trong cùng một đơn đặt hàng duy nhất. Thứ 2, một khách hàng có thể đặt các đơn hàng khác nhau trong cùng một ngày. Để có thể giải quyết được vấn đề này, chúng ta cần đưa thêm một thuộc tính mới vào quan hệ ORDER, đó là thuộc tính số hiệu đơn hàng Order_id, sẽ xác định duy nhất một đơn hàng do khách hàng đặt. Tuy nhiên, vì khách mua có thể đặt nhiều đầu sách khác nhau trong cùng một đơn hàng nên giá trị trong hai thuộc tính Order_id và Isbn kết hợp với nhau mới có thể xác định được các giá trị trong thuộc tính Quantity và Ship_date trong quan hệ ORDER.

Các đơn hàng sẽ được gán bởi các số hiệu đơn hàng khác nhau theo thứ tự tuần tự, và đơn hàng sau sẽ có giá trị số hiệu đơn hàng lớn hơn đơn hàng trước. Nếu có nhiều đơn hàng của cùng một khách trong cùng một ngày thì chúng cũng được phân biệt với nhau do các số hiệu đơn hàng là khác nhau. Câu lệnh SQL để tạo quan hệ ORDER này sẽ như sau:

```
Create table ORDER (Order_id integer,
                    Isbn char(10),
                    Cid integer,
```

Quantity integer,
Order_date date,
Ship_date date,
primary key (Order_id, Isbn),
foreign key (Isbn) references BOOK,
foreign key (Cid) references CUSTOMER)

4. Tinh chỉnh lược đồ

Tại bước này, chúng ta phải phân tích tập các quan hệ có thể gây ra sự dư thừa số liệu. Trong quan hệ BOOK chỉ có duy nhất một trường khoá đó là Isbn, và không có phụ thuộc hàm nào khác thoả trên quan hệ này. Do vậy quan hệ BOOK là đã ở dạng chuẩn 3NF. Quan hệ CUSTOMER cũng có duy nhất một thuộc tính khoá đó là Cid, bởi số hiệu thẻ tín dụng là duy nhất nên phụ thuộc hàm Cardnum → Cid cũng thoả. Không còn phụ thuộc hàm nào khác nên CUSTOMER cũng ở dạng chuẩn 3NF.

Như trên đã xác định cặp (Order_id, Isbn) là khoá của quan hệ ORDER. Thêm vào đó, mỗi đơn hàng được đặt bởi một khách mua của một ngày cụ thể nên 2 phụ thuộc hàm sau là thoả:

Order_id → Cid, và Order_id → Order_date

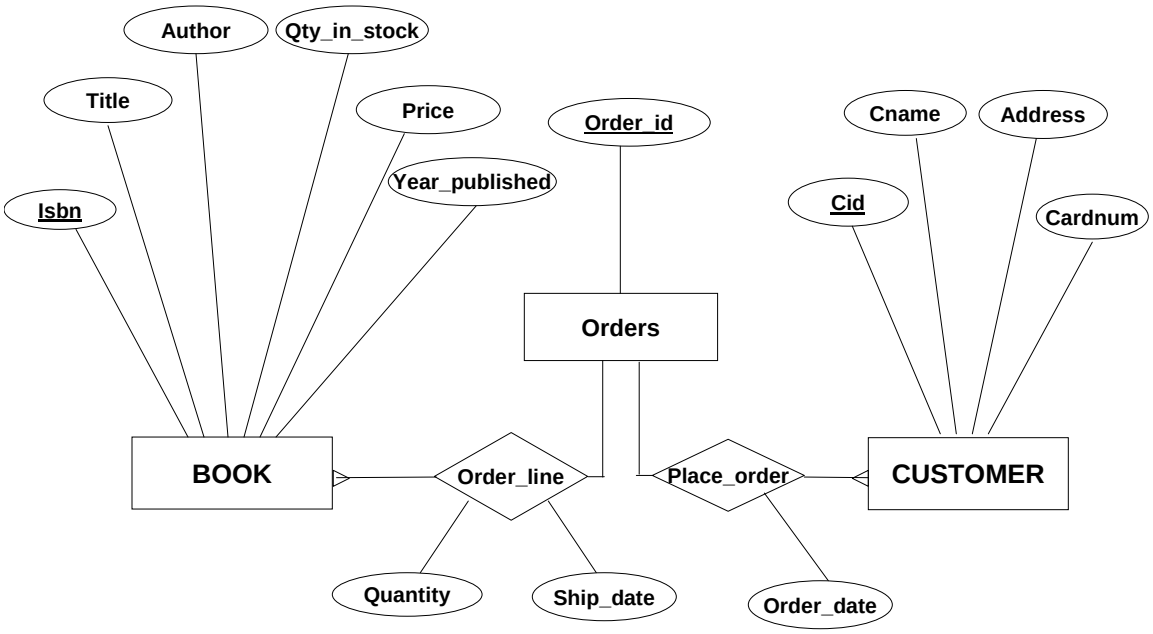
Do vậy chúng ta có thể kết luận rằng quan hệ ORDER không ở dạng chuẩn 3NF. Nên chúng ta phải tiến hành phân rã ORDER thành 2 quan hệ sau:

ORDER(Order_id, Cid, Order_date) và
ORDER_LINE(Order_id, Isbn, so_luong, Ship_date)

Như vậy cả 2 quan hệ trên đều ở dạng chuẩn 3NF. Và đến đây chúng ta có các câu lệnh SQL tạo quan hệ như sau:

```
Create table ORDER (Order_id integer,
        Cid integer,
        Order_date date
        primary key (Order_id),
        foreign key (Cid) references CUSTOMER)
Create table  ORDER_LINE (Order_id integer,
        Isbn char(10),
        Quantity integer,
        Ship_date date
        primary key (Order_id, Isbn),
        foreign key (Order_id) references ORDER,
        foreign key (Isbn) references BOOK)
```

Hình 6.4.1 là sơ đồ thực thể liên kết được sửa đổi theo cách thiết kế mới như trên. Chú ý rằng, chúng ta sẽ có sơ đồ này ngay nếu ngay từ đầu chúng ta coi ORDER là một kiểu thực thể thay vì là một liên kết. Mặc dù vậy, việc chúng ta chưa thể hiểu được đầy đủ các nghiệp vụ ngay từ lần gặp gỡ đầu tiên với khách hàng là một điều hết sức bình thường. Quá trình tinh chỉnh lược đồ này có thể được lặp đi lặp lại nhiều lần, đặc biệt đối với những bài toán lớn có nghiệp vụ phức tạp và đó thực sự là một qui trình phát sinh trong thực tế bởi hầu hết các thiết kế ban đầu đều được làm mịn dần trong quá trình xây dựng dự án.



Hình 6.4.1 Sơ đồ thực thể - liên kết được tinh chỉnh

Sau giai đoạn này chúng ta sẽ chuyển sang việc thiết kế cơ sở dữ liệu vật lý

5. Thiết kế cơ sở dữ liệu vật lý

Chủ cửa hàng sách mong muốn rằng khách hàng của họ sẽ tìm kiếm sách thông qua số ISBN trước khi đặt hàng mua sách. Việc đặt đơn hàng mua sách bao gồm: chèn 1 bản ghi vào quan hệ ORDER và một hoặc nhiều bản ghi vào quan hệ ORDER_LINE. Nếu số lượng sách yêu cầu có đủ trong cửa hàng, công việc giao hàng được chuẩn bị và Ship_date trong quan hệ ORDER_LINE được đặt giá trị tương ứng. Thêm vào đó, số lượng sách trong cửa hàng sẽ thay đổi

theo thời gian, tăng lên khi cửa hàng nhập thêm sách và giảm đi khi khách hàng đặt mua.

Như vậy chúng ta phải bắt đầu bằng việc xem xét đến vấn đề tìm kiếm sách thông qua giá trị ISBN. Vì Isbn là trường khoá nên việc tìm kiếm trên nó sẽ trả lại nhiều nhất là 1 bản ghi. Như vậy để tăng tốc độ tìm kiếm khi khách mua của cửa hàng tìm sách, chúng ta sẽ phải tạo một chỉ số băm không phân cụm (unclustered hash index) trên trường Isbn.

Tiếp theo chúng ta phải cân nhắc đến vấn đề cập nhật số lượng sách. Để cập nhật giá trị Qty_in_stock cho mỗi một đầu sách, đầu tiên phải tìm đến bản ghi đó thông qua giá trị ISBN, tệp chỉ số trên sẽ giúp công việc tìm kiếm nhanh hơn.

Ngoài ra, do khách mua cũng sẽ thường xuyên tìm kiếm sách thông qua tiêu đề hay tên tác giả, nên chúng ta quyết định xây dựng thêm các tệp chỉ số trên 2 trường Author và Title, hơn nữa việc bảo trì các tệp chỉ số này không phức tạp lắm do danh mục đầu sách ít khi thay đổi (chỉ khi có thêm đầu sách mới), mặc dù số lượng sách trong cửa hàng liên tục thay đổi.

Với quan hệ CUSTOMER, mỗi một khách mua được xác định bởi một giá trị định danh duy nhất. Do hầu hết các tìm kiếm trên CUSTOMER là dựa trên các giá trị này nên chúng ta quyết định tạo một tệp chỉ số băm phân cụm trên trường Cid để tăng tốc độ tìm kiếm.

Sang đến quan hệ ORDER, chúng ta nhận thấy rằng có 2 loại truy vấn: chèn thêm các đơn hàng mới và trả về các đơn hàng đã đặt. Cả 2 truy vấn này đều sử dụng thuộc tính Order_id như là khoá tìm kiếm, do vậy chúng ta quyết định xây dựng một tệp chỉ số trên nó. Tuy nhiên phải lựa chọn

giữa hai loại tệp chỉ số là B-tree hay băm. Trong trường hợp này giá trị các đơn hàng tăng theo thời gian, tương ứng với các giá trị trong thuộc tính Order_date, sắp xếp theo Order_id cũng hiệu quả như sắp xếp theo ngày đặt, do vậy chúng ta quyết định lựa chọn loại B-tree phân cụm làm tệp chỉ mục cho trường Order_id.

Trong quan hệ ORDER_LINE, hầu hết các thao tác trên quan hệ này là phép chèn, một số phép cập nhật trên thuộc tính Ship_date, và phép liệt kê danh mục các sách đặt mua của một đơn hàng đã cho. Nếu quan hệ này được sắp xếp theo Order_id, khi đó tất cả các phép chèn đơn giản chỉ là việc thêm vào cuối quan hệ và do đó rất hiệu quả. Một tệp chỉ mục phân cụm B-tree nên được xây dựng cho giá trị này, tệp chỉ số này cũng làm tăng tốc độ cho câu lệnh tìm kiếm liệt kê các sách đã đặt mua của một đơn hàng nào đó. Với phép cập nhật giá trị trong thuộc tính Ship_date, chúng ta cần phải tìm kiếm trên giá trị của 2 trường là Order_id và Isbn. Tệp chỉ số trên Order_id xây dựng phía trên cũng hữu ích cho phép tìm kiếm này, tuy nhiên nếu chúng ta có một tệp chỉ số trên cả 2 thuộc tính thì vẫn sẽ nhanh hơn.

Một số điều chỉnh đối với cơ sở dữ liệu:

Giả thiết sau khi cơ sở dữ liệu được đưa vào sử dụng và khách hàng bắt đầu đặt mua sách qua trang web, sau vài tháng, chúng ta nhận được một số phàn nàn của cửa hàng rằng khách mua của họ thấy việc xử lý các đơn đặt hàng của họ ngày càng chậm. Mặc dù các thao tác vẫn được thực hiện thành công, nhưng do số lượng bản ghi trong các quan hệ ORDER và ORDER_LINE ngày càng nhiều và các quan hệ này ngày càng lớn so với ban đầu nên ảnh hưởng đến tốc độ xử lý đơn hàng? Nếu điều này xảy ra, chúng ta phải xem

xét lại bản thiết kế. Chúng ta nhận thấy rằng, ở đây, có 2 loại đơn hàng, một loại đơn hàng đã được thực hiện xong tương ứng với các đầu sách trong đơn đã được giao và một loại còn chưa hoặc đang được thực hiện tương ứng với các đầu sách trong đơn hàng chưa hoặc đang được giao. Và hầu hết các khách mua có nhu cầu tìm kiếm trên loại đơn hàng thứ 2 (chưa hoặc đang được giao). Ngoài ra, cũng nhận thấy rằng, đối với loại đơn hàng này thì số lượng rất nhỏ so với loại đơn hàng đã được thực hiện xong. Do vậy chúng ta quyết định tiến hành điều chỉnh thêm đối với 2 quan hệ này, chúng ta tiến hành phân vùng (partition) ngang 2 quan hệ này thành 4 quan hệ mới đó là NEWORDER, OLDORDER, NEWORDER_LINE và OLDORDER_LINE. Rõ ràng, một đơn hàng luôn đi thành từng cặp hoặc cũ hoặc mới tương ứng với nhau. Đến đây chúng ta đã khắc phục được hiện tượng tìm kiếm trên những đơn hàng còn đang được thực hiện giao.

Tuy nhiên, đối với những truy vấn như tìm tất cả các đơn hàng của một khách hàng nào đó thì vẫn chậm, bởi khi đó hệ thống vẫn phải tìm kiếm trên cả 2 loại quan hệ cũ và mới trên, nhưng việc tìm kiếm này là có thể chấp nhận được và nó ít khi xảy ra, thường là các báo cáo tháng, quý, năm của cửa hàng.

6. Vấn đề an toàn bảo mật

Chúng ta đã hoàn tất việc thiết kế vật lý cho cơ sở dữ liệu. Tiếp theo chúng ta phải xem xét đến vấn đề an toàn dữ liệu.

Thông thường, có 3 nhóm người dùng có nhu cầu truy cập cơ sở dữ liệu, đó là: khách mua, nhân viên cửa hàng, và chủ cửa hàng. Ngoài ra, cũng cần có người quản trị hệ thống,

chịu trách nhiệm đối với các hoạt động của cơ sở dữ liệu và có quyền truy nhập vào toàn bộ cơ sở dữ liệu.

- Với chủ cửa hàng, anh ta sẽ có đầy đủ các quyền trên tất cả các quan hệ, tuy nhiên anh ta không được phép tạo dựng thêm các đối tượng dữ liệu mới. Do vậy người quản trị sẽ phân quyền thao tác đầy đủ trên các đối tượng dữ liệu trong cơ sở dữ liệu cho anh ta.
- Với khách mua, họ chỉ được phép tìm kiếm trên quan hệ BOOK và có thể đặt hàng trực tuyến, nhưng họ không được phép truy nhập đến các bản ghi của các khách hàng khác cũng như các đơn đặt hàng khác. Do vậy ở đây chúng ta phải giới hạn theo 2 cách. Cách thứ nhất, phải sử dụng các đặc trưng an toàn của cơ sở dữ liệu để giới hạn phạm vi truy suất đến các dữ liệu nhạy cảm. Ví dụ khi tạo lập một khoản mục mới cho khách hàng, người quản trị cơ sở dữ liệu sẽ tạo dựng luôn những khung nhìn chỉ cho phép anh ta xem các dữ liệu của riêng mình thông qua các khung nhìn đó. Cách thứ 2, chúng ta có thể giới hạn thông qua việc tạo các giao diện trong ứng dụng trên Web. Và mọi truy vấn đều được gắn thêm giá trị Cid của khách mua hiện thời.
- Với các nhân viên của cửa hàng, tùy từng đối tượng mà người quản trị sẽ phân cho họ các quyền thao tác trên từng đối tượng dữ liệu cụ thể. Ví dụ với người quản lý kho, anh ta chỉ được phép cập nhật trên quan hệ BOOK. Với người quản lý các thông tin về khách hàng thì chỉ có thể thao tác trên quan hệ CUSTOMER mà thôi. Còn với những người quản lý các đơn hàng, họ có các quyền cập nhật trên hai quan

hệ NEWORDER và NEWORDERLINE, nhưng với các quan hệ còn lại như BOOK, CUSTOMER thì chỉ có quyền xem mà thôi.

Cụ thể, để hạn chế quyền truy nhập của khách mua, chúng ta có thể tạo một tài khoản có tên là khachmua và gán cho tài khoản này các quyền sau:

```
Select on BOOK, NEWORDER, OLDORDER,
NEWORDER_LINE, OLDORDER_LINE
Insert on NEWORDER, NEWORDER_LINE
```

Với các nhóm nhân viên, như trên đã phân tích, họ có thể có quyền chèn thêm một đầu sách mới, cập nhật lại số lượng sách trong cửa hàng và sửa lại thông tin đặt hàng của khách trong một số trường hợp, cập nhật lại thông tin liên quan đến khách hàng ngoại trừ giá trị tại trường Cardnum. Trên thực tế, nhóm này còn không được phép biết đến các giá trị tại thuộc tính này. Do vậy chúng ta phải tạo thêm các khung nhìn sau:

```
Create view CUSTOMERINFO (Cid, Cname, Address) as
Select Cid, Cname, Address
From CUSTOMER
```

Và cấp các quyền sau cho nhóm nhân viên này:

```
Select on CUSTOMERINFO, BOOK, NEWORDER,
OLDORDER, NEWORDER_LINE, OLDORDER_LINE
Insert on CUSTOMERINFO, BOOK, NEWORDER,
OLDORDER, NEWORDER_LINE, OLDORDER_LINE
Update on CUSTOMERINFO, BOOK, NEWORDER,
OLDORDER, NEWORDER_LINE, OLDORDER_LINE
```

Delete on BOOK, NEWORDER, OLDORDER, NEWORDER_LINE, OLDORDER_LINE

Ngoài ra, do sử dụng trang web trên mạng, nên việc gửi nhận thông tin trên mạng có thể cần được mã hoá và nên sử dụng giao thức SSL (Security Socket Layer).

Tài liệu tham khảo

[1] C.Goble - J.Keane, *Advances in Databases*, Springer , **1997**.

[2] H.David, *Data Analysis for Database Design. Third ed*, Butterworth, **2003**.

[3] J.Teorey, Database Modeling & Design , Morgan Kauf, **1999**.

[4] R.McFadden and al., *Modern Database Management. (5th. Ed.)*, Addison-Wesley, **2000**.

[5] R.Raghu, G.Jahannes, *Database Management Systems*, McGraw-hill, **2000**.

[6] C.J.Date, *Nhập môn các hệ cơ sở dữ liệu* (Hồ Thuần dịch), Nhà xuất bản thống kê, **1986**.

[7] Hồ Thuần, *Đề cương chi tiết giáo trình cơ sở dữ liệu nâng cao*, Bộ GD&ĐT, **1998**.

[8] Nguyễn Kim Anh, *Nguyên lý của các hệ cơ sở dữ liệu*, Nhà xuất bản Đại học quốc gia Hà Nội, **2004**.

[9] Nguyễn Văn Ba, *Phân tích và thiết kế hệ thống thông tin*, Nhà xuất bản Đại học quốc gia Hà Nội, **2002**.

[10] Trần Thành Trai, *Phân tích và thiết kế hệ thống thông tin quản lý*, Nhà xuất bản thống kê, **1994**.

[11] Vũ Đức Thi, *Cơ sở dữ liệu kiến thức và thực hành* , Nhà xuất bản thống kê, **1997**.

[12] http://www.oracle.com

[13] http://metalink.oracle.com

[14] http://www.microsoft.com

[15] http://www.microsoft.com/sql/evaluation/overview/default.asp

[16] http://docs.rinet.ru/

[17] http://www.arch.su.edu.au/kcdc/teaching

KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG
ĐẠI HỌC CẦN THƠ

Giáo trình:

THỰC HÀNH
HỆ CƠ SỞ DỮ LIỆU

Biên soạn: Trần Ngân Bình – Bùi Đăng Hà Phương
Tháng 9/2012

PHẦN 1. TỔNG QUAN

I. MỤC ĐÍCH YÊU CẦU

Giáo trình Thực hành Hệ Cơ Sở Dữ Liệu (CSDL) là một phần của môn Hệ CSDL nhằm giúp sinh viên nắm bắt được nội dung lý thuyết về ngôn ngữ SQL, và thực hiện một cách thành thạo các câu lệnh truy vấn trên cơ sở dữ liệu.

Các hệ quản trị CSDL được sử dụng phổ biến hiện nay đều hỗ trợ ngôn ngữ SQL theo chuẩn ít nhất là SQL-92. Ngoài những tính năng mà chuẩn đã nêu, mỗi phần mềm hệ quản trị CSDL có thể có những tính năng riêng thêm vào.

Phần mềm được chọn để thực hành là hệ quản trị CSDL SQL Server 2005. Trong 30 tiết (6 buổi) thực hành, sinh viên sẽ được rèn luyện từng dạng câu hỏi truy vấn thông qua các bài tập cụ thể trên các cơ sở dữ liệu cụ thể.

Sau khi học xong môn này, sinh viên phải:

- Nhận biết được sự thông thương giữa các bảng trong một cơ sở dữ liệu. Để từ đó biết cách kết nối các bảng để truy xuất dữ liệu theo yêu cầu.
- Hiểu được câu hỏi truy vấn để từ đó có thể chọn lựa dạng câu hỏi truy vấn phù hợp.
- Vận dụng được các dạng câu truy vấn khác nhau để trả lời câu hỏi như: câu truy vấn lồng nhau, câu truy vấn cần gom nhóm các dòng, câu truy vấn có sử dụng các hàm hàm xử lý chuỗi, hàm xử lý ngày tháng, hàm chuyển kiểu, ...

II. NỘI DUNG CỐT LÕI

Tài liệu hướng dẫn gồm 3 phần:

- *Phần 1:* giới thiệu tổng quan về môn học, trong đó trình bày cụ thể các kiến thức, kỹ năng mà sinh viên cần đạt được sau khi học môn này. Ngoài ra, còn trình bày sơ lược về nội dung của tài liệu, kiến thức tiên quyết và phương pháp học tập.
- *Phần 2:* giới thiệu các nét cơ bản về hệ quản trị SQL Server 2005. Ngoài ra, trong phần này, giáo trình cũng trình bày một cách chi tiết các thao tác cần tiến hành để có thể thực hành.
- *Phần 3:* gồm 5 bài thực hành cụ thể. Mỗi bài sẽ có bố cục gồm 4 phần như sau:

1. Mục tiêu: nêu rõ mục tiêu rèn luyện của bài.
2. Lý thuyết: Nhắc lại một cách ngắn gọn câu lệnh SELECT cần dùng trong bài.
3. Bài tập có hướng dẫn: Gồm một số câu hỏi truy vấn trên một CSDL cho trước. Mỗi câu hỏi sẽ được phân tích để đi đến câu lệnh Select. Những điểm cần lưu ý sẽ được nhấn mạnh. Ngoài ra, một số câu hỏi nhỏ sẽ được đưa ra trong tình huống của câu hỏi giúp sinh viên hiểu rõ hơn vấn đề.
4. Bài tập tự làm: gồm một số câu hỏi truy vấn trên một CSDL khác với phần bài tập có hướng dẫn.

Trong 4 bài đầu tiên, mỗi bài sẽ tập trung rèn luyện một dạng câu hỏi truy vấn. Bài thực hành thứ 5 là bài tập tổng hợp gồm tất cả các dạng câu hỏi đã học ở các bài trước, vì vậy sinh viên sẽ tự thực hiện mà không có phần hướng dẫn. Mỗi buổi thực hành sẽ tương ứng với một bài trong tài liệu hướng dẫn này.

III. KIẾN THỨC TIÊN QUYẾT

Là một phần của môn học Hệ CSDL, phần thực hành này được đưa vào giảng dạy cho sinh viên với yêu cầu sinh viên đã học xong chương SQL của môn Hệ CSDL.

IV. PHƯƠNG PHÁP HỌC TẬP

Với mục tiêu nâng cao khả năng tự học tập và tự nghiên cứu của sinh viên, người soạn đã cố gắng lồng ghép vào nội dung các ví dụ minh họa đơn giản, cụ thể; đồng thời bố trí bố cục với mong muốn tạo sự dễ hiểu cho sinh viên và người đọc.

Để học tốt môn học này, sinh viên cần phải:

- ✓ Trước mỗi buổi thực hành, sinh viên cần dành khoảng 3 giờ tự học để:
  - Xem lại phần lý thuyết của câu lệnh sẽ thực hành (phần 1 và 2 của mỗi bài + tài liệu tham khảo nếu cần thiết)
  - Tự làm bài tập có hướng dẫn tại nhà. Sinh viên nên tự suy nghĩ tìm ra đáp án cho các câu hỏi bài tập trước khi nhìn vào đáp án của giáo viên cung cấp.
- ✓ Trong mỗi buổi thực hành, sinh viên thực hiện phần bài tập tự làm. Nếu không kịp, sinh viên tự sắp xếp thời gian để hoàn tất bài tập của buổi trước trước khi tiến hành bài thực hành kế tiếp.
- ✓ Khi thực thi câu lệnh, nếu hệ thống báo lỗi, SV cần phải đọc thông báo lỗi, tìm hiểu ý nghĩa và cách giải quyết lỗi đó. Một số lỗi thông dụng được giải thích tại trang 13.

MỤC LỤC

I. MỤC ĐÍCH YÊU CẦU 1
II. NỘI DUNG CỐT LÕI 1
III. KIẾN THỨC TIÊN QUYẾT 1
IV. PHƯƠNG PHÁP HỌC TẬP 2
PHẦN 2. CƠ BẢN VỀ SQL Server 2005 5
I. GIỚI THIỆU SƠ LƯỢC VỀ SQL SERVER 2005. 5
1. Kết nối vào Server: 5
2. Cách thức lưu trữ Cơ sở dữ liệu trong SQL Server: 6
3. Chọn hay mở CSDL làm việc: 6
4. Các kiểu dữ liệu thông dụng trong SQL: 6
5. Quan sát CSDL bằng Object Browser: 7
6. Soạn thảo và thực thi câu truy vấn: 9
7. Một số hàm thông dụng trong SQL Server: 10
II. CÁC LỖI THƯỜNG GẶP TRONG SQL 13
1. Tên cột không tìm thấy trong bảng: 13
2. Tên bảng không tìm thấy: 13
3. Lỗi cú pháp: 13
4. Tên cột có mặt ở nhiều bảng. 13
5. Phép so sánh không tương thích kiểu. 13
III. HƯỚNG DẪN THỰC HÀNH 14
1. Mở Query Analyzer đăng nhập vào SQL Server trên máy chủ. 14
2. Quan sát CSDL: 14
3. Mở CSDL muốn truy vấn: 14
4. Soạn thảo và thực hiện câu truy vấn. 14
5. Quan sát kết quả và kiểm chứng. 14
6. Lưu lại các câu truy vấn vào tập tin *.sql. 14
IV. CÁCH LƯU và XÓA BẢNG TẠM: 14
V. SAO LƯU/ PHỤC HỒI CSDL: 14
1. Sử dụng lệnh: 14
2. Sử dụng giao diện chương trình: 15
PHẦN 3. BÀI TẬP THỰC HÀNH 20
BÀI 1:
I. MỤC ĐÍCH 21
II. LÝ THUYẾT 21
III. BÀI TẬP CÓ HƯỚNG DẪN: 21
IV. BÀI TẬP TỰ LÀM: 25
BÀI 2:
I. MỤC ĐÍCH 27
II. LÝ THUYẾT 27
III. BÀI TẬP CÓ HƯỚNG DẪN: 28
IV. BÀI TẬP TỰ LÀM: 32
BÀI 3:
I. MỤC ĐÍCH: 34
II. LÝ THUYẾT: 34
III. BÀI TẬP CÓ HƯỚNG DẪN 34
IV. BÀI TẬP TỰ LÀM 37

BÀI 4:
I. MỤC ĐÍCH: 38
II. BÀI TẬP CÓ HƯỚNG DẪN: 38
III. BÀI TẬP TỰ LÀM 41
BÀI 5:
I. MỤC ĐÍCH: 43
II. LÝ THUYẾT: 43
III. BÀI TẬP CÓ HƯỚNG DẪN: 44
IV. BÀI TẬP TỰ LÀM: 46
BÀI 6:
I. MỤC ĐÍCH: 47
II. CSDL: QUẢN Lý DỰ Án (CSDL6) 47
III. CÂU HỎI: 47

TÀI LIỆU THAM KHẢO

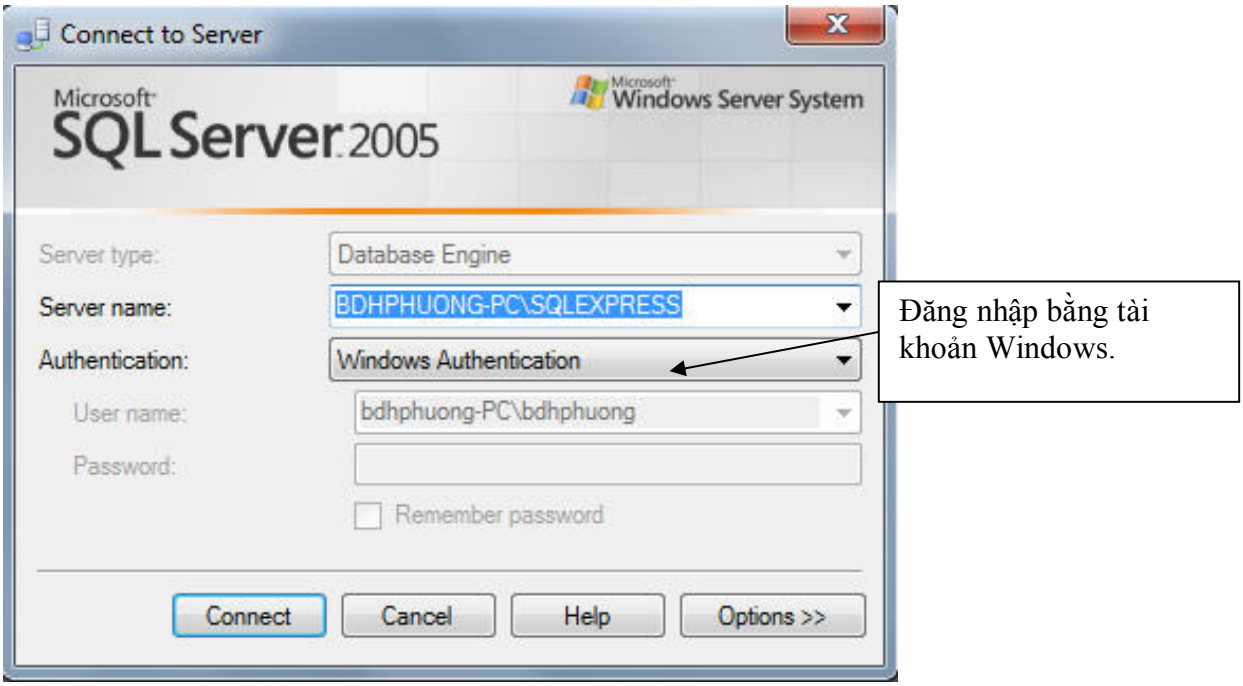
[1] **Giáo Trình CSQL -** Đinh Khắc Quyền. Khoa Công Nghệ Thông Tin, 2005
[2] **SQL Tutorial:** http://www.w3schools.com/sql/
[3] **SQL Server 2000 Book Online**

PHẦN 2. CƠ BẢN VỀ SQL SERVER 2005

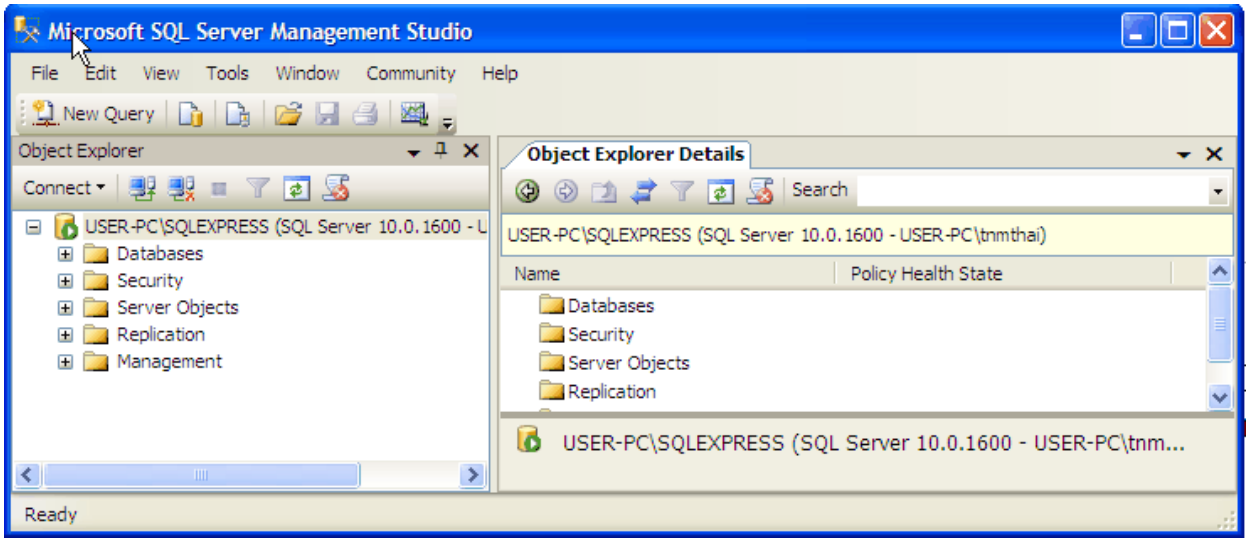
I. GIỚI THIỆU SƠ LƯỢC VỀ SQL SERVER 2005.

1. Kết nối vào Server:

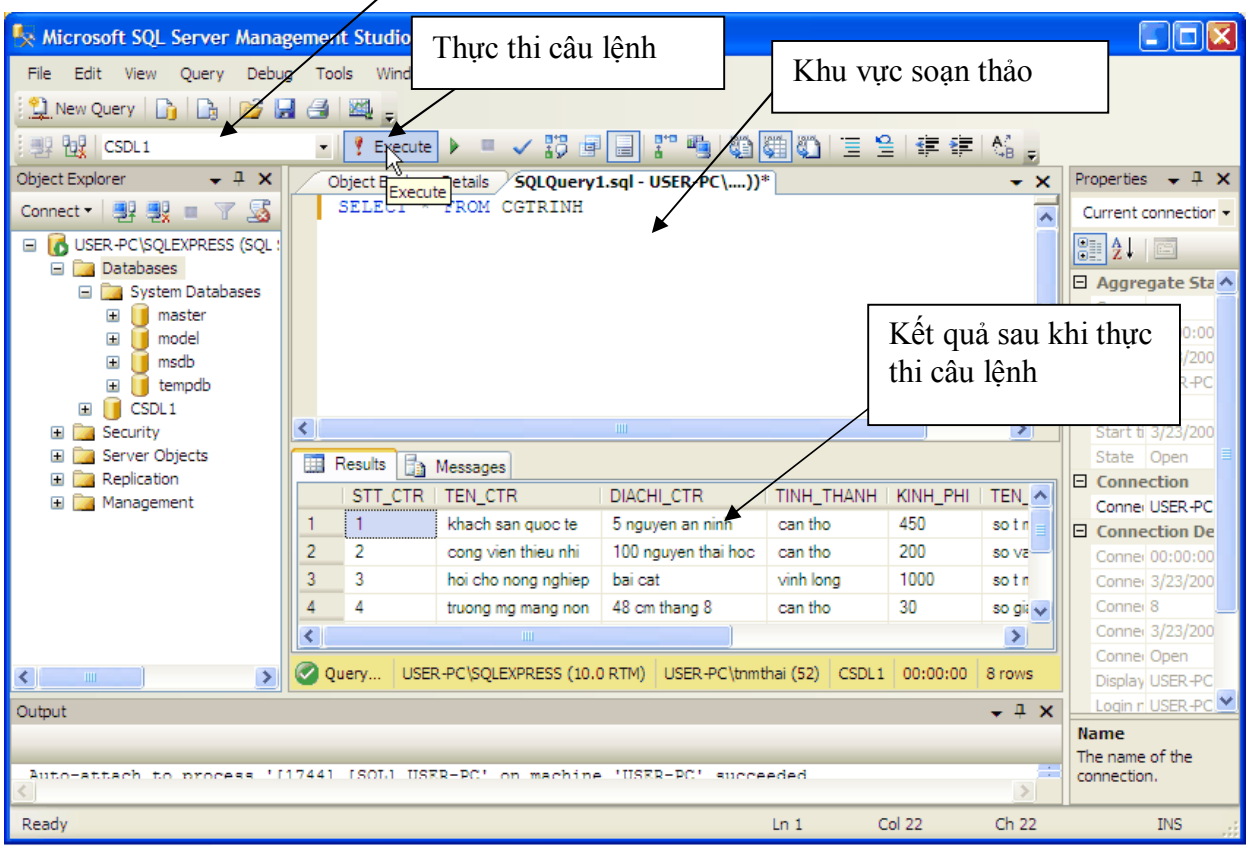
Chọn **Star/All Programs/Microsoft SQL Server 2005/ SQL Server Management Studio Express**. Cửa sổ đăng nhập sẽ xuất hiện như hình 1:



Sau khi chọn **Connect**, cửa sổ **SQL Server Management Studio** sẽ xuất hiện như hình 2:



Để soạn thảo và thực thi câu lệnh, click vào ***New Query*** như hình 3.



2. Cách thức lưu trữ Cơ sở dữ liệu trong SQL Server:

Một CSDL SQL Server bao gồm:

- 1 hoặc nhiều data file: 1 file dữ liệu chính (*.mdf) và không hoặc nhiều file dữ liệu phụ (*.ndf).
- 1 hoặc nhiều log file (*.ldf): đây chính là nhật ký giao tác trên CSDL.

SQL Server quản lý cùng lúc nhiều CSDL, chia làm hai loại:

- CSDL hệ thống (system databases): dùng để lưu trữ thông tin quản lý toàn bộ hệ thống SQL Server. Có bốn CSDL hệ thống, đó là: master, model, tempdb, msdb
- CSDL người dùng (user databases): là CSDL do người dùng tạo ra. Có 2 CSDL mẫu thông dụng trong SQL Server là: Northwind và pubs

3. Chọn hay mở CSDL làm việc:

Vì hệ thống quản lý nhiều CSDL nên khi làm việc với SQL Server, điều đầu tiên là phải chọn CSDL nào để làm việc trên đó. Có 2 cách:

Cách 1: Nhấp chọn CSDL từ danh sách xổ xuống của combo box trên thanh công cụ (xem hình 2)

Cách 2: Thực thi lệnh: USE <Tên CSDL>

4. Các kiểu dữ liệu thông dụng trong SQL:

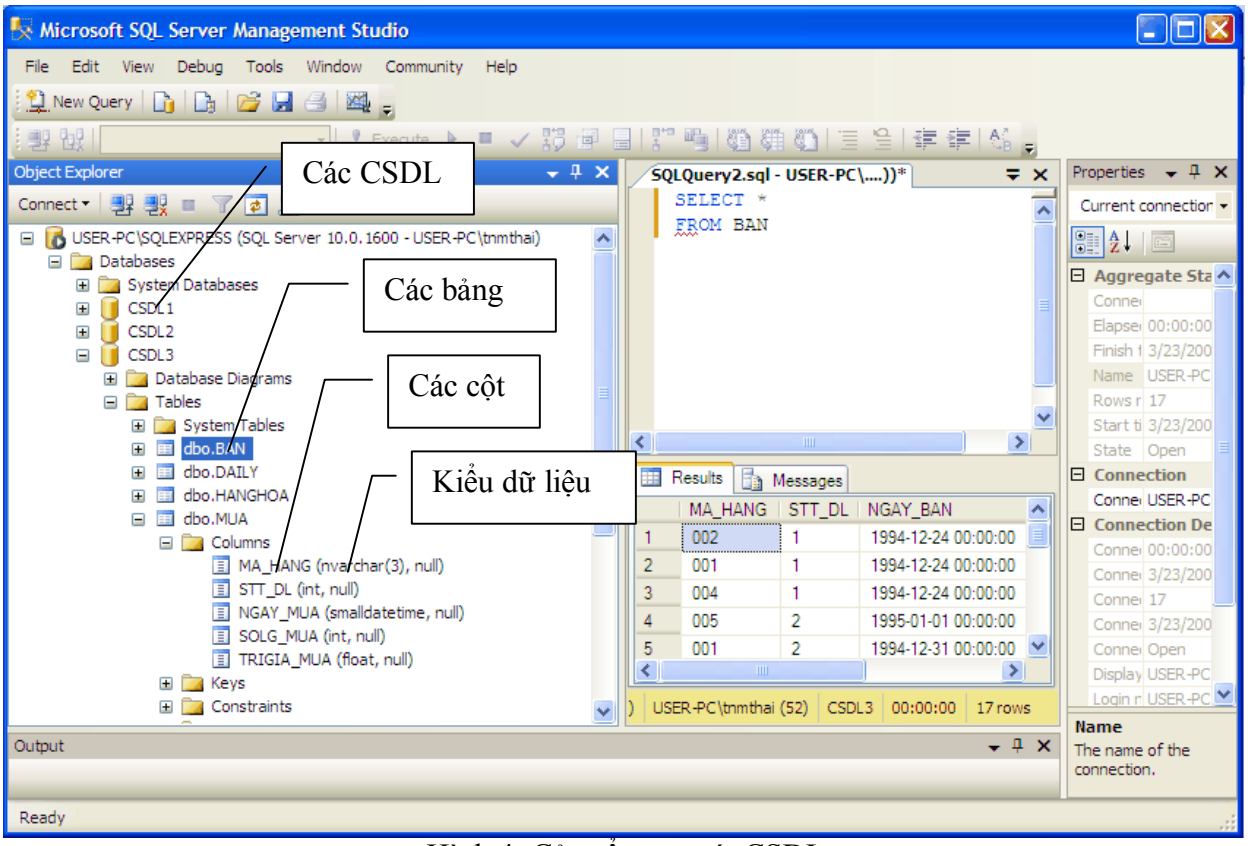
Tên kiểu	Giải thích	Biểu diễn hằng
Bit	Số nguyên 0 hoặc 1	`0, 1`
Int	Số nguyên từ -2^{31} (-2.147.483.648) đến 2^{31} - 1 (2.147.483.647)	`-1000, 23455`
SmallInt	Số nguyên từ -2^{15} (- 32.768) đến 2^{15} - 1	`32124, -764`

	(32.767)	
TinyInt	Số nguyên từ 0 đến 255	31, 45
Decimal/ Numeric	Các số thập phân từ -10^{38} - 1 đến 10^{38} - 1	1894.1204
Money	Tiền tệ từ -2^{63} đến 2^{63} - 1	234, -87.65 $12, $542023.14
SmallMoney	Tiền tệ từ -214.748,3648 đến 214.748,3647	-$45.56 -34.54
Float	Số thực từ -1,79 E +308 đến 1,79 E +308	101.5E5 0.5E-2
Real	Số thực từ -3.40 E + 38 đến 3.40E + 38	245.21E-10
DateTime	Ngày giờ từ 1/1/1753 đến 31/10/9999	'April 15, 1998' , '15 April, 1998' , '980415' , '04/15/98'
SmallDateTime	Ngày giờ từ 1/1/1900 đến 06/6/2079	như DateTime
Char	Ký tự có độ dài ô nhớ cố định và tối đa là 8000 kí tự (không hỗ trợ Unicode)	'50% complete.' 'O''Brien' "O'Brien"
nchar	Ký tự có độ dài ô nhớ cố định và tối đa là 4000 kí tự (hỗ trợ Unicode)	'Nguyễn Văn Minh' 'Lương Tâm'
varchar	Ký tự có độ dài ô nhớ không cố định và tối đa là 8000 kí tự (không hỗ trợ Unicode)	'50% complete.' 'O''Brien' "O'Brien"
nvarchar	Ký tự có độ dài ô nhớ không cố định và tối đa là 4000 kí tự (hỗ trợ Unicode)	'Nguyễn Văn Minh' 'Lương Tâm'

5. Quan sát CSDL bằng Object Browser:

a. Quan sát lược đồ CSDL:

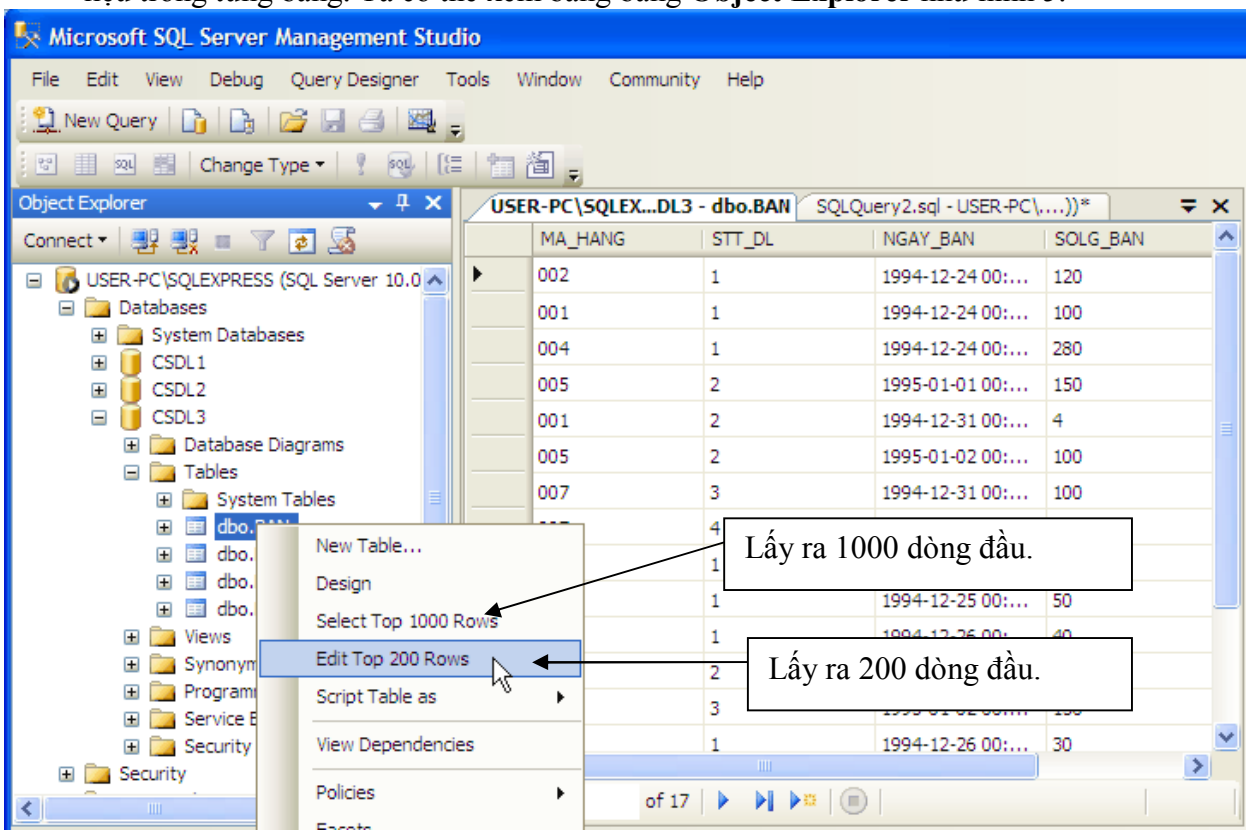
Để có thể truy vấn trên một CSDL nào đó, điều quan trọng là ta phải biết lược đồ của CSDL đó. Ta có thể quan sát lược đồ của các CSDL mà SQL Server đang quản lý bằng cách nhấp vào **View -> Object Explorer**. Khi đó sẽ có một cửa sổ dọc Object Explorer xuất hiện ở bên trái như hình 3 cho phép ta quan sát cấu trúc của các bảng trong CSDL như sau:



Hình 4: Cửa sổ quan sát CSDL.

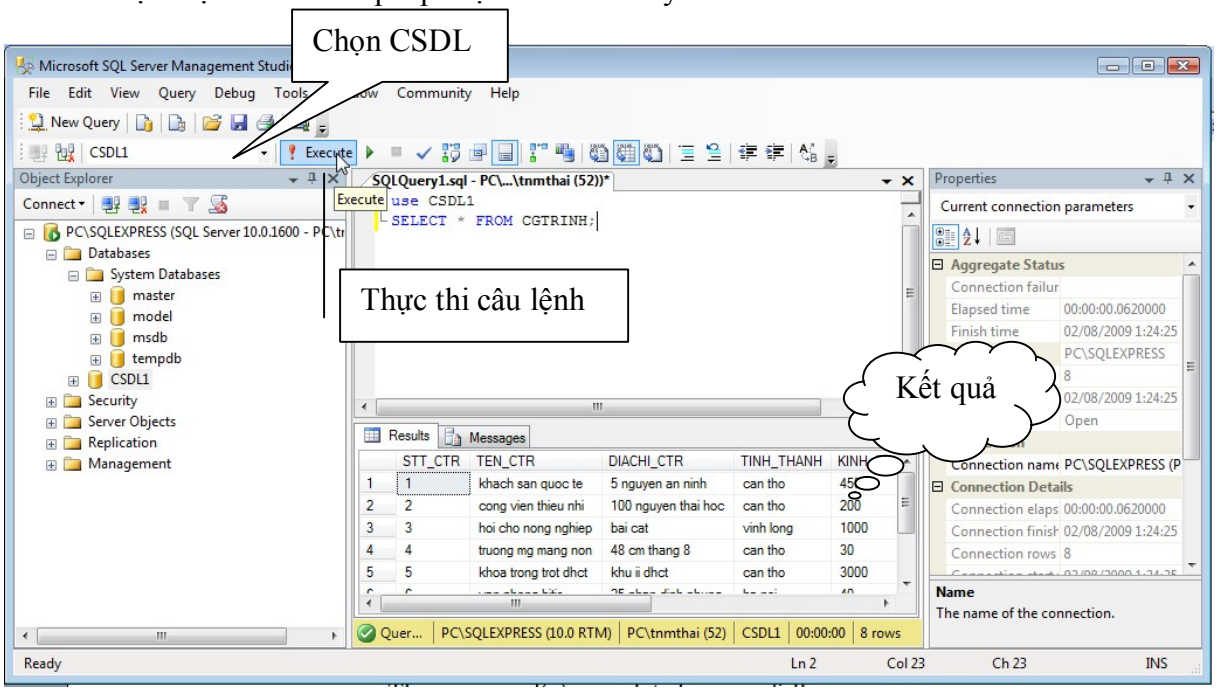
b.Mở bảng dữ liệu:

Để có thể truy vấn được chính xác, ngoài việc quan sát lược đồ CSDL, ta còn phải quan sát dữ liệu trong từng bảng. Ta có thể xem bảng bằng **Object Explorer** như hình 5.



6. Soạn thảo và thực thi câu truy vấn:

Nếu chưa kết nối với SQL Server thì ta không thể thực hiện truy vấn. Để kết nối, ta vào menu **File -> Connect Object Explorer**. Khi đó cửa sổ kết nối sẽ xuất hiện như hình 1 ở trang 4. Sau khi kết nối thì xuất hiện một cửa sổ cho phép soạn thảo câu truy vấn như hình 6.



Trong quá trình thao tác, có thể mở thêm nhiều tập tin truy vấn bằng cách vào menu File -> New hoặc click vào biểu tượng trên thanh công cụ.

a. Kiểm tra cú pháp câu lệnh:

Sau khi gõ câu lệnh vào phần nhập câu lệnh, có thể kiểm tra cú pháp câu lệnh bằng cách nhấp vào biểu tượng ☑ trên thanh công cụ (xem hình 5) hoặc nhấn Ctrl + F5. Nếu câu lệnh không có lỗi cú pháp thì cửa sổ nhỏ bên phải dưới sẽ hiển thị câu:

The command(s) completed successfully.

Nếu câu lệnh có lỗi cú pháp thì lỗi sẽ được hiển thị.

b. Thực thi câu lệnh:

Thực thi tất cả các câu lệnh trong tập tin script hiện hành bằng cách nhấp vào nút ! trên thanh công cụ (xem hình 5) hoặc nhấn F5.
Nếu chỉ muốn thực thi một hoặc một số câu lệnh nào thôi thì bôi đen các lệnh đó và nhấn F5.

c. Ghi chú trong SQL Query:

Khi cần ghi chú thích hoặc tạm đóng không thực thi một đoạn lệnh nào đó, ta có thể sử dụng kí hiệu ghi chú như sau:
Ghi chú một dòng: Đặt dấu -- trước dòng ghi chú
Ghi chú nhiều dòng: Đặt ghi chú trong cặp dấu /* ... */

d. Lưu lại truy vấn trong tập tin lệnh (script): (*.sql)

Sau khi nhập vào các câu truy vấn (như hình 5), các câu lệnh truy vấn có thể được lưu lại dưới dạng file text (*.sql) bằng cách chọn File > Save hoặc nhấn Ctrl + S.
Tập tin này sau khi lưu vào đĩa có thể được mở lên lại để thực thi bằng cách vào File > Open.

7. Một số hàm thông dụng trong SQL Server:

a. Xử lý chuỗi:

> ***Lưu ý***: *SQL Server không phân biệt chữ hoa và chữ thường. Vì vậy:*
> *điều kiện: lower(ten_ctr)='khach san QUOC TE' => vẫn cho ra kết quả*

Một số hàm xử lý chuỗi thông dụng:

- **LEN**(<chuỗi>): Trả về chiều dài chuỗi
- **LOWER**(<chuỗi>): đổi <chuỗi> sang dạng chữ thường

  Ví Dụ: Select LOWER("Tran PHI phap") => "tran phi phap"
- **UPPER**(<chuỗi>): đổi <chuỗi> sang dạng chữ hoa

  Ví Dụ: Select UPPER("Tran PHI phap") => "TRAN PHI PHAP"
- **LEFT**(<chuỗi>, <số n>): trả về chuỗi mới gồm n ký tự bên trái của <chuỗi>

  Ví Dụ: Select LEFT('Tin Hoc', 3) => 'Tin'
- **RIGHT**(<chuỗi>, <số n>): trả về chuỗi mới gồm n ký tự bên phải của <chuỗi>
- **SUBSTRING**(<chuỗi>, <số n> ,<số m>): trả về chuỗi con bằng cách lấy từ <chuỗi> m ký tự từ vị trí n.

  Ví Dụ: Select SUBSTRING('Tin Mới Học', 5, 3) => 'Mới'
- **STUFF**(<chuỗi 1>, <số n>, <số m>, <chuỗi 2>): thay m ký tự trong <chuỗi 1> từ vị trí n bằng <chuỗi 2>.

  Ví Dụ: Select STUFF ('Tin Học', 5, 0, 'Mới ') => 'Tin Mới Học' -- xen vào

  Select STUFF('Tin Hãy Học', 5, 3, 'Mới') => 'Tin Mới Học' -- thay thế

  Select STUFF('Tin Hãy Học', 5, 4, '') => 'Tin Hoc' -- xóa
- **CHARINDEX** (<chuỗi 1> , <chuỗi 2> [, <số n>]): trả về vị trí xuất hiện lần đầu tiên của chuỗi 1 trong chuỗi 2, bắt đầu tìm từ vị trí thứ n của chuỗi 2.

  Ví Dụ:Select CHARINDEX ('qua', 'noi qua ma hong qua') => 5

  Select CHARINDEX ('qua', 'noi qua ma hong qua', 6) => 17
- **PATINDEX** (<'%mẫu tìm kiếm%'> , <chuỗi>): trả về vị trí bắt đầu xuất hiện mẫu lần đầu tiên trong chuỗi, nếu không tìm thấy mẫu thì trả về 0.

  Ví Dụ :Select PATINDEX('%om%', 'lom khom duoi nui tieu vai chu') => 2

  Select PATINDEX('%__om %', 'lom khom duoi nui tieu vai chu') => 5

b. Xử lý ngày

> ***Lưu ý***: *- Kiểu DATETIME của SQL Server bao gồm nhiều thành phần: ngày / tháng / năm/ giờ / phút / giây / phần triệu của giây .*
> *- Để tránh nhầm lẫn: nên mô tả năm bằng 4 chữ số, mô tả tháng bằng tên tắt của tháng (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec).*

- **GETDATE()**: Cho kết quả là ngày hiện hành.
- **DATEPART**(<thành phần>, <ngày>): Trả về giá trị của <thành phần> trong <ngày>.

Các thành phần thông dụng của ngày gồm có:

Thành phần	Viết tắt	Ý nghĩa
Year	yy, yyyy	Năm
Quarter	qq, q	Quí
Month	mm, m	Tháng
Dayofyear	dy, y	Ngày của năm (1-> 366)
Day	dd, d	Ngày của tháng (1 -> 31)
Week	wk, ww	Tuần trong năm (1 -> 52)
WeekDay	Dw	Ngày trong tuần (1 -> 7)
Hour	hh	Giờ (1 -> 24)

Ví Dụ: Select DATEPART (year, GETDATE()) => 2006

Select DATEPART(week, '20 Aug, 2006') => 34

- **DATEADD** (<thành phần>,<số n>, <ngày>): cộng vào <thành phần> của <ngày> một số n.

Ví Dụ: Select DATEADD (week, 5, '20 Aug, 2006') => 2006-09-24

Cho biết 10 ngày sau kể từ ngày 05/08/2006 là ngày thứ mấy:

Select DATEPART (weekday, DATEADD (day, 10, '5 Aug, 2006')) => 3(thứ ba)

- **DATEDIFF** (<thành phần> , <ngày bắt đầu>, <ngày kết thúc>): trả về số khác biệt giữa 2 thành phần của ngày bắt đầu và ngày kết thúc.

Ví Dụ: Cho biết từ ngày 05/08/2006 đến ngày 11/9/2006 có bao nhiêu ngày:

Select DATEDIFF (day, '5 Aug, 2006', '11 Sep, 2006') => 37 ngày

Select DATEDIFF (hour, '5 Aug, 2006', '11 Sep, 2006') => 888 giờ

- **DATENAME**(<thành phần>, <ngày>): Trả về tên của <thành phần> trong <ngày>.

Ví Dụ: Select DATENAME (month, '5 Aug, 2006') => August

Select DATENAME (weekday, '5 Aug, 2006') => Saturday

c. Các hàm xử lý số

- **FLOOR**(<số>): Cho kết quả là số nguyên gần nhất nhỏ hơn hoặc bằng <số>

Ví Dụ: SELECT FLOOR(123.45), FLOOR(-123.45), FLOOR($123.45)

=> 123 -124 123.0000

- **CEILING**(<số>): Cho kết quả là số nguyên gần nhất lớn hơn hoặc bằng <số>

Ví Dụ: SELECT CEILING(123.45), CEILING(-123.45), CEILING($123.45)

=> 124 -123 124.0000

d. Hàm chuyển đổi kiểu

- CAST (<dữ liệu> AS <kiểu>): Trả về <dữ liệu> với <kiểu> mới.

Ví Dụ: SELECT 10 + cast ('34.5' as float) => 44.5

SELECT right(CAST(124 AS char(4)) , 2) => 4

e. Cấu Trúc Phân Nhánh CASE:

Đánh giá danh sách các điều kiện và trả về biểu thức phù hợp.
CASE có hai dạng:

- Dạng đơn giản: So sánh hai biểu thức để xác định kết quả trả về.

```
CASE <tên cột>
    WHEN <giá trị 1> THEN <kết quả 1>
    [WHEN <giá trị 2> THEN <kết quả 2> ...]
    [
            ELSE <kết quả ngoại lệ>
    ]
END
```

- Dạng tìm kiếm: Đánh giá các điều kiện để xác định kết quả trả về.

```
CASE
      WHEN <biểu thức điều kiện 1> THEN <kết quả trả về 1>
      [WHEN <biểu thức điều kiện 2> THEN <kết quả trả về 2> ...]
      [
          ELSE <kết quả trả về ngoại lệ>
      ]
END
```

Ví Dụ:

```
Select CASE
        When 10 > 2 THEN 'DUNG'
        When 10 = 3 THEN 'SAI'
    END
```

=> DUNG

II. CÁC LỖI THƯỜNG GẶP TRONG SQL

1. Tên cột không tìm thấy trong bảng:

Thông báo lỗi: Invalid column name <tên cột>.

Khi gặp lỗi này hãy xem lại tên cột (còn gọi là tên thuộc tính/ tên trường) trong câu lệnh đã đúng chưa.

2. Tên bảng không tìm thấy:

Thông báo lỗi: Invalid object name <tên bảng>.

Khi gặp lỗi này hãy xem lại tên bảng trong câu lệnh đã đúng chưa.

3. Lỗi cú pháp:

Thông báo lỗi: Incorrect Syntax near <từ gần lỗi nhất>.

Khi gặp lỗi này hãy xem lại **cú pháp** trong câu lệnh đã đúng chưa, có thể sai từ khóa (gỏ sai các từ SELECT, FROM, WHERE, ..), hoặc thiếu dấu ngoặc của hàm hoặc của select con, ...

4. Tên cột có mặt ở nhiều bảng

Thông báo lỗi: Ambiguous column name <tên cột>

Khi gặp lỗi này hãy xem lại <tên cột> gây lỗi trong câu lệnh đã có đặt tên bảng hoặc bí danh đứng trước chưa. Cú pháp: <tên bảng hoặc bí danh>.MA_HANG

5. Phép so sánh không tương thích kiểu

Thông báo lỗi: Error converting data type ...

hoặc Syntax error converting ...

Khi gặp lỗi này hãy xem lại trong câu lệnh đã có phép toán so sánh nào đó không tương thích kiểu dữ liệu.

III. HƯỚNG DẪN THỰC HÀNH

1. Mở Query Analyzer đăng nhập vào SQL Server trên máy chủ.

2. Quan sát CSDL:

Sử dụng Object Browser để xem cấu trúc của các bảng cũng như dữ liệu của từng trường và mở từng bảng dữ liệu để quan sát như hướng dẫn ở mục 5 ở trang 7 .

Khi quan sát CSDL cần lưu ý:

- Sự liên thông giữa các bảng dữ liệu. Tốt nhất là vẽ sơ đồ liên thông lên giấy nháp để có thể tham khảo trong quá trình truy vấn thông tin từ nhiều bảng.
- Kiểu dữ liệu của từng trường, để đặt điều kiện trên các cột cho đúng kiểu hoặc áp dụng các hàm xử lý dữ liệu phù hợp.

3. Mở CSDL muốn truy vấn:

Như đã hướng dẫn ở mục 3 ở trang 6

4. Soạn thảo và thực hiện câu truy vấn.

Như đã hướng dẫn ở mục **Error! Reference source not found.** ở trang **Error! Bookmark not defined.**

5. Quan sát kết quả và kiểm chứng.

Cần phải kiểm tra lại kết quả truy vấn xem có đúng hay không bằng cách mở lại các bảng để kiểm tra xem kết quả đạt được có đúng như mong muốn.

6. Lưu lại các câu truy vấn vào tập tin *.sql.

IV. CÁCH LƯU và XÓA BẢNG TẠM:

Trong khi thực hiện một số truy vấn, thỉnh thoảng ta cần lưu lại những kết quả truy vấn vào một bảng nào đó, thì ta dùng mệnh đề INTO sau mệnh đề SELECT và trước FROM.

Ví dụ: Select * **Into NV** From NhanVien;

Sau khi thực thi câu lệnh này, trong CSDL sẽ xuất hiện thêm một bảng NV lưu kết quả của câu truy vấn.

Trong trường hợp ta chỉ muốn bảng này tồn tại cho đến hết phiên làm việc (session) hiện tại thôi, thì ta có thể lưu vào bảng tạm (là những bảng có tên bắt đầu bằng dấu #), các bảng này sẽ được tạo trong CSDL tempdb của hệ thống. Khi client ngưng kết nối với Server thì các bảng này sẽ tự động bị xóa.

Ví Dụ:
```
Select STT_CTR, count(*) As TongSoCN  Into #CAU3E
From THAMGIA
Group by STT_CTR
```

Trong khi client vẫn còn đang kết nối thì những bảng tạm này vẫn còn trong CSDL tempdb. Vì vậy, sau khi chạy xong câu ví dụ trên, nếu ta chạy lại câu lệnh này thì SQL sẽ báo lỗi là bảng tạm này đã tồn tại. Nên ta phải xóa bảng tạm này trước khi muốn chạy lại câu lệnh trên bằng lệnh xóa bảng DROP TABLE <tên bảng>.

VD: DROP TABLE #CAU3E

V. SAO LƯU/ PHỤC HỒI CSDL:

Có thể chép CSDL từ máy này sang máy khác bằng cách sử dụng lệnh sao lưu / phục hồi CSDL của SQL Server. Có thể thực hiện bằng lệnh hoặc giao diện.

1. Sử dụng lệnh:

a. Sao lưu CSDL:

Để sao lưu một CSDL ta thực hiện các bước sau:

Bước 1. Mở CSDL muốn sao lưu bằng lệnh:

USE <Tên CSDL>

Bước 2. Tạo thiết bị sao lưu gắn với một tập tin sao lưu bằng lệnh:

EXEC sp_addumpdevice 'disk' , '<tên thiết bị>', '<đường dẫn và tên tập tin>'

Bước 3. Ra lệnh sao lưu CSDL vào thiết bị sao lưu:

BACKUP DATABASE <tên CSDL muốn chép> TO <Tên thiết bị>

VD: Sao lưu CSDL có tên là CSDL1 vào ổ đĩa D và đặt tên là CSDL1_backup.bak:

Mở CSDL cần chép:

USE CSDL1

Tạo thiết bị sao lưu tên BAK1 gắn với tập tin D:\CSDL1_backup.bak

EXEC sp_addumpdevice 'disk', 'BAK1', 'D:\CSDL1_backup.bak'

Ra lệnh sao lưu CSDL1 vào thiết bị sao lưu vừa tạo BAK1

BACKUP DATABASE CSDL1 TO BAK1

Lưu ý: *Không thể tạo hai thiết bị sao lưu cùng tên. Nghĩa là nếu đã tạo thiết bị BAK1 rồi thì những lần sao chép sau không cần phải thực hiện bước 2 nữa.*

b. Phục hồi CSDL từ tập tin đã sao lưu:

Giả sử ta muốn chép CSDL1 về máy nhà (đã có cài SQL Server). Thì ta sẽ thực hiện các bước sau:

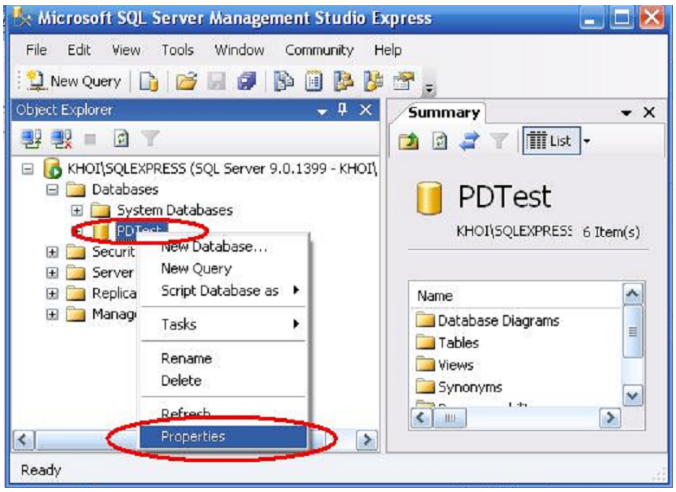
- Thực hiện sao lưu CSDL1 như đã hướng dẫn ở mục trên.
- Chép tập tin sao lưu về máy nhà.
- Mở Query Analyzer
- Ra lệnh phục hồi CSDL1 từ tập tin sao lưu:

RESTORE DATABASE <Tên CSDL> FROM DISK = '<đường dẫn và tên file sao lưu>'

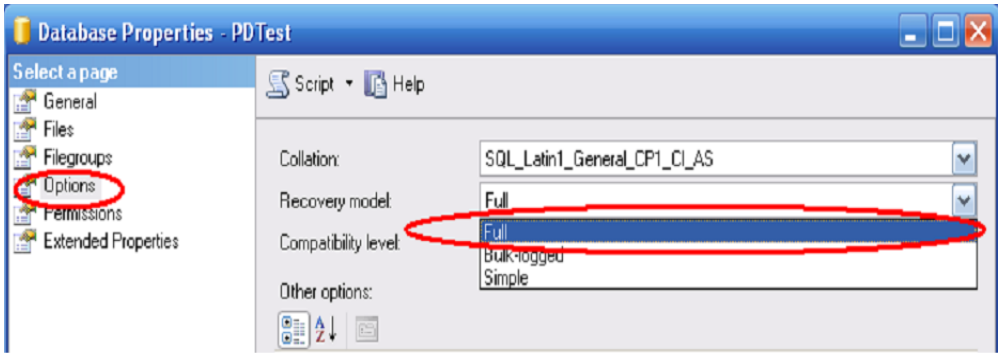
2. Sử dụng giao diện chương trình:

a. Sao lưu CSDL:

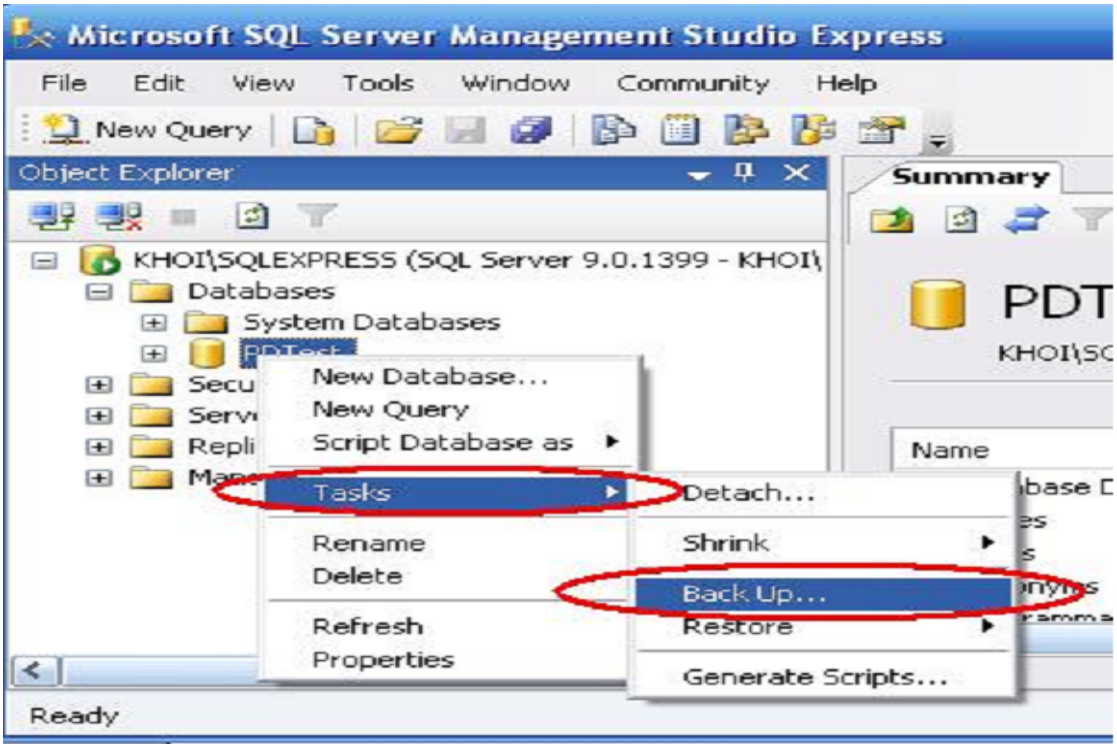
Bước 1: Từ nút "Databases", click phải lên CSDL muốn sao lưu sau đó chọn "Properties"



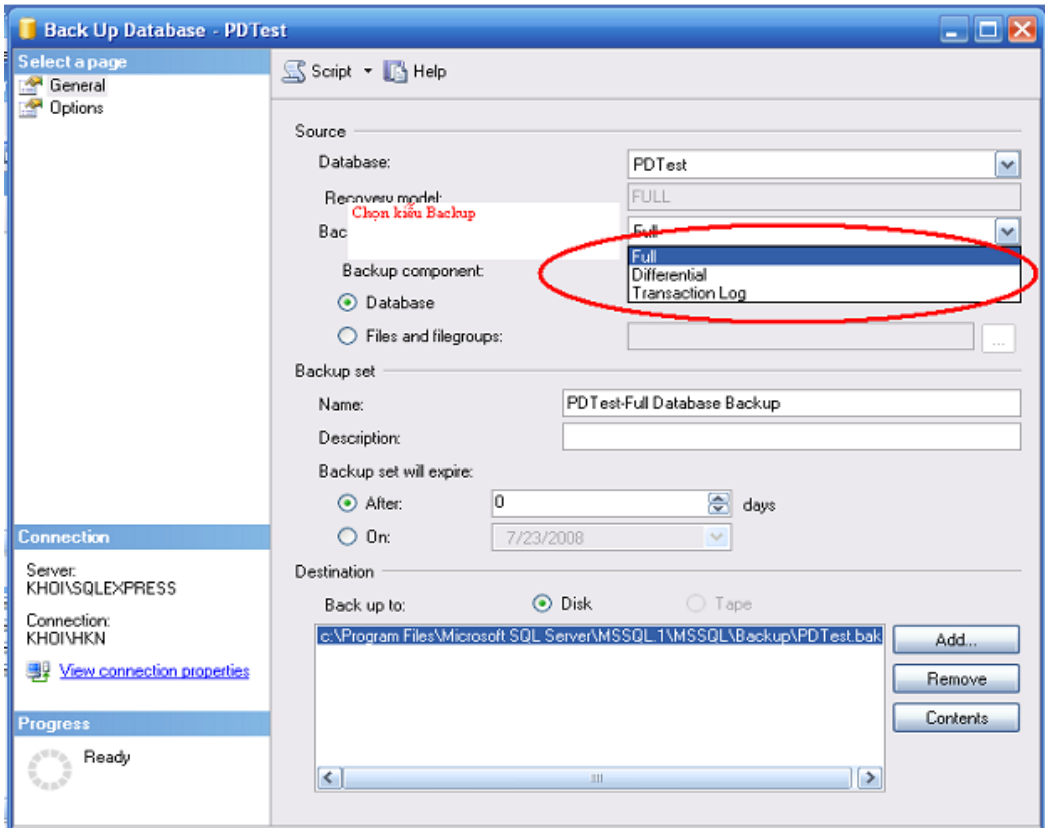
Chọn OK



Bước 2: Từ nút "Databases", click phải lên CSDL muốn sao lưu sau đó chọn "Tasks" và chọn "Backup" .

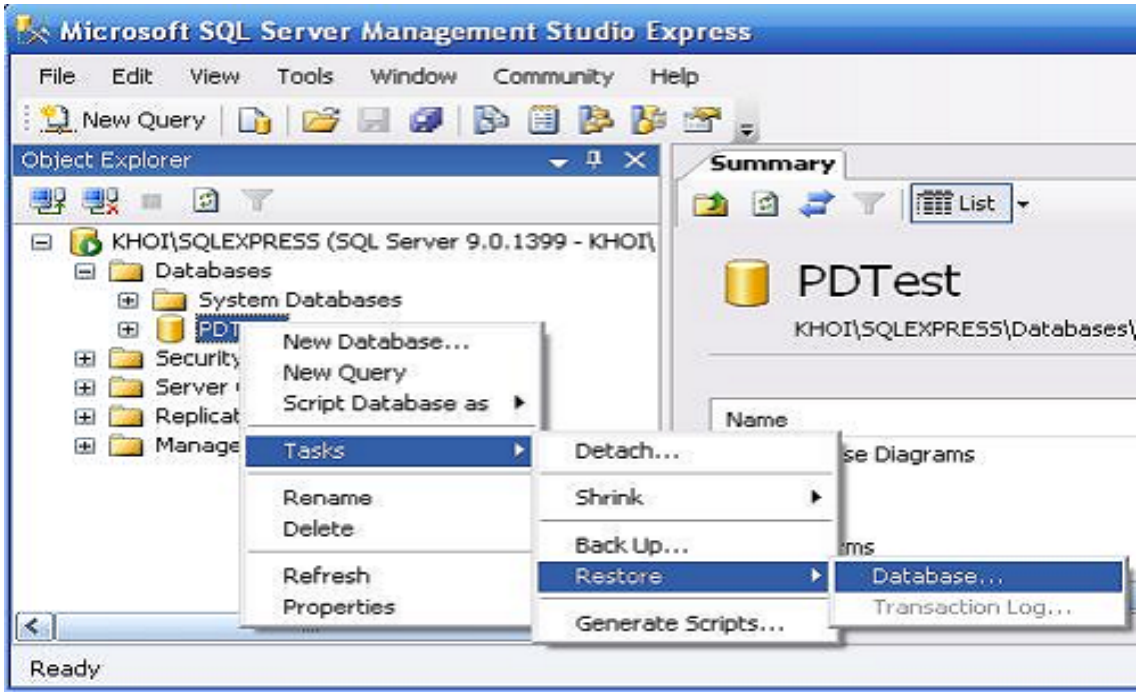


Bước 3: Nhấn chọn "Add.." và nhập vào đường dẫn và tên file sao lưu, nhấn OK để quá trình sao lưu hoàn tất

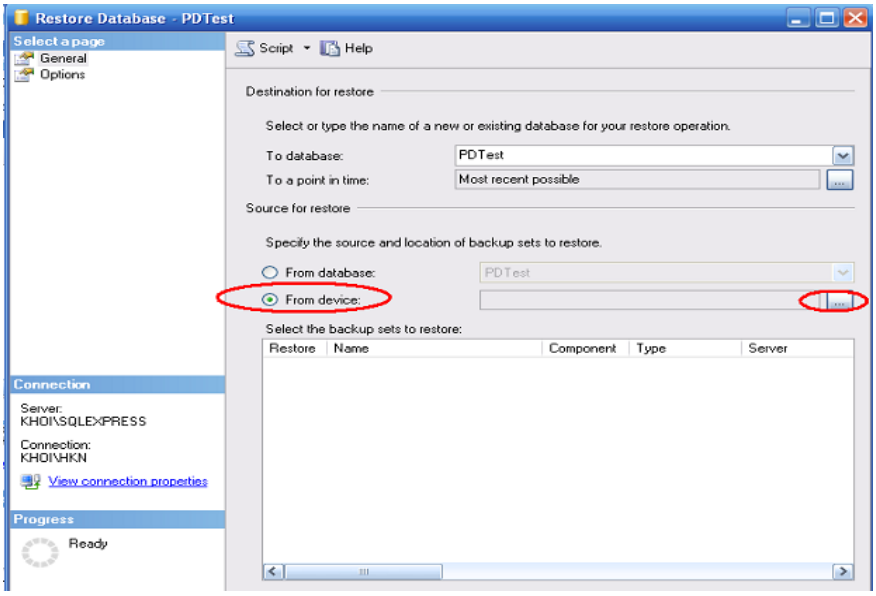


b.Phục hồi CSDL:

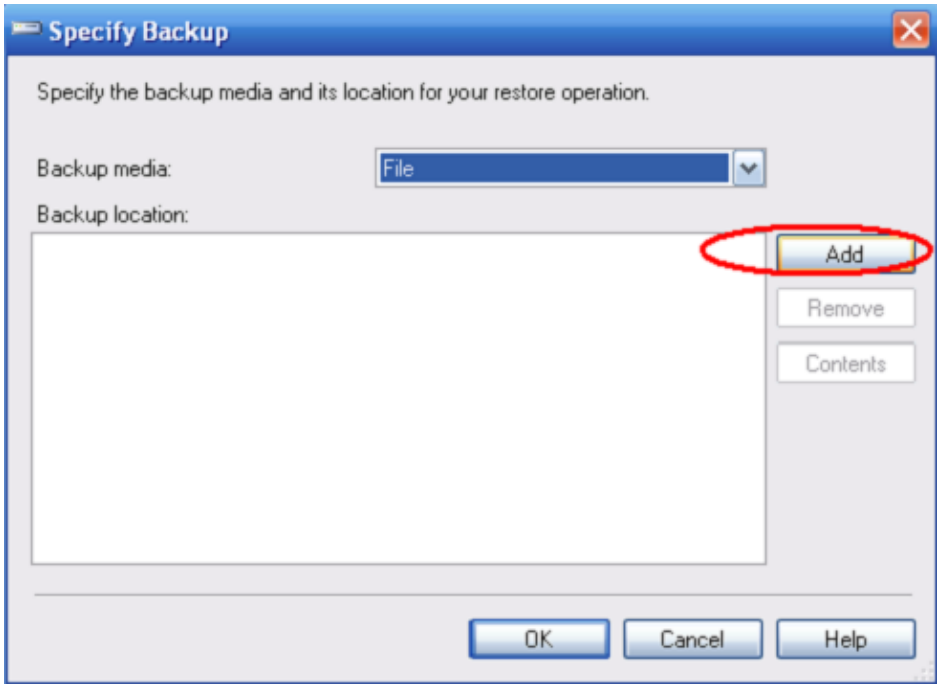
Bước 1: Click chuột phải lên "Databases" sau đó chọn "Tasks" và chọn "Restore", chọn "Database"



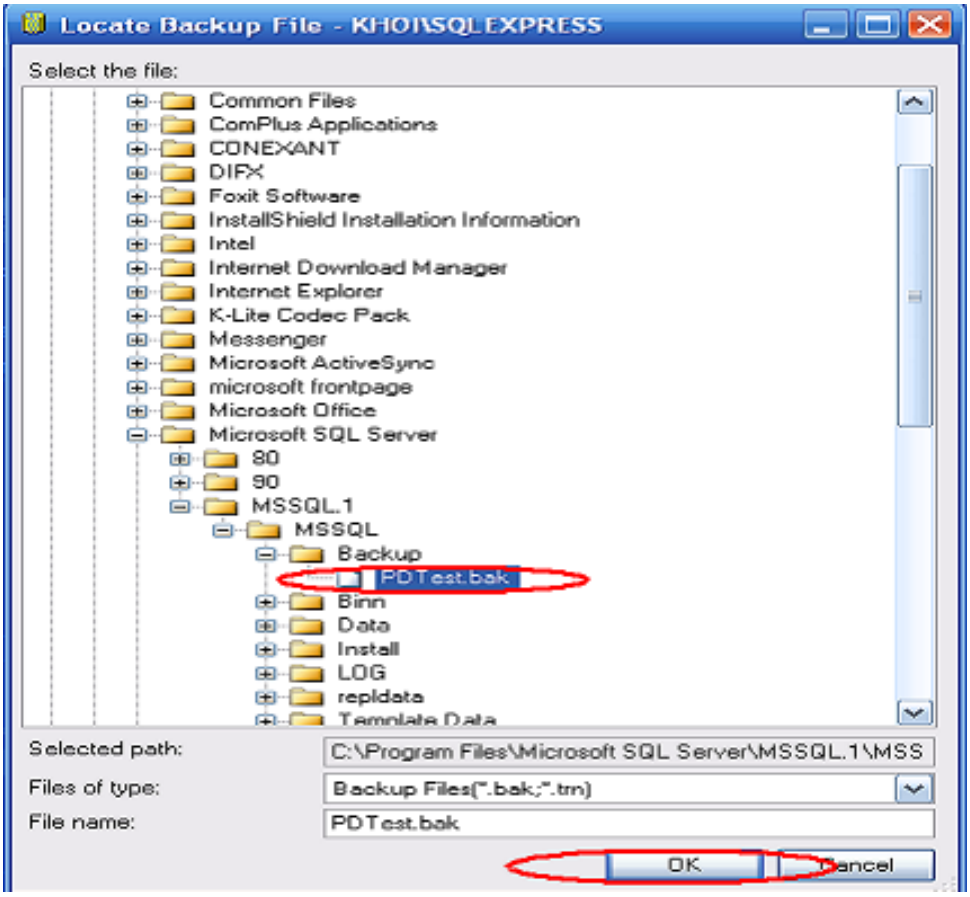
Bước 2: Chọn "From device" sau đó chọn đường dẫn đến file đã sao lưu CSDL trước đó



Nhấn Add



Nhấn OK


Locate Backup File - KHOI\SQLEXPRESS
Select the file:
Common Files
ComPlus Applications
CONEXANT
DIFX
Foxit Software
InstallShield Installation Information
Intel
Internet Download Manager
Internet Explorer
K-Lite Codec Pack
Messenger
Microsoft ActiveSync
microsoft frontpage
Microsoft Office
Microsoft SQL Server
80
90
MSSQL.1
MSSQL
Backup
PDTest.bak
Binn
Data
Install
LOG
repldata
Template Data
Selected path:
C:\Program Files\Microsoft SQL Server\MSSQL.1\MSS
Files of type:
Backup Files(*.bak;*.trn)
File name:
PDTest.bak
OK
Cancel

PHẦN 3. BÀI TẬP THỰC HÀNH

Phần thực hành được chia thành 6 bài. Trong 4 bài đầu tiên, mỗi bài sẽ tập trung rèn luyện một dạng câu hỏi truy vấn, bài 5 dành cho các câu lệnh tạo, sửa bảng, xen, sửa, xóa dòng dữ liệu. Mỗi bài sẽ có bố cục như sau:

1. Mục tiêu
2. Lý thuyết: Nhắc lại một cách ngắn gọn câu lệnh cần dùng trong bài.
3. Bài tập có hướng dẫn: Gồm một số câu hỏi truy vấn trên một CSDL cho trước. Mỗi câu hỏi sẽ được phân tích để đi đến câu lệnh. Những điểm cần lưu ý sẽ được nhấn mạnh. Ngoài ra, một số câu hỏi nhỏ sẽ được đưa ra trong tình huống của câu hỏi giúp sinh viên hiểu rõ hơn vấn đề.
4. Bài tập tự làm: gồm một số câu tương tự như trong phần bài tập có hướng dẫn nhưng không kèm đáp án.

Bài thực hành 6 dành cho bài tập tổng hợp gồm tất cả các dạng câu hỏi, và sinh viên sẽ tự thực hiện mà không có phần hướng dẫn.

BÀI 1

I. MỤC ĐÍCH

Luyện tập khả năng nhận biết sự thông thương trong một cơ sở dữ liệu, từ đó kết nối dữ liệu để truy xuất thông tin cần thiết.

II. LÝ THUYẾT

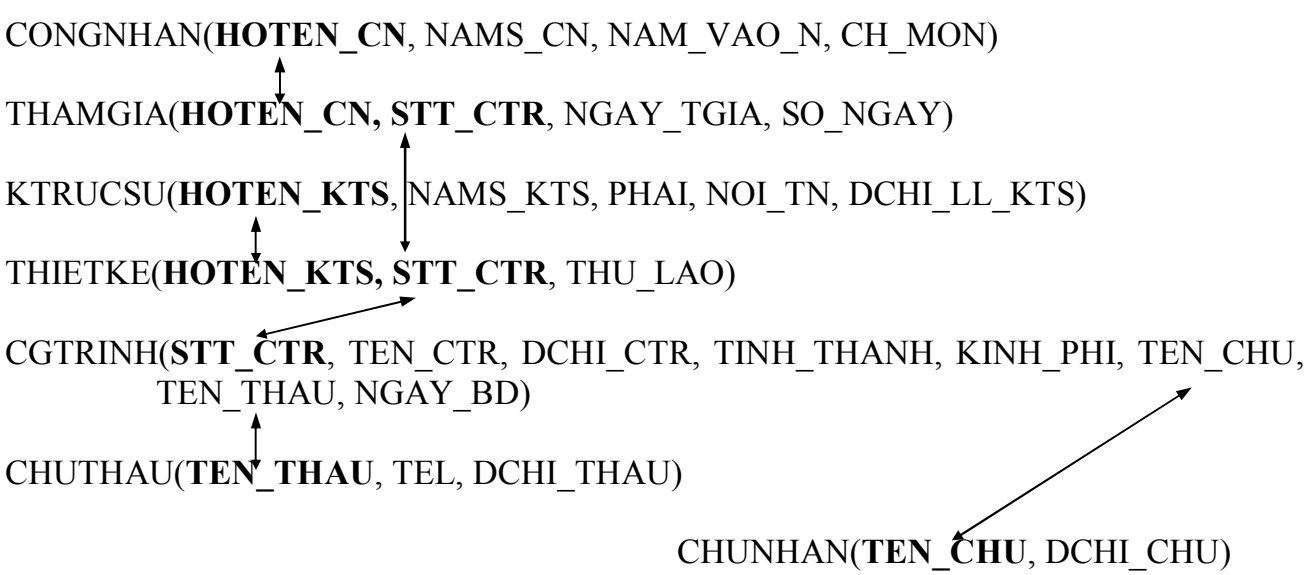
Câu lệnh SQL sử dụng trong bài này:
SELECT <DS tên cột>
FROM <Tên bảng> [<bí danh>] [, <Tên bảng> [<bí danh>] [,...]]
[WHERE <DS các ĐK AND hoặc OR với nhau>]

III. BÀI TẬP CÓ HƯỚNG DẪN:

1. CSDL: Quản lý công trình (CSDL1)

   KTRUCSU(**HOTEN_KTS**, NAMS_KTS, PHAI, NOI_TN, DCHI_LL_KTS)
   Mỗi kiến trúc sư có một họ tên, một năm sinh, phái, nơi tốt nghiệp, địa chỉ liên lạc. Giả sử không có hai kiến trúc sư trùng tên.
   CHUTHAU(**TEN_THAU**, TEL, DCHI_THAU)
   Mỗi chủ thầu xây dựng công trình đều có một tên thầu, một số điện thoại và một địa chỉ. Giả sử không có hai chủ thầu trùng tên.
   CHUNHAN(**TEN_CHU**, DCHI_CHU)
   Mỗi chủ nhân của một công trình đều có một tên chủ nhân và một địa chỉ. Giả sử không có hai chủ nhân trùng tên.
   CONGNHAN(**HOTEN_CN**, NAMS_CN, NAM_VAO_N, CH_MON)
   Mỗi công nhân tham gia xây dựng các công trình đều có một họ tên, một năm sinh, một năm vào nghề, và một chuyên môn nào đó. Giả sử không có hai công nhân trùng họ tên.
   CGTRINH(**STT_CTR,** TEN_CTR, DCHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)
   Mỗi công trình có một số thứ tự, một tên công trình, một địa chỉ, thuộc về một tỉnh thành nào đó, được xây dựng với một kinh phí (đơn vị tính là triệu đồng), thuộc về sở hữu của một chủ nhân, do một chủ thầu xây dựng, và được khởi công xây từ một ngày nào đó.
   THAMGIA(**HOTEN_CN, STT_CTR**, NGAY_TGIA, SO_NGAY)
   Một công nhân có thể tham gia xây dựng nhiều công trình, và một công trình cũng có nhiều công nhân tham gia. Khi một công nhân tham gia vào một công trình nào đó sẽ được ghi nhận lại ngày bắt đầu tham gia và số ngày tham gia.
   THIETKE(**HOTEN_KTS, STT_CTR**, THU_LAO)
   Một kiến trúc sư có thể thiết kế nhiều công trình, mỗi công trình cũng có thể do nhiều kiến trúc sư cùng thiết kế. Khi một kiến trúc sư thiết kế một công trình sẽ có một thù lao tương ứng.

2. Yêu cầu 1: Tìm sự thông thương giữa các bảng trong CSDL:



3. Yêu cầu 2: Mở các bảng dữ liệu để xem kiểu dữ liệu của từng trường và quan sát dữ liệu của từng bảng.

4. Yêu cầu 3: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:

a. Hãy cho biết tên và địa chỉ các công trình do chủ thầu Công ty xây dựng số 6 thi công.

Phân tích câu hỏi:

Yêu cầu cho biết: TEN_CTR, DCHI_CTR => lấy từ bảng: CGTRINH

Điều kiện: TEN_THAU = 'Cty xd so 6' =>Trong CSDL, tên thầu có mặt ở cả hai bảng CGTRINH và CHUTHAU, nên ta chỉ cần đặt ĐK trên bảng CGTRINH đã chọn ở bước trên.

***Lưu ý**: Sinh viên cần phải mở bảng CGTRINH để xem dữ liệu thực tế của bảng; như trong trường hợp này tên thầu Công Ty xây dựng số 6 được lưu trữ thành chuỗi 'Cty xd so 6'.*

Câu truy vấn:

```
Select      TEN_CTR, DCHI_CTR
From  CGTRINH
Where TEN_THAU = 'Cty xd so 6'
```

b. Tìm tên và địa chỉ liên lạc của các chủ thầu thi công công trình ở Cần Thơ do kiến trúc sư Lê Kim Dung thiết kế.

Phân tích câu hỏi:

Yêu cầu cho biết: TEN_THAU, DCHI_THAU => Lấy từ bảng: CHUTHAU

Điều kiện: TINH_THANH = 'Can tho' => Đặt ĐK trên bảng: CGTRINH

Và HOTEN_KTS = 'Le Kim Dung' => Trong CSDL, họ tên KTS có mặt ở cả hai bảng KTRUCSU vàTHIETKE, nhưng ta phải chọn bảng THIETKE, vì bảng này mới cho biết KTS nào thiết kế công trình nào, vì vậy ta đặt ĐK trên bảng: THIETKE

Như vậy, ta sẽ truy xuất 3 bảng: CHUTHAU, CGTRINH, THIETKE. Ta thấy các bảng này đều liên thông với nhau, nghĩa là chúng có cột chung, có thể nối kết tự nhiên được.

***Lưu ý**: Khi câu truy vấn truy xuất thông tin từ nhiều bảng thì nhớ phải:*

- Mô tả đầy đủ các điều kiện kết nối các bảng.

- Các cột chung phải đặt tên bảng (hoặc bí danh nếu bảng đã có bí danh) phía trước tên cột, phân cách bằng dấu chấm.

- Khi đặt bí danh cho bảng có thể có hoặc không có từ khóa AS

Câu truy vấn:

```
Select      a.TEN_THAU, DCHI_THAU
From  CHUTHAU as  a, CGTRINH b, THIETKE as c
Where             a.TEN_THAU = b.TEN_THAU
      And   b.STT_CTR = c.STT_CTR;
      And   TINH_THANH = 'Can tho'
      And   HOTEN_KTS = 'Le Kim Dung'
```

c. Hãy cho biết nơi tốt nghiệp của các kiến trúc sư đã thiết kế công trình Khách Sạn Quốc Tế ở Cần Thơ.

Phân tích câu hỏi:

Yêu cầu cho biết: NOI_TN => Lấy từ bảng: KTRUCSU

Điều kiện: TEN_CTR = 'KS Quoc Te' } => Đặt ĐK trên bảng: CGTRINH
Và TINH_THANH ='Can Tho' }

Như vậy, ta sẽ truy xuất 2 bảng: KTRUCSU và CGTRINH. Tuy nhiên, ta thấy hai bảng này không liên thông nhau hay không có cột chung, không thể nối kết tự nhiên được. Hay nói cách khác, hai bảng này không cho biết KTS nào đã thiết kế công trình Khách sạn quốc tế. Vậy, ta phải thêm vào bảng THIETKE trong truy vấn.

Câu truy vấn:

```
Select      NOI_TN
From  KTRUCSU as  a, CGTRINH as b, THIETKE as c
Where             a.HOTEN_KTS = c.HOTEN_KTS
      And   b.STT_CTR = c.STT_CTR;
      And   TEN_CTR = 'KS Quoc Te'
      And   TINH_THANH ='Can Tho'
```

d. Cho biết họ tên, năm sinh, năm vào nghề của các công nhân có chuyên môn hàn hoặc điện đã tham gia các công trình mà chủ thầu Lê Văn Sơn đã trúng thầu.

Phân tích câu hỏi:

Yêu cầu cho biết: HOTEN_CN, NAMS_CN, NAM_VAO_N => Lấy từ bảng: CONGNHAN

Điều kiện: CH_MON = 'Han' } => Đặt ĐK trên bảng: CONGNHAN
Hoặc CH_MON = 'Dien' }
Và TEN_THAU = 'Le Van Son' => Đặt ĐK trên bảng: CGTRINH

Như vậy, ta sẽ truy xuất 2 bảng: CONGNHAN và CGTRINH. Tương tự như câu c, hai bảng này không cho biết công nhân nào đã tham gia công trình nào. Vậy, ta phải thêm vào bảng THAMGIA trong truy vấn.

***Lưu ý**: Trong danh sách các điều kiện vừa có AND vừa có OR, ta phải dùng dấu ngoặc để chỉ rõ thứ tự kiểm tra điều kiện.*

Câu truy vấn:

```
Select      HOTEN_CN, NAMS_CN, NAM_VAO_N
From  CONGNHAN a, CGTRINH b, THAMGIA c
Where       a.HOTEN_CN = c.HOTEN_CN And   b.STT_CTR = c.STT_CTR
And   ( CH_MON = 'Han'  or  CH_MON = 'Dien'      )
      And   TEN_THAU = 'Le Van Son'
```

e. Những công nhân nào đã bắt đầu tham gia công trình Khách sạn Quốc Tế ở Cần Thơ trong giai đoạn từ ngày 15/12/94 đến ngày 31/12/94, số ngày tương ứng là bao nhiêu.

Phân tích câu hỏi:

Yêu cầu cho biết: HOTEN_CN, NGAY_TGIA, SO_NGAY => Ta lấy từ bảng: THAMGIA

(Ta không cần lấy từ bảng CONGNHAN vì không cần các thông tin khác của công nhân)

Điều kiện: NGAY_TGIA thuộc '12/15/94' -> '12/31/94' => Đặt ĐK trên THAMGIA
TEN_CTR = 'KS Quoc Te' } => Đặt ĐK trên bảng: CGTRINH
Và TINH_THANH ='Can Tho'
Như vậy, ta sẽ truy xuất 2 bảng: THAMGIA và CGTRINH.

***Lưu ý**: Cần lưu ý về cách biểu diễn ngày tháng mà hệ thống đang sử dụng. Chẳng hạn 'mm/dd/yy'. An toàn nhất là biểu diễn tháng bằng tên; vd: '15 Dec, 1994' hoặc 'Dec 15, 1994'*

Câu truy vấn:

```
Select      HOTEN_CN, NGAY_TGIA, SO_NGAY
From  CGTRINH as b, THAMGIA as c
Where       b.STT_CTR = c.STT_CTR
And   NGAY_TGIA  between  'Dec 15, 1994' And 'Dec 31, 1994'
      And   TEN_CTR = 'KS Quoc Te'
      And   TINH_THANH ='Can Tho'
```

f. Cho biết tên và địa chỉ của công trình mà công nhân Nguyễn Hông Vân đang tham gia vào ngày 18/12/94.

Phân tích câu hỏi:

Yêu cầu cho biết: TEN_CTR, DCHI_CTR => Ta lấy từ bảng: CGTRINH

Điều kiện: Đang tham gia vào ngày 18/12/94 có nghĩa là: ngày 18/12/94 phải nằm trong khoảng từ ngày bắt đầu và ngày kết thúc tham gia (ngày kết thúc tức là ngày bắt đầu tham gia + số ngày tham gia).

=> NGAY_TGIA <= '12/18/94'
NGAY_TGIA + SO_NGAY >= '12/18/94' } => Đặt ĐK trên THAMGIA
HOTEN_CN = 'Nguyen Hong Van'

***Lưu ý**: Để cộng ngày tháng trong SQL Server, ta dùng hàm DATEADD. Tham khảo tại phần giới thiệu các hàm xử lý ngày tháng ở trang 10*

Câu truy vấn:

```
Select TEN_CTR, DCHI_CTR      From  CGTRINH as b, THAMGIA as c
Where  b.STT_CTR = c.STT_CTR     And      NGAY_TGIA <=  'Dec 18, 1994'
      And   DATEADD(dd, SO_NGAY, NGAY_TGIA) >= 'Dec 18, 1994'
            And   HOTEN_CN = 'Nguyen Hong Van'
```

g. Cho biết họ tên và năm sinh của các kiến trúc sư đã tốt nghiệp ở thành phố Hồ Chí Minh và đã thiết kế ít nhất một công trình có kinh phí đầu tư trên 400 triệu đồng.

Phân tích câu hỏi:

Yêu cầu cho biết: HOTEN_KTS, NAMS_KTS => Lấy từ bảng: KTRUCSU

Điều kiện: NOI_TN = 'TP HCM' => Đặt ĐK trên bảng: KTRUCSU

Và KINH_PHI > 400 => Đặt ĐK trên bảng: CGTRINH

Như vậy, ta sẽ truy xuất 2 bảng: KTRUCSU và CGTRINH. Tương tự câu c, vì hai bảng này không liên thông nhau nên ta phải thêm vào bảng THIETKE trong truy vấn.

***Lưu ý**: - Vì đơn vị tính của kinh phí trong CSDL là triệu đồng nên 400 triệu sẽ là 400.*
-Từ 'ít nhất một' trong câu truy vấn có thể bỏ qua vì nếu một KTS không có công trình nào thoả điều kiện thì khi đặt ĐK chọn và kết nối, KTS đó sẽ tự động bị loại bỏ. Quy luật này áp dụng đúng cho mọi câu truy vấn.

Câu truy vấn:

```
Select   HOTEN_KTS, NAMS_KTS
From  KTRUCSU a, CGTRINH b, THIETKE c
Where          a.HOTEN_KTS = c.HOTEN_KTS
   And   b.STT_CTR = c.STT_CTR;
   And   NOI_TN = 'TP HCM'
         And   KINH_PHI > 400
```

IV. BÀI TẬP TỰ LÀM:

1. CSDL: Quản lý báo chí (CSDL2)

   BAIBAO(**STT_BAI**, TUA, MA_THLOAI)
   Mỗi bài báo được đánh một số thứ tự phân biệt, có một tựa bài và cho biết nó thuộc thể loại nào.
   BAO_TCHI(**MA_BAO_TC**, TEN, DCHI_TS, TEL, DINH_KY)
   Mỗi tờ báo hoặc tạp chí có một mã riêng biệt, có một tên, một địa chỉ toà soạn, một số điện thoại, và được xuất bản theo một định kỳ nào đó.
   THE_LOAI(**MA**, DGIAI)
   Mỗi thể loại bài báo có một mã thể loại và một diễn giải.
   DANG_BAI(**STT_BAI,** MA_BAO_TC, SO_BAO_TC, TRANG)
   Mỗi bài báo khi được đăng trong một báo hoặc tạp chí nào đó, thì được ghi nhận là đã đăng trong số báo hoặc số tạp chí nào và đăng tại trang số mấy.
   PHATHANH(**MA_BAO_TC, SO_BAO_TC**, NGAY)
   Mỗi số báo hoặc số tạp chí khi được phát hành sẽ được ghi nhận ngày phát hành.
   VIETBAI(**STT_BAI**, BUT_HIEU)
   Mỗi bài báo được viết bởi một tác giả mà ta ghi nhận bằng bút hiệu.

2. Yêu cầu 1: Tìm sự thông thương giữa các bảng trong CSDL2

3. Yêu cầu 2: Mở các bảng dữ liệu để xem kiểu dữ liệu của từng trường và quan sát dữ liệu của từng bảng.

4. Yêu cầu 3: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:

   a. Tìm tên các báo / tạp chí có đăng bài của tác giả Tô Phan. Cho biết đăng trên số nào? Được phát hàng vào những ngày nào?

   b. Nội dung của tạp chí Kiến thức ngày nay số 156 gồm những bài tựa gì? Đăng từ trang nào?

   c. Cho biết tựa và thể loại tất cả các bài báo do tác giả Nguyễn Trọng Vinh sáng tác và đã được đăng trên các báo / tạp chí xuất bản hàng tuần. Cho biết đồng thời tên và số của tờ báo / tạp chí tương ứng mỗi bài được đăng.

   d. Cho biết các số báo / tạp chí và các ngày phát hành tương ứng trong năm 1994 của tờ báo Lao động.

   e. Tờ Tuổi trẻ chủ nhật số 23 có những bài bút ký nào? (Cho biết tựa của chúng)

   f. Cho biết ngày đăng, tên và số của tờ báo / tạp chí đã đăng bài "Chúc vui năm Lợn".

   g. Cho biết tên, địa chỉ và số điện thoại của tờ báo tạp chí đã đăng bài "Tìm hiểu chất độc sarin" trong tuần lễ từ 23/04/95 đến 30/04/1995.

   h. Tác giả Hàn Ngọc Cẩm có những truyện nào đã được đăng trên báo / tạp chí? Cho biết tựa của chúng.

   i. Cho biết số lượng bài báo mà tạp chí Kiến thức ngày nay đã đăng trên tất cả các số báo.

   j. Cho biết thể loại của các bài báo do các tác giả có họ Lê sáng tác.

   k. Tìm tựa các bài thơ đã được đăng trên các báo/tạp chí trong tuần lễ từ 23/4/95 đến 30/4/95.

l. Tìm tên các tác giả được đăng bài trên báo Cựu Chiến Binh.

m. Tìm tên những báo/tạp chí có đang xã luận. In ra tên các bài xã luận đó và các số báo/tạp chí tương ứng.

BÀI 2

I. MỤC ĐÍCH

Luyện tập câu lệnh SELECT lồng nhau để trả lời cho các câu hỏi cần sử dụng phép giao hoặc phép trừ. Ngoài ra, còn luyện tập khả năng vận dụng các hàm kết tập max, min, sum, count, avg.

II. LÝ THUYẾT

Câu lệnh SQL sử dụng trong bài này:

SELECT <Tên các cột>
FROM <Tên bảng>
WHERE <điều kiện chứa select con>
(SELECT <Tên cột>
FROM <Tên bảng> [WHERE <Điều kiện>])
[AND | OR <điều kiện chứa select con>
(SELECT <Tên cột>
FROM <Tên bảng> [WHERE <Điều kiện>])
[...]]

*** <u>Điều kiện chứa select con có các dạng sau đây:</u>**

<Tên cột> <so sánh> (<select con>):
ĐK đúng khi giá trị của cột so sánh đúng với giá trị trả về từ select con.

<Tên cột> <so sánh> ALL (<select con>):
ĐK đúng khi giá trị của cột so sánh đúng với tất cả các giá trị trả về từ select con.

<Tên cột> <so sánh> ANY|SOME (<select con>):
ĐK đúng khi giá trị của cột so sánh đúng với bất kỳ một giá trị nào trả về từ select con.

<Tên cột> [NOT] IN (<select con>):
ĐK đúng khi giá trị của cột nằm trong tập hợp các giá trị trả về của select con.

[NOT] EXISTS (<select con>):
Đk đúng khi kết quả trả về của select con khác rỗng.

Khi sử dụng các dạng này, câu lệnh select con cần phải tuân theo quy tắc sau:

Select con trả về:	Một cột	Nhiều cột
Một dòng	Dạng 1	Dạng 5
Nhiều dòng	Dạng 2, 3, 4	

***<u>Lưu ý</u>**: - Câu lệnh SELECT trong SQL Server 2000 cho phép lồng đến tối đa là 32 mức.*
- Mệnh đề HAVING cũng có thể chứa SELECT con như mệnh đề WHERE.

III. BÀI TẬP CÓ HƯỚNG DẪN:

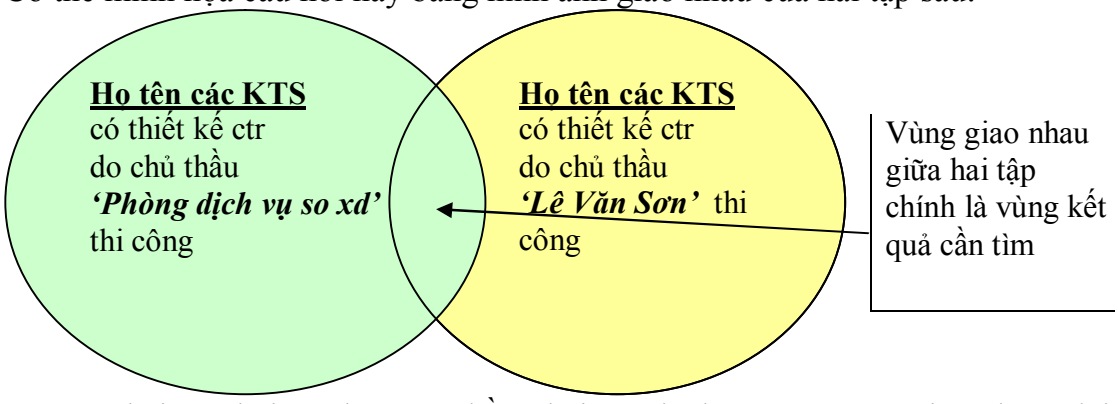
1. CSDL: Quản lý công trình (CSDL1)

   KTRUCSU(**HOTEN_KTS**, NAMS_KTS, PHAI, NOI_TN, DCHI_LL_KT)
   CHUTHAU(**TEN_THAU**, TEL, DCHI_THAU)
   CHUNHAN(**TEN_CHU**, DCHI_CHU)
   CONGNHAN(**HOTEN_CN**, NAMS_CN, NAM_VAO_N, CH_MON)
   CGTRINH(**STT_CTR**, TEN_CTR, DCHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)
   THAMGIA(**HOTEN_CN, STT_CTR**, NGAY_TGIA, SO_NGAY)
   THIETKE(**HOTEN_KTS, STT_CTR**, THU_LAO)
   Tham khảo trang 21 để xem diễn giải cho từng bảng.

2. Yêu cầu 1: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:

   a. Cho biết họ tên các kiến trúc sư *vừa* thiết kế các công trình do Phòng Dịch vụ Sở Xây dựng thi công, *vừa* thiết kế các công trình do chủ thầu Lê Văn Sơn thi công.

   Phân tích câu hỏi:
   Có thể minh họa câu hỏi này bằng hình ảnh giao nhau của hai tập sau:



   Vậy ta phải tìm hai tập hợp này bằng hai câu lệnh SELECT và giao chúng lại với nhau. Tuy nhiên, vì phép giao không được hỗ trợ trực tiếp nên ta sử dụng dạng câu SELECT lồng nhau trong đó sử dụng từ khóa IN để kết nối select cha và select con với nhau (tức là dạng điều kiện chứa select con số 4 đã đề cập ở trên).

> ***Lưu ý***: *Cách nhận biết dạng câu hỏi sử dụng phép GIAO để trả lời:*
> *Yêu cầu tìm một <đối tượng nào đó>* ***vừa*** *thực hiện <một hành động> này hay sở hữu một <tính chất> này, lại* ***vừa*** *thực hiện <một hành động> khác hay sở hữu một <tính chất> khác*

   Câu truy vấn:

```
Select distinct HOTEN_KTS From THIETKE a, CGTRINH b
Where a.STT_CTR = b.STT_CTR
and TEN_THAU = 'phong dich vu so xd'
and HOTEN_KTS  IN
(Select HOTEN_KTS From THIETKE a, CGTRINH b
  Where a.STT_CTR = b.STT_CTR
and TEN_THAU = 'le van son')
```

> ***Lưu ý***: *Trong câu SELECT con dạng này (dùng từ khóa IN) ta chỉ được select một cột duy nhất (không được select nhiều cột), và cột select phải là cột ta đặt trong điều kiện chứa select con. Trong ví dụ trên đó là cột hoten_kts.*

***Thắc mắc**: Select cha va select con của câu truy vấn trên có thể đổi chỗ cho nhau được không?*

b. Cho biết tên công trình có kinh phí cao nhất.

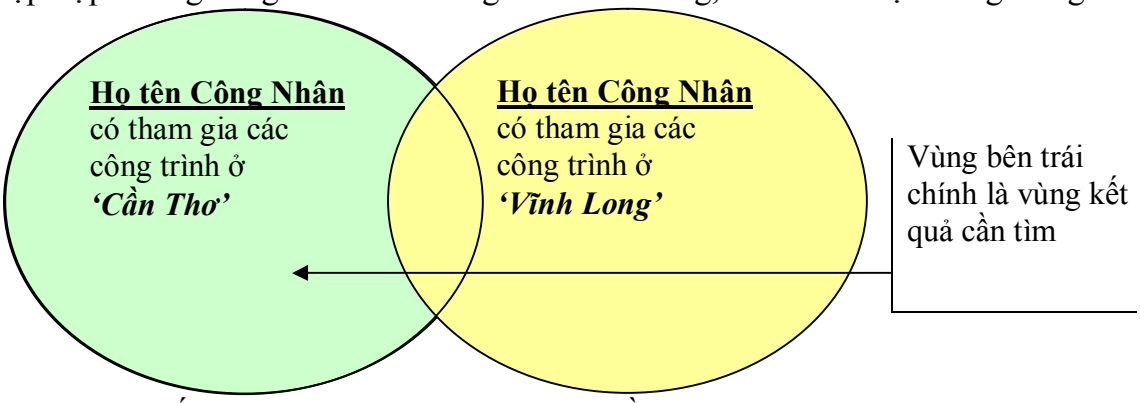
Phân tích câu hỏi:
Vì ta chưa biết kinh phí cao nhất là bao nhiêu nên ta phải dùng một select con kết hợp với hàm max để tìm ra con số này. Vì chỉ có duy nhất một con số lớn nhất, nên ta có thể sử dụng dạng điều kiện chứa select con số 1.
Câu truy vấn:

```
Select TEN_CTR From CGTRINH
Where KINH_PHI =
(Select max(KINH_PHI) From CGTRINH)
```

***Lưu ý**: Đối với các câu hỏi dạng câu hỏi:*
Tìm <đối tượng> sở hữu một <thông số> < lớn / nhỏ / nhiều / ...>. nhất.
Ta đều phải sử dụng select con trong câu truy vấn.

c. Cho biết họ tên các công nhân có tham gia các công trình ở Cần Thơ, nhưng không có tham gia công trình ở Vĩnh Long.

Phân tích câu hỏi:
Như ta biết CSDL của ta chỉ lưu trữ những thông tin có hay đã xảy ra. Vì vậy để trả lời được cây hỏi này, ta chỉ có thể lấy tập hợp những công nhân có tham gia ở Cần Thơ trừ đi tập hợp những công nhân có tham gia ở Vĩnh Long, như minh họa trong trong hình sau:



Vậy trước hết ta phải tìm hai tập hợp này bằng hai câu lệnh SELECT và sau đó trừ tập thứ nhất cho tập thứ hai. Tuy nhiên, vì phép trừ không được hỗ trợ trực tiếp nên ta sử dụng dạng câu SELECT lồng nhau trong đó sử dụng từ khóa NOT IN để kết nối select cha và select con với nhau (tức là dạng điều kiện chứa select con số 4 đã đề cập ở trên).

***Lưu ý**: Cách nhận biết dạng câu hỏi sử dụng phép TRỪ để trả lời:*
*Yêu cầu tìm một <đối tượng nào đó> **có** thực hiện <một hành động> này hay sở hữu một <tính chất> này, **nhưng không** (hoặc **chưa**) thực hiện <một hành động> khác hay sở hữu một <tính chất> khác*

Câu truy vấn:

```
Select distinct    HOTEN_CN      From THAMGIA a, CGTRINH b
Where a.STT_CTR = b.STT_CTR
and TINH_THANH = 'Can Tho'
and HOTEN_CN  NOT IN
   (Select HOTEN_CN    From THAMGIA a, CGTRINH b
    Where a.STT_CTR = b.STT_CTR
         and TINH_THANH = 'Vinh Long')
```

***Thắc mắc**: Select cha va select con của câu truy vấn trên có thể đổi chỗ cho nhau được không?*

d. Cho biết tên của các chủ thầu đã thi công các công trình có kinh phí lớn hơn tất cả các công trình do chủ thầu Phòng Dịch vụ Sở Xây dựng thi công.

Phân tích câu hỏi:
Trước hết ta phải dùng một select để tìm tất cả các kinh phí của các công trình do chủ thầu Phòng dịch vụ sở xây dựng thi công. Sau đó một select thứ 2 để tìm các chủ thầu thi công các công trình lớn hơn tất cả các kinh phí này. Ta thấy hai câu select này có thể lồng nhau select thứ nhất sẽ là select con vì nó được thực hiện trước, còn select thứ 2 sẽ là select cha, và select cha sẽ dùng điều kiện chứa select con dạng thứ 2 (có từ khóa ALL).
Câu truy vấn:

```
Select TEN_THAU From CGTRINH
Where KINH_PHI   > ALL
   (Select KINH_PHI From CGTRINH
     Where TEN_THAU = 'Phong dich vu so xd')
```

***Thắc mắc**: Câu này ta có thể viết lại bằng cách sử dụng dạng điều kiện chứa select con số 1 được không?*

e. Cho biết họ tên các kiến trúc sư có thù lao thiết kế cho một kiến trúc sư của một công trình nào đó dưới giá trị trung bình thù lao thiết kế của một kiến trúc sư cho một công trình.

Phân tích câu hỏi:
Trước hết ta phải dùng một select để tìm giá trị trung bình thù lao thiết kế của một công trình. Sau đó dùng một select thứ 2 để tìm các kiến trúc sư có thù lao thiết kế một công trình nào đó nhỏ hơn con số tìm được. Ta thấy hai câu select này có thể lồng nhau: select thứ nhất sẽ là select con vì nó được thực hiện trước, còn select thứ 2 sẽ là select cha, và select cha sẽ dùng điều kiện chứa select con dạng thứ 1.
Câu truy vấn:

```
Select HOTEN_KTS From THIETKE
Where  THU_LAO <
   (Select Avg(THU_LAO) From THIETKE)
```

f. Tìm tên và địa chỉ những thầu đã trúng thầu công trình có kinh phí thấp nhất.

Phân tích câu hỏi:
Vì ta chưa biết kinh phí thấp nhất là bao nhiêu nên ta phải dùng một select con kết hợp với hàm min để tìm ra con số này. Vì chỉ có duy nhất một con số nhỏ nhất, nên ta có thể sử dụng dạng điều kiện chứa select con số 1.
Câu truy vấn:

```
Select TEN_THAU, DCHI_THAU From CHU_THAU a, CGTRINH b
Where a.TEN_THAU = b.TEN_THAU And KINH_PHI =
(Select min(KINH_PHI) From CGTRINH)
```

***Thắc mắc:** - Dạng câu hỏi này giống câu hỏi nào trong bài thực hành này?*
- Câu này có thể sử dụng dạng điều kiện chứa select con số 3 hay không?

g. Tìm họ tên và chuyên môn của các công nhân tham gia các công trình do kiến trúc sư 'Le Thanh Tung' thiết kế.

Phân tích câu hỏi:

Trước hết ta phải dùng một select để tìm số thứ tự các công trình do kiến trúc sư Lê Thanh Tùng thiết kế. Sau đó dùng một select thứ 2 để tìm các công nhân đã tham gia vào các công trình tìm được. Ta thấy hai câu select này có thể lồng nhau: select thứ nhất sẽ là select con vì nó được thực hiện trước, còn select thứ 2 sẽ là select cha, và select cha sẽ dùng điều kiện chứa select con dạng thứ 4.

Câu truy vấn:

```
Select   a.HOTEN_CN, CH_MON
From   CONGNHAN a, THAMGIA b
Where  a.HOTEN_CN  =  b.HOTEN_CN
   and b.STT_CTR  IN
         (Select STT_CTR From THIETKE
          Where HOTEN_KTS = 'Le Thanh Tung')
```

h. Tìm các cặp tên của chủ thầu có trúng thầu các công trình tại cùng một thành phố.

Phân tích câu hỏi:

Vì mỗi dòng trong bảng CGTRINH chỉ cho biết chủ thầu nào thi công công trình nào, nên để tìm được những cặp chủ thầu trúng thầu các công trình tại cùng thành phố, thì ta phải mở bảng CGTRINH hai lần với hai bí danh khác nhau, kết nối chúng bằng các điều kiện khác tên thầu nhưng cùng tỉnh thành:

a.TEN_THAU <> b.TEN_THAU (1)
a.TINH_THANH = b.TINH_THANH

Tuy nhiên, với điều kiện kết nối như vậy, ta sẽ nhận được hai dòng cho mỗi cặp với thứ tự tên thầu ngược nhau; chẳng hạn: (Le Van Son, Cty xd so 6) và (Cty xd so 6, Le Van Son). Để khắc phục tình trạng này, ta sửa lại điều kiện kết nối (1) bằng:

a.TEN_THAU < b.TEN_THAU

Câu truy vấn:

```
Select   Distinct  c1.TEN_THAU, c2.TEN_THAU, c1.TINH_THANH
From     CGTRINH c1, CGTRINH c2
Where    c1.TEN_THAU < c2.TEN_THAU
         and c1.TINH_THANH = c2.TINH_THANH
```

i. Tìm các cặp tên của các công nhân có làm việc chung với nhau trong ít nhất là hai công trình.

Phân tích câu hỏi:

Trước hết ta phải dùng một select để tìm các cặp họ tên công nhân có tham gia cùng công trình. Sau đó dùng một select thứ hai nhóm từng cặp lại và đếm số công trình họ tham gia chung với nhau, chọn ra những cặp có số công trình tham gia chung lớn hơn hoặc bằng 2. Hai select này không thể lồng nhau, vì select thứ hai nhóm dữ liệu trên kết quả của select thứ nhất. Vì vậy bắt buộc ta phải dùng 2 select. Select thứ nhất sẽ đưa kết quả vào bảng #TAM bằng mệnh đề INTO <Tên Bảng>

Lưu ý: *Các bảng có tên bắt đầu bằng dấu # sẽ được lưu vào CSDL tempdb. Đây là CSDL nháp, vì vậy, sau khi ta đóng kết nối với SQL Server thì các bảng này sẽ bị xóa.*

Câu select thứ nhất hoàn toàn tương tự như câu h ở trên. Vì mỗi dòng trong bảng THAMGIA chỉ cho biết công nhân nào tham gia công trình nào, nên để tìm được những cặp công nhân tham gia cùng công trình, thì ta phải mở bảng THAMGIA hai lần với hai bí danh khác nhau, kết nối chúng bằng các điều kiện:

a.HOTEN_CN <> b.HOTEN_CN (1)
a.STT_CTR = b.STT_CTR

Tuy nhiên, với điều kiện kết nối như vậy, ta sẽ nhận được hai dòng cho mỗi cặp với thứ tự họ tên ngược nhau; chẳng hạn: (Nguyen Van A, Tran Thi B) và (Tran Thi B, Nguyen Van A). Để khắc phục tình trạng này, ta sửa lại điều kiện kết nối (1) bằng:

a.HOTEN_CN < b.HOTEN_CN

Câu truy vấn:

```
Select t1.HOTEN_CN As cn1, t2.HOTEN_CN As cn2, t1.STT_CTR
Into     #CAU_2i
From THAMGIA t1, THAMGIA t2
Where t1.HOTEN_CN < t2.HOTEN_CN
         and t1.STT_CTR = t2.STT_CTR
Order by cn1, cn2
```

Lưu ý: *Mặc dù hai cột lấy từ hai bảng với bí danh khác nhau nhưng có tên giống nhau, nên ta phải đặt lại tên cột bằng từ khóa AS thì mới có thể lưu lại vào bảng mới.*

```
Select cn1, cn2      From  #CAU_2i
Group by cn1, cn2
Having count(STT_CTR) >=2
```

IV. BÀI TẬP TỰ LÀM:

1. CSDL: Quản lý mua bán hàng hóa (CSDL3)

   HANGHOA(**MA_HANG**,TEN_HG)
   Mỗi hàng hóa được gán một mã hàng riêng biệt để dễ phân biệt, và có một tên hàng.
   DAILY(**STT_DL**, TEN_DL, DCHI_DL)
   Mỗi đại lý được gán một số thứ tự phân, có một tên đại lý và một địa chỉ liên lạc.
   MUA(**MA_HANG, STT_DL, NGAY_MUA**, SOLG_MUA, TRIGIA_MUA)
   Khi một đại lý mua một mặt hàng nào đó (MA_HANG), vào một ngày nào đó (NGAY_MUA), sẽ được ghi nhận lại số lượng mua và trị giá mua tổng cộng.
   BAN(**MA_HANG, STT_DL, NGAY_BAN**, SOLG_BAN, TRIGIA_BAN)
   Khi một đại lý bán một mặt hàng nào đó (MA_HANG), vào một ngày nào đó (NGAY_BAN), sẽ được ghi nhận lại số lượng bán và trị giá bán tổng cộng.

2. Yêu cầu 1: Tìm sự thông thương giữa các bảng trong CSDL3

3. Yêu cầu 2: Mở các bảng dữ liệu để xem kiểu dữ liệu của từng trường và quan sát dữ liệu của từng bảng.

4. Yêu cầu 3: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:

   a. **Tìm tên những đại lý vừa có bán coca cola vừa có bán pepsi cola.**

   b. **Tìm tên những mặt hàng được mua nhưng chưa được bán.**

   c. **Tìm tên và địa chỉ những đại lý có mua ít nhất cùng một mặt hàng với Vạn Lợi mua.**

   d. **Cho biết tên các mặt hàng được ít nhất là hai đại lý mua vào với số lượng trên 50.**

   e. **Tìm tổng trị giá mua Coca Cola ở đại lý Tân Hiệp Hưng.**

   f. **Tìm tổng trị giá bán Coca Cola ở đại lý Tân Hiệp Hưng.**

   g. **Tìm tên những mặt hàng được bán ở đại lý Tân Hiệp Hưng vào cả hai ngày 15/12/94 và ngày 31/12/94.**

   h. **Có bao nhiêu mặt hàng được mua vào nhưng chưa được bán ra ở đại lý Tân Hiệp Hưng trong tháng 12 năm 1994.**

   i. **Tìm tên các mặt hàng có mua và bán trong cùng một ngày ở cùng một đại lý.**

   j. **Tìm tên và địa chỉ của các đại lý và những mặt hàng có số lượng mua và bán bằng nhau trong cùng một ngày.**

   k. **Tìm tên đại lý đã mua coca cola với số lượng nhiều hơn tất cả các đại lý khác.**

BÀI 3

I. MỤC ĐÍCH:

Luyện tập câu lệnh SELECT có GROUP BY, HAVING có kết hợp với các dạng SELECT đã thực tập ở các bài trước.

II. LÝ THUYẾT:

Câu lệnh SQL sử dụng trong bài này:

SELECT <các cột phân nhóm>, <hàm-kết-tập(<biểu thức>)>
FROM <bảng>
[WHERE <Điều kiện chọn dòng>]
[GROUP BY <Cột để phân nhóm> [, <Cột để phân nhóm> [,...]]]
[HAVING <điều kiện chọn nhóm>]

III. BÀI TẬP CÓ HƯỚNG DẪN

1. CSDL: Quản lý Công Trình (CSDL 1)
   1. KTRUCSU(**HOTEN_KTS**, NAMS_KTS, PHAI, NOI_TN, DCHI_LL_KT)
   2. CHUTHAU(**TEN_THAU**, TEL, DCHI_THAU)
   3. CHUNHAN(**TEN_CHU**, DCHI_CHU)
   4. CONGNHAN(**HOTEN_CN**, NAMS_CN, NAM_VAO_N, CH_MON)
   5. CGTRINH(**STT_CTR**, TEN_CTR, DCHI_CTR, TINH_THANH, KINH_PHI, TEN_CHU, TEN_THAU, NGAY_BD)
   6. THAMGIA(**HOTEN_CN, STT_CTR**, NGAY_TGIA, SO_NGAY)
   7. THIETKE(**HOTEN_KTS, STT_CTR**, THU_LAO)

   Tham khảo trang 21 để xem diễn giải cho từng bảng.

2. Yêu cầu 1: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:

   a. Tìm tổng kinh phí của tất cả các công trình theo từng chủ thầu.

   Phân tích câu hỏi:
   Vì mỗi dòng dữ liệu trong bảng CGTRINH chỉ cho biết kinh phí của từng công trình, nên muốn biết tổng kinh phí theo từng chủ thầu thì ta phải nhóm các dòng dữ liệu theo cột TEN_THAU, sau đó dùng hàm sum để cộng kinh phí cho mỗi nhóm.
   Câu truy vấn:

```
Select   TEN_THAU, sum(KINH_PHI) As TongKP
From     CGTRINH
Group by TEN_THAU
```

   Thắc mắc: *Nếu trong mệnh đề Select ta thêm cột TEN_CTR thì có được không?Tại sao?*

   b. Cho biết họ tên các kiến trúc sư có tổng thù lao thiết kế các công trình lớn hơn 25 triệu.

   Phân tích câu hỏi:
   Vì mỗi dòng dữ liệu trong bảng THIETKE chỉ cho biết thù lao thiết kế của một kiến trúc sư cho một công trình nào đó, nên muốn biết tổng thù lao thiết kế các công trình của từng kiến trúc sư, thì ta phải nhóm các dòng dữ liệu theo cột HOTEN_KTS, sau đó dùng hàm sum để cộng thù lao cho mỗi nhóm. Sau cùng, dùng mệnh đề HAVING để chọn lại các nhóm (hay các kiến trúc sư) có tổng thù lao > 25.
   Câu truy vấn:

```
Select   HOTEN_KTS, sum(THU_LAO) As TongTL
```

```
From     THIETKE
Group by HOTEN_KTS
Having sum(THU_LAO) > 25
```

c. Cho biết số lượng các kiến trúc sư có tổng thù lao thiết kế các công trình lớn hơn 25 triệu.

Phân tích câu hỏi:

Câu hỏi này tương tự như câu b, nhưng thay vì liệt kê họ tên các kiến trúc sư thỏa điều kiện, thì ta chỉ cần cho biết có tổng cộng bao nhiêu kiến trúc sư thỏa điều kiện như vậy. Vì vậy, trước hết ta phải tìm danh sách các kiến trúc sư thỏa điều kiện trước rồi mới đếm trên danh sách tìm được. Hay nói khác hơn, ta phải thực hiện 2 bước bằng 2 lệnh select khác nhau:

Câu 1. giống hệt câu b, nhưng kết quả xuất ra sẽ được lưu vào bảng #CAU_3C

Câu 2. Đếm số dòng trên tập tin #CAU_3C

Câu Truy Vấn

```
Select   HOTEN_KTS, sum(THU_LAO) As TongTL   Into   #CAU_3C
From     THIETKE
Group by HOTEN_KTS
Having sum(THU_LAO) > 25

Select  count(*) As Solg_KTS
From     #CAU_3C
```

Thắc mắc: *Ở bước 2 ta có cần phải đếm phân biệt không?*

d. Tìm tổng số công nhân đã tham gia của mỗi công trình.

Phân tích câu hỏi:

Vì mỗi dòng dữ liệu trong bảng THAMGIA chỉ cho công nhân nào tham gia công trình nào, nên muốn biết tổng số công nhân đã tham gia theo từng công trình, thì ta phải nhóm các dòng dữ liệu theo cột STT_CTR, sau đó dùng hàm count để đếm số dòng (mỗi dòng là một công nhân) trong mỗi nhóm để có được tổng số công nhân tham gia.

Câu truy vấn:

```
Select   STT_CTR, count(*) As TongSoCN
From     THAMGIA
Group by STT_CTR
```

Thắc mắc: *Trong mệnh đề Select của câu này có thể thêm cột HOTEN_CN không?*

e. Tìm tên và địa chỉ công trình có tổng số công nhân tham gia nhiều nhất.

Phân tích câu hỏi:

Câu hỏi này tương tự như câu d, nhưng thay vì cho biết tổng số công nhân tham gia của tất cả các công trình, thì ở đây ta chỉ yêu cầu kết xuất ra công trình nào có tổng số công nhân tham gia nhiều nhất.

Vì vậy, trước hết ta phải thực hiện câu d trước đưa kết quả vào bảng #CAU_3E, từ bảng #CAU_3E ta mới có thể chọn ra công trình có tổng số công nhân tham gia nhiều nhất được. Vậy, ta phải thực hiện 2 bước bằng 2 lệnh select khác nhau:

Câu 1. giống hệt câu d, nhưng kết quả xuất ra sẽ được lưu vào bảng #CAU_3E

Câu 2. Chọn ra dòng có TongSoCN cao nhất từ tập tin #CAU_3E, đồng thời kết nối với bảng CGTRINH để lấy thông tin về tên và địa chỉ công trình.

Câu Truy Vấn:

```
Select   STT_CTR, count(*) As TongSoCN       Into   #CAU_3E
From     THAMGIA
```

```
Group by STT_CTR

Select  TEN_CTR, DIACHI_CTR, TongSoCN
From #CAU_3E a, CGTRINH b
Where a.STT_CTR = b.STT_CTR
   And   TongSoCN  =
         (Select  Max(TongSoCN)
           From        #CAU_3E)
```

f. Cho biết tên các thành phố và kinh phí trung bình cho mỗi công trình của từng thành phố tương ứng.

Phân tích câu hỏi:

Vì mỗi dòng dữ liệu trong bảng CGTRINH chỉ cho biết kinh phí của mỗi công trình và cho biết công trình đó nào thuộc tỉnh thành nào, nên muốn biết kinh phí trung bình cho mỗi công trình của từng thành phố, thì ta phải nhóm các dòng dữ liệu theo cột TINH_THANH, sau đó dùng hàm avg để tính kinh phí trung bình cho mỗi nhóm (tức là mỗi thành phố).

Câu truy vấn:

```
Select   TINH_THANH, Avg(KINH_PHI) As KinhPhiTB
From     CGTRINH     Group By  TINH_THANH
```

g. Cho biết họ tên các công nhân có tổng số ngày tham gia vào các công trình lớn hơn tổng số ngày tham gia của công nhân Nguyen Hong Van.

Phân tích câu hỏi:

Để trả lời được câu hỏi này trước hết ta phải biết tổng số ngày tham gia các công trình của mỗi công nhân. Sau đó mới chọn ra những công nhân có tổng số ngày này lớn hơn tổng số ngày của công nhân Nguyễn Hồng Vân. Vì vậy, ta phải thực hiện hai bước này bằng 2 lệnh Select khác nhau:

Câu 1. Nhóm dữ liệu trong bảng THAMGIA theo HOTEN_CN, và tính tổng số ngày tham gia. Lưu kết quả vào bảng #CAU_3G

Câu 2. Chọn các dòng trên bảng #CAU_3G thỏa điều kiện.

Câu truy vấn:

```
Select   HOTEN_CN, Sum(SO_NGAY) As TongSoNgay      Into   #CAU_3G
From     THAMGIA
Group By  HOTEN_CN

Select HOTEN_CN, TongSoNgay      From  #CAU_3G
Where TongSoNgay     >
         (Select TongSoNgay   From   #CAU_3G
           Where HOTEN_CN = 'Nguyen Hong Van')
```

h. Cho biết tổng số công trình mà mỗi chủ thầu đã thi công tại mỗi thành phố.

Phân tích câu hỏi:

Vì mỗi dòng dữ liệu trong bảng CGTRINH chỉ cho công trình nào thuộc tỉnh thành nào và do chủ thầu nào thi công, nên muốn biết tổng số công trình mà mỗi chủ thầu đã thi công tại mỗi tỉnh thành, thì ta phải nhóm các dòng dữ liệu theo 2 cột TEN_THAU và TINH_THANH, sau đó dùng hàm count để đếm số dòng chính là số công trình của mỗi nhóm.

Câu Truy Vấn:

```
Select   TEN_THAU,  TINH_THANH, Count (STT_CTR)  As  TongSoCTR
From     CGTRINH
Group By   TEN_THAU,  TINH_THANH
```

i. Cho biết họ tên công nhân có tham gia ở tất cả các công trình.

Phân tích câu hỏi:
Vì mỗi dòng dữ liệu trong bảng THAMGIA chỉ cho công nhân nào tham gia công trình nào, nên muốn biết công nhân nào có tham gia hết tất cả các công trình trước hết ta đếm xem mỗi công nhân đã tham gia tổng cộng bao nhiêu công trình, đưa vào bảng #CAU_3i. Sau đó, chọn từ bảng #CAU_3i những công nhân có tổng số công trình tham gia bằng với tổng số công trình ta có, bằng cách đếm số dòng trong bảng CGTRINH.
Câu Truy Vấn:

```
Select   HOTEN_CN,  Count (STT_CTR)  As  TongSoCtrTG
Into     #CAU_3i
From     THAMGIA
Group by       HOTEN_CN

Select   HOTEN_CN          From   #CAU_3i
Where    TongSoCtrTG =
         (Select Count(*) From CGTRINH)
```

IV. BÀI TẬP TỰ LÀM

1. CSDL: Quản lý mua bán hàng hóa (CSDL3)
   1. HANGHOA(MA_HANG,TEN_HG)
   2. DAILY(STT_DL, TEN_DL, DCHI_DL)
   3. MUA(MA_HANG, STT_DL, NGAY_MUA, SOLG_MUA, TRIGIA_MUA)
   4. BAN(MA_HANG, STT_DL, NGAY_BAN, SOLG_BAN, TRIGIA_BAN)

   Tham khảo trang 32 để xem diễn giải cho từng bảng.
2. Yêu cầu 1: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:
   a. Tìm đơn giá mua trung bình của bia Sài gòn tính trên tất cả các đại lý.
   b. Tìm đơn giá mua trung bình của bia Sài gòn tính trên trên từng đại lý
   c. Hiển thị số lượng tồn kho của từng mặt hàng theo từng đại lý.
   d. Tìm tên, địa chỉ của đại lý có tổng giá trị mua trong một ngày lớn hơn 700000.
   e. Với mỗi đại lý, hãy cho biết ngày nào có số lượng mặt hàng bán ra là ít nhất.
   f. Tìm tên và địa chỉ của những đại lý có tổng trị giá bán cao nhất.
   g. Tìm tổng só tiền mà mỗi đại lý đã chi để mua hàng theo từng tháng.
   h. Tìm tên và tổng số lượng bán ra của các mặt hàng đã bán được với số lượng nhiều nhất của từng đại lý.
   i. Cho biết tổng số món hàng mà mỗi đại lý kinh doanh (mua và bán).
   j. Tìm tên đại lý đã mua vào mặt hàng bia Heineken với đơn giá mua là cao nhất.

BÀI 4

I. MỤC ĐÍCH:

Luyện tập câu lệnh SELECT có sử dụng các hàm hàm xử lý chuỗi, hàm xử lý ngày tháng, hàm chuyển kiểu, ...; kết hợp với các dạng câu SELECT của các bài trước.

II. BÀI TẬP CÓ HƯỚNG DẪN:

1. CSDL: Quản lý giảng dạy (CSDL5)

   CHUCDANH(**MACD**, TENCD, GIOCHUAN, TIEN_1TIET)
   Mỗi chức danh có một mã chức danh phân biệt, một tên chức danh, có một số lượng giờ chuẩn tương ứng, và số tiền một tiết trả theo chức danh đó.
   GIAOVIEN(**MAGV**, HOTEN, PHAI, MACD)
   Mỗi giáo viên có một mã giáo viên phân biệt, một họ tên, phái và một mã chức danh tương ứng.
   GD_0506(**MAGV, MONDAY, LOPDAY**, SOTIET, SISO)
   Thông tin về việc giảng dạy của giáo viên trong năm học 05-06 bao gồm mã giáo viên, tên môn dạy, tên lớp dạy, số tiết dạy, sĩ số lớp.
   TAMUNG(**MAGV, NGAY_TA**, SOTIEN)
   Mỗi giáo viên có thể tạm ứng tiền. Khi tạm ứng, thông tin về ngày tạm ứng và số tiền tạm ứng sẽ được ghi nhận lại.

2. Yêu cầu 1: Tìm sự thông thương giữa các bảng trong CSDL

3. Yêu cầu 2: Mở các bảng dữ liệu để xem kiểu dữ liệu của từng trường và quan sát dữ liệu của từng bảng

4. Yêu cầu 3: Hãy viết câu lệnh SQL để trả lời các câu hỏi sau:

   a. In ra danh sách mã giáo viên, họ tên các cán bộ cùng với danh xưng là 'Thầy' nếu cán bộ là nam, và 'Cô' nếu cán bộ là nữ.

   Phân tích câu hỏi:

   Ta thấy danh xưng Thầy hay Cô sẽ phụ thuộc vào cột PHAI trong bảng GIAOVIEN. Ta sẽ dùng cấu trúc CASE đơn giản để kiểm tra cột PHAI và trả về thông tin phù hợp. Tham khảo cú pháp của CASE ở trang 11.

   Câu Truy Vấn:

```
Select MAGV,  Case PHAI
               When 0      Then  'Thay ' + HOTEN
               When 1      Then  'Co ' + HOTEN
End      As   HT
From     GIAOVIEN
```

   b. In ra danh sách gồm 2 cột: Cột 'Họ giáo viên' và cột 'Chữ Lót+Tên giáo viên'

   Phân tích câu hỏi:

   Cột HOTEN trong bảng GIAOVIEN bao gồm cả họ và tên. Để tách họ và tên ra thành 2 cột thì ta dùng các hàm left để lấy họ, hàm right để lấy ra tên. Để tách họ ra, ta cần biết chiều dài của họ, hay nói khác hơn là vị trí khoảng trắng đầu tiên trong chuỗi HOTEN. Ta sẽ sử dụng hàm CharIndex để tìm vị trí này. Còn tên ta sẽ lấy từ bên phải sang một số lượng kí tự bằng chiều dài họ tên trừ đi chiều dài họ. Tham khảo cú pháp của các hàm này ở trang 10.

   Câu Truy Vấn:

```
Select Left(HOTEN, CharIndex(' ', HOTEN)) As Ho_Gv,
         Right (HOTEN, Len(HOTEN) - CharIndex(' ', HOTEN)) As Ten_GV
From GIAOVIEN
```

c. Cho biết danh sách họ tên các giáo viên chỉ dạy cho các lớp điện tử.

Phân tích câu hỏi:

Từ ***chỉ'***trong câu hỏi cho ta biết đây là dạng câu hỏi sử dụng phép Trừ hai tập hợp. Tập hợp thứ nhất (T1) là tập hợp các giáo viên có dạy cho lớp Điện tử, tập hợp thứ hai (T2) là tập hợp các giáo viên có dạy các lớp ghép Điện tử và Tin Học hoặc các lớp khác (như Lý, Chế biến,...). Lấy T1 trừ cho T2. Có nghĩa là T1 là select cha, còn T2 là select con. Điều kiện chứa select con sử dụng từ khóa NOT IN.

Câu Truy Vấn:

```
Select distinct a.MAGV, HOTEN From GD_0506 a, GIAOVIEN b
Where a.MAGV = b.MAGV
   And LOPDAY Like '%DT%'
   And a.MAGV Not In
         (Select MAGV From GD_0506
           Where LOPDAY Like '%TH%'
               or LOPDAY not like '%DT%')
```

d. Tìm các giáo viên có chức danh là Giảng Viên đã dạy cho lớp DT20. (lưu ý: lớp DT-TH20 là lớp ghép của DT20 và TH20).

Câu Truy Vấn:

```
Select HOTEN, LOPDAY From GIAOVIEN a, GD_0506 b, CHUCDANH c
Where a.MAGV = b.MAGV      And a.MACD = c.MACD
   And TENCD = 'Giang Vien'      And   LOPDAY Like '%DT%20%'
```

e. Tìm họ tên các giáo viên có tên lót là Văn, và có tạm ứng tiền ít nhất là 2 lần.

Phân tích câu hỏi:

Vì mỗi dòng của bảng tạm ứng chỉ cho biết giáo viên nào tạm ứng ngày nào, với số tiền bao nhiêu thôi, nên muốn biết giáo viên nào tạm ứng ít nhất hai lần thì ta phải nhóm dữ liệu theo cột MAGV, sau đó đếm số dòng của mỗi nhóm và chỉ chọn nhóm có số dòng lớn hơn 1.

Câu Truy Vấn:

```
Select HOTEN, Count(NGAY_TA) As SoLanTA
From GIAOVIEN a, TAMUNG b
Where a.MAGV = b.MAGV      And   HOTEN   Like  '% Van %'
Group By b.MA_GV, HOTEN
Having count(NGAY_TA) > 1
```

f. Cho biết họ tên các giáo viên đã tạm ứng trong tháng 1 năm 2006?

Câu Truy Vấn:

```
Select HOTEN, NGAY_TA
From GIAOVIEN a, TAMUNG b
Where   a.MAGV = b.MAGV
And   DatePart(month, NGAY_TA) = 1
   And   DatePart(year, NGAY_TA) = 2006
```

g. Cho biết họ tên các giáo viên đã tạm ứng trong 3 tháng gần đây?

Câu Truy Vấn:

```
Select HOTEN, NGAY_TA
From GIAOVIEN a, TAMUNG b
Where   a.MAGV = b.MAGV
   And   DateDiff(month, NGAY_TA, GETDATE()) <= 3
```

h. Cho biết tổng số tiết (đã qui chuẩn) mà giáo viên có tên Nam đã giảng dạy.

Biết rằng:

+ Số tiết qui chuẩn = SOTIET * HESO;

trong đó HESO được tính như sau:

Nếu sĩ số lớp <= 80 thì HESO =1; Nếu 80 < sĩ số lớp <= 120 thì HESO =1.2; Nếu 120 < sĩ số lớp <= 150 thì HESO =1.4; Ngược lại, sĩ số lớp > 150 thì HESO =1.5

Phân tích câu hỏi:

Ở đây ta phải dùng cấu trúc CASE dạng tìm kiếm để tính số tiết qui chuẩn. Tham khảo trang 11

Câu truy vấn:

```
Select Sum( Case
            When SISO <= 80 Then SOTIET
            When SISO <= 120 Then SOTIET * 1.2
            When SISO <= 150  Then SOTIET * 1.4
            Else SOTIET * 1.5
      END)
From GIAOVIEN a, GD_0506 b
Where   a.MAGV = b.MAGV
      And   HOTEN Like '% Nam'
```

i. Tìm họ tên các giáo viên có dạy vượt giờ chuẩn

Phân tích câu hỏi:

Tương tự như câu h, ta cũng phải tính số tiết qui chuẩn của từng giáo viên sau đó chọn ra những giáo viên có tổng số tiết qui chuẩn lớn hơn giờ chuẩn theo chức danh qui định.

Câu truy vấn.

-- Tính tổng số số tiết qui chuẩn cho từng giáo viên, đưa vào bảng tạm #CAU_4i

```
Select MAGV, Sum( Case
            When SISO <= 80 Then SOTIET
            When SISO <= 120 Then SOTIET * 1.2
            When SISO <= 150  Then SOTIET * 1.4
            Else SOTIET * 1.5
      END)   As TongSTQC      Into   #CAU_4i
From GD_0506 b
Group By MAGV
```

-- Hiển thị họ tên các giáo viên có tổng số tiết dạy vượt giờ chuẩn.

```
Select HOTEN, TongSTQC, GIOCHUAN
From  #CAU_4i a, GIAOVIEN b, CHUCDANH c
Where   a.MAGV = b.MAGV And   b.MACD = c.MACD
      And TóngTQC > GIOCHUAN
```

j. Hãy tính số tiền thực lĩnh của tất cả các cán bộ trong trường.

Biết rằng:

+ Cán bộ chỉ được lãnh tiền giảng dạy khi đã dạy vượt giờ chuẩn (nghĩa là số tiết trong giờ chuẩn không được tính tiền).

+ Số tiền giảng dạy thực tế cho một giáo viên là:

Số tiết vượt giờ qui chuẩn * TIEN_1TIET.

+ Thực lĩnh của cán bộ được tính theo công thức như sau:

Nếu cán bộ có dạy vượt giờ và có tạm ứng thì: Thực lĩnh = số tiền giảng dạy - tổng số tiền tạm ứng.Nếu cán bộ có dạy vượt giờ mà không có tạm ứng thì: Thực lĩnh = số tiền giảng dạy Nếu cán bộ không có dạy vượt giờ mà có tạm ứng thì: Thực lĩnh = - tổng số tiền tạm ứng.Nếu cán bộ không có dạy vượt giờ và không có tạm ứng thì: Thực lĩnh = 0.

Phân tích câu hỏi :

Tương tự như câu i, ta cũng tính tổng số giờ dạy chuẩn của mỗi giáo viên, đưa vào bảng tạm #CAU_4j_1.
Tính tổng số tiền tạm ứng của mỗi cán bộ, đưa vào bảng tạm #CAU_4j_2.
Kết nối hai bảng này để tính số tiền thực lĩnh. Tuy nhiên, nếu kết nối tự nhiên bình thường, thì những người có mặt ở một trong hai bảng sẽ biến mất (nghĩa là những người chỉ có dạy hoặc chỉ có tạm ứng), đây là điều ta không muốn, hoặc có những người không có dạy cũng không hề tạm ứng cũng phải xuất ra kết quả. Vì vậy, ở đây ta phải sử dụng kết nối mở rộng.

Lưu ý : *SQL Server 2000 hổ trợ kết nối mở rộng về một phía bằng cách dùng từ khóa left / right outer join.*

Câu hỏi truy vấn :

```
Select MAGV, Sum( Case
      When SISO <= 80 Then SOTIET
      When SISO <= 120 Then SOTIET * 1.2
      When SISO <= 150  Then SOTIET * 1.4
      Else SOTIET * 1.5
END)   As TongSTQC      Into   #CAU_4j_1
From GD_0506 b    Group By MAGV

Select MAGV, Sum(SOTIEN) As TongTA Into #CAU_4j_2
From TAMUNG Group By MAGV

Select a.MAGV, CASE
      When  (TongSTQC > GIOCHUAN) And TongTA is null
            Then TIEN_1TIET * (TongSTQC - GIOCHUAN)
      When  (TongSTQC > GIOCHUAN) And TongTA is not null
            Then (TIEN_1TIET * (TongSTQC - GIOCHUAN) ) - TongTA
      When  (TongSTQC <= GIOCHUAN) And TongTA is null
            Then 0
      Else  -TongTA End
      As ThucLinh
From #CAu_4j_1 a right outer join  GIAOVIEN b on
      a.MAGV = b.MAGV join CHUCDANH c on b.MACD = c.MACD
      left outer join #CAU_4j_2 d on d.MAGV = a.MAGV
```

III. BÀI TẬP TỰ LÀM

1. CSDL: Quản lý câu lạc bộ (CSDL4)

CLB(**MA_CLB**, TEN_CLB, DCHI_CLB)
Mỗi câu lạc bộ có một mã câu lạc bộ phân biệt, một tên và một địa chỉ câu lạc bộ.
DOI (**MA_DOI, MA_CLB**, LT, PHAI)
Trong cùng một câu lạc bộ, mỗi đội có một mã đội phân biệt. Nhưng hai câu lạc bộ có thể có hai đội trùng mã. Mỗi đội thuộc một lứa tuổi, một phái nào đó và thuộc 1 câu lạc bộ.
NHA_TD (**MA_NHATD**, DCHI_NHATD, TG_SOSAN)
Mỗi nhà thi đấu có một mã nhà thi đấu phân biệt, một địa chỉ và tổng số sân nó có.
TGIAN_TD (**LUATUOI**, TGIAN_TD)
Thời gian thi đấu trong một trận đấu được qui định theo lứa tuổi của đội thi đấu.
TRANDAU (**MA_TD**, NGAY_TD, GIO_TD, MA_NHATD)
Mỗi trận đấu có một mã trận đấu phân biệt, thi đấu vào ngày, giờ nào đó và diễn ra tại một nhà thi.
THAMGIA(**MA_TD, MA_DOI, MA_CLB**)
Khi một đội của một câu lạc bộ tham gia thi đấu trong một trận đấu nào đó. Thì thông tin sẽ được lưu lại trong bảng này.
VDV (**MA_VDV**, TEN_VDV, DCHI_VDV, PHAI, LUATUOI, MA_DOI, MA_CLB)

Mỗi vân động viên có một mã vận động viên phân biệt, một tên, một địa chỉ, thuộc một phái, một lứa tuổi nào đó, đồng thời phải cho biết vận động viên đó chơi cho đội nào của câu lạc bộ nào.

2. Yêu cầu 1:

a. Tìm tên tất cả các vận động viên có tham gia thi đấu trong tháng 1 năm 1990.

b. Cho biết thông tin về các trận đấu gồm: mã trận đấu, địa chỉ nhà thi đấu nơi diễn ra trận đấu, ngày thi đấu, thời điểm bắt đầu và kết thúc trận đấu.

c. Tìm các cặp mã trận đấu kết thúc vào cùng ngày giờ.

d. Tìm tên các vận động viên có thi đấu tại nhà thi đấu số 27 trong ngày 900616, từ 10h sáng đến 4h30 chiều.

e. Tìm địa chỉ nhà thi đấu, nơi mà vận động viên Tuấn đang thi đấu vào 3h chiều ngày 901206.

f. Tìm danh sách các trận đấu bắt đầu hoặc kết thúc trong khoảng thời gian từ 12h đến 16h diễn ra trên một sân của nhà thi đấu số 49 trong ngày 900805.

g. Tìm tên các vận động viên có tham gia thi đấu trong các tháng ít ra là như vận động viên Tuấn.

h. Tìm số lượng trận đấu được tổ chức theo từng năm của từng nhà thi đấu.

i. Tìm danh sách các câu lạc bộ có đội thi đấu tại nhà thi đấu số 01 vào các ngày trong tháng 1 năm 90.

j. Tìm danh sách các đội có vận động viên có tên bắt đầu bằng ký tự H, và có thi đấu trong tháng 8 năm 90.

BÀI 5

I. MỤC ĐÍCH:

Luyện tập câu lệnh tạo CSDL, tạo bảng với một số ràng buộc cơ bản, lệnh xen, sửa, xóa dữ liệu.

II. LÝ THUYẾT:

Câu lệnh SQL sử dụng trong bài này:

1. Tạo CSDL:
   CREATE DATABASE <tên CSDL mới>
2. Chọn CSDL hiện hành:
   USE <tên CSDL>
3. Tạo bảng:
   Lệnh tạo bảng với ràng buộc trên cột:

```
CREATE TABLE  <tên_bảng_mới> (
    { <tên_cột>  {<kiểu> | <tên_miền_trị>}
          [ NOT NULL ]
          [ PRIMARY KEY | UNIQUE ]
          [ DEFAULT <trị_mặc_định>]
          [ CHECK (<điều_kiện>)]
          [ REFERENCES <tên bảng cha>
                [(<danh sách tên cột của khóa chính trong bảng cha>)]
                [ON UPDATE <hành_động>]
                [ON DELETE <hành_động>]      ]
      [,...]   }    )
```

   Lệnh tạo bảng với ràng buộc trên bảng:

```
CREATE TABLE  <tên_bảng> (      { <tên_cột>   {<kiểu> | <tên_miền_trị>} [,...]    }
    [, PRIMARY KEY (<danh_sách_cột>) ]
    {  [, UNIQUE (<danh_sách_cột>)]        [, ...]   }
    {   [, FOREIGN KEY (<ds_cột_khóa_ngoại>)
          REFERENCES  <tên_bảng_cha> [(danh sách thuộc tính khóa)],
          [ON UPDATE <hành_động>]
          [ON DELETE <hành_động>] ]
          [,...]    }
    {   [, CHECK (<điều_kiện>)                [,...]    }
)
```

4. Sửa bảng:

```
ALTER TABLE <tên_bảng>
    [ ADD  <tên_cột>    {<kiểu | <miền_trị>}  <ràng buộc trên cột>       ]
|   [DROP  <tên_cột>
|    [ALTER COLUMN <tên_cột> <kiểu mới>
|   [ADD [CONSTRAINT [tên_RB] ] <định_nghĩa_RB>]
|   [DROP CONSTRAINT tên_RB]
```

5. Xóa bảng:
   DROP TABLE <tên_bảng>

6. Xen, sửa, xóa dữ liệu:

Xen *một dòng* vào bảng đã tồn tại:

INSERT INTO <tên bảng>
[(<tên cột 1> [,<tên cột 2> [,...]])]
VALUES (<biểu thức 1> [,<biểu thức 2> [,...]])

Xen kết quả truy vấn vào bảng đã tồn tại:

INSERT INTO <Tên bảng>
[(<tên cột 1> [,<tên cột 2> [,...]])]
<lệnh SELECT>

Cập nhật dữ liệu:

UPDATE <Tên bảng>
SET <Tên Cột> = <biểu thức>
[, <Tên cột> = <biểu thức> [,..]]
[WHERE <Điều kiện chọn các dòng cập nhật>]

Xóa dữ liệu:

DELETE <Tên bảng>
[WHERE <Điều kiện chọn dòng muốn xóa>]

III. BÀI TẬP CÓ HƯỚNG DẪN:

1. Cài đặt cơ sở dữ liệu quản lý bán hàng cùng với các ràng buộc khóa chính, khóa ngoại, ràng buộc miền trị,…

Khach (makh, tenkh, dchi_kh, dthoai_kh, congno)

Mỗi khách hàng có một mã phân biệt, một tên, địa chỉ, điện thoại và công nợ.

Hoadon (so_hd, ngay_hd, ngayxuat, trigia, makh)

Mỗi hóa đơn có một số hóa đơn phân biệt, có một ngày ghi hóa đơn, ngày xuất hàng, trị giá hóa đơn và xuất cho một khách hàng duy nhất. Hàng hóa chỉ xuất sau khi đã lập hóa đơn.

Ct_hd (so_hd, mahh, gia_ban, sl, thanhtien)

Một hóa đơn bán nhiều mặt hàng, mỗi mặt hàng có một số lượng bán, giá bán và thành tiền. Thành tiền chính là giá bán x số lượng.

Phieuthu (so_pt, makh, ngaythu, sotien)

Mỗi khi thu tiền của một khách hàng, một phiếu thu được xuất với số phân biệt, có ghi nhận ngày thu và số tiền thu.

Tạo CSDL và chọn CSDL hiện hành:

```
CREATE DATABASE QLBH;
Go    -- Kết thúc gói (batch), thực hiện xong trước khi thực hiện lệnh kế
USE QLBH;
```

Các câu lệnh tạo bảng:

```
CREATE TABLE Khach (makh INT PRIMARY KEY IDENTITY,
      tenkh char(40) not null, dchi_kh char(50),
      dthoai_kh char(15) CHECK (dthoai_kh LIKE
```

```
'([0-9][0-9][0-9])-[0-9][0-9][0-9][0-9][0-9][0-9][0-9]', congno money)
      CREATE TABLE Hoadon (so_hd INT PRIMARY KEY IDENTITY,
            ngay_hd DATETIME CHECK (ngay_hd <= getdate()),
            ngayxuat DATETIME CHECK (ngayxuat <= getdate()),
            trigia MONEY NOT NULL, makh INT NOT NULL REFERENCES Khach,
            CHECK(ngay_hd <= ngayxuat)     )
      CREATE TABLE Ct_hd (so_hd INT REFERENCES Hoadon NOT NULL,
            mahh CHAR(5) NOT NULL, gia_ban MONEY,
            sl INT CHECK (sl>0), thanhtien AS gia_ban * sl PERSISTED,
            PRIMARY KEY (so_hd, mahh))
      CREATE TABLE PhieuThu (So_pt INT PRIMARY KEY IDENTITY,
            makh INT REFERENCES Khach NOT NULL,
            ngay_thu DATETIME NOT NULL CHECK (ngay_thu <=getdate()),
            sotien MONEY)
```

Lưu ý:
- Kiểu dữ liệu của khóa chính và khóa ngoại phải giống nhau.
- Từ khóa IDENTITY qui định giá trị của cột là do hệ thống tự tăng.
- Cột thanhtien của bảng ct_hd là cột có giá trị được tính toán được (computed column) từ biểu thức đặc tả sau từ khóa AS, PERSISTED qui định cột này sẽ được lưu vật lý.

2. Sửa bảng thêm ràng buộc duy nhất cho cột số điện thoại của bảng Khach, đặt trị mặc định cho các cột ngày ghi hóa đơn và ngày xuất hàng là ngày hiện tại.

```
ALTER TABLE Khach ADD CONSTRAINT uq_sdt UNIQUE (dthoai_kh);
ALTER TABLE Hoadon ADD CONSTRAINT df_ngay_hd DEFAULT getdate() FOR ngay_hd;
ALTER TABLE hoadon ADD CONSTRAINT df_ngay_xuat DEFAULT getdate() FOR ngayxuat;
```

3. Xóa đi ràng buộc vừa tạo

```
ALTER TABLE Hoadon DROP CONSTRAINT df_ngay_hd
```

4. Thêm dữ liệu vào các bảng bằng lệnh

```
insert into Khach (tenkh, dthoai_kh) values ('Nguyen van an','0123456789');

select * from Khach;

insert into Hoadon (ngay_hd, ngayxuat,trigia, makh)
               values('1-feb-2011', '2-feb-2011', 1000, 1);
insert into Hoadon (trigia, makh)
               values (2000, 1);
select * from Hoadon;

insert into Ct_hd values (1, 'BGOMO', 50,10);
insert into Ct_hd values (1, 'BGTID', 100,5);
```

Lưu ý: Trong lệnh insert, cột tính toán 'thành tiền' sẽ không được đưa giá trị vào.

```
select * from Ct_hd;  --xem cột thành tiền đã có giá trị chưa?
```

5. Cập nhật lại ngày xuất hàng của hóa đơn 1 là ngày 3-feb-2011

```
Update Hoadon set ngayxuat = '3-feb-2011' where so_hd = 1
```

6. Xóa toàn bộ dữ liệu vừa nhập

```
delete ct_hd;
delete Hoadon;
delete Khach;
```

Hỏi: Có thể xóa dữ liệu trong các bảng theo trình tự các lệnh delete khác đi không? Tại sao?

IV. BÀI TẬP TỰ LÀM:

Sử dụng CSDL1:

1. Hãy cài đặt tất cả các ràng buộc khóa chính, khóa ngoại, khóa duy nhất, miền giá trị cho từng bảng.
2. Trong bảng CongNhan, hãy cập nhật giá trị cho các cột năm sinh và năm vào nghề thành 4 con số (VD: 60 -> 1960).

   Trong bảng ThamGia:

3. Hãy thêm cột ThamNien (số năm làm việc tính đến thời điểm tham gia công trình, >0), DonGia (đơn giá ngày công, >0) và cột TienCong (tiền công) ràng buộc >0 và là cột tính toán từ đơn giá ngày công và số ngày tham gia.
4. Cập nhật dữ liệu cho cột ThamNien theo ngày tham gia và năm vào nghề.
5. Cập nhật dữ liệu cho cột DonGia theo thâm niên như sau:

   Thâm niên >20 năm => đơn giá = 100

   Thâm niên >15 năm => đơn giá = 80

   Thâm niên>10 năm => đơn giá = 50
6. Quan sát cột TienCong trong bảng xem có giá trị chưa.

BÀI 6

I. MỤC ĐÍCH:

Bài tập tổng hợp

II. CSDL: QUẢN Lý DỰ Án (CSDL6)

NHA_CCAP(**MA_NCC**, TEN_NCC, VON, TH_PHO)
Mỗi nhà cung cấp có một mã nhà cung cấp, một tên, một số vốn điều lệ (đơn vị: triệu đồng), phải có vốn điều lệ tối thiểu là 10 triệu, và đóng tại một thành phố nào đó, mặc định có giá trị là 'Can Tho'.
HANG_HOA(**MA_HANG**, TEN_HANG, MAU, TR_LUONG)
Mỗi hàng hóa hay mặt hàng có một mã hàng, một tên hàng, có một màu và một trọng lượng duy nhất (phải >0).
DU_AN(**MA_DU_AN**, TEN_DU_AN, TH_PHO)
Mỗi dự án có một mã, một tên và thuộc về một thành phố nào đó.
CUNG_UNG(**MA_NCC, MA_HANG, MA_DU_AN**, SO_LUONG)
Một nhà cung cấp sẽ cung ứng nhiều mặt hàng cho một dự án với số lượng cung cấp khác nhau (phải >0).

III. CÂU HỎI:

1. Sau khi phục hồi CSDL6, hãy thêm tất cả các ràng buộc cho từng bảng.
2. Trong bảng Cung Ứng, hãy thêm cột đơn giá (>0) và cột thành tiền trong đó cột thành tiền là cột có giá trị được tính toán từ đơn giá và số lượng.
3. Trong bảng Cung Ứng, hãy cập nhật đơn giá cho mặt hàng p1 là 10, quan sát cột thành tiền.
4. Tìm thông tin đầy đủ gồm mã, tên và thành phố của mỗi dự án ở TP HCM.
5. Tìm mã và tên các mặt hàng được cung ứng cho dự án Tân an.
6. Tìm tên và vốn của các nhà cung cấp đã cung ứng mặt hàng Sơn Bạch Tuyết cho dự án Tân Phú. Cột vốn cần ghi thêm đơn vị tiền tệ đi kèm.
7. Tìm tên các dự án do nhà cung cấp S1 cung ứng.
8. Tìm tên nhà cung cấp đã cung ứng các mặt hàng màu đỏ cho dự án J1.
9. Tìm tên của các mặt hàng mà không có mặt hàng nào có trọng lượng nhỏ hơn.
10. Tìm tên nhà cung cấp đã cung ứng cho cả hai dự án J1 và J2.
11. Hiển thị số mặt hàng mà mỗi nhà cung cấp đã cung ứng cho các dự án.
12. Tìm số lượng trung bình các mặt hàng mã mỗi nhà cung cấp có thể cung ứng.
13. Với mỗi mặt hàng, hãy hiển thị số lượng cung ứng mà mỗi nhà cung cấp đã cung ứng cho các dự án.
14. Đối với mỗi mặt hàng cung ứng cho một dự án, tìm mã hàng, tên dự án và tổng số lượng hàng tương ứng.
15. Đối với mỗi mặt hàng cung ứng cho một dự án, tìm tên hàng, tên dự án và tổng số lượng hàng tương ứng.
16. Tìm tên nhà cung cấp có vốn đầu tư cao nhất.
17. Tìm tên các nhà cung cấp đã cung ứng cho một dự án bất kỳ ở thành phố Hồ Chí Minh.
18. Tìm mã của các nhà cung cấp đã cung ứng cho ít nhất hai dự án có cùng thành phố với nhà cung cấp đó.
19. Tìm tên các dự án không được cung ứng bất cứ mặt hàng màu đỏ nào bởi một hãng ở thành phố Hồ Chí Minh.

20. Tìm mã của các nhà cung cấp cung ứng ít nhất là một mặt hàng, các mặt hàng này được cung ứng ít nhất bởi một nhà cung cấp, nhà cung cấp này cung cấp ít nhất là một mặt hàng màu đỏ.
21. Tìm mã của các dự án đang sử dụng ít nhất là một mặt hàng mà S1 có khả năng cung ứng.
22. Tìm tên dự án đã được cung ứng mặt hàng Xi Măng Hoàng Thạch với số lượng lớn nhất.
23. Tìm mã các dự án được cung ứng hoàn toàn bởi hãng S2.
24. Tìm mã các dự án chỉ sử dụng những mặt hàng mà hãng S1 có khả năng cung ứng.
25. Tìm tổng số lượng của mỗi mặt hàng mà mỗi dự án đã được cung cấp.
26. Tìm tổng số dự án được hãng S3 cung ứng.
27. Tìm tổng số dự án mà mỗi nhà cung cấp đã cung ứng.
28. Tìm tổng số lượng mặt hàng P1 do hãng S1 cung ứng.
29. Tìm trung bình số lượng cho mặt hàng P1 được cung ứng cho một dự án.
30. Tìm trung bình số lượng cho mặt hàng P1 được cung ứng bởi một hãng.
31. Tìm trung bình số lượng cho mặt hàng P1 được cung ứng cho một dự án bởi một hãng.
32. Tìm tên nhà cung cấp đã cung ứng cho ít nhất là 3 dự án.
33. Tìm số lượng các nhà cung cấp đã cung ứng cho các dự án tại ít nhất là 3 thành phố.
34. Tim số lượng trung bình các bao xi măng (bất kể hiệu gì, màu gì) mã mỗi dự án đã tiêu thụ.
35. Tìm tên dự án đã được cung ứng mặt hàng sơn (bất kể hiệu gì, màu gì) với tổng số lượng cao nhất.
36. Tìm mã của những hãng nào cung ứng cùng một mặt hàng cho tất cả các dự án.
37. Tìm mã các dự án được hãng S1 cung ứng cho tất cả các mặt hàng mà hãng S1 có khả năng cung ứng.
38. Tìm mã các dự án được cung ứng bởi tất cả những hãng nào cung ứng mặt hàng màu đỏ.
39. Tìm mã các mặt hàng được cung ứng cho tất cả các dự án ở TP HCM.

TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA TOÁN - TIN HỌC

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

(*Bài giảng tóm tắt*)

NGƯỜI BIÊN SOẠN

TẠ THỊ THU PHƯỢNG

Đà Lạt 2009

MỤC LỤC

Chương 1: Tổng quan về Hệ quản trị cơ sở dữ liệu ..... Trang 1
I. Giới thiệu 1
II. Cấu trúc và thành phần của hệ quản trị cơ sở dữ liệu 2
Chương 2: Xây dựng, quản lý và khai thác cơ sở dữ liệu 5
I. Khái niệm cơ sở dữ liệu 5
II. Tạo cơ sở dữ liệu. 5
III. Kiểu dữ liệu. 6
IV. Tạo và quản lý bảng. 7
V. Các thao tác trên dữ liệu 11
VI. Truy vấn dữ liệu 12
VII. Tạo và sử dụng khung nhìn (View) 14
VIII. Tạo và sử dụng chỉ mục (Index) 14
IX. Chuyển đổi dữ liệu với các ứng dụng khác 18
Chương 3: T-SQL nâng cao 19
I. Khai báo và sử dụng biến 19
II. Cấu trúc điều khiển 20
III. Thủ tục thường trú (Stored Procedures) 22
IV. Kiểu dữ liệu cursor 26
V. Hàm người dùng (User Defined Functions) 32
VI. Triggers và cài đặt ràng buộc dữ liệu 35
Chương 4: Bảo mật và an toàn dữ liệu 40
I. Bảo mật trong hệ quản trị cơ sở dữ liệu 40
II. Bản sao dữ liệu 46
III. Sao lưu và khôi phục dữ liệu 59
IV. Quản lý giao dịch 61
Chương 5: Lập trình cơ sở dữ liệu 92
I. Lập trình ADO.NET 92
II. Thiết kế chức năng đọc/ ghi dữ liệu 95
III. Tạo báo biểu với Crystal Report 98
Bài tập 105

http://www.ebook.edu.vn

Chương 1
TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

I. Giới thiệu

Thông tin là nguồn tài nguyên quý giá của một tổ chức. Các phần mềm máy tính là những công cụ hiệu quả để xử lý thông tin và hệ quản trị cơ sở dữ liệu là công cụ phổ biến cho phép lưu trữ và rút trích thông tin một cách hiệu quả.

Hệ quản trị cơ sở dữ liệu quan hệ là hệ quản trị cơ sở dữ liệu phổ biến nhất hiện nay và được hỗ trợ bởi nhiều nhà cung cấp phần mềm. Tính hiệu quả của các ứng dụng phụ thuộc vào chất lượng của việc tổ chức dữ liệu. Những cải tiến trong kỹ thuật và xử lý cơ sở dữ liệu đưa đến các cơ hội sử dụng thông tin một cách linh hoạt và hiệu quả khi dữ liệu được tổ chức và lưu trữ trong các cấu trúc quan hệ. Hệ quản trị cơ sở dữ liệu là một thành công trong lĩnh vực thương mại.

Mục tiêu của hệ quản trị cơ sở dữ liệu.

Hệ quản trị cơ sở dữ liệu phải đảm bảo các mục tiêu sau: dữ liệu sẵn dùng (*data availability*), tính toàn vẹn dữ liệu (*data integrity*), an toàn dữ liệu (*data secutity*), và độc lập dữ liệu (*data independency*).

- Dữ liệu sẵn dùng (*data availability*): dữ liệu được tổ chức sao cho mọi người dùng có thể truy cập dễ dàng theo chức năng và nhiệm vụ của họ.
- Tính toàn vẹn dữ liệu (*data integrity*): dữ liệu lưu trữ trong cơ sở dữ liệu là đúng đắn, đáng tin cậy.
- An toàn dữ liệu (*data secutity*): Chỉ những người dùng được phép mới có thể truy cập dữ liệu. Nếu nhiều người dùng truy cập chung một mục dữ liệu cùng lúc thì hệ quản trị cơ sở dữ liệu không cho phép họ thực hiện những thay đổi gây mâu thuẫn dữ liệu.
- Độc lập dữ liệu (*data independency*): hệ quản trị cơ sở dữ liệu phải cho phép tất cả mọi người dùng được phép lưu trữ, cập nhật và rút trích dữ liệu hiệu quả mà không cần nắm chi tiết về cấu trúc của cơ sở dữ liệu được biểu diễn và cài đặt.

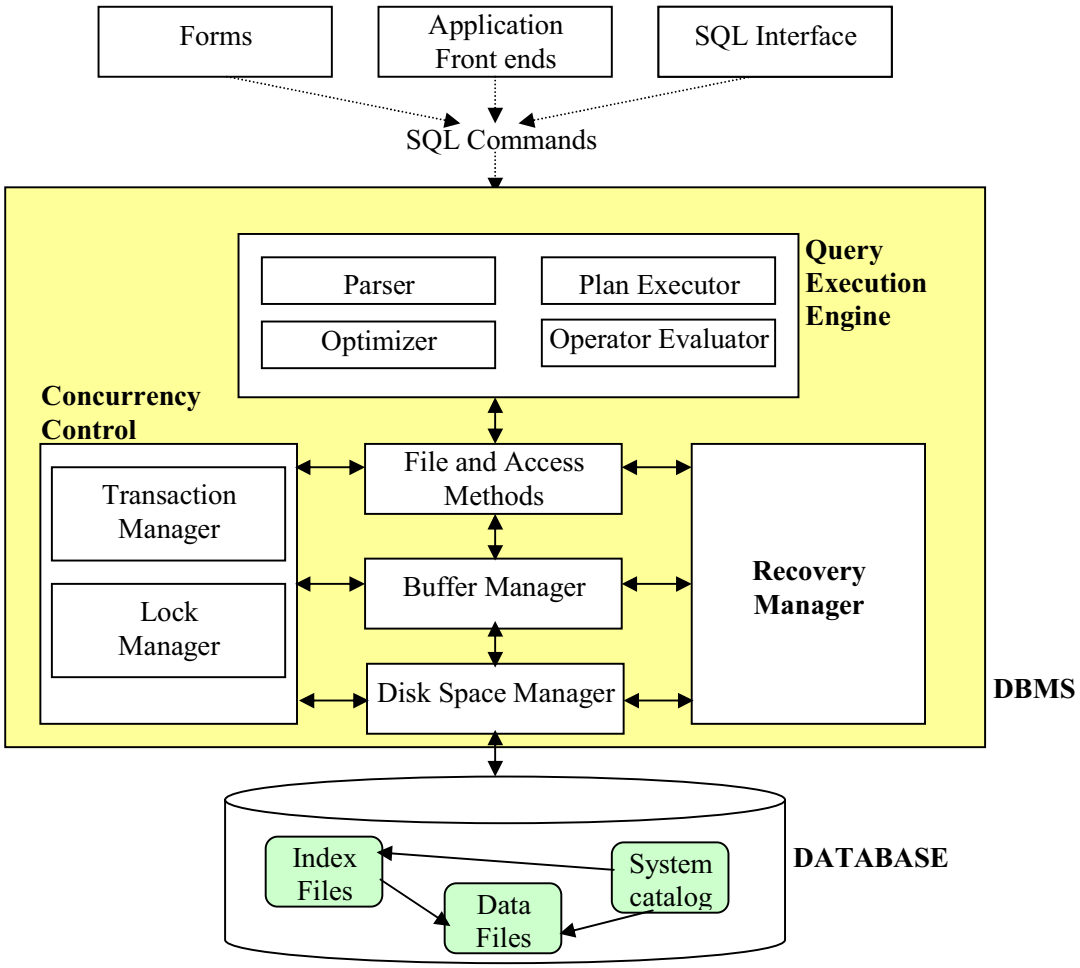
Quá trình phát triển của hệ quản trị cơ sở dữ liệu.

Quá trình phát triển của DBMS như sau:
- Flat files: 1960s – 1980s
- Hierarchical: 1970s –1990s
- Network : 1970s – 1990s
- Relational: 1980s – đến nay
- Object-oriented: 1990s – đến nay

http://www.ebook.edu.vn

- Object-relational: 1990s – đến nay
- Data warehousing: 1980s – đến nay
- Web-enabled: 1990s – đến nay

II. Cấu trúc và thành phần của hệ quản trị cơ sở dữ liệu.



Hình 1.1 Kiến trúc của DBMS

Kiến trúc của hệ quản trị cơ sở dữ liệu gồm 2 thành phần chức năng:

- Bộ quản lý lưu trữ (Storage manager).
- Bộ Xử lý truy vấn (Query Processor).

1. Bộ quản lý lưu trữ

Bộ quản lý lưu trữ có nhiệm vụ lưu trữ, rút trích và cập nhật dữ liệu vào cơ sở dữ liệu. Bộ quản lý lưu trữ gồm có các đơn vị sau:

- Kiểm tra chứng thực và toàn vẹn.
- Quản lý giao dịch .
- Quản lý file.

http://www.ebook.edu.vn

- Quản lý vùng đệm.

Quản lý giao dịch (Transaction management)

Thông thường, một số thao tác trên cơ sở dữ liệu tạo thành một đơn vị logic công việc. Ta hãy xét ví dụ chuyển khoản, trong đó một số tiền x được chuyển từ tài khoản A (A:=A-x) sang một tài khoản B (B:=B+x). Một yếu tố cần thiết là cả hai thao tác này hoặc cùng xảy ra hoặc không hoạt động nào xảy ra cả. Việc chuyển khoản phải xảy ra trong tính toàn thể của nó hoặc không. Yêu cầu ***toàn thể-hoặc-không*** này được gọi là tính nguyên tố (atomicity). Một yếu tố cần thiết khác là sự thực hiện việc chuyển khoản bảo toàn tính nhất quán của cơ sở dữ liệu: *giá trị của tổng A + B phải được bảo toàn*. Yêu cầu về tính chính xác này được gọi là tính nhất quán (consistency). Cuối cùng, sau khi thực hiện thành công hoạt động chuyển khoản, các giá trị của các tài khoản A và B phải bền vững cho dù có thể có sự cố hệ thống. Yêu cầu về tính bền vững này được gọi là tính lâu bền (durability).

Một giao dịch là một tập các hoạt động thực hiện chỉ một chức năng logic trong một ứng dụng cơ sở dữ liệu. Mỗi giao dịch là một đơn vị mang cả tính nguyên tố lẫn tính nhất quán. Như vậy, các giao dịch phải không được vi phạm bất kỳ ràng buộc nhất quán nào: ***Nếu cơ sở dữ liệu là nhất quán khi một giao dịch khởi động thì nó cũng phải là nhất quán khi giao dịch kết thúc thành công***. Tuy nhiên, trong khi đang thực hiện giao dịch, phải cho phép sự không nhất quán tạm thời. Sự không nhất quán tạm thời này tuy là cần thiết nhưng lại có thể dẫn đến các khó khăn nếu xảy ra sự cố.

Trách nhiệm của người lập trình là xác định đúng đắn các giao dịch sao cho bảo toàn tính nhất quán của cơ sở dữ liệu.

Đảm bảo tính nguyên tố và tính lâu bền là trách nhiệm của hệ cơ sở dữ liệu nói chung và của ***thành phần quản trị giao dịch (transaction-management component)*** nói riêng. Nếu không có sự cố, tất cả giao dịch hoàn tất thành công và tính nguyên tố được hoàn thành dễ dàng. Tuy nhiên, do sự hiện diện của các sự cố, một giao dịch có thể không hoàn tất thành công sự thực hiện của nó. Nếu tính nguyên tố được đảm bảo, một giao dịch thất bại không gây ảnh hưởng đến trạng thái của cơ sở dữ liệu. Như vậy, cơ sở dữ liệu phải được hoàn lại trạng thái của nó trước khi giao dịch bắt đầu. Hệ quản trị cơ sở dữ liệu phải có trách nhiệm phát hiện sự cố hệ thống và trả lại cơ sở dữ liệu về trạng thái trước khi xảy ra sự cố.

Khi một số giao dịch tương tranh cập nhật cơ sở dữ liệu, tính nhất quán của dữ liệu có thể không được bảo toàn, ngay cả khi mỗi giao dịch là chính xác. ***Bộ quản trị điều khiển tương tranh (concurency-control manager)*** có trách nhiệm điều khiển các tương tác giữa các giao dịch đồng thời để đảm bảo tính thống nhất của CSDL.

http://www.ebook.edu.vn

***Thành phần Kiểm tra chứng thực và toàn vẹn* (*Authorization and Integrity Manager*)**

Kiểm tra ràng buộc toàn vẹn và quyền truy cập dữ liệu của người dùng cơ sở dữ liệu.

***Thành phần quản lý giao dịch* (*Transaction manager*)**

Thành phần này đảm bảo rằng cơ sở dữ liệu luôn ở trạng thái nhất quán. Nó quản lý việc thực thi các yêu cầu thao tác dữ liệu và đảm bảo các truy cập dữ liệu đồng thời không dẫn đến mâu thuẫn.

Thành phần quản lý file (File manager): quản lý việc cấp phát không gian trên đĩa. Các file được dùng để chứa tập các dữ liệu tương tự nhau. Hệ quản lý file quản lý các file độc lập, giúp đỡ nhập và lấy các mẩu tin. Thành phần quản lý file thiết lập và duy trì danh sách các cấu trúc và chỉ mục được định nghĩa trong lược đồ trong. Thành phần quản lý file có thể:

- Tạo file.
- Xóa file.
- Cập nhật mẩu tin trong file.
- Lấy một mẩu tin từ một file.

***Thành phần quản lý vùng đệm* (*Buffer Manager*)**: có trách nhiệm chuyển dữ liệu từ đĩa lưu trữ vào bộ nhớ chính theo yêu cầu của chương trình.

2. **Bộ xử lý truy vấn (Query Processor)**

Thực hiện câu truy vấn nhận được từ người dùng qua các giai đoạn phân tích (parser), tối ưu hóa câu hỏi (query optimizer), lập kế hoạch thực hiện (plan executor) và thực hiện tính toán (operator evaluator).

http://www.ebook.edu.vn

Chương 2
XÂY DỰNG, QUẢN LÝ VÀ KHAI THÁC CƠ SỞ DỮ LIỆU

I. Khái niệm cơ sở dữ liệu

- Ở mức logic, một cơ sở dữ liệu (CSDL) gồm:
  - Các bảng (tables) chứa dữ liệu có cấu trúc và các ràng buộc (constraint) định nghĩa trên các bảng.
  - Các khung nhìn (view).
  - Các thủ tục/ hàm.
  - Các vai trò (role) và người dùng (user).
  - …
- Ở mức lưu trữ vật lý, một database của SQL Server được lưu trữ bởi 3 loại tập tin:
  - Tập tin dữ liệu (data file) gồm có:
    - ✓ 1 tập tin dữ liệu chính (primary data file), thường có phần mở rộng "mdf": chứa các dữ liệu khởi đầu của database.
    - ✓ 0-n tập tin dữ liệu thứ cấp (secondary data file), thường có phần mở rộng "ndf": chứa các dữ liệu không lưu trữ hết trong tập tin dữ liệu chính.
  - Tập tin nhật ký giao tác (transaction log file) gồm có 1-n tập tin nhật ký, thường có phần mở rộng "ldf": *chứa các thông tin về nhật ký giao tác, dùng để phục hồi database sau khi xảy ra sự cố*.

II. Tạo cơ sở dữ liệu

1. Cú pháp lệnh tạo CSDL

Create Database *database_name*
[On [Primary]
{ *file_spec* [,…n] }
]
[Log on
{ *file_spec* [,…n] }
]

với

file_spec :: = (Name = *logical_file_name*,
Filename = '*os_file_name* '

http://www.ebook.edu.vn

[, Size = *size*]
[, Maxsize = { *max_size* | Unlimited }]
[, Filegrowth = *growth_increment*])

Mặc định, các tập tin dữ liệu và log được lưu trong thư mục MSSQL\ Data của thư mục cài đặt SQL Server.

Ví dụ

- Ví dụ 1: tạo CSDL QLSinhVien theo các quy định mặc định của SQL Server

```
Create Database QLSinhVien
```

- Ví dụ 2: tạo CSDL QLSinhVien với khai báo tên file logic, thư mục lưu tập tin dữ liệu chính, kích thước, …

```
Create Database QLSinhVien
On
(       Name = QLSV_Data
        Filename  = 'C:\ ...\ QLSV_Data.mdf ',
        Size = 1,
        Filegrowth = 10% )
```

- Ví dụ 3

```
Create Database QLSinhVien
On
(       Name = QLSV_Data1,
        Filename  = 'C:\ ...\ QLSV_Data.mdf ',
        Size = 1,
        Maxsize = 10 MB,
        Filegrowth = 1 MB   ) ,
(       Name = QLSV_Data2 ,
        Filename = 'C:\...\QLSV_Data1.ndf' )
Log on
(       Name = QLSV_Log,
        Filename = 'D:\...\QLSV_Log.ldf')
```

2. Xoá một CSDL đã tồn tại

```
Drop Database database_name
```

3. Thay đổi một CSDL

```
Alter Database database_name ….
```

Dùng để:
- Thêm/xoá/thay đổi các tập tin.
- Thay đổi các tùy chọn cho CSDL.

III. Kiểu dữ liệu

SQL Server cung cấp các kiểu dữ liệu:

1. Số

- Số nguyên: bit, tinyint, smallint, int, bigint.

http://www.ebook.edu.vn

– Số thực
  ✓Floating point:
    o *float(n)*
    o *real = float(24)*
  ✓Fixed point
    o *Decimal(p,s)*
    o *Numeric(p,s)*

2. Chuỗi

✓char(n): chuỗi có độ dài cố định.

✓ nchar(n): chuỗi (theo mã Unicode) có độ dài cố định.

✓varchar(n): chuỗi có độ dài thay đổi.

✓nvarchar(n): chuỗi (theo mã Unicode) có độ dài thay đổi.

✓text: kiểu dữ liệu cho phép chứa chuỗi có kích thước hơn 8KB.

✓ntext: kiểu dữ liệu cho phép chứa chuỗi (theo mã Unicode) có kích thước hơn 8KB.

3. Ngày giờ

✓Datetime.

✓Smalldatetime

4. *Kiểu người dùng tự định nghĩa*

a. Định nghĩa một kiểu dữ liệu:

sp_addtype *type_name*, *system_type* [, '*null_type*'][, '*owner*']

Ví dụ: định nghĩa kiểu dữ liệu Code là kiểu chuỗi gồm 10 ký tự cho phép để trống

Exec sp_addtype Code, char(10), 'NULL'

b. Xóa một kiểu dữ liệu người dùng định nghĩa:

sp_droptype '*type_name'*

IV. Tạo và quản lý bảng

1. Tạo bảng

– Xác định các cột (các thuộc tính) của bảng.

– Xác định khóa chính.

– Xác định các thuộc tính null/ not null.

– Xác định thuộc tính identity (nếu có) (phải là kiểu số nguyên).

Lưu ý:

– *Luôn tạo khóa chính cho một bảng.*

– *Ràng buộc khóa ngoại nên được tạo sau khi đã tạo xong tất cả các bảng liên quan.*

a. Cú pháp lệnh tạo bảng

http://www.ebook.edu.vn

Create table *Table_name*
(
{ *Column_name* *Data_type* [null | not null]
[default *default_value*]
[identity [(*seed*, *increment*)]]
} [,…n]
[, constraint *constraint_name* primary key (*Column_name* [,…n])]
)

Ví dụ: Tạo bảng học sinh có khóa chính là (STT, Lop)

```
Create table HOCSINH
(       STT         tinyint not null,
        Lop         char(5) not null  default  '11A1' ,
        HoTen       nvarchar(30) not null,
        NgaySinh    datetime not null,
        DiaChi      nvarchar(100),
        constraint  pk_HS primary key (STT, Lop)
)
```

b. Thay đổi cấu trúc bảng / xóa bảng

✓ Thay đổi cấu trúc bảng là thực hiện:

- Thêm/ xoá/ cập nhật kiểu dữ liệu của một cột (column).
- Thêm/ xoá/ kiểm tra/ không kiểm tra ràng buộc (constraint).
- Cho phép/ không cho phép trigger hoạt động.

Cú pháp: Alter table <tên_bảng>

…

✓ Xóa bảng: xoá dữ liệu và cấu trúc của bảng

Cú pháp: Drop table <tên_bảng>

Ví dụ

- Thêm thuộc tính DanToc vào bảng HOCSINH:

```
Alter table HOCSINH
        Add DanToc nvarchar(20) null default 'Kinh'
```

- Sửa kiểu dữ liệu của thuộc tính NgaySinh thành kiểu SmallDatetime**:**

```
Alter table HOCSINH
        Alter column NgaySinh SmallDatetime not null
```

2. Quản lý bảng

- Các tên bảng, tên ràng buộc không được trùng nhau trong cùng một database.

http://www.ebook.edu.vn

- Tên các cột trong cùng một bảng không được trùng nhau.
- Thông tin về các bảng, các ràng buộc được lưu trong *bảng hệ thống* ***sysobjects***

  Ví dụ: đọc thông tin về các bảng trong database hiện hành:

  Select * from sysobjects where type = 'U'
- Một số thủ tục SQL Server cung cấp để quản lý bảng và cấu trúc bảng:
  - sp_databases
  - sp_tables ['*table_name*'] [, '*owner*'][,'*database_name*'][, " '*type*' "]

    Ví dụ: Exec sp_tables null, null, null, " 'TABLE' "
  - sp_help [*object_name*]
    - ✓ sp_help cho biết các thông tin về đối tượng bất kỳ trong database (đối tượng có chứa trong sysobjects).

      Ví dụ: Exec sp_help HOCSINH
  - sp_columns *object* [, *owner*] [, *database*] [,*column*]
  - sp_helpconstraint '*table_name*'
  - …

3. Cài đặt ràng buộc toàn vẹn đơn giản

SQL Server cung cấp sẵn cơ chế để kiểm tra các loại ràng buộc toàn vẹn (RBTV) sau:

- Khóa chính (primary key constraint).
- Khóa ngoại (foreign key constraint).
- Giá trị duy nhất (unique constraint).
- Check constraint (Kiểm tra ràng buộc miền giá trị).

Có thể khai báo ràng buộc trong lúc tạo bảng hoặc khi bảng đã tồn tại. *Thông thường nên khai báo ràng buộc toàn vẹn trước khi nhập dữ liệu.*

a. Khai báo ràng buộc trong lúc tạo bảng

Cú pháp:

```
Create table Table_name
(   …
        [, constraint Constraint_name
                    {  primary key (Column_name [,…n])
                     | unique ( Column_name [,…n])
                       | check ( logical_expression ) }
        ] […n]
)
```

Ví dụ

http://www.ebook.edu.vn

```
Create table SinhVien
(
        MaSV        char(10) not null,
        HoTen       nvarchar(30) not null,
        Nam         tinyint,
        CMND        char(10),
        Khoa        char(5),
        constraint pk_SV primary key (MaSV),
        constraint u_CMND unique (CMND),
        constraint chk_Nam check (Nam > 0 and Nam <= 4)
)
```

b. Khai báo ràng buộc trên bảng đã tồn tại

Cú pháp:

```
Alter table table_name
        [with  check| with nocheck] Add
                { constraint  constraint_name
                        {   primary key ( column_name [,…n] )
                          | unique ( column_name [,…n] )
                          | check ( logical_expression )
                          | foreign key ( column_name [,…n] )
                              references ref_table ( ref_column [,…n] )
                                      [ on delete {cascade| no action} ]
                                      [ on update {cascade| no action} ]
                } [,…n]
```

Ví dụ

```
/* giả sử đã tồn tại bảng KHOA( MaKhoa, …) */
Alter table SINHVIEN
        with check add
          constraint u_CMND unique (CMND),
          constraint chk_Nam check (Nam in (1, 2, 3, 4) ),
          constraint fk_SV_maKhoa foreign key (Khoa),
                references KHOA(MaKhoa)
```

c. Kiểm tra / không kiểm tra ràng buộc

Cú pháp:

```
        Alter table Table_name
                {Check| Nocheck} constraint { All | constraint_name [,…n] }
```

Ví dụ:

```
        alter table SINHVIEN
                nocheck constraint u_CMND, chk_Nam
```

d. Xoá ràng buộc

Cú pháp:

http://www.ebook.edu.vn

Alter table *table_name*

Drop { [constraint] *constraint_name* } [,…n]

Ví dụ:

Alter table SINHVIEN

drop constraint u_CMND, chk_Nam

e. Rule

- Rule là một qui định chung được tạo ra trong một CSDL.
- Một rule có thể được áp dụng cho nhiều thuộc tính của nhiều bảng khác nhau, hoặc cho các kiểu dữ liệu người dùng định nghĩa trong database.

Tạo rule

Cú pháp:

Create rule *rule_name*
as *logical_expression*

(trong đó "logical_expression" phải chứa một biến. Biến này tương ứng với đối tượng sẽ được áp dụng rule).

Ví dụ:

create rule r_SoDuong
as @value >0

Kết buộc/ gỡ kết buộc rule

Kết buộc rule: dùng thủ tục:

sp_bindrule '*rule_name*', '*object*', ['futureonly']

trong đó:

- ✓ Tùy chọn *futureonly* chỉ dùng khi kết buộc rule với kiểu dữ liệu người dùng định nghĩa, có nghĩa các cột thuộc kiểu dữ liệu này trước đó không bị ảnh hưởng bởi rule.

Ví dụ: sp_bindrule 'r_SoDuong', 'SinhVien.Nam'

- ✓ Rule mới kết buộc sẽ ngầm gỡ rule cũ trên đối tượng.

Gỡ kết buộc

sp_unbindrule '*object*', ['futureonly']

Ví dụ: sp_unbindrule 'SinhVien.Nam'

Xoá rule

Cú pháp: Drop rule {*rule_name*} [,…n]

Lưu ý: *Chỉ xóa được rule khi nó không còn kết buộc với đối tượng nào.*

V. Các thao tác trên dữ liệu

Chú ý khi thêm/ xóa/ cập nhật dữ liệu:

http://www.ebook.edu.vn

– Dữ liệu nhập phải phù hợp với kiểu dữ liệu.

– Đảm bảo các ràng buộc toàn vẹn.

– Định dạng giá trị kiểu chuỗi unicode, kiểu datetime.

– Nhập giá trị rỗng (Null).

1. Các dạng lệnh insert

– *Thêm từng dòng dữ liệu vào bảng*

Insert [into] *Table_name*[(*column_name*[,…n])]
values (*value* [,…n])

– *Thêm 0-n dòng dữ liệu từ bảng khác/ từ kết quả của một câu truy vấn*

Insert [into] *Table_name*
Select_statement

Lưu ý: *trong câu select, ta có thể đọc dữ liệu từ các bảng trong database khác. Khi đó, tên bảng được viết đầy đủ như sau:*

Database_name.Owner.Table_name

Ví dụ: select * from QLSinhVien.dbo.SinhVien

2. Lệnh cập nhật dữ liệu

update *table_name*
set *column_name_*1= value1,…, *column_name_m= value_m*
[where *conditional_expression*]

3. Lệnh xoá dữ liệu

delete [from] *table_name*
[where *conditional_expression*]

VI. Truy vấn dữ liệu

1. Câu truy vấn tổng quát

Cú pháp tổng quát của câu truy vấn dữ liệu:

SELECT [tính chất] <danh sách các thuộc tính_1>
FROM <danh sách các table hoặc query/view [as alias] >
[**WHERE** <điều kiện_1>]
[**GROUP BY** <danh sách các thuộc tính_2>]
[**HAVING** <điều kiện_2>]
[**ORDER BY** <danh sách các thuộc tính_3> [ASC | DESC]]

trong đó:

– ***Tính chất*** là một trong các từ khóa: **ALL** (chọn ra tất cả các dòng trong bảng), **DISTINCT** (loại bỏ các dòng trùng lắp thông tin), **TOP <n>** (chọn n dòng đầu tiên thỏa mãn điều kiện).

http://www.ebook.edu.vn

- ***Danh sách các thuộc tính_1***: tên các thuộc tính cho biết thông tin cần lấy.

Chú ý:

- ✓ Các thuộc tính cách nhau bởi dấu ‘,’.
- ✓ Nếu lấy tất cả các thuộc tính của 1 bảng R thì dùng: R.*
- ✓ Nếu sau FROM chỉ có 1 bảng và lấy tất cả các cột của bảng đó thì dùng select *.
- ✓ Nếu tồn tại 1 thuộc tính sau select xuất hiện ở 2 bảng sau FROM thì phải chỉ định rõ thuộc tính đó thuộc bảng nào.

- ***Danh sách các table/query/view***: các bảng, câu truy vấn, hoặc khung nhìn chứa thông tin cần lấy. Khi tìm kiếm thông tin trên nhiều hơn 2 bảng/truy vấn thì phải kết các bảng lại với nhau (có thể đặt điều kiện kết đặt sau where hoặc đặt trong mệnh đề **From… join/ left join/ right join/full join … on …**).
- ***Alias***: bí danh (tên tắt) của bảng dùng cho các bảng có tên quá dài, hoặc một bảng được dùng nhiều lần trong mệnh đề from của câu truy vấn.
- ***điều kiện_1***: là điều kiện để lọc dữ liệu (chọn các bộ thoả điều kiện).
- ***danh sách các thuộc tính_2***: dữ liệu sẽ được gom nhóm theo các cột này, độ ưu tiên tính từ trái sang.
- ***điều kiện_2***: điều kiện lọc các nhóm theo một tiêu chí đại diện cho cả nhóm.
- ***danh sách các thuộc tính_3***:sắp xếp dữ liệu theo cột nào, thứ tự là tăng (ASC) hoặc giảm (DESC). Mặc định là dữ liệu được sắp theo thứ tự tăng dần. Việc sắp xếp được thực hiện theo thứ tự ưu tiên từ trái qua phải.

Lưu ý:

- *Nếu câu truy vấn không có mệnh đề Group By thì cũng không có mệnh đề Having.*
- *Nếu câu truy vấn có chứa mệnh đề Group By thì* ***Danh sách các thuộc tính_1*** *chỉ chứa các thuộc tính hoặc biểu thức liên quan đến các thuộc tính trong* ***danh sách các thuộc tính_2*** *và các hàm gộp (max, min, avg, sum, count).*

2. Các hàm thường dùng

- Các hàm gộp (Aggregate functions): max, min, sum, avg, count
- Các hàm thời gian.
- Các hàm toán học.
- Các hàm xử lý chuỗi.
- ….

(*Sinh viên có thể tra cứu theo từ khóa trong Books Online*).

http://www.ebook.edu.vn

VII. Tạo và sử dụng khung nhìn (View)

1. Khái niệm khung nhìn:

Khung nhìn (View) là một bảng ảo, có cấu trúc như một bảng, khung nhìn không lưu trữ dữ liệu mà dữ liệu của nó được tạo ra khi sử dụng, khung nhìn là đối tượng thuộc CSDL. Khung nhìn được tạo ra từ câu lệnh truy vấn dữ liệu (lệnh Select), truy vấn từ một hoặc nhiều bảng dữ liệu.

2. Sử dụng khung nhìn

- Khung nhìn được sử dụng khai thác dữ liệu như một bảng dữ liệu, có thể được chia sẻ bởi nhiều người dùng, an toàn trong khai thác.
- Có thể thực hiện truy vấn dữ liệu trên cấu trúc của khung nhìn.
- Các khung nhìn được tạo từ nhiều bảng hoặc trong khung nhìn có chứa từ khóa DISTINCT, hàm gộp, mệnh đề **group by** đều không cho phép cập nhật dữ liệu từ khung nhìn vào các bảng gốc trong cơ sở dữ liệu.

Cú pháp tạo khung nhìn:

Create View *view_name*

As *Select_statement*

VIII. Tạo và sử dụng chỉ mục (Index)

Chỉ mục (Index) là một phần quan trọng đối với CSDL, đặc biệt là cơ sở dữ liệu lớn. Chỉ mục được thiết lập từ một hoặc nhiều cột dữ liệu của bảng dữ liệu. Các giá trị của Chỉ mục sẽ được sắp xếp và lưu trữ theo một danh sách (bảng khác). Mỗi giá trị chỉ mục là duy nhất trong danh sách và nó sẽ liên kết đến giá trị trong bảng dữ liệu (liên kết dạng con trỏ). Việc lưu trữ dữ liệu của bảng có khóa chỉ mục được thực hiện theo cấu trúc B-Cây nhằm tăng tốc độ truy xuất dữ liệu đối với ổ đĩa (thiết bị thứ cấp).

Khi tìm kiếm một giá trị trong cột dữ liệu, mà cột này tham gia tạo Chỉ mục, đầu tiên câu lệnh xác định vị trí của giá trị nằm trong Chỉ mục bằng phép duyệt cây, sau đó thực hiện tìm theo liên kết đến bản ghi chứa giá trị tương ứng với khóa trong bảng.

1. Lựa chọn chỉ mục

- Không có chỉ mục, hệ quản trị CSDL thực hiện truy vấn bằng cách duyệt qua từng dòng trong bảng.
- Cài đặt các chỉ mục cho bảng giúp truy vấn thông tin nhanh hơn (tìm kiếm trên B-Cây).
- Khóa chính và các ràng buộc unique hiển nhiên là các chỉ mục của bảng.
- Cơ sở để chọn cài đặt chỉ mục: dựa vào các nhu cầu truy vấn thực hiện *thường xuyên* trên CSDL.

http://www.ebook.edu.vn

- Nên cài đặt chỉ mục cho các trường hợp sau:
  - Trường hợp 1: Có nhu cầu truy vấn thường xuyên các bộ của bảng Q theo một số (tập) thuộc tính nào đó.

    Ví dụ: GiaoDich(MãGD, …,NgàyGD): Có nhu cầu truy xuất *thường xuyên* các bộ của giao dịch trong một ngày hoặc trong một khoảng thời gian nhất định: cài đặt chỉ mục trên thuộc tính NgayGD của quan hệ GiaoDich.
  - Trường hợp 2: tập thuộc tính tham gia vào phép kết của một câu truy vấn xảy ra thường xuyên.

    Ví dụ: cho 2 lược đồ quan hệ:

    HocSinh(STT, Lop, HoTen,…)

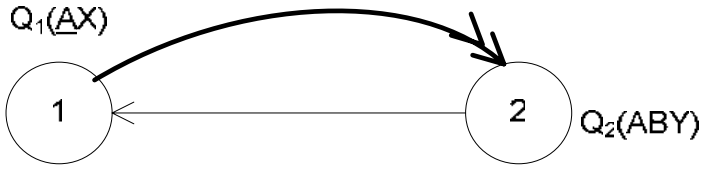
    KetQua(STT, Lop, Mon, Diem)

    Thường xuyên có nhu cầu truy vấn: cho biết kết quả học tập của một học sinh. Câu lệnh truy vấn như sau:

```
select hs.STT, hs.Lop, hs.HoTen, kq.Mon, kq.Diem
from HocSinh hs join KetQua kq on hs.STT = kq.STT
        and hs.Lop = kq.Lop
```

    → Cài đặt chỉ mục (STT, Lop) cho quan hệ KetQua

    Tổng quát: trên mô hình quan hệ, xác định các *con đường truy xuất thường xuyên:*



    → Từ một bộ của Q_1(một giá trị cụ thể a của A) có nhu cầu truy xuất thường xuyên các bộ của Q_2 tương ứng (tìm kiếm các bộ của Q_2 với A = a): khai báo chỉ mục (A) cho Q_2.

Lưu ý: *một chỉ mục (AB) khác với hai chỉ mục (A) và (B).*

2. Các loại chỉ mục

Có hai loại chỉ mục:

- Clustered index
- Nonclustered index

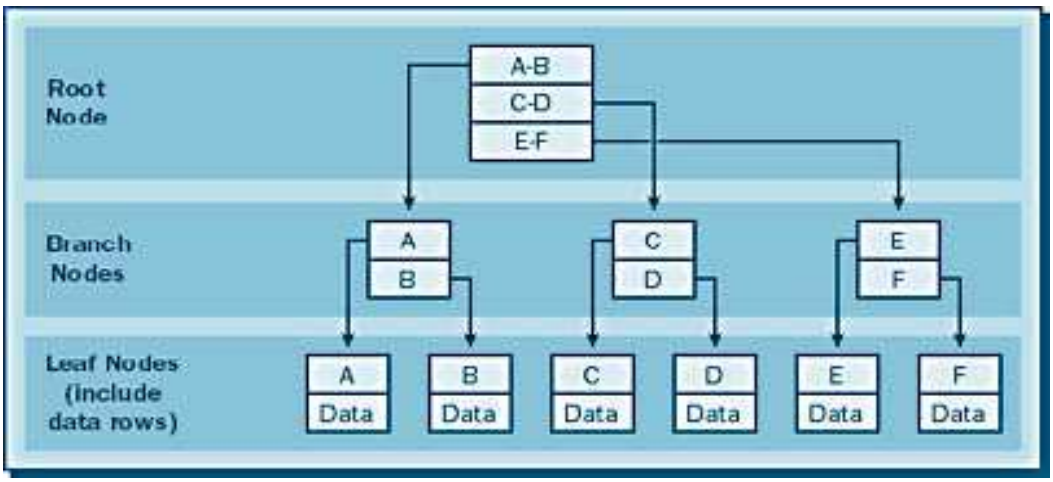
Clustered index:

- Dữ liệu thật sự được sắp xếp vật lý theo chỉ mục (thật sự nằm ở nút lá của cây).
- Mỗi bảng chỉ có thể có một clustered chỉ mục, thường là khóa chính.

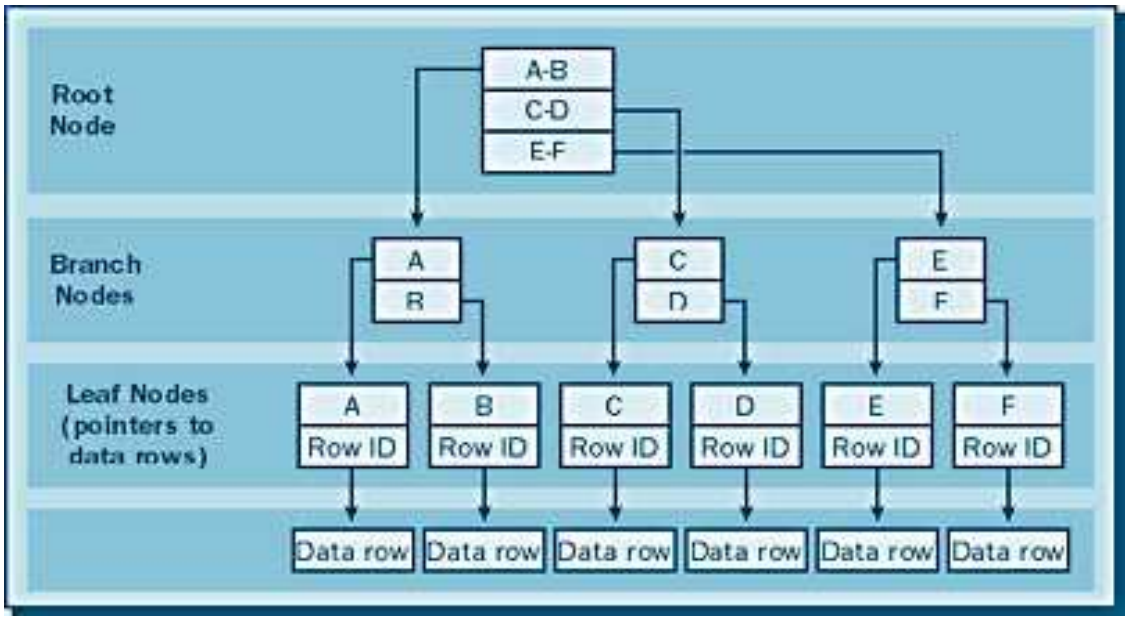
http://www.ebook.edu.vn

Nonclustered index:

- Chỉ mục logic, dữ liệu thật sự không được sắp xếp vật lý theo chỉ mục.
- Nút lá là con trỏ trỏ đến vị trí của bộ dữ liệu, hoặc trỏ đến giá trị của clustered chỉ mục (trong trường hợp bảng có clustered index).

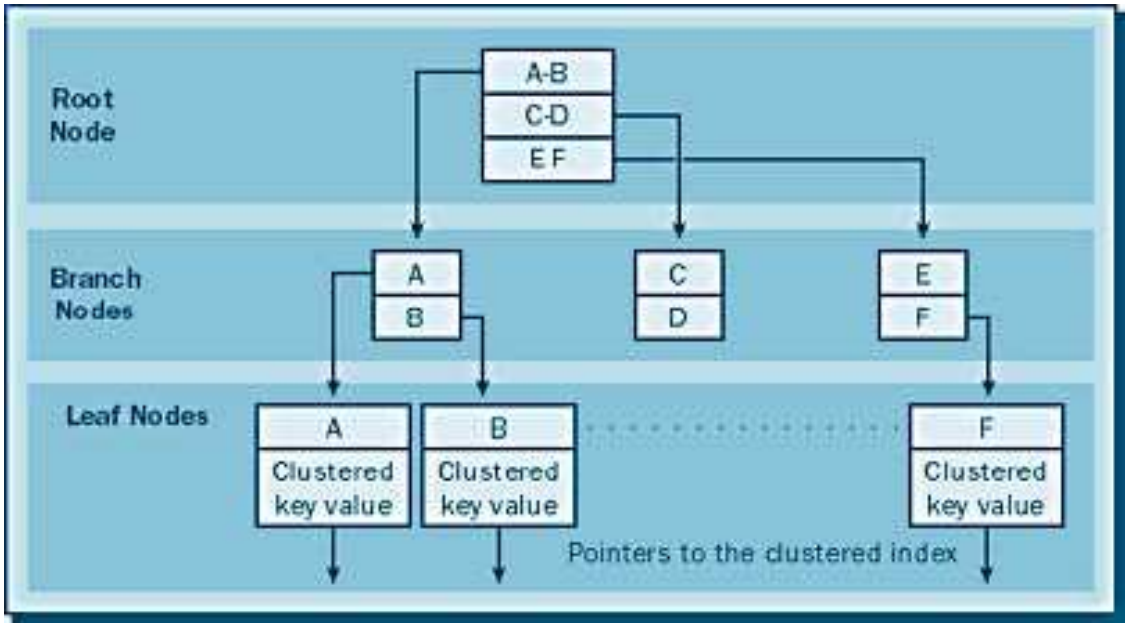


– Không có clustered index:



– Có clustered index

http://www.ebook.edu.vn



- Một số cân nhắc khi chọn chỉ mục:
  - Sử dụng nhiều chỉ mục tăng tốc độ truy vấn, nhưng làm giảm hiệu quả của các thao tác thêm/xoá/cập nhật dữ liệu.
  - Không nên tạo chỉ mục trên các bảng quá nhỏ (vài trăm dòng).
  - Chỉ nên chọn chỉ mục mà mỗi giá trị của nó tương ứng với một số ít bộ. Nếu mỗi giá trị chỉ mục ứng với trên 20% số lượng bộ trong bảng, thực hiện truy vấn bình thường bằng cách duyệt qua các dòng trong bảng sẽ hiệu quả hơn.
  - Các giá trị chỉ mục phải phân bố đều các bộ trong bảng.
  - Cố gắng dùng các chỉ mục với số thuộc tính ít (chiếm ít không gian và cần ít chi phí duy trì hơn chỉ mục với số thuộc tính lớn).
  - Clustered index phải nhỏ (số thuộc tính ít, kích thước nhỏ), vì các chỉ mục nonclustered đều phải gắn kết tới nó.

3. Cài đặt chỉ mục với SQL Server

Một số qui định:

1. Một bảng có tối đa 249 nonclustered chỉ mục (bao gồm cả những chỉ mục ngầm định khi khai báo khóa chính và chỉ mục).
2. Kích thước tối đa của một chỉ mục (tổng kích thước các thuộc tính tham gia vào chỉ mục) không quá 900 bytes.
3. Mặc định: chỉ mục clustered được khai báo ngầm định cùng với khai báo khóa chính, các trường hợp khác là nonclustered (tất nhiên có thể chỉ định khác đi).

Cú pháp khai báo chỉ mục:

Create [Unique][Cluster| Nonclustered] Chỉ mục *chỉ mục_name*

http://www.ebook.edu.vn

On {*table* | *view* } (*column* [Asc | Desc] [,...*n*])

Ví dụ:

Create nonclustered chỉ mục idx_STTHS_Lop

On KETQUA (STTHS, Lop)

Cú pháp xóa chỉ mục:

Drop Chỉ mục *table_name* (*chỉ mục_name*)

Ví dụ:

Drop Chỉ mục KETQUA(idx_STTHS_Lop)

IX. Chuyển đổi dữ liệu với các ứng dụng khác

(xem các tài liệu hướng dẫn thực hành SQL Server kèm theo)

http://www.ebook.edu.vn

Chương 3
T-SQL NÂNG CAO

I. Khai báo và sử dụng biến

1. Biến cục bộ

- Là một đối tượng có thể chứa giá trị thuộc một kiểu dữ liệu nhất định, tên biến bắt đầu bằng một ký tự @.
- Biến cục bộ có giá trị trong một *query batch* hoặc trong một thủ tục thường trú (stored procedure) hoặc hàm (function).
- Khai báo biến cục bộ bằng lệnh declare: cung cấp tên biến và kiểu dữ liệu:

Declare *tên_biến Kiểu_dữ_liệu*

Ví dụ:

Declare @MaSinhVien *char(10)*
Declare @HoTen *nvarchar(30)*
Declare @Sum *float*, @Count *int*

- Để gán giá trị cho một biến cục bộ dùng lệnh ***set***. Giá trị gán cho biến phải phù hợp với kiểu dữ liệu của biến.

Set t*ên_biến = giá_trị*
Set *tên_biến = tên_biến*
Set *tên_biến* = b*iểu_thức*
Set *tên_biến = kết_quả_truy_vấn*

Ví dụ:

```
Set @MaLop = 'TH2001'
Set @SoSV = (select count (*) from SinhVien)
Set @MaLop = 'TH'+Year(@NgayTuyenSinh)
```

Đưa kết quả truy vấn vào biến:

```
SV(MaSV: int; HoTen: nvarchar(30), Tuoi int)
Select @Var1 = HoTen, @Var1 = Tuoi  from SV
  where MaSV = 1
```

Lưu ý: *nếu câu truy vấn trả về nhiều dòng, các biến chỉ nhận giá trị tương ứng của dòng đầu tiên.*

2. Biến toàn cục

- Là các biến hệ thống do SQL Server cung cấp, tên biến bắt đầu bằng 2 ký tự @
- SQL tự cập nhật giá trị cho các biến này, người sử dụng không thể gán giá trị trực tiếp.
- **Một số biến hệ thống thường dùng**
  - **@@error:** thông báo mã lỗi, nếu @@error = 0: thao tác thực hiện thành công.

http://www.ebook.edu.vn

- **@@rowcount:** cho biết số dòng bị ảnh hưởng bởi lệnh cuối (insert, update, delete).
- **@@trancount:** cho biết số giao dịch đang hoạt động trên kết nối hiện tại.
- **@fetch_status:** cho biết thao tác lấy dữ liệu từ cursor có thành công không.

II. Cấu trúc điều khiển

1. Lệnh If…else

- Chức năng: xét điều kiện để quyết định những lệnh T-SQL nào sẽ được thực hiện
- Cú pháp:

```
If  biểu_thức_điều kiện
        Lệnh| Khối_lệnh
[Else Lệnh| Khối_lệnh]
```

Khối lệnh là một hoặc nhiều lệnh nằm trong cặp từ khóa **begin…end**

Ví dụ: xét 2 lược đồ quan hệ (LĐQH)

HocPhan(MaHP, TenHP, SiSo)

DangKy(MaSV, MaHP)

Viết lệnh để thêm một đăng ký mới cho sinh viên có mã số 001 vào học phần HP01 (giả sử học phần này đã tồn tại trong bảng HocPhan). Lời giải như sau:

```
Declare @SiSo int
select @SiSo = SiSo from HocPhan where MaHP= 'HP01'
if @SiSo < 50
Begin
    insert into DANG_KY(MaSV, MaHP)
            values('001', 'HP01')
    print N'Đăng ký thành công'
End
Else
    print N'Học phần đã đủ SV'
```

2. Lệnh While

- Chức năng*:* thực hiện lặp lại một đoạn lệnh T-SQL khi điều kiện còn đúng.
- Cú pháp:

http://www.ebook.edu.vn

While *biểu_thức_điều_kiện*

Lệnh| Khối lệnh

– Có thể sử dụng *Break* và *Continue* trong khối lệnh của while

- ✓ Break: thoát khỏi vòng while hiện hành.
- ✓ Continue : trở lại đầu vòng while, bỏ qua các lệnh sau đó.

Ví dụ: xét lược đồ quan hệ SinhVien(MaSV: int, HoTen: nvarchar(30))

Viết lệnh xác định một mã sinh viên mới theo qui định: mã sinh viên tăng dần, nếu có chỗ trống thì mã mới xác định sẽ chèn vào chỗ trống đó. Chẳng hạn, nếu trong bảng sinhvien đã có các mã sinh viên 1, 2, 3, 7 → mã sinh viên mới là 4.

Giải:

```
Declare @STT int
Set @STT = 1
While exists(select * from SV where MaSV = @STT)
        set @STT = @STT+1
Insert into SV(MaSV, HoTen) values(@STT, 'Nguyen Van A')
```

3. Lệnh Case

– Chức năng: kiểm tra một dãy các điều kiện và trả về kết quả phù hợp với điều kiện đúng. Lệnh case được sử dụng như một hàm trong câu select.

– Cú pháp: Có hai dạng:

- Dạng 1 (simple case):

```
Case Biểu_thức_đầu_vào
        When Giá_trị  then kết_quả
        [...n]
        [ Else kết_quả_khác]
End
```

- Dạng 2 (searched case):

```
Case
        When biểu_thức_điều kiện  then kết_quả
        [...n]
        [ Else kết_quả_khác]
End
```

Ví dụ: xét LĐQH NHAN_VIEN(MaNV, HoTen, NgaySinh, CapBac,Phai)

http://www.ebook.edu.vn

Cho biết những nhân viên đến tuổi nghỉ hưu biết rằng tuổi về hưu của nam là 60, của nữ là 55).

Giải:

```
select * from NHAN_VIEN
        where datediff(yy, NgaySinh, getdate()) > =
            Case Phai
                when 'Nam' then 60
                when 'Nu' then 55
            End
```

Cho biết mã NV, họ tên và loại nhân viên (cấp bậc <=3:bình thường, cấp bậc = null: chưa xếp loại, còn lại: cấp cao).

Giải:

```
Select MaNV, HoTen, 'Loai' =  Case
        when CapBac<=3 then 'Binh Thuong'
        when CapBac is null then 'Chua xep loai'
        else 'Cap Cao'    End
From NhanVien
```

III. Thủ tục thường trú (Stored Procedures)

1. Khái niệm

Thủ tục thường trú (Stored Procedures - SP) chứa các lệnh T_SQL. Tương tự như một thủ tục trong các ngôn ngữ lập trình, SP trong SQL Server có thể truyền tham số, có tính tái sử dụng. Các thủ tục này được dịch và lưu trữ thành một đối tượng trong CSDL.

Ý nghĩa:

- Tính tái sử dụng, tính uyển chuyển nhờ hệ thống tham số.
- Khi biên dịch SP, các lệnh trong của nó được tối ưu hóa nó sao cho thực thi hiệu quả nhất. Kết quả tối ưu hóa được lưu bền vững. Khi gọi thực thi thủ tục không cần biên dịch và tối ưu hóa lại → lời gọi thủ tục tiết kiệm thời gian và tài nguyên hơn khối lệnh tương đương thân thủ tục.
- Trong ứng dụng triển khai theo môi trường client/server, client gửi lời gọi SP lên server thì chiếm đường truyền ít hơn rất nhiều lần so với việc gửi khối lệnh tương đương trong thân thủ tục → Giảm khối lượng thông tin trao đổi khi ứng dụng gửi yêu cầu thực hiện công việc về cho server do đó tránh nghẽn đường truyền, giảm trì trệ.

http://www.ebook.edu.vn

- Đóng gói chỉ các thao tác cho phép trên CSDL vào các SP và quy định truy xuất dữ liệu phải thông qua SP. Ngoài ra còn có thể phân quyền trên SP → Hỗ trợ tốt hơn cho việc đảm bảo an toàn (security) cho CSDL.
- SP giúp cho việc kết xuất báo biểu bằng Crystal Report trở nên đơn giản và hiệu quả hơn rất nhiều so với việc kết xuất dữ liệu trực tiếp từ các bảng và khung nhìn.

2. Khai báo và sử dụng thủ tục

Cú pháp khai báo:

Create {proc | procedure} *procedure_name*

{*Parameter_name DataType* [=*default*] [output] }[,…n]

As

{ khối lệnh }

Go

Lưu ý:

- ✓ *Tên tham số đặt theo qui tắc như tên biến cục bộ.*
- ✓ *Giá trị trả về của SP dùng một (hay một số) tham số* output.

Ví dụ:

- *Xây dựng SP cho biết danh sách sinh viên của một lớp có mã cho trước*

```
Create proc DS_Lop @MaLop varchar(10)
As
        Select SV.MaSV, SV.HoVaTen, SV.NgaySinh
        From SinhVien SV where SV.Lop = @MaLop
Go
```

- *Xây dựng SP tính toán giá trị cho đơn hàng có mã cho trước với quan hệ DonHang như sau:*

DonHang(Ma, SoLuong, DonGia, ThueSuat, ChietKhau, ThanhTien)

```
Create proc TongTien @MaDH varchar(10)
As
Declare @ThanhTien float
Declare @TienThue float
Declare @TienChietKhau float
Declare @DonGia float,@SoLuong int
Set @SoLuong = (select SoLuong from DonHang where Ma = @MaDH)
```

http://www.ebook.edu.vn

```
Set @DonGia = (select DonGia from DonHang where Ma = @MaDH)
Set @TienThue = (select ThueSuat from DonHang where Ma = @MaDH)
Set @TienChietKhau = (select ChietKhau from DonHang
                        where Ma = @MaDH)
Set @ThanhTien = @DonGia*@SoLuong
Set @TienThue = @ThanhTien*@TienThue/100
Set @ThanhTien = @ThanhTien + @TienThue
Set @TienChietKhau = @ThanhTien*@TienChietKhau/100
Set @ThanhTien = @ThanhTien - @TienChietKhau
Update DonHang set ThanhTien = @ThanhTien where Ma = @MaDH
Go
```

– *Viết thủ tục thêm một đăng ký của sinh viên vào một học phần (tổng quát ví dụ trong phần If ...else)*

```
Create procedure usp_ThemDangKy
    @MaSV char(5), @MaHP char(5),
    @SiSo int = 0 output
As
select @SiSo = SiSo from HocPhan where MaHP= @MaHP
if @SiSo < 50
 Begin
       insert into DANG_KY(MaSV, MaHP)
             values(@MaSV, @MaHP)
       set @SiSo = @SiSo+1
       return 1
    End
 return 0
Go
```

- *Xây dựng SP tính điểm trung bình và xếp loại cho sinh viên thuộc lớp cho trước. Giả sử có các quan hệ như sau:*

SinhVien (MaSV, HoTen, DTB, XepLoai, Lop)

MonHoc (MaMH, TenMH)

KetQua (MaMH, MaSV, LanThi, Diem)

trong đó:

http://www.ebook.edu.vn

- Điểm thi chỉ tính lần thi sau cùng.
- Xếp loại: Xuất sắc [9, 10], Giỏi [8, 8.9], Khá [7, 7.9], Trung bình [5.0, 6.9], Yếu [0,4.9].
- Kết quả xuất dạng tham số output, không ghi xuống CSDL.

Giải

```
Create proc XepLoaiSV @MaSV varchar(10), @DTB float out put,
                              @XL nvarchar(20) out put
As
        Set @DTB = (Select avg(Diem) from KetQua Kq
                        Where MaSV = @MaSV
                                and  not exists (select * from KetQua Kq1
                                                        where Kq1.MaSV = @MaSV
                                                                and Kq1.MaMH=Kq.MaMH
                                                                and Kq1.LanThi > Kq.LanThi))
        If @DTB >= 9
                Set @XL = N'Xuất sắc'
        Else if @DTB >= 8
                Set @XL = N'Giỏi'
        Else if @DTB >= 7
                Set @XL = N'Khá'
        Else if @DTB >= 5
                Set @XL = N'Trung bình'
        Else
                Set @XL = N'Yếu'
Go
```

Cú pháp gọi thực hiện thủ tục:

EXEC| EXECUTE

{ [*@return_status* =] *procedure_name*

{ [*@parameter _name* =] *value* [OUTPUT] } [,...*n*]

Lưu ý:

- *Khi gọi thực hiện SP, dùng từ khóa Exec và cần truyền đủ tham số với kiểu dữ liệu phù hợp và thứ tự chính xác như khai báo trong định nghĩa SP.*
- *Có thể truyền giá trị cho tham số đầu vào (input) là một hằng hoặc một biến đã gán giá trị, không truyền được một biểu thức.*
- *Để nhận được giá trị kết quả (thông qua tham số đầu ra), cần truyền vào một biến và có từ khóa output.*

Ví dụ:

- Gọi thủ tục usp_ThemDangKy:

http://www.ebook.edu.vn

```
Exec usp_ThemDangKy  '001', 'HP01'
```

hoặc

```
Exec usp_ThemDangKy @MaHP = 'HP01', @MaSV = '001'
```

- Gọi thủ tục usp_ThemDangKy có nhận kết quả đầu ra:

```
Declare @SiSo int
Exec usp_ThemDangKy '001','HP01', @SiSo output
Print  @SiSo
```

- Gọi thủ tục usp_ThemDangKy có nhận kết quả đầu ra và kết quả trả về từ thủ tục :

```
Declare @SiSo int, @KetQua int
Exec @KetQua = usp_ThemDangKy '001','HP01', @SiSo output
```

- Gọi thực hiện thủ tục xếp loại sinh viên:

```
Declare@MaSinhVien varchar(10)
Declare@DiemTB varchar(10)
Declare@XepLoai varchar(10)
Set@MaSinhVien = '0712345'
Exec XepLoaiSV @MaSinhVien,@DiemTB out put,@XepLoai out put
Exec XepLoaiSV '0713478', @DiemTB out put,@XepLoai out put
```

Sửa thủ tục

Thay từ khóa Create trong lệnh tạo thủ tục bằng từ khóa Alter.

Xóa thủ tục

```
Drop {procedure|proc} procedure_name
```

Ví dụ: Drop procedure usp_ThemDangKy

3. Stored procedure hệ thống

- Là những thủ tục do SQL Server cung cấp sẵn để thực hiện các công việc: quản lý CSDL, quản lý người dùng, cấu hình CSDL,…
- Các thủ tục này có tên bắt đầu bằng "sp_" → Khi xây dựng thủ tục, tránh đặt tên thủ tục bắt đầu với "sp_".

IV. Kiểu dữ liệu cursor

1. Khái niệm Cursor

- Là một cấu trúc dữ liệu ánh xạ đến một tập các dòng dữ liệu kết quả của một câu truy vấn (select).

http://www.ebook.edu.vn

- Cho phép duyệt tuần tự qua tập các dòng dữ liệu và đọc giá trị từng dòng.
- Thể hiện của cursor là 1 biến, nhưng tên biến này không bắt đầu bằng '@'.
- Vị trí hiện hành của cursor có thể được dùng như điều kiện trong mệnh đề where của lệnh update hoặc delete: cho phép cập nhật/xoá dữ liệu (dữ liệu thật sự trong CSDL) tương ứng với vị trí hiện hành của cursor.

2. Khai báo và sử dụng Cursor

Khai báo Cursor

Có thể sử dụng cú pháp chuẩn SQL 92 hoặc cú pháp T_SQL mở rộng.

– Cú pháp SQL 92 chuẩn:

Declare *cursor_name* [Insensitive] [Scroll] Cursor

For *select_statement*

[For {Read only| Update [of *column_name* [,…n]] }]

– Cú pháp T_SQL mở rộng

Declare *cursor_name* Cursor

[Local | Global]

[Forward_only| Scroll]

[Static| Dynamic]

[Read_only]

For select_statement

[For Update [of column_name [,…n]]]

Lưu ý: *Tên cursor trong các cách khai báo không bắt đầu bằng ký tự "@".*

Ý nghĩa các tham số tùy chọn trong khai báo:

- *Insensitive/ static:* nội dung của cursor không thay đổi trong suốt thời gian tồn tại, trong trường hợp này cursor chỉ là read only.
- *Dynamic:* trong thời gian tồn tại, nội dung của cursor có thể thay đổi nếu dữ liệu trong các bảng liên quan có thay đổi.
- *Local:* cursor cục bộ, chỉ có thể sử dụng trong phạm vi một khối (query batch) hoặc một thủ tục/ hàm.
- *Global:* cursor toàn cục, có thể sử dụng trong một thủ tục/hàm hay một query batch bất kỳ hoặc đến khi bị hủy một cách tường minh.
- *Forward_only:* cursor chỉ có thể duyệt một chiều từ đầu đến cuối.
- *Scroll:* có thể duyệt lên xuống cursor tùy ý (duyệt theo đa chiều).

http://www.ebook.edu.vn

- *Read only:* chỉ có thể đọc từ cursor, không thể sử dụng cursor để update dữ liệu trong các bảng liên quan (ngược lại với "for update…").

Mặc định khi khai báo cursor nếu không chỉ ra các tùy chọn thì cursor có các tính chất:

- Global
- Forward_only
- Read only hay "for update" tùy thuộc vào câu truy vấn
- Dynamic

Duyệt cursor

Dùng lệnh Fetch để duyệt tuần tự qua cursor theo cú pháp:

Fetch

[[Next| Prior| First| Last| Absolute n| Relative n]

From] Tên_cursor

[Into Tên_biến [,…n]]

- Mặc định: fetch next.
- Đối với cursor dạng forward_only, chỉ có thể fetch next.
- Biến hệ thống @@fetch_status cho biết lệnh fetch vừa thực hiện có thành công hay không, giá trị của biến này cơ sở để biết đã duyệt đến cuối cursor hay chưa.

Quy trình sử dụng Cursor

- Khai báo cursor.
- "Mở" cursor bằng lệnh *Open*

  Open *tên_cursor*

- Khai báo các biến tạm để chứa phần tử hiện hành (đang được xử lý) của cursor:
  - ✓ Các biến tạm phải cùng kiểu dữ liệu với các trường tương ứng của phần tử trong cursor.
  - ✓ Có n trường trong phần tử của cursor thì phải có đủ n biến tạm tương ứng.
- Fetch (next,…) cursor để chuyển đến vị trí phù hợp:
  - ✓ Có thể đưa các giá trị của dòng hiện hành vào các biến thông qua mệnh đề into của lệnh fetch.
  - ✓ Nếu không có mệnh đề into, các giá trị của dòng hiện hành sẽ được hiển thị ra cửa sổ kết quả (result pane) sau lệnh fetch.
  - ✓ Có thể sử dụng vị trí hiện tại như là điều kiện cho mệnh đề where của câu delete/ update (nếu cursor không là read_only).

http://www.ebook.edu.vn

- Lặp lại việc duyệt và sử dụng cursor, có thể sử dụng biến @@fetch_status để biết đã duyệt qua hết cursor hay chưa. **@@FETCH_STATUS = 0 : lấy dữ liệu thành công, @@FETCH_STATUS < 0 : không lấy được dữ liệu.**
- Đóng cursor bằng lệnh Close

  Close *Tên_cursor*

  Lưu ý: *Sau khi đóng, vẫn có thể mở lại nếu cursor chưa bị hủy.*
- Hủy cursor bằng lệnh deallocate

  Deallocate *Tên_cursor*

Ví dụ: xét hai LĐQH

SINHVIEN (MaSV, HoTen, MaKhoa)

KHOA(MaKhoa, TenKhoa)

- Duyệt và đọc giá trị từ cursor

Cập nhật lại giá trị MaSV = Viết tắt tên Khoa + MaSV hiện tại cho tất cả sinh viên:

```
declare cur_DSKhoa cursor
for select MaKhoa, TenKhoa from Khoa
open cur_DSKhoa
declare @MaKhoa int,
        @TenKhoa varchar(30), @TenTat varchar(5)
fetch next from cur_DSKhoa into @MaKhoa, @TenKhoa
while @@fetch_status = 0
begin
    -- xác định tên tắt của Khoa dựa vào @TenKhoa…
    update   SinhVien set MaSV = @TenTat+MaSV
          Where MaKhoa = @MaKhoa
    fetch next from cur_DSKhoa into @MaKhoa, @TenKhoa
end
Close  cur_DSKhoa
Deallocate cur_DSKhoa
```

- Dùng cursor để xác định dòng cập nhật

http://www.ebook.edu.vn

```
declare cur_DSKhoa cursor scroll
for select MaKhoa, TenKhoa from Khoa
open cur_DSKhoa
fetch absolute 2 from cur_DSKhoa
if (@@fetch_status = 0)
        update Khoa
                set TenKhoa = 'aaa'
                where current of cur_DSKhoa
Close  cur_DSKhoa
Deallocate cur_DSKhoa
```

3. Biến cursor

- Ta có thể khai báo một biến kiểu cursor và gán cho nó tham chiếu đến một cursor đang tồn tại.
- Biến cursor có thể được xem như là con trỏ cursor.
- Biến cursor là một biến cục bộ.
- Biến cursor sau khi gán giá trị được sử dụng như một cursor thông thường.

Ví dụ :

```
Declare @cur_var cursor
set @cur_var = my_cur      -- my_cur là một cursor đang tồn tại
```

hoặc:

```
Declare @cur_var cursor
set @cur_var = cursor for select_statement
```

Kết hợp cursor với stored procedure

Xây dựng SP tính điểm trung bình và xếp loại cho sinh viên thuộc lớp cho trước. Giả sử có các quan hệ như sau:

SinhVien (MaSV, HoTen, DTB, XepLoai, Lop)

MonHoc (MaMH, TenMH)

KetQua (MaMH, MaSV, LanThi, Diem)

Biết rằng

- ✓ Điểm thi chỉ tính lần thi sau cùng
- ✓ Xếp loại: Xuất sắc [9, 10], Giỏi [8, 8.9], Khá [7, 7.9], Trung bình [5.0, 6.9], Yếu [0, 4.9].
- ✓ Kết quả ghi xuống CSDL, đồng thời xuất ra tổng số sinh viên xếp loại giỏi của lớp đó.

- **Phân tích ví dụ:**

http://www.ebook.edu.vn

- Lớp cần xét có nhiều sinh viên, từng sinh viên cần được xử lý thông qua 3 bước:
  - ✓ Tính điểm trung bình cho sinh viên, điểm trung bình phải là điểm của lần thi sau cùng. Có thể tái sử dụng thủ tục XepLoaiSVLop.
  - ✓ Dựa vào điểm trung bình của sinh viên để xác định xếp loại.
  - ✓ Cập nhật điểm và xếp loại vào bảng sinh viên.
- Mọi sinh viên đều lặp lại 3 bước trên.

Từ phân tích trên ta thấy:

- ✓ Cần xử lý nhiều phần tử (các sinh viên).
- ✓ Mỗi phần tử xử lý tương đối phức tạp (truy vấn, tính toán, gọi thủ tục khác, điều kiện rẽ nhánh, cập nhật dữ liệu, …).
- ✓ Cách xử lý các phần tử là như nhau.

⇒ Sử dụng cursor là thích hợp

- ✓ Cursor chứa các sinh viên của lớp cần xét, chỉ cần chứa mã sinh viên là được.

- **Xây dựng thủ tục**

```
Create procedure  XepLoaiSVLop
    @Lop nvarchar(10), @SoSVGioi int out
As
    Declare @DTB float
    Declare @XepLoai nvarchar(20)
    Declare @MaSV nvarchar(10)
    Declare cur_SV cursor
    For  (select MaSV from SinhVien where Lop=@Lop)
    Open cur_SV
    Fetch Next from cur_SV into @MaSV
    While  @@FETCH_STATUS = 0
    Begin
            Exec XepLoaiSV @MaSV, @DTB output, @XepLoai output
            Update SinhVien set DTB = @DTB, XepLoai=@XepLoai
            Where MaSV = @MaSV
            Fetch Next from cur_SV into @MaSV
    End
    Close cur_SV
    Deallocate cur_SV
```

http://www.ebook.edu.vn

```
Set @SoSVGioi = (select count(*) from sinhvien
                    where lop = @Lop and XepLoai = N'Giỏi')
Go
```

V. Hàm người dùng (User Defined Functions)

1. Khái niệm hàm người dùng

- Giống stored procedure:
  - mã lệnh có thể tái sử dụng.
  - Chấp nhận các tham số input.
  - Biên dịch một lần và từ đó có thể gọi khi cần.
- Khác stored procedure:
  - Chấp nhận nhiều kiểu giá trị trả về (chỉ một giá trị trả về).
  - Không chấp nhận tham số output.
  - Khác về cách gọi thực hiện.
- Có thể xem hàm người dùng thuộc về 3 loại tùy theo giá trị trả về của nó:
  - Giá trị trả về là kiểu dữ liệu cơ sở (int, varchar, float, datetime…).
  - Giá trị trả về là Table có được từ một câu truy vấn.
  - Giá trị trả về là table mà dữ liệu có được nhờ tích lũy dần sau một chuỗi thao tác xử lý và insert.

2. Khai báo và sử dụng

Khai báo hàm người dùng

Loại 1: Giá trị trả về là kiểu dữ liệu cơ sở

```
Create function func_name
( {parameter_name  DataType [= default ] } [,…n])
returns DataType
As
Begin
…
Return {value | variable | expression}
End
```

Ví dụ:

http://www.ebook.edu.vn

```
Create function SoLonNhat
(@a int,@b int,@c int) return int
As
Begin
        declare @max int
        set @max = @a
        if @b > max  set  @max = @b
        if @c > max  set  @max = @c
        return @max
End
```

Loại 2: Giá trị trả về là một bảng có được từ một câu truy vấn

```
Create function func_name
        ( {parameter_name  DataType [= default ] } [,…n])
        returns Table
As
Return [ ( ]select_statement [ ) ]
Go
```

Ví dụ: Viết hàm in danh sách các mặt hàng của một mã đơn hàng cho trước

```
Create function DanhSachMatHang
( @MaDonHang varchar(10) ) returns Table
As
Return
(Select MH.TenHang,MH.DonGia
From ChiTietDH CT, MatHang MH
Where CT.MaDH = @MaDonHang
and CT.MaMH = MH.MaMH)
Go
```

Loại 3: Giá trị trả về là một bảng mà dữ liệu có được nhờ tích lũy dần sau một chuỗi thao tác xử lý và insert.

http://www.ebook.edu.vn

```
Create function func_name
( {parameter_name  DataType [= default ] } [,…n])
returns TempTab_name Table(Table_definition)
As
Begin
...
Return
End
Go
```

Ví dụ:

```
Create function DanhSachLop ()
returns @DS  Table(@MaLop varchar(10),@SoSV int)
As
--các xử lý insert dữ liệu vào bảng DS
return
Go
```

Lưu ý: *Trong thân hàm không được sử dụng các hàm hệ thống bất định (Built-in nondeterministic functions), bao gồm :*

- GETDATE
- GETUTCDATE
- NEWID
- RAND
- TEXTPTR
- @@TOTAL_ERRORS, @@CPU_BUSY, @@TOTAL_READ, @@IDLE, @@TOTAL_WRITE, @@CONNECTIONS …

Sử dụng hàm người dùng

Các hàm người dùng được sử dụng trong câu truy vấn, trong biểu thức… phù hợp kiểu dữ liệu trả về của nó.

Ví dụ:

- Select dbo.SoLonNhat(3,5,7)
- Select * from DanhSachLop()

Lưu ý:

- *Nếu dùng giá trị mặc định của tham số, phải dùng từ khóa default.*

http://www.ebook.edu.vn

– *Khi gọi hàm loại 1 (trả về giá trị cơ bản), phải có tên owner của hàm đi kèm (ví dụ dbo.SoLonNhat(5,8,-10)).*

Thay đổi hàm người dùng

Thay từ khóa create trong các lệnh tạo hàm bằng từ khóa alter

Xóa hàm người dùng

Drop function *tên_hàm_cần _xóa*

Ví dụ:

Drop function DanhSachMatHang

3. Các hàm hệ thống

Ngoài các hàm do người dùng định nghĩa, SQL Server còn cung cấp các hàm xây dựng sẵn của hệ thống. Các hàm này cung cấp tiện ích như xử lý chuỗi, xử lý thời gian, xử lý số học…

Sinh viên tìm hiểu thêm về các hàm này trong Books on-line và các tài liệu tham khảo.

- Để tạo hàm hệ thống cần tiến hành theo các bước sau:
  - ✓ Tạo hàm trong cơ sở dữ liệu **Master**
  - ✓ Tên hàm bắt đầu bởi **fn*_functionName***
  - ✓ Thay đổi chủ nhân của hàm bằng thủ tục sp_changeobjectowner như sau:

    EXEC sp_changeobjectowner 'fn_*functionName*' , 'system_function_schema'

Ví dụ: *Tạo hàm hệ thống thực chuyển đổi một biến kiểu ngày tháng sang kiểu chuỗi.*

```
--Tạo hàm fn_doingay
        create function fn_doingay(@ngay datetime)
        returns char(10)
        as
        begin
                return convert(nchar(10),@ngay,103)
        end
--Thay đổi chủ nhân của hàm
        EXEC sp_changeobjectowner 'fn_doingay' , 'system_function_schema'
```

Sau lệnh này hàm fn_doingay có thể dùng được cho CSDL bất kỳ.

```
        Select manv, hoten, fn_doingay(ngaysinh)
        From nhanvien
```

VI. Triggers và cài đặt ràng buộc dữ liệu

1. Giới thiệu

- Trigger là một loại stored procedure đặc biệt có các đặc điểm sau:

http://www.ebook.edu.vn

- Tự động thực hiện khi có thao tác insert, delete hoặc update trên dữ liệu.
- Thường dùng để kiểm tra các ràng buộc toàn vẹn của CSDL hoặc các qui tắc nghiệp vụ.
- Một trigger được định nghĩa trên một bảng, nhưng các xử lý trong trigger có thể sử dụng nhiều bảng khác.

- Xử lý của trigger thường cần sử dụng đến hai bảng tạm:
  - **Inserted**: chứa các dòng vừa mới được thao tác insert/ update thêm vào bảng.
  - **Deleted**: chứa các dòng vừa mới bị xóa khỏi bảng bởi thao tác update/delete.

  Lưu ý: *update = delete dòng chứa giá trị cũ + insert dòng chứa giá trị mới*
  - Inserted và deleted là các bảng trong bộ nhớ chính:
    - Cục bộ cho mỗi trigger.
    - Có cấu trúc giống như bảng (table) mà trigger định nghĩa trên đó
    - Chỉ tồn tại trong thời gian trigger đang xử lý.
  - Nếu thao tác insert/ delete/ update thực hiện trên nhiều dòng, trigger cũng chỉ được gọi một lần →Bảng inserted/ deleted có thể chứa nhiều dòng.

2. Sử dụng Trigger

Khai báo trigger

- Cú pháp:

```
Create trigger tên_trigger
On {tên_bảng|tên_view}
{For| After| Instead of } { [delete] [,] [insert] [,] [update] }
As
                { các lệnh T-sql }
Go
```

rong đó:

For | After:

- Trigger được gọi thực hiện sau khi thao tác delete/ insert/ update tương ứng đã được thực hiện thành công:
  - ✓ Các dòng mới được thêm chứa đồng thời trong bảng dữ liệu và bảng inserted.
  - ✓ Các dòng bị xoá chỉ nằm trong bảng deleted (đã bị xoá khỏi bảng dữ liệu).
- Có thể xử lý quay lui thao tác đã thực hiện bằng lệnh *rollback transaction.*

Instead of:

- Trigger được gọi thực hiện thay cho thao tác delete/ insert/ update tương ứng:

http://www.ebook.edu.vn

- ✓ Các dòng mới được thêm chỉ chứa trong bảng inserted.
- ✓ Các dòng bị chỉ định xoá nằm đồng thời trong bảng deleted và bảng dữ liệu (dữ liệu không bị xoá).

– Trigger *Instead of* thường được dùng để xử lý cập nhật trên khung nhìn.

Lưu ý:

– Lệnh tạo trigger phải là lệnh đầu tiên trong một *query batch.*

– Trên một bảng có thể định nghĩa nhiều trigger *for/after* cho mỗi thao tác nhưng chỉ có thể định nghĩa một trigger *instead of* cho mỗi thao tác.

– Không thể định nghĩa trigger *instead of update/ delete* trên bảng có cài đặt khóa ngoại dạng *update cascade/ delete cascade.*

– Trong thân trigger, có thể sử dụng hàm *Update(tên_cột)* để kiểm tra xem việc cập nhật được thực hiện trên cột nào.

Update(*tên_cột*) = *true* : có thực hiện cập nhật trên cột *tên_cột*

Sử dụng trigger cài đặt một số loại ràng buộc

Ví dụ 1- Ràng buộc liên thuộc tính – liên quan hệ

Cho CSDL:

DatHang(MaPDH, NgayDH,…)

GiaoHang(MaPGH, MaPDH, NgayGH,…)

Ràng buộc: Ngày giao hàng không thể nhỏ hơn ngày đặt hàng tương ứng

Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
DatHang	-	-	+ (NgayDH)
GiaoHang	+	-	+ (NgayGH, MaPDH)

→Cần cài đặt trigger cho thao tác sửa trên bảng DatHang, và thêm/sửa trên bảng Giaohang

Trigger cho thao tác thêm và sửa trên giao hàng:

```
Create trigger tr_GH_ins_upd_NgayGH
On GIAOHANG  for insert, update
As
if  update(MaPDH) or update (NgayGH)
        if exists(select * from inserted i, DatHang d
                where i.MaPDH = d.MaPDH
```

http://www.ebook.edu.vn

```
                    and i.NgayGH<d.NgayDH)
        begin
            raiserror (N'Ngày GH không thể nhỏ hơn ngày ĐH',0,1)
            rollback tran
        end
go
```

Bài tập: Trigger cho thao tác sửa trên đặt hàng.

Ví dụ 2 – Ràng buộc toàn vẹn liên bộ

Xét LĐQH: KetQua(MASV, MAMH, LANTHI, DIEM)

Tạo trigger kiểm tra RBTV: "Sinh viên chỉ được thi tối đa 2 lần cho một môn học"

- Xác định bảng tầm ảnh hưởng:

	Insert	Delete	Update
KetQua	+	-	+ (MASV, MAMH)

- Cài đặt trigger dựa trên bảng tầm ảnh hưởng:

```
-- Tao trigger ung voi thao tac insert tren bang KetQua
Create trigger trg_KetQua_insert
on KetQua
for insert
as
    declare @SoLanThi int
    select @SoLanThi=count(*)
    from KetQua K, inserted I
    where I.MaSV=K.MaSV and I.MaMH = K.MaMH
    if @SoLanThi > 2
    begin
            raiserror('So lan thi phai <= 2', 0,1)
            rollback transaction
    end
```

Bài tập: Tạo trigger ứng với thao tac Update trên bảng KetQua.

Ví dụ 3: Trigger cho việc thực hiện một thao tác cập nhật dữ liệu nào đó.

Cho quan hệ

CHI_TIET_HOA_DON(**MaHD,STT**, MaMH, SoLuong, DonGia, ThanhTien).

http://www.ebook.edu.vn

- Viết trigger thực hiện cập nhật giá trị của ThanhTien khi thêm một chi tiết hóa đơn

```
CREATE TRIGGER trgTINHTIEN_CTHD
ON CHI_TIET_HOA_DON
FOR INSERT ,UPDATE
AS
BEGIN
    Update CHI_TIET_HOA_DON
    Set ThanhTien=SoLuong*DonGia
    Where exists (SELECT *
                  FROM INSERTED I
                  WHERE CHI_TIET_HOA_DON.MAHD = I.MAHD and
                        CHI_TIET_HOA_DON.STT = I.STT)
END
```

http://www.ebook.edu.vn

Chương 4
BẢO MẬT VÀ AN TOÀN DỮ LIỆU

I. Bảo mật trong hệ quản trị cơ sở dữ liệu

1. Khái niệm cơ bản về bảo mật

Nhằm bảo vệ hệ thống CSDL không bị xâm nhập, người quản trị cơ sở dữ liệu phải quyết định cho phép hay không cho phép người dùng truy cập và thao tác trên cơ sở dữ liệu dựa vào nhiệm vụ của người dùng trên hệ CSDL. Người quản trị thường dựa trên nền tảng lý thuyết bảo mật của hệ cơ sở dữ liệu đa người dùng, nhằm tìm ra phương pháp bảo mật theo đúng với nhu cầu của bảo mật dữ liệu.

Với mục đích tăng tính bảo mật dữ liệu, SQL Server hỗ trợ các tính năng cho phép người quản trị thiết lập cơ chế bảo vệ cơ sở dữ liệu trong môi trường đa người dùng, bao gồm các yếu tố chính sau:

- Vai trò của người dùng trong hệ thống và cơ sở dữ liệu.
- Quyền sử dụng các ứng dụng cơ sở dữ liệu trong SQL Server.
- Quyền tạo và sửa đổi cấu trúc các đối tượng CSDL.
- Quyền truy cập, xử lý dữ liệu.

Khi đăng nhập vào một hệ thống CSDL đa người dùng, người sử dụng cần phải cung cấp UserID (tài khoản) và Password (mật khẩu). Dựa trên UserID hệ thống có khả năng kiểm soát tất cả các hành vi của người sử dụng trên CSDL SQL Server.

Để thực hiện được chức năng này, người quản trị CSDL cần phải thiết lập các quyền xử lý và truy cập vào CSDL khi tạo ra UserID, ngoài ra còn có một số thuộc tính khác của SQL Server như quyền backup dữ liệu, trao đổi dữ liệu với các ứng dụng CSDL khác, …

Khi nói đến bảo mật, người quản trị cần quan tâm đến các thông tin sau của người dùng:

- Một người dùng chỉ có một UserID và một mật khẩu.
- Thời gian có hiệu lực của mật khẩu.
- Giới hạn chiều dài của mật khẩu.
- Giới hạn người sử dụng theo license hay mở rộng.
- Thông tin về người sử dụng.

Khi tạo người sử dụng, tên tài khoản cần rõ ràng, dễ hiểu dễ gợi nhớ, và không cho phép các ký tự đặc biệt, không nên có khoảng trắng.

2. Lựa chọn bảo mật

http://www.ebook.edu.vn

Khi tạo ra một người dùng (login user) trong SQL Server, có 3 cách để tăng tính bảo mật cho người sử dụng đó:

- Giao tiếp với hệ điều hành: sử dụng UserID và Password của hệ điều hành Windows để đăng nhập SQL Server. Với loại bảo mật này, người dùng truy cập vào mạng và có thể sử dụng CSDL SQL Server, đồng thời một người dùng có UserID và Password có thể sử dụng tài nguyên trên mạng.
- Bảo mật chuẩn: với loại này, người sử dụng có UserID và Password tách rời với hệ điều hành mạng, ứng với loại bảo mật này người sử dụng chỉ có hiệu lực trong CSDL SQL Server, không thể sử dụng tài nguyên trên mạng.
- Tổng hợp cả hai trường hợp trên: một số người dùng sử dụng quyền sử dụng trên hệ điều hành và SQL Server, một số khác chỉ sử dụng quyền truy cập vào SQL Server.

Lưu ý: *Tài khoản người dùng có giá trị trên SQL Server hiện hành, khi sang một SQL Server khác phải tạo ra tài khoản người dùng trên server đó.*

- SQL Server cung cấp các chức năng hay các thủ tục tạo mới và quản trị người dùng CSDL SQL Server như sau:
  - ✓ Sử dụng thủ tục sp_addlogin.
  - ✓ Sử dụng công cụ Enterprise Manager.

- Sử dụng thủ tục sp_addlogin

  Cú pháp

```
sp_addlogin [ @loginame = ] 'login'
   [ , [ @passwd = ] 'password' ]
   [ , [ @defdb = ] 'database' ]
   [ , [ @deflanguage = ] 'language' ]
   [ , [ @sid = ] sid ]
   [ , [ @encryptopt = ] 'encryption_option' ]
```

trong đó các tham số có ý nghĩa như sau:

- **@loginame:** tên tài khoản sẽ tạo.
- **@passwd:** mật khẩu cho người dùng có tài khoản trên.
- **@defdb:** cơ sở dữ liệu mặc định khi người dùng đăng nhập vào SQL Server.
- **@deflanguage:** ngôn ngữ mặc nhiên cho người dùng SQL Server.
- **@sid:** số nhận dạng hệ thống khi người dùng đăng nhập vào.
- **@encryptopt:** khi tạo tài khoản người dùng trong CSDL, các thông tin về tài khoản, mật khẩu được lưu trữ trong bảng sysusers của CSDL Master, nếu bạn cung cấp tham số skip_enctription thì không mã hoá mật khẩu trước khi lưu vào bảng sysusers, nếu không cung cấp tham số hay để trống, SQL Server sẽ mã hoá mật khẩu trước khi lưu trữ.

http://www.ebook.edu.vn

Ví dụ: tạo người dùng có tên ‘nam’, mật khẩu ‘123’, cơ sở dữ liệu mặc định ‘QLKyNang’

Exec sp_addlogin ‘nam’, ‘123’, ‘QLKyNang’

- Thay đổi mật khẩu

sp_password [[**@old =**] '*old_password*' **,**]
{ [**@new =**] '*new_password*' }
[**,** [**@loginame =**] '*login*']

3. Quyền người dùng và quản trị quyền người dùng

Quyền của người dùng được định nghĩa như mức độ người dùng có thể hay không thể thực thi trên CSDL, quyền được chia thành 4 loại như sau:

- Quyền truy cập vào SQL Server.
- Quyền truy xuất vào CSDL.
- Quyền thực hiện trên các đối tượng của CSDL.
- Quyền xử lý dữ liệu.

Cấp phát quyền truy cập vào CSDL

Cú pháp:

Use *db_name*
Go
sp_grantdbaccess [@loginame =] '*login*'
[,[@name_in_db =] '*name_in_db*' [OUTPUT]]

Các tham số:

- ✓ @loginame: tài khoản của người sử dụng đăng nhập vào SQL Server
- ✓ @name_in_db: tạo bí danh (tên khác) của tài khoản người dùng khi truy cập vào CSDL *db_name* được chỉ định, nếu không chỉ rõ CSDL muốn cho phép người dùng truy cập thì người dùng được cấp quyền trên CSDL hiện hành.

Loại bỏ quyền truy cập vào CSDL db_name của người dùng

Use *db_name*
Go

sp_revokedbaccess [@loginame =] '*login*'

Cấp phát quyền thực thi trên cơ sở dữ liệu

Sau khi cấp phát quyền cho người dùng truy cập vào CSDL, kế tiếp cho phép người dùng đó có quyền truy cập và xử lý các đối tượng trong CSDL cũng như xử lý dữ liệu trên các đối tượng đó.

Các quyền truy cập trên các đối tượng trong một CSDL:

http://www.ebook.edu.vn

Quyền	Diễn giải
SELECT	Cho phép người sử dụng nhìn thấy dữ liệu, nếu người sử dụng có quyền này thì họ chỉ có thể thực thi những phát biểu select để truy vấn dữ liệu trên các bảng hay các view được cho phép.
INSERT	Cho phép người sử dụng thêm dữ liệu, nếu người sử dụng có quyền này, họ có thể thực hiện phát biểu Insert, đối với một số hệ thống CSDL khác, muốn thực thi phát biểu Insert, người sử dụng phải có quyền Select, trong SQL Server quyền Insert không liên quan đến quyền truy vấn Select.
UPDATE	Quyền này cho phép người sử dụng chỉnh sửa dữ liệu bằng phát biểu Update.
DELETE	Quyền này cho phép người sử dụng xóa dữ liệu bằng phát biểu Delete.
REFERENCE	Cho phép người sử dụng thêm dữ liệu vào bảng có khóa ngoại bằng phát biểu Insert, trong SQL Server quyền Insert không liên quan đến quyền truy vấn Select.
EXECUTE	Quyền này cho phép người sử dụng thực thi các thủ tục (SP) trong CSDL.

Thủ tục cấp quyền GRANT

GRANT ALL | < PERMISSION> [, …n]

ON

<table hoặc view name> [(<column name> [, … n])]

| <stored hoặc extension procedure name>

TO

<login or role name> [, … n]

[WITH GRANT OPTION]

[AS <role name>]

Trong đó: từ khóa ALL cho phép người sử dụng có tất cả các quyền. PERMISSION là một trong các quyền: SELECT, INSERT, UPDATE, DELETE, REFERENCE, EXECUTE. Chỉ rõ những bảng dữ liệu, view hoặc thủ tục nào cho phép người dùng truy cập và xử lý.

Từ chối quyền truy vấn và xử lý dữ liệu

DENY ALL | < PERMISSION> [, …n]

ON

<table hoặc view name> [(<column name> [, … n])]

| <stored hoặc extension procedure name>

http://www.ebook.edu.vn

TO

<login or role name> [, … n]

[Cascade]

Loại bỏ quyền truy vấn và xử lý dữ liệu

```
REVOKE [ GRANT OPTION FOR ]
   { ALL [ PRIVILEGES ] | permission [ ,...n ] }
   {
      [ ( column [ ,...n ] ) ] ON { table | view }
      | ON { table | view } [ ( column [ ,...n ] ) ]
      | ON { stored_procedure | extended_procedure }
      | ON { user_defined_function }
   }
{ TO | FROM }
   security_account [ ,...n ]
[ CASCADE ]
[ AS { group | role } ]
```

Quyền tạo đối tượng trong CSDL

Trong CSDL có một số đối tượng và các chức năng như sao lưu dữ liệu mà mỗi người sử dụng trên CSDL tùy theo chức năng và nhiệm vụ cụ thể được phép hay không được phép tạo các đối tượng như table, view, stored procedure, … và tạo CSDL.

Các quyền tạo các đối tượng như sau:

- ✓ Create Database.
- ✓ Create Table.
- ✓ Create View.
- ✓ Create Procedure.
- ✓ Create Rule.
- ✓ Create Default.
- ✓ Backup Database.
- ✓ Backup Log.

Để phân quyền tạo đối tượng trong CSDL cho người dùng, trong SQL Server có thể sử dụng thủ tục GRANT như sau:

GRANT < ALL | Statement [, … n]> TO <login ID> [, … n]

http://www.ebook.edu.vn

4. Vai trò của người sử dụng trong SQL Server và cơ sở dữ liệu

Vai trò trên SQL Server:

Vai trò (Role)	Diễn giải
sysadmin	Có các quyền tương đương với sa.
serveradmin	Cấu hình một số tham số và tắt server.
setupadmin	Bị giới hạn bớt một số chức năng liên kết server và khởi động một số thủ tục.
securityadmin	Quản lý người dùng và tạo CSDL.
processadmin	Được phép dừng các giao tác đang thực hiện trên CSDL và một số quá trình thực hiện khác của SQL Server.
dbcreator	Được phép tạo CSDL.
Diskadmin	Quản lý các tập tin liên quan đến CSDL SQL Server.

Vai trò trên CSDL:

Vai trò	Diễn giải
db_owner	Với vai trò này, người sử dụng (NSD) thuộc nhóm sở hữu CSDL mới có thể truy cập vào CSDL.
db_accessadmin	Thực hiện các chức năng giống như securityadmin.
db_datareader	NSD được phép select trên các bảng dữ liệu của các người dùng khác trong CSDL.
db_datawriter	NSD được phép insert, update, delete trên các bảng dữ liệu của các người dùng khác trong CSDL.
db_ddladmin	NSD có thể thêm hay chỉnh sửa các đối tượng của CSDL.
db_securityadmin	NSD có quyền tương đương với quyền của securityadmin.
db_backupoperator	NSD có thể thực hiện chức năng backup dữ liệu.
db_denydareader	Không cho phép sử dụng phát biểu SELECT trên tất cả các bảng dữ liệu của CSDL.
db_denydawriter	Không cho phép sử dụng phát biểu INSERT, UPDATE, DELETE trên tất cả các bảng dữ liệu của CSDL.

Sử dụng các thủ tục hệ thống để tạo một role mới, thêm một người dùng vào một role, loại bỏ người sử dụng ra khỏi một role.

- *Tạo một role*

  sp_addrole [**@rolename =**] '*role_name*'
  [**,** [**@ownername =**] '*owner*']

  Trong đó:

http://www.ebook.edu.vn

@rolename: tên role mới

@ownername: chủ sở hữu của role mới, mặc định là dbo

Sau khi tạo role mới cần phải gán một số quyền truy cập và xử lý trên các bảng dữ liệu nào đó trong CSDL cho role mới đó.

- *Thêm người sử dụng vào Role*

  sp_addrolemember [**@rolename** =] '*role_name*' ,
  [**@membername** =] '*login_ID*'

- *Loại bỏ người sử dụng ra khỏi một role*

  sp_droprolemember [**@rolename** =] '*role_name*' ,
  [**@membername** =] '*login_ID* '

II. Bản sao dữ liệu

Phần này sẽ giới thiệu kỹ thuật làm giảm lưu lượng dữ liệu giao dịch với SQL Server khi đã cấu hình nhiều Server trên mạng.

1. Giới thiệu về nhân bản dữ liệu

Nhân bản dữ liệu (Replication) là công cụ được sử dụng để sao chép một hoặc nhiều CSDL đến một hoặc nhiều server (SQL Server) khác. Các Server được đặt trong mạng máy tính nội bộ (LAN). Người khai thác có thể thực hiện truy cập đến CSDL có trong Server chứa dữ liệu được nhân bản. Dữ liệu giữa các máy được thực hiện đồng bộ với nhau theo lịch hoặc theo sự kiện, khi có yêu cầu. Nhân bản dữ liệu có những ưu điểm sau:

- Dữ liệu được lưu trữ ở nhiều nơi, hiệu quả trong việc có nhiều ứng dụng cùng truy cập, khai thác.
- Thích hợp với các ứng dụng phân tích dữ liệu của hệ thống xử lý giao dịch trực tuyến (OLTP) trong kho dữ liệu (Data warehouse).
- Có thể khai thác dữ liệu khi không kết nối đến Server.
- Giảm thiểu xung đột do số lượng lớn các giao dịch trên mạng.
- Là một giải pháp an toàn khi Server bị lỗi hoặc bảo dưỡng.

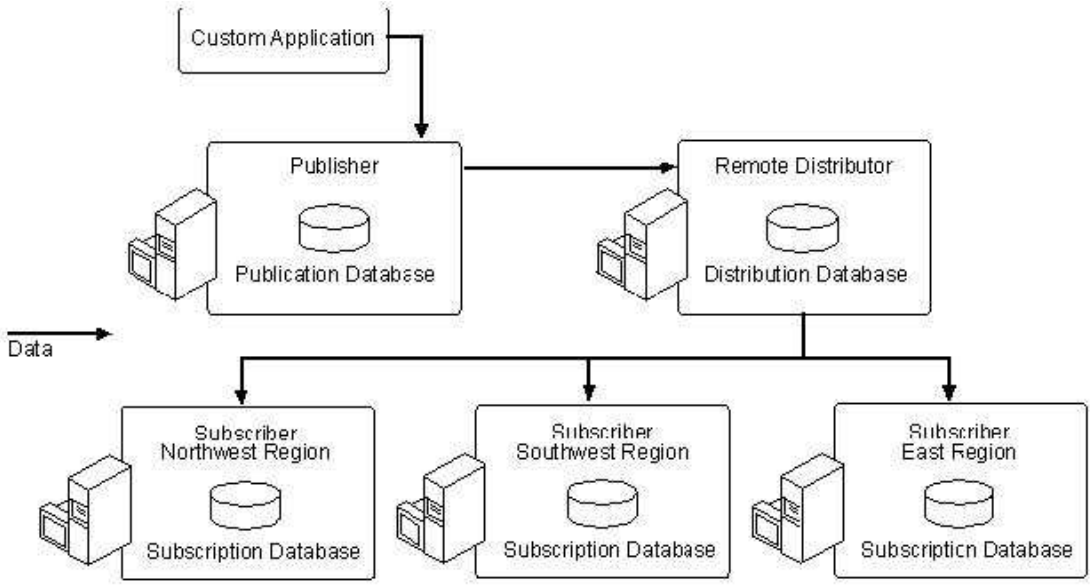
Mô hình nhân bản.

Dịch vụ nhân bản dữ liệu gồm các thành phần cơ bản sau: Publisher, Distributor, Subscribers, Publications, Articles, Subscriptions.

- *Publisher*: là server cung cấp dữ liệu nhân bản cho các server khác. Một publisher có thể thiết lập nhiều bộ dữ liệu nhân bản (gọi là publication).

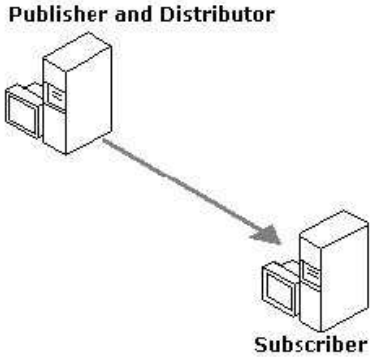
http://www.ebook.edu.vn

- *Distributor*: là server quản lý các thông tin nhân bản, lưu trữ dữ liệu trong các giao dịch thực hiện nhận và chuyển dữ liệu từ Publisher đến các Subscriber. Remote distributor là server tách rời khỏi publisher và được cấu hình là distributor. Local distributor là một server được cấu hình là Publisher và Distributor.
- *Subscriber*: Là server nhận dữ liệu nhân bản. Subscriber gắn liền với publication (là máy chủ nhận dữ liệu nhân bản của một bộ dữ liệu cấu hình nhân bản).
- *Article*: Là một bảng, tập dữ liệu hoặc đối tượng của CSDL cấu hình để nhân bản.
- *Publication*: Là một tập gồm một hoặc nhiều article.
- *Subscription*: Là một giao dịch yêu cầu bản sao bộ dữ liệu hoặc các đối tượng của CSDL thực hiện nhân bản. Trong mỗi giao dịch publisher thực hiện đẩy (push subscription) dữ liệu, subscriber thực hiện kéo (pull subscription).



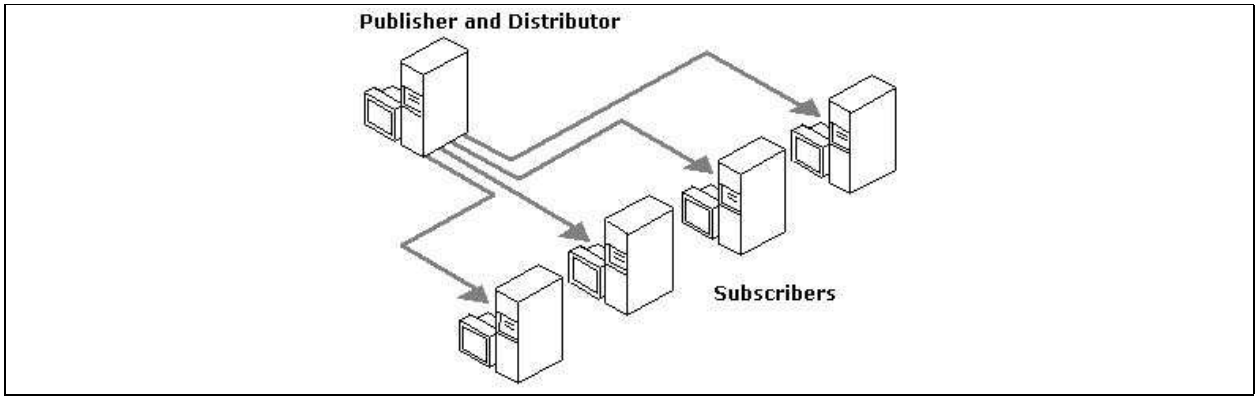
Nhân bản dữ liệu được thực hiện theo những mô hình cơ bản sau:

- *Central Publisher*: Là mô hình Publisher và Distributor thiết lập trên một máy. Gồm các mô hình sau:
  - ✓ Một Publisher và một Subscriber:

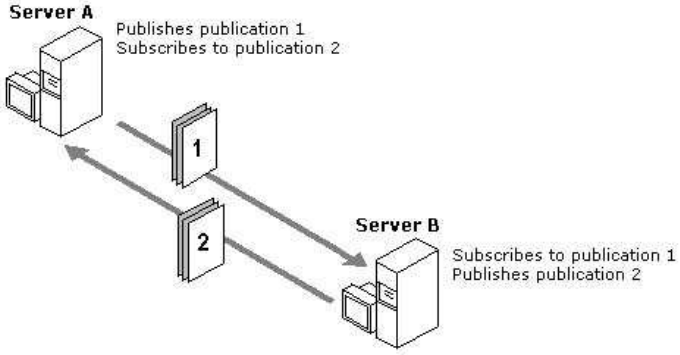


http://www.ebook.edu.vn

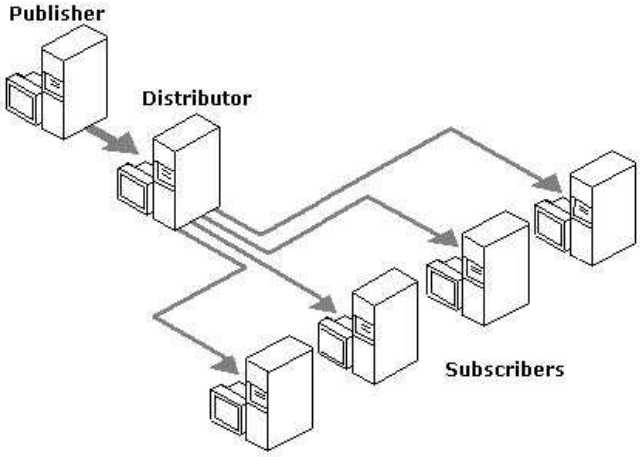
✓ Một Publisher và nhiều Subscriber.



✓ Publisher và Subscriber được thiết lập trên một máy:

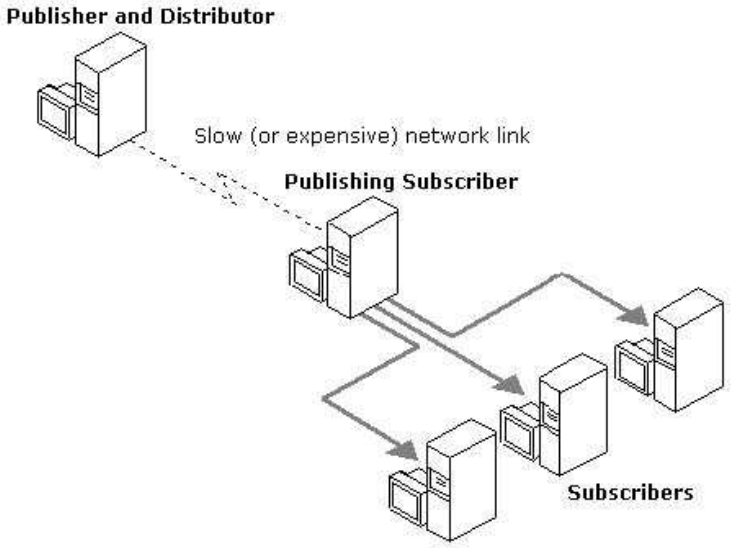


– *Publisher và Distributor không thiết lập trên một máy*:

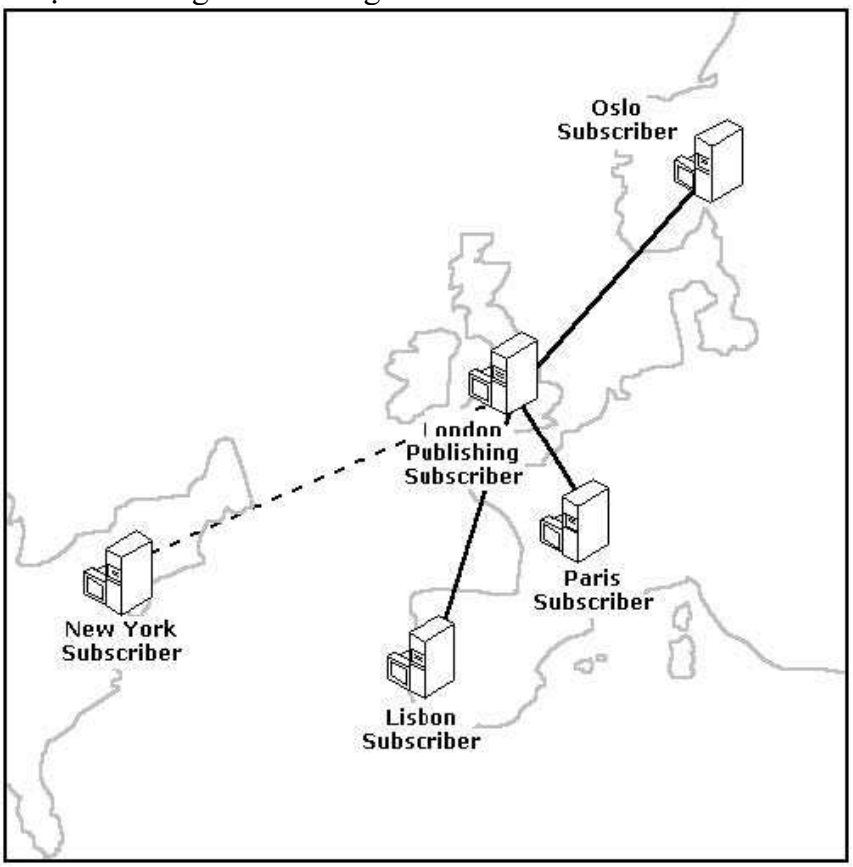


– *Republisher*: Là mô hình Publisher xuất bản dữ liệu đến Subscriber, sau đó Subscriber được thiết lập là Publisher xuất bản dữ liệu đến Subscriber khác.

http://www.ebook.edu.vn

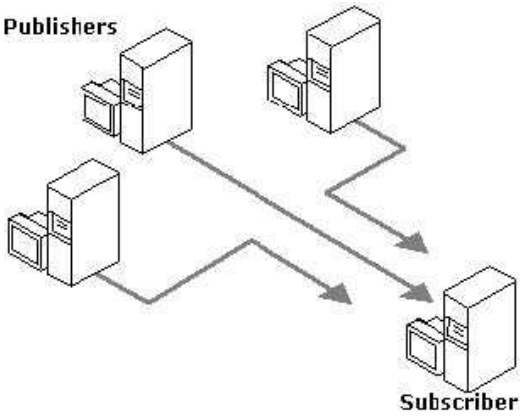


Đường truyền giữa hai máy được thiết lập là Publisher có thể tốc độ thấp, phù hợp với vị trí xa nhau. Ví dụ mô hình giữa các vùng cách xa nhau:



+ *Central Subscriber*: Là mô hình Subscriber thiết lập nhận dữ liệu xuất bản từ nhiều Publisher.

http://www.ebook.edu.vn

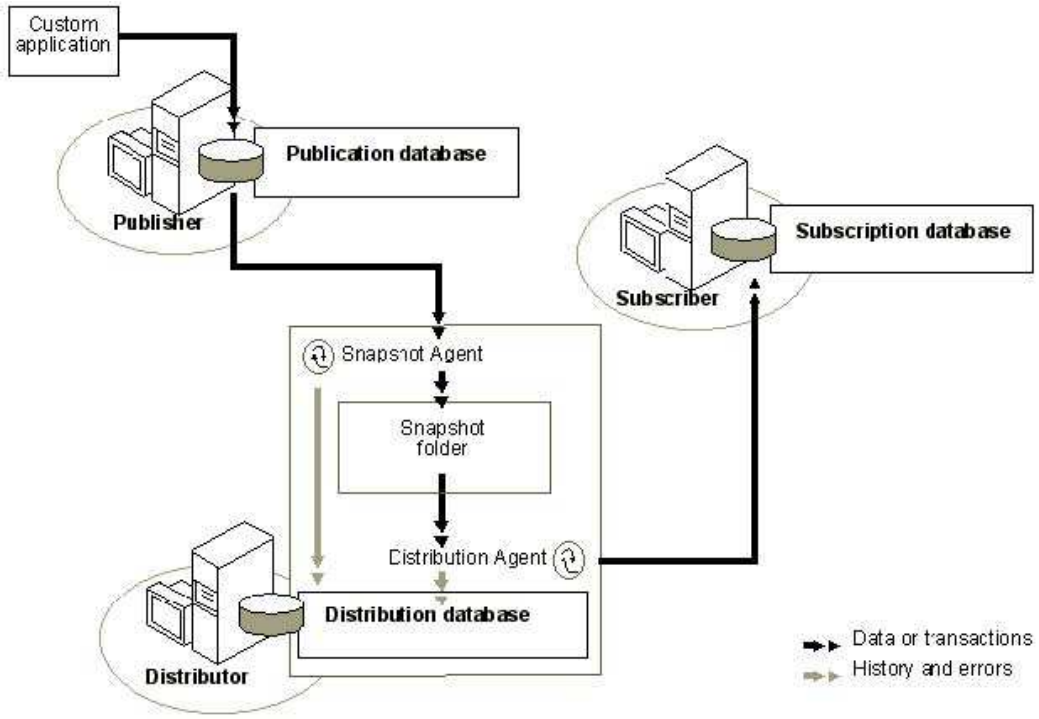


Những kiểu nhân bản dữ liệu.

Có 3 kiểu nhân bản dữ liệu: Snaphot, Transaction, Merge.

Snapshot replication: là kiểu nhân bản thực hiện sao chép, phân tán dữ liệu hoặc các đối tượng của CSDL tại một thời điểm. Snapshot thường được sử dụng cho những tình huống sau:

- ✓ Dữ liệu thường là tĩnh, ít thay đổi.
- ✓ Nhân bản số lượng dữ liệu nhỏ.

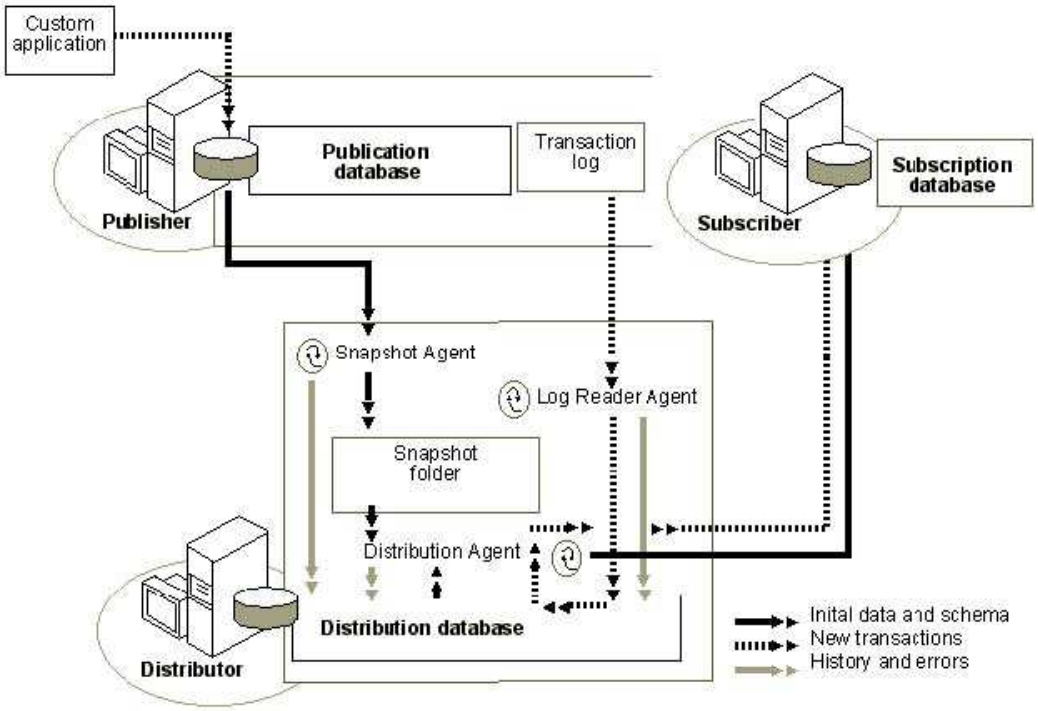


Transaction replication: là kiểu nhân bản mà bắt đầu bằng nhân bản Snapshot, sau đó sẽ thực hiện nhân giao dịch dữ liệu theo các sự kiện insert, update, delete và những thay đổi liên quan đến thực hiện thủ tục, khung nhìn chỉ mục. Nhân bản kiểu này cho phép thực hiện lọc dữ liệu tại Publisher, cho phép người sử dụng sửa đổi dữ liệu nhân bản tại

http://www.ebook.edu.vn

subscriber và chuyển dữ liệu đã sửa đổi đến Publisher hoặc Subscriber khác, dữ liệu sửa đổi này có thể coi là dữ liệu được xuất bản. Nhân bản kiểu này được thực hiện khi:

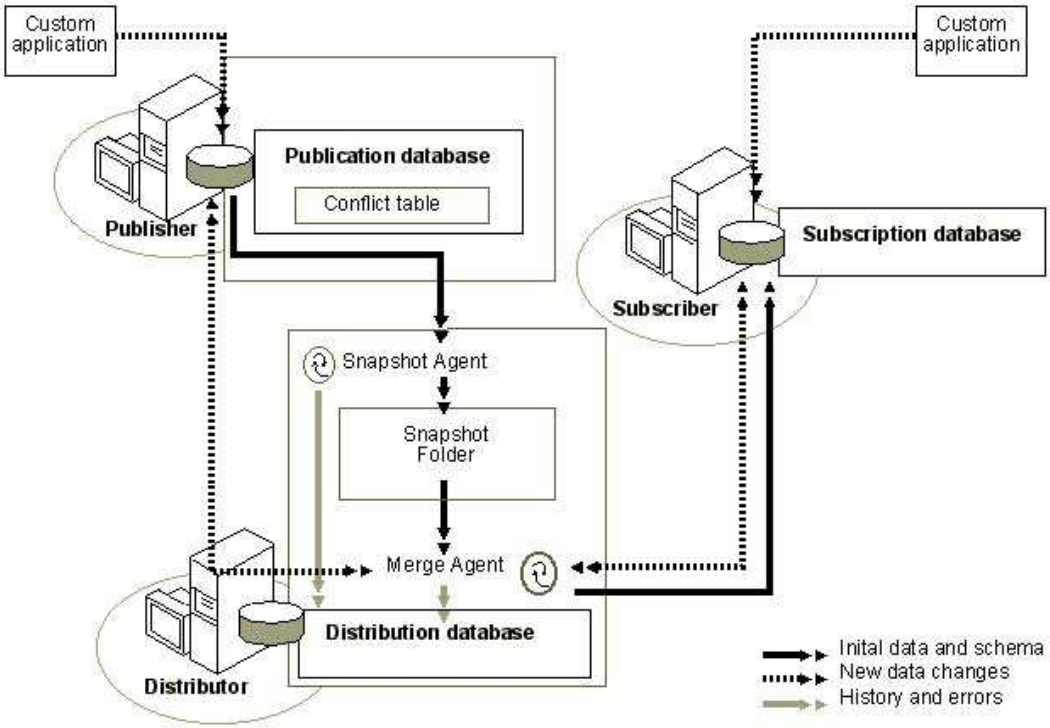
- ✓ Muốn sửa đổi dữ liệu được xuất bản chuyển đến Subscriber, thời gian thực hiện theo giây, hoặc tức thời.
- ✓ Cần giao dịch trên toàn bộ hệ thống nhân bản dữ liệu (dữ liệu có thể chuyển đến tất cả các Subscriber hoặc không chuyển đến Subscriber nào).
- ✓ Subscriber thường xuyên kết nối với Publisher.



Merge replication: là kiểu nhân bản dữ liệu cho phép thực hiện nhân bản sửa đổi dữ liệu trên nhiều Subscriber, có thể kết nối (online) hoặc không kết nối (offline) đến Publisher. Dữ liệu sẽ được đồng bộ theo lịch hoặc theo yêu cầu, dữ liệu cập nhật có thời điểm sau sẽ được chấp nhận. Kiểu nhân bản này thực hiện khi:

- ✓ Nhiều Subscriber có nhu cầu cập nhật dữ liệu và chuyển dữ liệu cập nhật đến Publisher hoặc Subscriber khác.
- ✓ Subscriber yêu cầu nhận hoặc chuyển dữ liệu khi offline, đồng bộ dữ liệu với các Subscriber và Publisher sau.

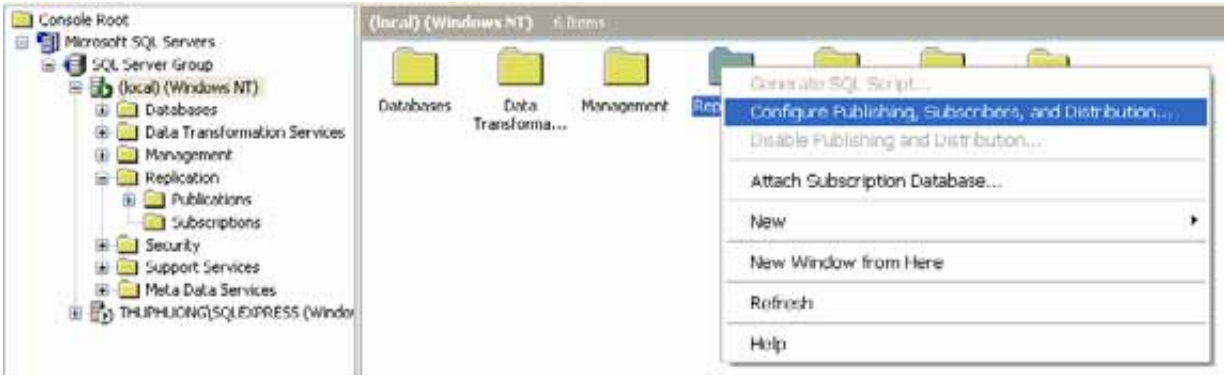
http://www.ebook.edu.vn



2. *Cấu hình Publisher và Distributor.*

Trước khi thực hiện cấu hình các máy thành Publisher hay Distributor ta phải thực hiện chạy dịch vụ SQL Server Agent trong chức năng Service manager. Các bước cấu hình như sau:

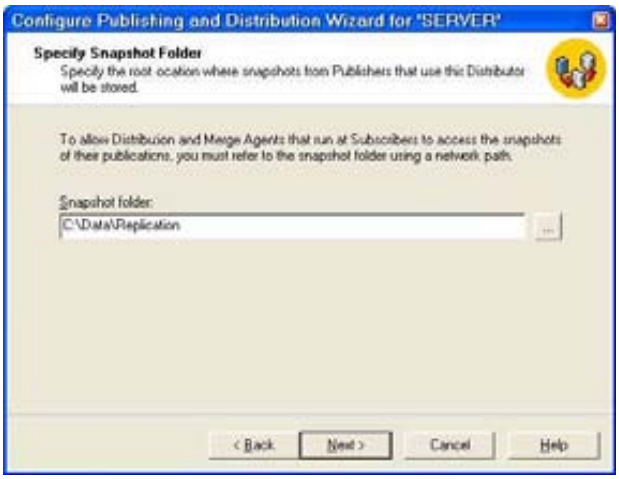
- Chọn Server cần cấu hình -> Replication
- Nhấp phải chuột -> Configure Publishing Subscription and Distribution...



- Thực hiện theo các bước:

+ Chọn thư mục Snapshot

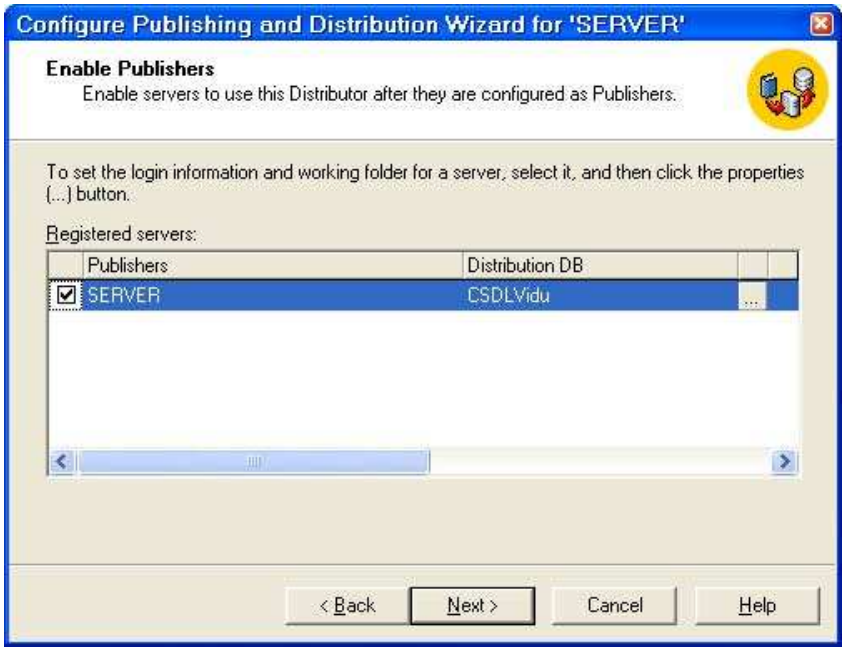
http://www.ebook.edu.vn



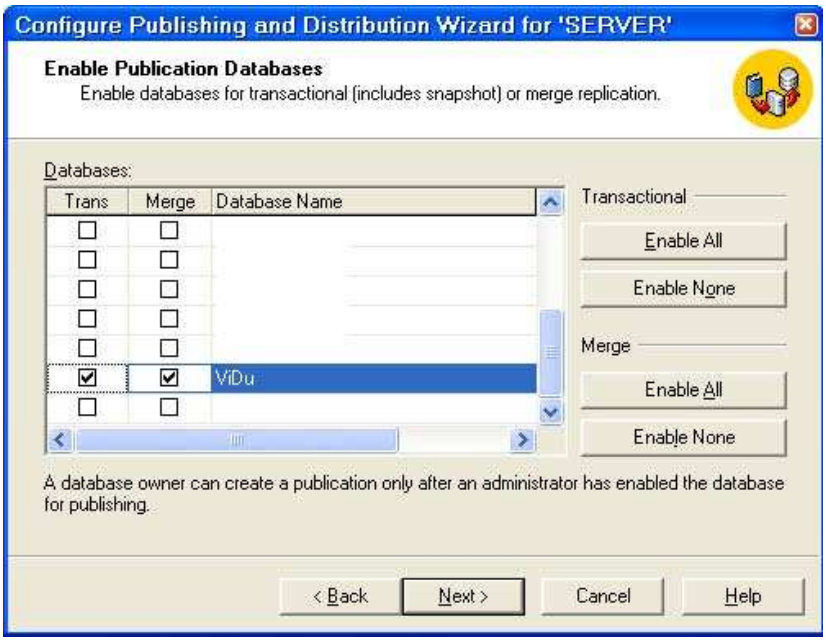
- Đặt tên CSDL của Distribution.

- Chọn Server cấu hình thành Publisher.

http://www.ebook.edu.vn



- Chọn CSDL tham gia nhân bản, kiểu nhân bản.



- Chọn Server được cấu hình là Subscriber của Publisher đang thiết lập.
- Kết thúc.

Tạo Publication

Bước này sẽ thực hiện tạo Publication, cách thực hiện như sau:

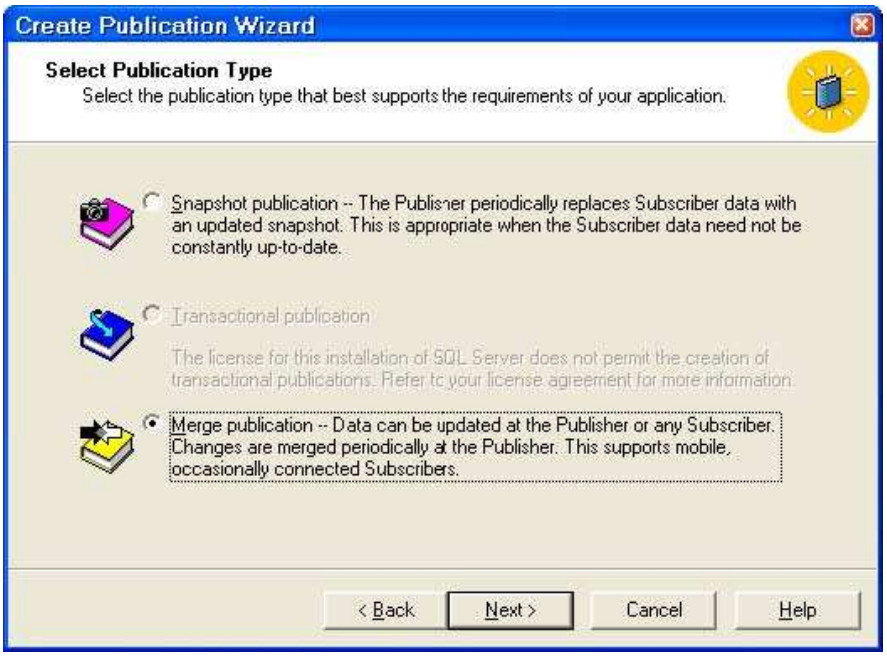
+ Chọn Publication trong Replication của Publisher.

+ Nhấn phải chuột → chọn New Publication...

http://www.ebook.edu.vn

+ Thực hiện theo các bước:

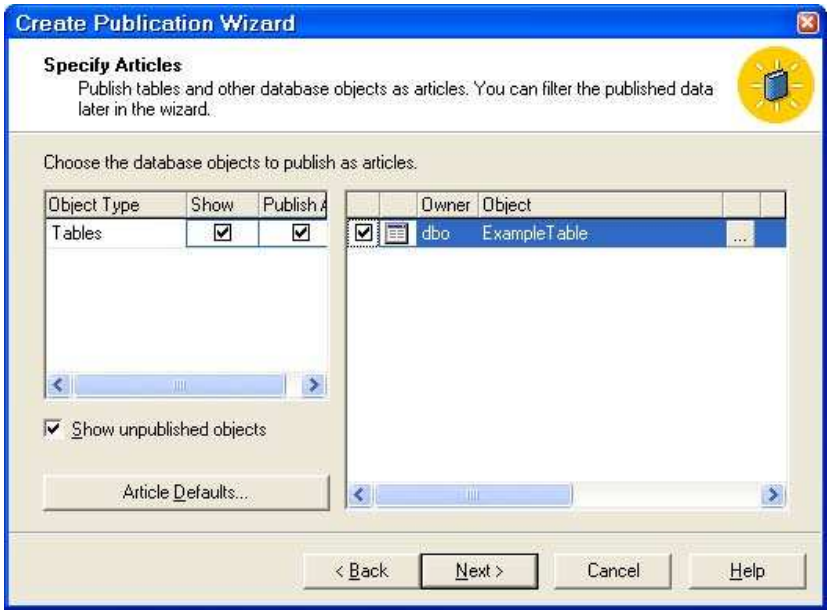
- Chọn CSDL cần xuất bản dữ liệu hoặc đối tượng.
- Chọn kiểu nhân bản (trong ví dụ này thực hiện kiểu Merge).



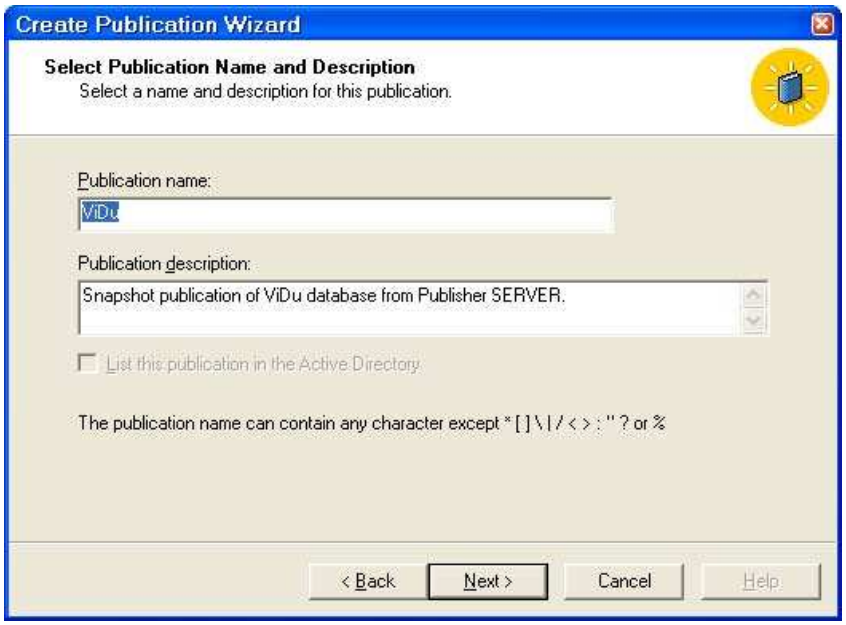
- Chọn phiên bản SQL Server của Subscriber.

- Chọn Article tham gia Publication.

http://www.ebook.edu.vn



- Đặt tên cho Publication.



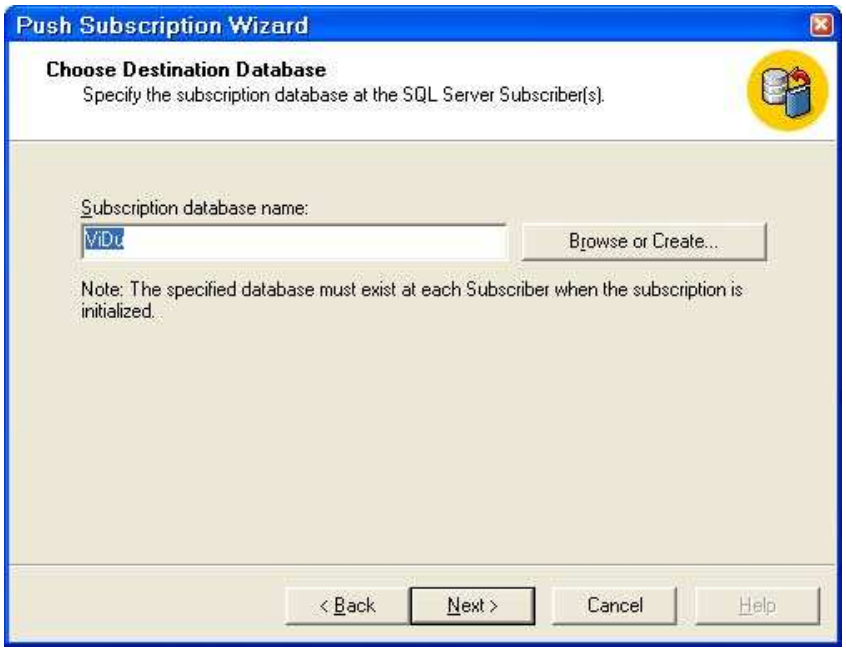
- Kết thúc.

***Tạo Push Subscription*.**

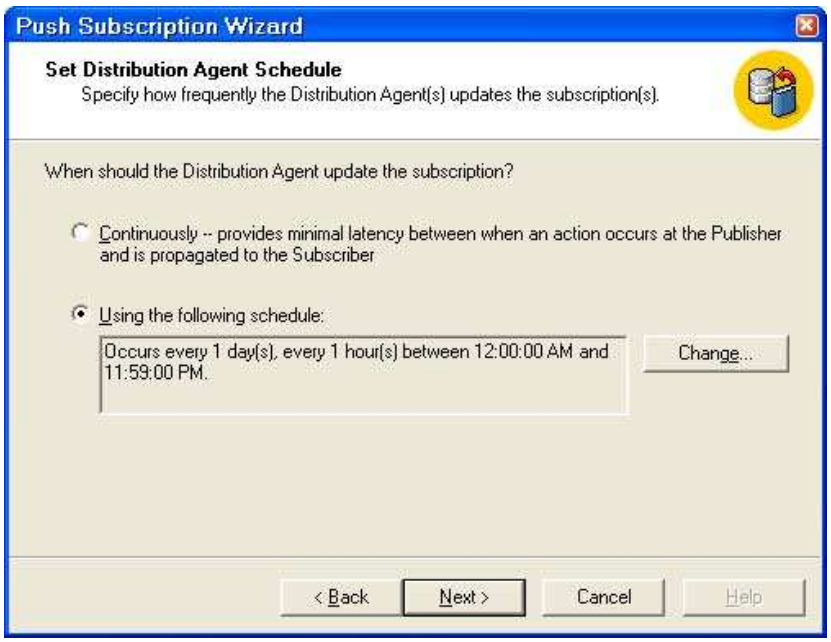
Bước này thực hiện tạo thủ tục đẩy (push) từ Publisher (Distributor trong ví dụ này) đến Subscriber, được thực hiện trên Publisher. Các bước thực hiện như sau:

http://www.ebook.edu.vn

- Chọn Publication của Publisher -> Nhấn phải chuột -> Push new Subscription...
- Chọn Subscriber.
- Chọn CSDL trên Subscriber nếu đã có, nếu chưa có thực hiện chọn chức năng tạo mới.



- Chọn lịch thực hiện đồng bộ dữ liệu.
- Kết thúc. Sau khi thiết lập xong trên Subscriber sẽ có CSDL theo tên đã tạo.



Tạo Pull Subscription

http://www.ebook.edu.vn

Bước này thực hiện tạo công cụ kéo dữ liệu nhân bản từ Publisher về Subscriber, được thực hiện trên Subscriber.

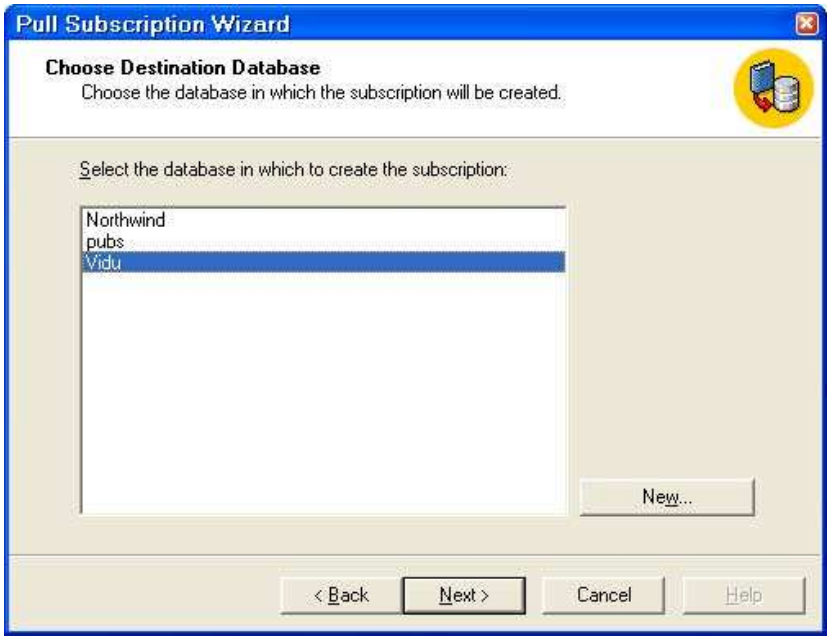
- Chọn Subscription của Subscriber → Nhấn phải chuột → New Pull Supcription...

- Thực hiện theo các bước:

  + Chọn Publication.

  + Chọn Agent tham gia kết nối Publisher.

- Chọn CSDL đích.



http://www.ebook.edu.vn

- Thực hiện tiếp các bước và kết thúc. Nếu đã tạo Push Subscription với một CSDL sẽ không được tạo Pull Subscription với CSDL đó.

Thực hiện đồng bộ dữ liệu

Sau khi thiết lập theo các mô hình nhân bản xong, có thể thực hiện đồng bộ dữ liệu bằng cách:

- Thực hiện theo lịch.

- Theo yêu cầu: Chọn Subscription (Push hoặc Pull) → Nhấn phải chuột → Start Synchronizing.

Sau khi thực hiện xong dữ liệu sẽ được đồng bộ giữa Publisher và Subscriber. Ngoài thực hiện theo công cụ ta có thể tìm hiểu thực hiện nhân bản theo câu lệnh T-SQL hoặc Stored Procedure.

III. Sao lưu và khôi phục dữ liệu

1. Lý do phải sao lưu và khôi phục dữ liệu

Trong quá trình thực hiện quản trị CSDL SQL Server thì một số nguyên nhân sau đây bắt buộc bạn phải xem xét đến kỹ thuật sao lưu và khôi phục dữ liệu:

- ✓ Ổ đĩa bị hỏng (chứa các tập tin CSDL).
- ✓ Server bị hỏng.
- ✓ Nguyên nhân bên ngoài (thiên nhiên, hỏa hoạn, mất cắp,...).
- ✓ User vô tình xóa dữ liệu.
- ✓ Bị vô tình hay cố ý làm thông tin sai lệch.
- ✓ Bị hack.

2. Các loại sao lưu dữ liệu

Sao lưu (Backup) dữ liệu trong SQL Server gồm các loại sau:

- ✓ Full Database Backups: sao chép toàn bộ CSDL (các tập tin bao gồm các bảng, khung nhìn, các đối tượng khác).
- ✓ Differential Database Backups: sao chép những dữ liệu thay đổi trong Data file kể từ lần full backup gần nhất.
- ✓ File or file group backups: sao chép một file đơn hay file group.
- ✓ Differential File or File Group Backups: thực hiện như Differential Database nhưng copy phần dữ liệu thay đổi của file đơn hoặc file group.
- ✓ Transaction log backups: Ghi nhận tất cả các transaction chứa trong transaction log file kể từ lần transaction log backup gần nhất. Với loại sao lưu này ta có thể khôi phục dữ liệu tại một thời điểm.

3. Các mô hình khôi phục dữ liệu

http://www.ebook.edu.vn

- ✓ Full Recovery model: Là mô hình phục hồi toàn bộ hoạt động giao dịch của dữ liệu (Insert, Update, Delete, hoạt động bởi lệnh bcp, bulk insert). Với mô hình này ta có thể phục hồi dữ liệu tại một thời điểm trong quá khứ đã được lưu trong transaction log file.
- ✓ Bulk-Logged Recovery Model: Mô hình này được thực thi cho các thao tác bcp, bulk insert, create chỉ mục, writetext, updatetext, các hoạt động này chỉ nhật ký sự kiện vào log để biết mà không sao lưu toàn bộ dữ liệu, chi tiết như trong full recover. Các sự kiện Insert, Update, Delete vẫn được nhật ký và khôi phục bình thường.
- ✓ Simple Recovery Model: Với mô hình này bạn chỉ phục hồi lại thời điểm sao lưu gần nhất mà không theo thời điểm khác trong quá khứ.
- ✓ Cách đặt mô hình khôi phục:

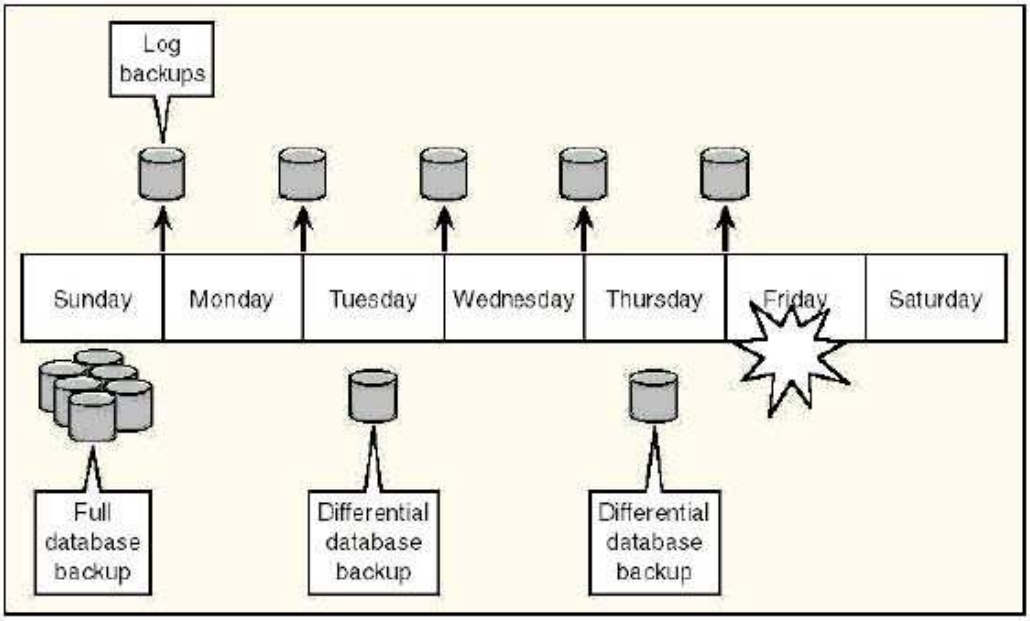
  - Chọn CSDL.

  - Nhấn nút phải chuột → Properties → Options → Recovery → chọn Model.

Xét ví dụ sau: Giả sử ta có một CSDL được backup theo chiến lược như hình vẽ bên dưới

Nhìn hình ta thấy CSDL được lập lịch Full Database Backup vào ngày chủ nhật, Differential Database Backup vào ngày thứ ba và thứ năm, còn Log Database Backup vào 5 ngày trong tuần, ngày thứ sáu có sự cố với CSDL, data file bị hỏng. Vấn đề đặt ra là phải phục hồi dữ liệu và CSDL hoạt động bình thường. Ta phải làm các bước sau:

- ✓ Thực hiện Backup log file (giả sử log file không bị hỏng).
- ✓ Khôi phục Full Database của ngày chủ nhật.
- ✓ Phục hồi Differential Database của ngày thứ năm.
- ✓ Khôi phục Transaction log backup ngày thứ năm.



http://www.ebook.edu.vn

4. Sao lưu cơ sở dữ liệu (Backup Database)

Trước khi xem xét kỹ thuật sao lưu CSDL, ta thống nhất một số thuật ngữ bằng tiếng Anh như sau:

- ✓ Backup: Là quá trình copy toàn bộ hoặc một phần database, transaction log, file, file group thành lập một backup set được chứa trong backup media (disk hoặc tape) bằng cách sử dụng một backup device (tape drive name hoặc physical filename).
- ✓ Backup Device: Một file vật lý hoặc một drive tape.
- ✓ Backup file: Một file chứa Backup set.
- ✓ Backup media: là Disk hoặc tape.
- ✓ Backup set: Một bộ backup một lần backup đơn chứa trên backup media.
- ✓ Các bước thực hiện backup như sau:
  - Chọn CSDL cần backup.
  - Nhấn phải chuột → All Tasks → Backup Database…
  - Nhập các tham số, lựa chọn kiểu.

5. Khôi phục dữ liệu (Restore Database)

Là chức năng thực hiện khôi phục dữ liệu đã sao lưu, tùy theo chiến lược backup mà người quản trị có thể phục hồi đến thời điểm nào, thu được bộ dữ liệu trong quá khứ như thế nào. Khôi phục dữ liệu được thực hiện theo thứ tự backup, thông tin này được lưu trữ trong msdb. Các bước thực hiện như sau:

- Chọn mục Databases → Nhấn nút phải chuột → All Tasks → Restore Database…
- Nhập tham số, chọn mô hình khôi phục.

IV. Quản lý giao dịch

1. Các khái niệm

Một giao dịch là một đơn vị thực hiện chương trình truy xuất và có thể cập nhật nhiều mục dữ liệu. Một giao dịch thường là kết quả của sự thực hiện một chương trình người dùng được viết trong một ngôn ngữ thao tác dữ liệu mức cao hoặc một ngôn ngữ lập trình (SQL, COBOL, PASCAL ...), và được phân cách bởi các câu lệnh (hoặc các lời gọi hàm) có dạng **begin transaction** và **end transaction**. Giao dịch bao gồm tất cả các hoạt động được thực hiện giữa begin và end transaction.

Để đảm bảo tính toàn vẹn của dữ liệu, ta yêu cầu hệ CSDL duy trì các tính chất sau của giao dịch:

- **Tính nguyên tử (Atomicity)**. Hoặc toàn bộ các hoạt động của giao dịch được phản ánh đúng đắn trong CSDL hoặc không có gì cả.
- **Tính nhất quán (consistency)**. Sự thực hiện của một giao dịch là cô lập (Không

http://www.ebook.edu.vn

có giao dịch khác thực hiện đồng thời) để bảo tồn tính nhất quán của CSDL.

- **Tính cô lập (Isolation)**. Cho dù nhiều giao dịch có thể thực hiện đồng thời, hệ thống phải đảm bảo rằng đối với mỗi cặp giao dịch T_i, T_j , hoặc T_j kết thúc thực hiện trước khi T_i khởi động hoặc T_j bắt đầu sự thực hiện sau khi T_i kết thúc. Như vậy mỗi giao dịch không cần biết đến các giao dịch khác đang thực hiện đồng thời trong hệ thống.
- **Tính bền vững (Durability)**. Sau một giao dịch hoàn thành, các thay đổi đã được tạo ra đối với CSDL vẫn còn ngay cả khi xảy ra sự cố hệ thống.

Các tính chất này thường được gọi là các tính chất ACID (Các chữ cái đầu của bốn tính chất). Ta xét một ví dụ: một hệ thống ngân hàng gồm một số tài khoản và một tập các giao dịch truy xuất và cập nhật các tài khoản. Tại thời điểm hiện tại, ta giả thiết rằng CSDL nằm trên đĩa, nhưng một vài phần của nó đang nằm tạm thời trong bộ nhớ. Các truy xuất CSDL được thực hiện bởi hai hoạt động sau:

- **READ(X)**: chuyển mục dữ liệu X từ CSDL đến buffer của giao dịch thực hiện hoạt động **READ** này.
- **WRITE(X)**: chuyển mục dữ liệu X từ buffer của giao dịch thực hiện **WRITE** đến CSDL.

Trong hệ CSDL thực, hoạt động **WRITE** không nhất thiết dẫn đến sự cập nhật trực tiếp dữ liệu trên đĩa; hoạt động **WRITE** có thể được lưu tạm thời trong bộ nhớ và được thực hiện trên đĩa muộn hơn. Trong ví dụ, ta giả thiết hoạt động **WRITE** cập nhật trực tiếp CSDL.

T_i là một giao dịch chuyển 50 từ tài khoản A sang tài khoản B. Giao dịch này có thể được xác định như sau:

T_i : READ(A);
A:=A - 50;
WRITE(A)
READ(B);
B:=B + 50;
WRITE(B);

Ta xem xét mỗi yêu cầu trong ACID:

- **Tính nhất quán**: Đòi hỏi nhất quán ở đây là tổng của A và B là không thay đổi bởi sự thực hiện giao dịch. Nếu không có yêu cầu nhất quán, tiền có thể được tạo ra hay mất bởi giao dịch. Dễ dàng kiểm nghiệm rằng nếu CSDL nhất quán trước một thực hiện giao dịch, nó vẫn nhất quán sau khi thực hiện giao dịch. Đảm bảo tính nhất quán

http://www.ebook.edu.vn

cho một giao dịch là trách nhiệm của người lập trình ứng dụng người đã viết ra giao dịch. Nhiệm vụ này có thể được làm cho dễ dàng bởi kiểm thử tự động các ràng buộc toàn vẹn.

- **Tính nguyên tử**: giả sử rằng ngay trước khi thực hiện giao dịch T_i, giá trị của các tài khoản A và B tương ứng là 1000 và 2000. Giả sử rằng trong khi thực hiện giao dịch T_i, một sự cố xảy ra cản trở T_i hoàn tất thành công sự thực hiện của nó. Ta cũng giả sử rằng sự cố xảy ra sau khi hoạt động **WRITE**(A) đã được thực hiện, nhưng trước khi hoạt động **WRITE**(B) được thực hiện. Trong trường hợp này giá trị của tài khoản A và B là 950 và 2000. Ta đã mất 50$. Tổng A+B không còn được bảo toàn. Như vậy, kết quả của sự cố là trạng thái của hệ thống không còn phản ánh trạng thái của thế giới thực mà CSDL được giả thiết nắm giữ. Ta gọi trạng thái như vậy là trạng thái không nhất quán. Ta phải đảm bảo rằng tính không nhất quán này không xuất hiện trong một hệ CSDL. Tuy nhiên, ở tại một vài thời điểm, hệ thống có thể ở trong trạng thái không nhất quán. Ví dụ giao dịch T_i, trong quá trình thực hiện cũng tồn tại thời điểm tại đó giá trị của tài khoản A là 950 và tài khoản B là 2000 – một trạng thái không nhất quán. Trạng thái này được thay thế bởi trạng thái nhất quán khi giao dịch đã hoàn tất. Như vậy, nếu giao dịch không bao giờ khởi động hoặc được đảm bảo sẽ hoàn tất, trạng thái không nhất quán sẽ không bao giờ xảy ra. Đó chính là lý do có yêu cầu về tính nguyên tử: Nếu tính chất nguyên tử được đảm bảo, ***tất cả các hành động của giao dịch được phản ánh trong CSDL hoặc không có gì cả***. ý tưởng cơ sở để đảm bảo tính nguyên tử là như sau: hệ CSDL lưu vết (trên đĩa) các giá trị cũ của bất kỳ dữ liệu nào trên đó giao dịch đang thực hiện viết, nếu giao dịch không hoàn tất, giá trị cũ được khôi phục để đặt trạng thái của hệ thống trở lại trạng thái trước khi giao dịch diễn ra. Đảm bảo tính nguyên tử là trách nhiệm của hệ CSDL, và được quản lý bởi một thành phần được gọi là thành phần quản trị giao dịch (transaction-management component).

- **Tính bền vững**: tính chất bền vững đảm bảo rằng mỗi khi một giao dịch hoàn tất, tất cả các cập nhật đã thực hiện trên cơ sở dữ liệu vẫn còn đó, ngay cả khi xảy ra sự cố hệ thống sau khi giao dịch đã hoàn tất. Ta giả sử một sự cố hệ thống có thể gây ra việc mất dữ liệu trong bộ nhớ chính, nhưng dữ liệu trên đĩa thì không mất. Có thể đảm bảo tính bền vững bởi việc đảm bảo hoặc ***các cập nhật được thực hiện bởi giao dịch đã được viết lên đĩa trước khi giao dịch kết thúc*** hoặc ***thông tin về sự cập nhật được thực hiện bởi giao dịch và được viết lên đĩa đủ cho phép CSDL xây dựng lại các cập nhật khi hệ CSDL được khởi động lại sau sự cố***. Đảm bảo tính bền vững là trách nhiệm của một thành phần của hệ CSDL được gọi là thành phần quản trị phục hồi (recovery-management component). Hai thành phần quản trị giao dịch và quản trị phục hồi quan hệ mật thiết với nhau.

http://www.ebook.edu.vn

- **Tính cô lập:** ngay cả khi tính nhất quán và tính nguyên tử được đảm bảo cho mỗi giao dịch, trạng thái không nhất quán vẫn có thể xảy ra nếu trong hệ thống có một số giao dịch được thực hiện đồng thời và các hoạt động của chúng đan xen theo một cách không mong muốn. Ví dụ, CSDL là không nhất quán tạm thời trong khi giao dịch chuyển khoản từ A sang B đang thực hiện, nếu một giao dịch khác thực hiện đồng thời đọc A và B tại thời điểm trung gian này và tính A+B, nó đã tham khảo một giá trị không nhất quán, sau đó nó thực hiện cập nhật A và B dựa trên các giá trị không nhất quán này, như vậy CSDL có thể ở trạng thái không nhất quán ngay cả khi cả hai giao dịch hoàn tất thành công. Một giải pháp cho vấn đề các giao dịch thực hiện đồng thời là ***thực hiện tuần tự các giao dịch***, tuy nhiên giải pháp này làm giảm hiệu năng của hệ thống. Các giải pháp khác cho phép nhiều giao dịch thực hiện tương tranh đã được phát triển ta sẽ thảo luận về chúng sau này. Tính cô lập của một giao dịch đảm bảo rằng sự thực hiện đồng thời các giao dịch dẫn đến một trạng thái hệ thống tương đương với một trạng thái có thể nhận được bởi thực hiện các giao dịch này một tại một thời điểm theo một thứ tự nào đó. Đảm bảo tính cô lập là trách nhiệm của một thành phần của hệ CSDL được gọi là thành phần quản trị tương tranh (concurrency-control component).

Trạng thái giao dịch

Nếu không có sự cố, tất cả các giao dịch đều hoàn tất thành công. Tuy nhiên, một giao dịch trong thực tế có thể không thể hoàn tất sự thực hiện của nó. Giao dịch như vậy được gọi là bị bỏ dở. Nếu ta đảm bảo được tính nguyên tử, một giao dịch bị bỏ dở không được phép làm ảnh hưởng tới trạng thái của CSDL. Như vậy, bất kỳ thay đổi nào từ giao dịch bị bỏ dở này đều phải bị hủy bỏ. Mỗi khi các thay đổi do giao dịch bị bỏ dở bị hủy bỏ, ta nói rằng giao dịch bị cuộn lại (rolled back). Khi một giao dịch hoàn tất một cách thành công sự thực hiện của nó được gọi là được bàn giao (committed). Một giao dịch được bàn giao, các lệnh cập nhật sẽ biến đổi CSDL sang một trạng thái nhất quán mới và nó là bền vững ngay cả khi có sự cố. Mỗi khi một giao dịch là được bàn giao, ta không thể hủy bỏ các hiệu quả của nó bằng cách bỏ dở nó. Cách duy nhất để hủy bỏ các hiệu quả của một giao dịch được bàn giao là thực hiện một giao dịch bù (compensating transaction), nhưng không phải luôn luôn có thể tạo ra một giao dịch bù. Do vậy trách nhiệm viết và thực hiện một giao dịch bù thuộc về người sử dụng và không được quản lý bởi hệ CSDL.

Một giao dịch phải ở trong một trong các trạng thái sau:

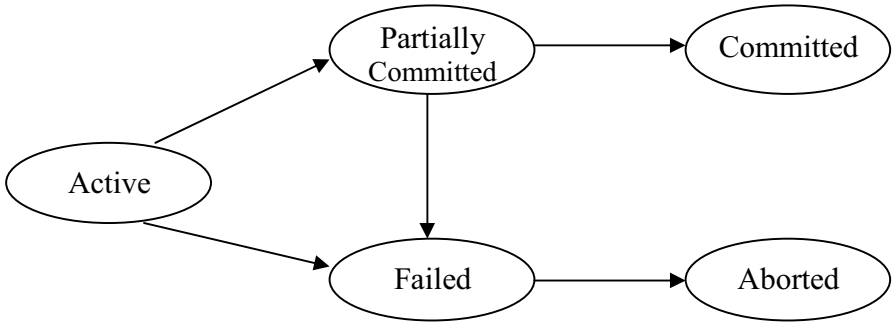
- **Hoạt động (Active)**: trạng thái khởi đầu; giao dịch giữ trong trạng thái này trong khi nó đang thực hiện.
- **Được bàn giao bộ phận (Partially Committed):** sau khi lệnh cuối cùng được thực hiện.

http://www.ebook.edu.vn

- **Thất bại (Failed):** sau khi phát hiện rằng sự thực hiện không thể tiếp tục được nữa.
- **Bỏ dở (Aborted):** sau khi giao dịch đã bị cuộn lại và CSDL đã phục hồi lại trạng thái của nó trước khi khởi động giao dịch.
- **Được bàn giao (Committed):** sau khi hoàn thành thành công giao dịch.

Ta nói một giao dịch đã được bàn giao (committed) chỉ nếu nó đã đi đến trạng thái Committed, tương tự, một giao dịch bị bỏ dở nếu nó đã đi đến trạng thái Aborted. *Một giao dịch được gọi là kết thúc nếu nó hoặc là committed hoặc là Aborted*. Một giao dịch khởi đầu bởi trạng thái *Active*. Khi nó kết thúc lệnh sau cùng của nó, nó chuyển sang trạng thái partially committed. Tại thời điểm này, giao dịch đã hoàn thành sự thực hiện của nó, nhưng nó vẫn có thể bị bỏ dở do đầu ra hiện tại vẫn có thể trú tạm thời trong bộ nhớ chính và như thế một sự cố phần cứng vẫn có thể ngăn cản sự hoàn tất của giao dịch. Hệ CSDL khi đó đã kịp viết lên đĩa đầy đủ thông tin giúp việc tái tạo các cập nhật đã được thực hiện trong quá trình thực hiện giao dịch, khi hệ thống tái khởi động sau sự cố. Sau khi các thông tin sau cùng này được viết lên đĩa, giao dịch chuyển sang trạng thái committed.

Biểu đồ trạng thái tương ứng với một giao dịch như sau:



Với giả thiết sự cố hệ thống không gây ra sự mất dữ liệu trên đĩa, một giao dịch đi đến trạng thái Failed sau khi hệ thống xác định rằng giao dịch không thể tiến triển bình thường được nữa (do lỗi phần cứng hoặc phần mềm). Như vậy, giao dịch phải được cuộn lại rồi chuyển sang trạng thái bỏ dở. Tại điểm này, hệ thống có hai lựa chọn:

- ***Khởi động lại giao dịch***, dùng lựa chọn này chỉ nếu giao dịch bị bỏ dở là do lỗi phần cứng hoặc phần mềm nào đó không liên quan đến logic bên trong của giao dịch. Giao dịch được khởi động lại được xem là một giao dịch mới.
- ***Huỷ giao dịch*** thường được tiến hành hoặc do lỗi logic bên trong giao dịch, lỗi này cần được chỉnh sửa bởi viết lại chương trình ứng dụng hoặc do đầu vào xấu hoặc do dữ liệu mong muốn không tìm thấy trong CSDL.

http://www.ebook.edu.vn

Ta phải thận trọng khi thực hiện thao tác viết ngoài khả quan sát (observable external **Write** - như viết ra terminal hay máy in). Mỗi khi một viết như vậy xảy ra, nó không thể bị xoá do nó có thể phải giao tiếp với bên ngoài hệ CSDL. Hầu hết các hệ thống cho phép các thao tác viết như thế xảy ra chỉ khi giao dịch đã di vào trạng thái committed. Một cách để thực thi một sơ đồ như vậy là cho hệ CSDL lưu trữ tạm thời bất kỳ giá trị nào kết hợp với các thao tác viết ngoài như vậy trong lưu trữ không hay thay đổi và thực hiện các thao tác viết hiện tại chỉ sau khi giao dịch đã đi vào trạng thái committed. Nếu hệ thống thất bại sau khi giao dịch đi vào trạng thái committed nhưng trước khi hoàn tất các tháo tác viết ngoài, hệ CSDL sẽ làm các thao tác viết ngoài này (sử dụng dữ liệu trong lưu trữ không hay thay đổi) khi hệ thống khởi động lại.

Trong một số ứng dụng, có thể muốn cho phép giao dịch hoạt động trình bày dữ liệu cho người sử dụng, đặc biệt là các giao dịch kéo dài trong vài phút hay vài giờ. Ta không thể cho phép xuất ra dữ liệu khả quan sát như vậy trừ phi ta buộc phải làm tổn hại tính nguyên tử giao dịch. Hầu hết các hệ thống giao dịch hiện hành đảm bảo tính nguyên tử và do vậy cấm dạng trao đổi với người dùng này.

Thực thi tính nguyên tử và tính bền vững

Thành phần quản trị phục hồi của một hệ CSDL hỗ trợ tính nguyên tử và tính bền vững. Trước tiên ta xét một sơ đồ đơn giản (song cực kỳ thiếu hiệu quả). Sơ đồ này giả thiết rằng chỉ một giao dịch là hoạt động tại một thời điểm và được dựa trên việc tạo bản sao của CSDL được gọi là các bản sao khuất (shadow copies). Sơ đồ giả thiết rằng CSDL chỉ là một file trên đĩa. Một con trỏ được gọi là db_pointer được duy trì trên đĩa, nó trỏ tới bản sao hiện hành của CSDL.

Trong sơ đồ CSDL khuất (shadow-database), một giao dịch muốn cập nhật CSDL, đầu tiên tạo ra một bản sao đầy đủ của CSDL. Tất cả các thao tác cập nhật được làm trên bản sao này, không đụng tới bản gốc. Nếu tại một thời điểm bất kỳ giao dịch bị bỏ dở, bản sao mới bị xoá. Bản sao cũ của CSDL không bị ảnh hưởng. Nếu giao dịch hoàn tất, nó được được bàn giao như sau: đầu tiên, hỏi hệ điều hành để đảm bảo rằng tất cả các trang của bản sao mới đã được viết lên đĩa (flush). Sau khi flush con trỏ db_pointer được cập nhật để trỏ đến bản sao mới; bản sao mới trở thành bản sao hiện hành của CSDL. Bản sao cũ bị xoá đi. Giao dịch được gọi là đã được được bàn giao tại thời điểm sự cập nhật con trỏ db_pointer được ghi lên đĩa. Ta xét kỹ thuật này quản lý sự cố giao dịch và sự cố hệ thống ra sao? Trước tiên, ta xét sự cố giao dịch. Nếu giao dịch thất bại tại thời điểm bất kỳ trước khi con trỏ db_pointer được cập nhật, nội dung cũ của CSDL không bị ảnh hưởng. Ta có thể bỏ dở giao dịch bởi xoá bản sao mới. Mỗi khi giao dịch được được bàn giao, tất cả các cập nhật mà nó đã thực hiện là ở trong CSDL được trỏ bởi db_pointer. Như vậy, hoặc tất cả các cập nhật của giao dịch đã được phản ánh hoặc không kết quả nào được phản ánh,

http://www.ebook.edu.vn

bất chấp tới sự cố giao dịch. Bây giờ ta xét sự cố hệ thống. Giả sử sự cố hệ thống xảy ra tại thời điểm bất kỳ trước khi db_pointer đã được cập nhật được viết lên đĩa. Khi đó, khi hệ thống khởi động lại, nó sẽ đọc db_pointer và như vậy sẽ thấy nội dung gốc của CSDL – không kết quả nào của giao dịch được nhìn thấy trên CSDL. Bây giờ lại giả sử rằng sự cố hệ thống xảy ra sau khi db_pointer đã được cập nhật lên đĩa. Trước khi con trỏ được cập nhật, tất cả các trang được cập nhật của bản sao mới đã được viết lên đĩa. Từ giả thiết file trên đĩa không bị hư hại do sự cố hệ thống. Do vậy, khi hệ thống khởi động lại, nó sẽ đọc db_pointer và sẽ thấy nội dung của CSDL sau tất cả các cập nhật đã thực hiện bởi giao dịch. Sự thực thi này phụ thuộc vào việc viết lên db_pointer, việc viết này phải là nguyên tử, có nghĩa là hoặc tất cả các byte của nó được viết hoặc không byte nào được viết. Nếu chỉ một số byte của con trỏ được cập nhật bởi việc viết nhưng các byte khác thì không thì con trỏ trở thành vô nghĩa và cả bản cũ lẫn bản mới của CSDL có thể tìm thấy khi hệ thống khởi động lại. May mắn thay, hệ thống đĩa cung cấp các cập nhật nguyên tử toàn bộ khối đĩa hoặc ít nhất là một sector đĩa. Như vậy hệ thống đĩa đảm bảo việc cập nhật con trỏ db_pointer là nguyên tử. Tính nguyên tử và tính bền vững của giao dịch được đảm bảo bởi việc thực thi bản sao bóng của thành phần quản trị phục hồi. Sự thực thi này cực kỳ thiếu hiệu quả trong ngữ cảnh CSDL lớn, do sự thực hiện một giao dịch đòi hỏi phải sao toàn bộ CSDL. Hơn nữa sự thực thi này không cho phép các giao dịch thực hiện đồng thời với các giao dịch khác. Phương pháp thực thi tính nguyên tử và tính lâu bền mạnh hơn và đỡ tốn kém hơn được trình bày trong chương *hệ thống phục hồi*.

Các thực hiện tương tranh

Hệ thống xử lý giao dịch thường cho phép nhiều giao dịch thực hiện đồng thời. Việc cho phép nhiều giao dịch cập nhật dữ liệu đồng thời gây ra những khó khăn trong việc bảo đảm sự nhất quán dữ liệu. Bảo đảm sự nhất quán dữ liệu mà không quan tâm tới sự thực hiện tương tranh các giao dịch sẽ cần thêm các công việc phụ. Một phương pháp tiến hành là cho các giao dịch thực hiện tuần tự: đảm bảo rằng một giao dịch khởi động chỉ sau khi giao dịch trước đã hoàn tất. Tuy nhiên có hai lý do hợp lý để thực hiện tương tranh là:

- Một giao dịch gồm nhiều bước. Một vài bước liên quan tới hoạt động I/O; các bước khác liên quan đến hoạt động CPU. CPU và các đĩa trong một hệ thống có thể hoạt động song song. Do vậy hoạt động I/O có thể được tiến hành song song với xử lý tại CPU. Sự song song của hệ thống CPU và I/O có thể được khai thác để chạy nhiều giao dịch song song. Trong khi một giao dịch tiến hành một hoạt động đọc/viết trên một đĩa, một giao dịch khác có thể đang chạy trong CPU, một giao dịch thứ ba có thể thực hiện đọc/viết trên một đĩa khác ... như vậy sẽ tăng lượng đầu vào hệ thống có nghĩa là tăng số lượng giao dịch có thể được thực hiện trong một lượng thời gian đã cho, cũng

http://www.ebook.edu.vn

có nghĩa là hiệu suất sử dụng bộ xử lý và đĩa tăng lên.

- Có thể có sự trộn lẫn các giao dịch đang chạy trong hệ thống, cái thì dài cái thì ngắn. Nếu thực hiện tuần tự, một quá trình ngắn có thể phải chờ một quá trình dài đến trước hoàn tất, mà điều đó dẫn đến một sự trì hoãn không lường trước được trong việc chạy một giao dịch. Nếu các giao dịch đang hoạt động trên các phần khác nhau của CSDL, sẽ tốt hơn nếu ta cho chúng chạy đồng thời, chia sẻ các chu kỳ CPU và truy xuất đĩa giữa chúng. Thực hiện tương tranh làm giảm sự trì hoãn không lường trước trong việc chạy các giao dịch, đồng thời làm giảm thời gian đáp ứng trung bình: *Thời gian để một giao dịch được hoàn tất sau khi đã được đệ trình*.

Động cơ để sử dụng thực hiện tương tranh trong CSDL cũng giống như động cơ để thực hiện đa chương trong hệ điều hành. Khi một vài giao dịch chạy đồng thời, tính nhất quán CSDL có thể bị vi phạm cho dù mỗi giao dịch là đúng. Một giải pháp để giải quyết vấn đề này là sử dụng thời lịch (schedule). Hệ CSDL phải điều khiển sự trao đổi giữa các giao dịch tương tranh để ngăn ngừa chúng phá huỷ sự nhất quán của CSDL. Các cơ chế cho điều đó được gọi là sơ đồ điều khiển tương tranh (concurrency-control scheme).

Xét hệ thống ngân hàng đơn giản, nó có một số tài khoản và có một tập hợp các giao dịch, chúng truy xuất, cập nhật các tài khoản này. Giả sử T_1 và T_2 là hai giao dịch chuyển khoản từ một tài khoản sang một tài khoản khác. Giao dịch T_1 chuyển 50$ từ tài khoản A sang tài khoản B. Giao dịch T_2 chuyển 10% số dư từ tài khoản A sang tài khoản B, và được xác định như sau:

T_1 : **Read**(A); A:=A-50; **Write**(A); **Read**(B); B:=B+50; **Write**(B);	T_2 : **Read**(A); Temp:=A*0.1; A:=A-temp; **Write**(A); **Read**(B); B:=B+temp; **Write**(B);

Giả sử giá trị hiện tại của A và B tương ứng là 1000$ và 2000$. Giả sử rằng hai giao dịch này được thực hiện theo trình tự:

Trường hợp 1: thực hiện xong giao dịch T_1 rồi đến giao dịch T_2

Trường hợp 2: thực hiện xong giao dịch T_2 rồi đến giao dịch T_1

http://www.ebook.edu.vn

T_1	T_2
Read(A); A:=A-50; **Write**(A); **Read**(B); B:=B+50; **Write**(B);	
	Read(A); Temp:=A*0.1; A:=A-temp; **Write**(A); **Read**(B); B:=B+temp; **Write**(B);
Thời lịch 1: Giá trị sau cùng của A là 855, B là 2145, tổng 2 tài khoản (A+B) là không đổi	

T_1	T_2
	Read(A); Temp:=A*0.1; A:=A-temp; **Write**(A); **Read**(B); B:=B+temp; **Write**(B);
Read(A); A:=A-50; **Write**(A); **Read**(B); B:=B+50; **Write**(B);	
Thời lịch 2: Giá trị sau cùng của A là 850, B là 2150, tổng 2 tài khoản (A+B) là không đổi	

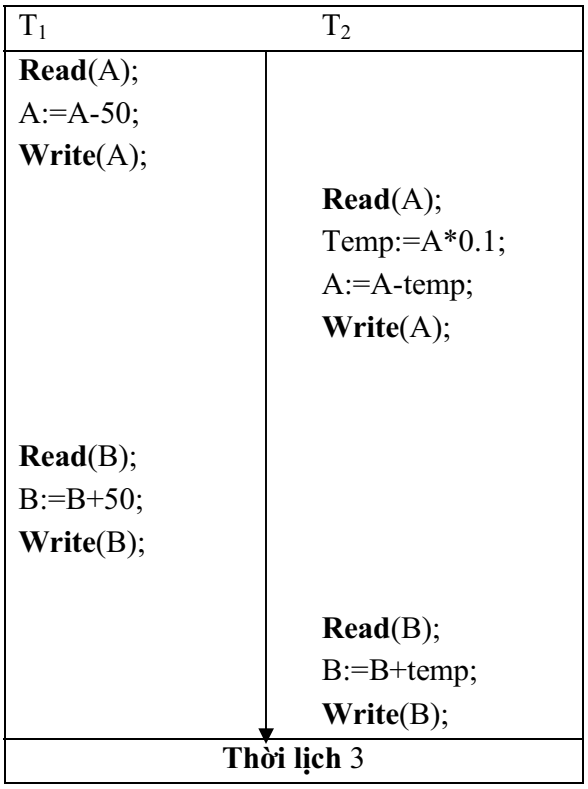
Thời lịch (schedule): là một dãy các thao tác (lệnh) của các giao tác được sắp xếp theo trình tự thời gian.

Một thời lịch đối với một tập các giao dịch phải bao gồm tất cả các chỉ thị của các giao dich này và phải bảo tồn thứ tự các chỉ thị xuất hiện trong mỗi một giao dịch. Ví dụ, đối với giao dịch T_1, chỉ thị **Write**(A) phải xuất hiện trước chỉ thị **Read**(B), trong bất kỳ thời lịch hợp lệ nào. Các thời lịch 1 và Thời lịch 2 là những thời lịch tuần tự.

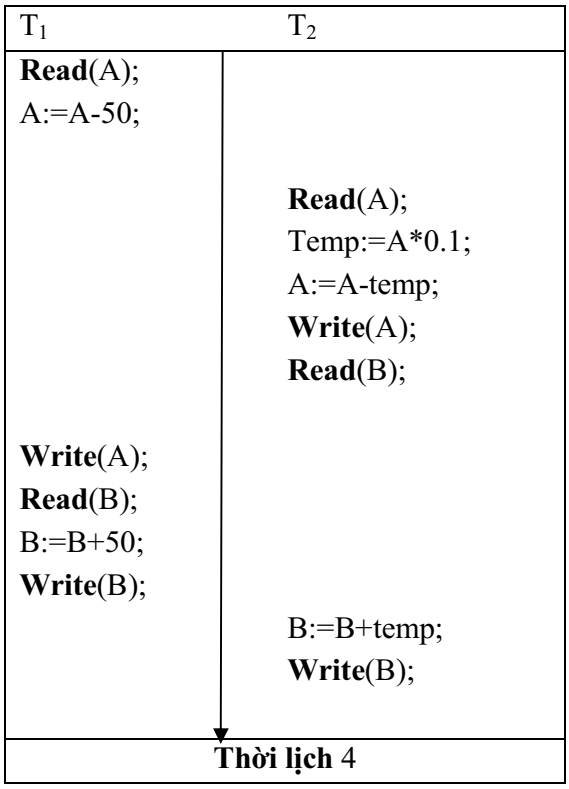
Thời lịch tuần tự gồm một dãy các chỉ thị từ các giao dịch, trong đó các chỉ thị thuộc về một giao dịch xuất hiện cùng nhau trong thời lịch. Như vậy, đối với một tập n giao dịch, có n! thời lịch tuần tự hợp lệ khác nhau.

Khi một số giao dịch được thực hiện đồng thời, thời lịch tương ứng không nhất thiết là tuần tự. Nếu hai giao dịch đang chạy đồng thời, hệ điều hành có thể thực hiện một giao dịch trong một khoảng ngắn thời gian, sau đó chuyển đổi ngữ cảnh, thực hiện giao dịch thứ hai một khoảng thời gian sau đó lại chuyển sang thực hiện giao dịch thứ nhất một khoảng và cứ như vậy (hệ thống chia sẻ thời gian). Thời lịch 3 và Thời lịch 4 là các ví dụ.

http://www.ebook.edu.vn



Thời lịch 3



Thời lịch 4

Tuy nhiên, không phải tất cả các thực hiện tương tranh đều cho ra một trạng thái đúng. Ví dụ xét thời lịch 4:

Sau khi thực hiện giao dịch này, ta đạt tới trạng thái trong đó giá trị cuối của A và B tương ứng là 950$ và 2100$. Trạng thái này là một trạng thái không nhất quán (A+B trước khi thực hiện giao dịch là 3000$ nhưng sau khi giao dịch là 3050$). Như vậy, nếu giao phó việc điều khiển thực hiện tương tranh cho hệ điều hành, sẽ có thể dẫn tới các trạng thái không nhất quán. Nhiệm vụ của hệ CSDL là đảm bảo rằng một thời lịch được phép thực hiện tương tranh sẽ đưa CSDL sang một trạng thái nhất quán. Thành phần của hệ CSDL thực hiện nhiệm vụ này được gọi là thành phần điều khiển tương tranh (concurrency-control component). Ta có thể đảm bảo sự nhất quán của CSDL với thực hiện tương tranh bằng cách nắm chắc rằng một thời lịch được thực hiện có cùng hiệu quả như một thời lịch tuần tự.

Tính khả tuần tự (Serializability)

Hệ CSDL phải điều khiển sự thực hiện tương tranh các giao dịch để đảm bảo rằng trạng thái CSDL giữ nguyên ở trạng thái nhất quán. Trước khi ta kiểm tra hệ CSDL có thể thực hiện nhiệm vụ này như thế nào, đầu tiên ta phải hiểu các thời lịch nào sẽ đảm bảo tính nhất quán và các thời lịch nào không. Vì các giao dịch là các chương trình, nên thật khó xác định các hoạt động chính xác được thực hiện bởi một giao dịch là hoạt động gì và

http://www.ebook.edu.vn

những hoạt động nào của các giao dịch tác động lẫn nhau. Vì lý do này, ta sẽ không giải thích kiểu hoạt động mà một giao dịch có thể thực hiện trên một mục dữ liệu. Thay vào đó, ta chỉ xét hai hoạt động: **Read** và **Write**. Ta cũng giả thiết rằng giữa một chỉ thị **Read(Q)** và một chỉ thị **Write(Q)** trên một mục dữ liệu **Q,** một giao dịch có thể thực hiện một dãy tuỳ ý các hoạt động trên bản sao của Q được lưu trú trong buffer cục bộ của giao dịch. Vì vậy ta sẽ chỉ nêu các chỉ thị **Read** và **Write** trong thời lịch, nếu biểu diễn với quy ước như vậy của thời lịch 3 sẽ là:

T_1	T_2
Read(A);	
Write(A);	
	Read(A);
	Write(A);
Read(B);	
Write(B);	
	Read(B);
	Write(B);

Tuần tự xung đột (Conflict Serializability)

Xét thời lịch S trong đó có hai chỉ thị liên tiếp I_i và I_j của các giao dịch T_i , T_j tương ứng ($i \neq j$). Nếu I_i và I_j tham khảo đến các mục dữ liệu khác nhau, ta có thể đổi chỗ I_i và I_j mà không làm ảnh hưởng đến kết quả của bất kỳ chỉ thị nào trong thời lịch. Tuy nhiên, nếu I_i và I_j tham khảo cùng một mục dữ liệu Q, khi đó thứ tự của hai bước này có thể rất quan trọng. Do ta đang thực hiện chỉ các chỉ thị **Read** và **Write**, nên ta có bốn trường hợp cần phải xét sau:

1. I_i = **Read**(Q); I_j = **Read**(Q): Thứ tự của I_i và I_j không gây ra vấn đề nào, do T_i và T_j đọc cùng một giá trị Q bất kể đến thứ tự giữa I_i và I_j.
2. I_i = **Read**(Q); I_j = **Write**(Q): thứ tự thực hiện của I_i và I_j là quan trọng.
3. I_i = **Write**(Q); I_j = **Read**(Q): thứ tự thực hiện của I_i và I_j là quan trọng.
4. I_i = **Write**(Q); I_j = **Write**(Q): Cả hai chỉ thị là hoạt động **Write**, thứ tự của hai chỉ thị này không ảnh hưởng đến cả hai giao dịch T_i và T_j . Tuy nhiên, giá trị nhận được bởi chỉ thị **Read** kế trong S sẽ bị ảnh hưởng do kết quả phụ thuộc vào chỉ thị **Write** được thực hiện sau cùng trong hai chỉ thị **Write** này. Nếu không còn chỉ thị **Write** nào sau I_i và I_j trong S, thứ tự của thứ tự thực hiện của I_i và I_j sẽ ảnh hưởng trực tiếp đến giá trị cuối của Q trong trạng thái CSDL kết quả (của thời lịch S).

http://www.ebook.edu.vn

Như vậy chỉ trong trường hợp cả I_i và I_j là các chỉ thị **Read**, thứ tự thực hiện của hai chỉ thị này (trong S) là không gây ra vấn đề.

*Ta nói I_i và I_j xung đột nếu các hoạt động này nằm trong các giao dịch khác nhau, tiến hành trên cùng một mục dữ liệu và có ít nhất một hoạt động là **Write**.*

Ví dụ, trong thời lịch schedule - 3: Chỉ thị **Write(A)** trong T_1 xung đột với **Read(A)** trong T_2. Tuy nhiên, chỉ thị **Write(A)** trong T_2 không xung đột với chỉ thị **Read(B)** trong T_1 do các chỉ thị này truy xuất các mục dữ liệu khác nhau.

I_i và I_j là hai chỉ thị liên tiếp trong thời lịch S. Nếu I_i và I_j là các chỉ thị của các giao dịch khác nhau và không xung đột, khi đó ta có thể đổi thứ tự của chúng mà không làm ảnh hưởng gì đến kết quả xử lý và như vậy ta nhận được một thời lịch mới S' tương đương với S. Chẳng hạn, do chỉ thị **Write(A)** của T_2 không xung đột với chỉ thị **Read(B)** của T_1, ta có thể đổi chỗ các chỉ thị này để được một thời lịch tương đương – thời lịch 5 dưới đây

T_1	T_2
Read(A);	
Write(A);	
	Read(A);
Read(B);	
	Write(A);
Write(B);	
	Read(B);
	Write(B);

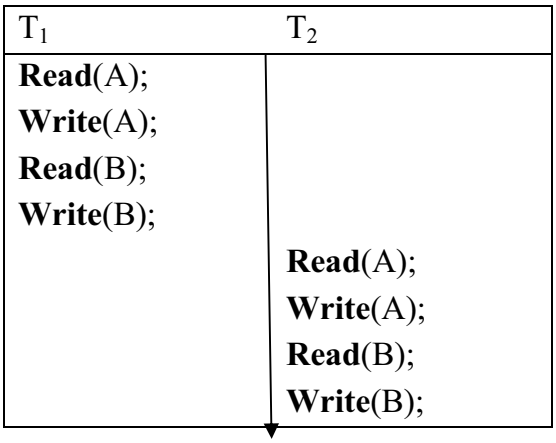
Thời lịch 5

Ta tiếp tục đổi chỗ các chỉ thị không xung đột như sau:

- Đổi chỗ chỉ thị **Read(B)** của T_1 với chỉ thị **Read(A)** của T_2
- Đổi chỗ chỉ thị **Write(B)** của T_1 với chỉ thị **Write(A)** của T_2
- Đổi chỗ chỉ thị **Write(B)** của T_1 với chỉ thị **Read(A)** của T_2

Kết quả cuối cùng của các bước đổi chỗ này là một thời lịch mới (thời lịch 6 –thời lịch tuần tự) tương đương với thời lịch ban đầu (thời lịch 3):

http://www.ebook.edu.vn



Thời lịch 6

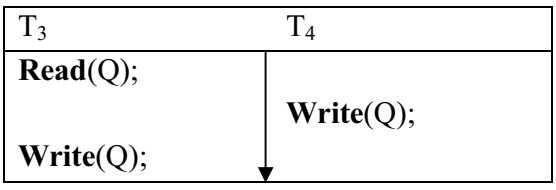
Sự tương đương này cho ta thấy: bất chấp trạng thái hệ thống ban đầu, thời lịch-3 sẽ sinh ra cùng trạng thái cuối như một thời lịch tuần tự nào đó.

Tương đương xung đột (conflict equivalent)*:Nếu một thời lịch S có thể biến đổi thành một thời lịch S' bởi một dãy các thao tác đổi chỗ các chỉ thị không xung đột, ta nói S và S' là tương đương xung đột.*

Ví dụ, trong các thời lịch đã được nêu ở trên, ta thấy thời lịch-1 tương đương xung đột với thời lịch-3.

Khả tuần tự xung đột (conflict serializable): M*ột thời lịch S là khả tuần tự xung đột (conflict serializable) nếu nó tương đương xung đột với một thời lịch tuần tự.*

Ví dụ, thời lịch 3 là khả tuần tự xung đột còn thời lịch 7 dưới đây không tương đương xung đột với một thời lịch tuần tự nào do vậy nó không là khả tuần tự xung đột:

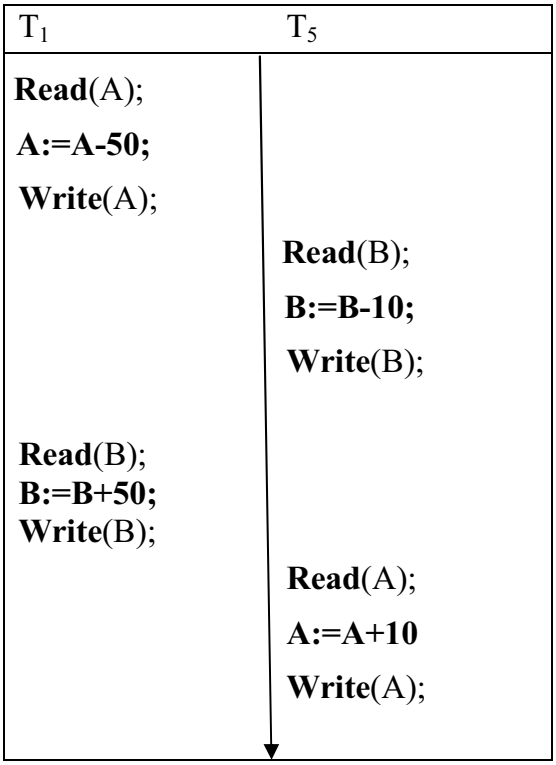


Thời lịch 7

Có thể có hai thời lịch sinh ra cùng kết quả, nhưng không tương đương xung đột. Ví dụ, giao dịch T_5 chuyển 10$ từ tài khoản B sang tài khoản A. Ta xét thời lịch 8 như dưới đây, thời lịch này không tương đương xung đột với thời lịch tuần tự < T_1, T_5 > do trong thời lịch 8 chỉ thị **Write(B)** của T_5 xung đột với chỉ thị **Read(B)** của T_1 như vậy ta không thể di chuyển tất cả các chỉ thị của T_1 về trước các chỉ thị của T_5 bởi việc hoán đổi liên tiếp các chỉ thị không xung đột. Tuy nhiên, các giá trị sau cùng của tài khoản A và B sau khi thực hiện thời lịch 8 hoặc sau khi thực hiện thời lịch tuần tự <T_1, T_5 > là như nhau A là

http://www.ebook.edu.vn

960 và B là 2040 tương ứng. Qua ví dụ này ta thấy cần thiết phải phân tích cả sự tính toán được thực hiện bởi các giao dịch mà không chỉ các hoạt động **Read** và **Write.** Tuy nhiên sự phân tích như vậy sẽ nặng nề và phải trả một giá tính toán cao hơn.



Schedule - 8

Tuần tự View (View Serializability)

Xét hai thời lịch S và S', trong đó cùng một tập hợp các giao dịch tham gia vào cả hai thời lịch. Các thời lịch S và S' được gọi là *tương đương view* nếu ba điều kiện sau được thỏa mãn:

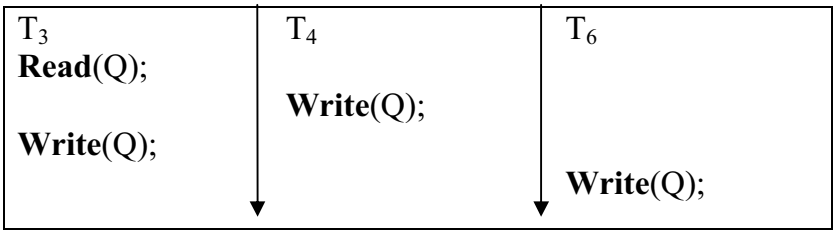
1. Đối với mỗi mục dữ liệu Q, nếu giao dịch T_i đọc giá trị khởi đầu của Q trong thời lịch S, thì giao dịch T_i phải cũng đọc giá trị khởi đầu của Q trong thời lịch S'.
2. Đối với mỗi mục dữ liệu Q, nếu giao dịch T_i thực hiện **Read(Q)** trong thời lịch S và giá trị đó được sản sinh ra bởi giao dịch T_j thì T_i cũng phải đọc giá trị của Q được sinh ra bởi giao dịch T_j trong S'.
3. Đối với mỗi mục dữ liệu Q, giao dịch thực hiện hoạt động **Write(Q)** sau cùng trong thời lịch S, phải thực hiện hoạt động **Write(Q)** sau cùng trong thời lịch S'.

Điều kiện 1 và 2 đảm bảo mỗi giao dịch đọc cùng các giá trị trong cả hai thời lịch và do vậy thực hiện cùng tính toán. Điều kiện 3 đi cặp với các điều kiện 1 và 2 đảm bảo cả hai thời lịch cho ra kết quả là trạng thái cuối cùng của hệ thống như nhau. Trong các ví dụ trước, thời lịch-1 là không tương đương view với thời lịch 2 do trong thời lịch-1, giá trị

http://www.ebook.edu.vn

của tài khoản A được đọc bởi giao dịch T_2 được sinh ra bởi T_1, trong khi điều này không xảy ra trong thời lịch-2. Thời lịch-1 tương đương view với thời lịch-3 vì các giá trị của các tài khoản A và B được đọc bởi T_2 được sinh ra bởi T_1 trong cả hai thời lịch.

Quan niệm tương đương view đưa đến quan niệm tuần tự view. *Ta nói thời lịch S là khả tuần tự view (view serializable) nếu nó tương đương view với một thời lịch tuần tự.* Ta xét thời lịch sau:



Thời lịch 9

Nó tương đương view với thời lịch tuần tự < T_3, T_4, T_6 > do chỉ thị **Read(Q)** đọc giá trị khởi đầu của Q trong cả hai thời lịch và T_6 thực hiện **Write** sau cùng trong cả hai thời lịch như vậy thời lịch 9 khả tuần tự view.

Mỗi thời lịch khả tuần tự xung đột là khả tuần tự view, nhưng có những thời lịch khả tuần tự view không khả tuần tự xung đột (ví dụ thời lịch 9).

Trong thời lịch 9 các giao dịch T_4 và T_6 thực hiện các hoạt động **Write(Q)** mà không thực hiện hoạt động **Read(Q)**, Các Write dạng này được gọi là các Write mờ (blind write). Các Write mờ xuất hiện trong bất kỳ thời lịch khả tuần tự view không khả tuần tự xung đột.

Tính khả phục hồi (Recoverability)

Ta đã nghiên cứu các thời lịch có thể chấp nhận dưới quan điểm sự nhất quán của CSDL với giả thiết không có giao dịch nào thất bại. Ta sẽ xét hiệu quả của thất bại giao dịch trong thực hiện tương tranh.

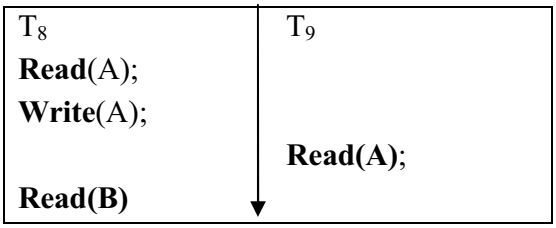
Nếu giao dịch T_i thất bại vì lý do nào đó, ta cần hủy bỏ hiệu quả của giao dịch này để đảm bảo tính nguyên tử của giao dịch. Trong hệ thống cho phép thực hiện tương tranh, cũng cần thiết đảm bảo rằng bất kỳ giao dịch nào phụ thuộc vào T_i cũng phải bị bỏ. Để thực hiện sự chắc chắn này, ta cần bố trí các hạn chế trên kiểu thời lịch được phép trong hệ thống.

Thời lịch khả phục hồi (Recoverable Schedule)

Xét thời lịch 10 trong đó T_9 là một giao dịch chỉ thực hiện một chỉ thị **Read(A)**. Giả sử hệ thống cho phép T_9 bàn giao ngay sau khi thực hiện chỉ thị **Read(A).** Như vậy T_9 bàn

http://www.ebook.edu.vn

giao trước T_8. Giả sử T_8 thất bại trước khi bàn giao, vì T_9 đã đọc giá trị của mục giữ liệu A được viết bởi T_8 , ta phải bỏ dở T_9 để đảm bảo tính nguyên tử giao dịch. Tuy nhiên T_9 đã được bàn giao và không thể bỏ dở được. Ta có tình huống trong đó không thể khôi phục đúng sau thất bại của T_8.



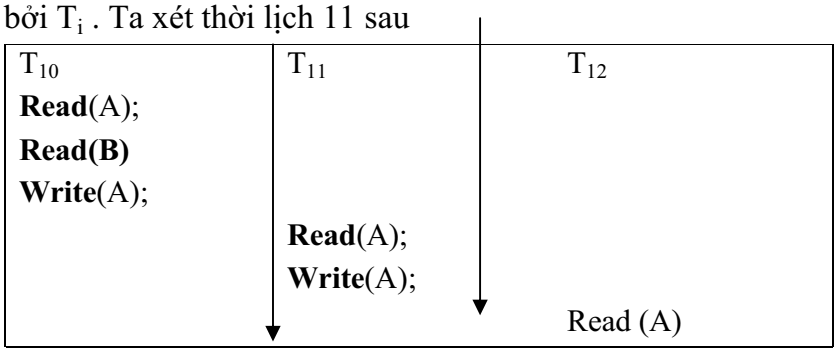
Thời lịch 10

Thời lịch 10 là một ví dụ về thời lịch không phục hồi được và không được phép. Hầu hết các hệ CSDL đòi hỏi tất cả các thời lịch phải phục hồi được.

Một thời lịch khả phục hồi là thời lịch trong đó, đối với mỗi cặp giao dịch T_i , T_j , nếu T_j đọc mục dữ liệu được viết bởi T_i thì hoạt động bàn giao của T_j phải xảy ra sau hoạt động bàn giao của T_i .

Thời lịch cascadeless (Cascadeless Schedule)

Ngay cả khi thời lịch là khả phục hồi, để phục hồi đúng sau thất bại của một giao dịch T_i ta phải cuộn lại một vài giao dịch. Tình huống như thế xảy ra khi các giao dịch đọc dữ liệu được viết bởi T_i . Ta xét thời lịch 11 sau



Thời lịch 11

Giao dịch T_{10} viết một giá trị được đọc bởi T_{11}. Giao dịch T_{12} đọc một giá trị được viết bởi T_{11}. Giả sử rằng tại điểm này T_{10} thất bại. T_{10} phải cuộn lại, do T_{11} phụ thuộc vào T_{10} nên T_{11} cũng phải cuộn lại và cũng như vậy với T_{12}. Hiện tượng trong đó một giao dịch thất bại kéo theo một dãy các giao dịch phải cuộn lại được gọi là sự cuộn lại hàng loạt (cascading rollback).

Cuộn lại hàng loạt dẫn đến việc huỷ bỏ một khối lượng công việc đáng kể. Phải hạn chế các thời lịch để việc cuộn lại hàng loạt không thể xảy ra. Các thời lịch như vậy được gọi là các thời lịch cascadeless.

http://www.ebook.edu.vn

Một thời lịch cascadeless là một thời lịch trong đó mỗi cặp giao dịch T_i , T_j nếu T_j đọc một mục dữ liệu được viết trước đó bởi T_i, hoạt động bàn giao của T_i phải xuất hiện trước hoạt động đọc của T_j . Một thời lịch cascadeless là khả phục hồi.

Thực thi cô lập (Implementation of Isolation)

Có nhiều sơ đồ điều khiển tương tranh có thể được sử dụng để đảm bảo các tính chất một thời lịch phải có (nhằm giữ CSDL ở trạng thái nhất quán, cho phép quản lý các giao dịch...), ngay cả khi nhiều giao dịch thực hiện tương tranh, chỉ các thời lịch có thể chấp nhận được sinh ra, bất kể hệ điều hành chia sẻ thời gian tài nguyên như thế nào giữa các giao dịch.

Ví dụ ta xét một sơ đồ điều khiển tương tranh sau: một giao dịch được cấp một khóa (lock) trên toàn bộ CSDL trước khi nó khởi động và tháo khóa khi nó đã bàn giao. Trong khi giao dịch này giữ khóa, không giao dịch nào khác được cấp khóa và như vậy phải chờ đến tận khi khóa được tháo. Trong cơ chế khóa, chỉ một giao dịch được thực hiện tại một thời điểm và như vậy chỉ thời lịch tuần tự được sinh ra. Sơ đồ điều khiển tương tranh này cho ra một hiệu năng tương tranh thấp. Ta nói nó cung cấp một bậc tương tranh thấp (poor degree of concurrency).

Mục đích của các sơ đồ điều khiển tương tranh là cung cấp một bậc cạnh tranh cao trong khi vẫn đảm bảo các thời lịch được sinh ra là khả tuần tự xung đột hoặc khả tuần tự view và cascadeless.

2. Thuật toán kiểm tra tính khả tuần tự

Thuật toán kiểm tra xung đột tương đương (Tính khả tuần tự xung đột)

Cho S là một thời lịch. Ta dùng một đồ thị định hướng, được gọi là đồ thị trình tự (precedence graph), biểu diễn từ S=(V, E). Trong đó:

V là tập các đỉnh và E là tập các cung.

Tập các đỉnh V bao gồm tất cả các giao dịch tham gia vào thời lịch.

Tập các cung E bao gồm tất cả các cung dạng $T_i \rightarrow T_j$ sao cho một trong các điều kiện sau được thoả mãn:

(i). T_i thực hiện **Write(Q)** trước T_j thực hiện **Read(Q).**

(ii). T_i thực hiện **Read(Q)** trước khi T_j thực hiện **Write(Q).**

(iii). T_i thực hiện **Write(Q)** trước khi T_j thực hiện **Write(Q).**

Nếu một cung $T_i \rightarrow T_j$ tồn tại trong đồ thị trình tự, thì trong bất kỳ thời lịch tuần tự S' nào tương đương với S, T_i phải xuất hiện trước T_j.

http://www.ebook.edu.vn

Ví dụ:

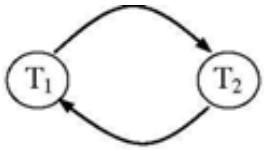
Thời lịch 1 được biểu diễn bằng đồ thị

vì tất cả các chỉ thị của T_1 được thực hiện trước chỉ thị đầu tiên của T_2.

Thời lịch 2 được biểu diễn bằng đồ thị

vì tất cả các chỉ thị của T_2 được thực hiện trước chỉ thị đầu tiên của T_1.

Thời lịch 4 được biểu diễn bằng đồ thị chứa các cung $T_1 \rightarrow T_2$ vì T_1 thực hiện **Read(A)** trước T_2 thực hiện **Write(A).** Nó cũng chứa cung $T_2 \rightarrow T_1$ vì T_2 thực hiện **Read(B)** trước khi T_1 thực hiện **Write(B):**



Nếu đồ thị trình tự đối với S có chu trình, khi đó thời lịch S không là khả tuần tự xung đột. *Như vậy bài toán kiểm tra tính khả tuần tự xung đột của thời lịch S đưa về bài toán kiểm tra chu trình trên đồ thị trình tự biểu diễn của S.*

Thuật toán kiểm tra khả tuần tự view

Không có thuật toán nào hiệu quả để kiểm tra tính khả tuần tự view của một thời lịch. Một cách tiếp cận là cải tiến thuật toán kiểm tra xung đột tương đương để kiểm tra tính khả tuần tự view.

Giả sử S là thời lịch gồm các giao dịch $\{T_1 , T_2 , ... , T_n\}$. T_b và T_f là hai giao dịch giả sao cho: T_b bắt nguồn lệnh **Write(Q)** đối với mỗi mục dữ liệu Q được truy xuất trong S, T_f bắt nguồn lệnh **Read(Q)** đối với mỗi mục dữ liệu Q được truy xuất trong S. Ta xây dựng thời lịch mới S' từ S bằng cách xen T_b ở bắt đầu của S và T_f ở cuối của S. Biểu diễn S' bằng đồ thị trình tự có gán nhãn trên cung được xây dựng dựa trên các quy tắc:

1. Gán nhãn 0 cho cung $T_i \rightarrow T_j$, nếu T_j đọc giá trị của mục dữ liệu Q được viết bởi T_i
2. Xóa tất cả các cung liên quan tới các giao dịch không dùng. Một giao dịch T_i được gọi là không dùng nếu không có con đường nào trong đồ thị trình tự dẫn từ T_i đến T_f .
3. Đối với mỗi mục dữ liệu Q sao cho T_j ***đọc giá trị của Q được viết (Write) bởi*** T_i và T_k thực hiện **Write(Q),** $T_k \neq T_b$ tiến hành các bước sau :
    a. Nếu $T_i = T_b$ và $T_j \neq T_f$, gán nhãn 0 cho cung $T_j \rightarrow T_k$ trong đồ thị trình tự gán

http://www.ebook.edu.vn

nhãn

b. Nếu $T_i \neq T_b$ và $T_j = T_f$ gán nhãn 0 cho cung $T_k \rightarrow T_i$ trong đồ thị trình tự gán nhãn

c. Nếu $T_i \neq T_b$ và $T_j \neq T_f$ gán nhãn p cho cả hai cung $T_k \rightarrow T_i$ và $T_j \rightarrow T_k$ trong đồ thị trình tự gán nhãn, trong đó p là một số nguyên duy nhất > 0 mà chưa được sử dụng trước đó để gán nhãn cung.

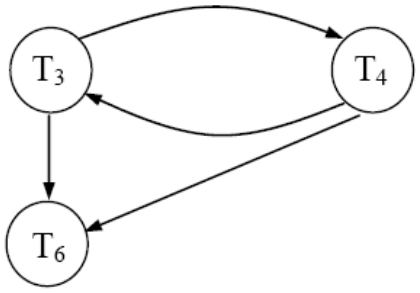
Quy tắc 3c phản ánh rằng nếu T_i viết mục dữ liệu được đọc bởi T_j thì một giao dịch T_k viết cùng mục dữ liệu này phải hoặc đi trước T_i hoặc đi sau T_j. Quy tắc 3a và 3b là trường hợp đặc biệt là kết quả của sự kiện T_b và T_f cần thiết là các giao dịch đầu tiên và cuối cùng tương ứng.

Ví dụ: xét thời lịch 9

T_3	T_4	T_6
Read(Q);		
	Write(Q);	
Write(Q);		
		Write(Q);

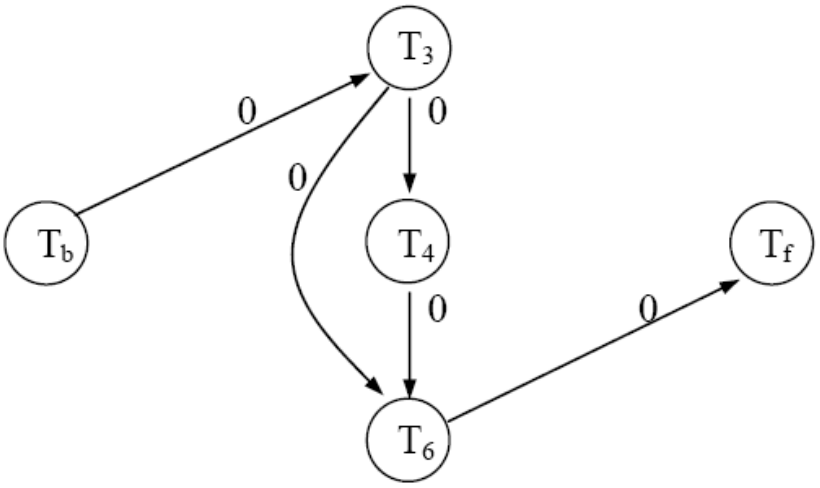
Thời lịch 9

Biểu diễn đồ thị của S như sau:



Và đồ thị trình tự có gán nhãn biểu diễn S’ như sau:

http://www.ebook.edu.vn



Nếu đồ thị trình tự gán nhãn không chứa chu trình, thời lịch tương ứng là khả tuần tự view, như vậy thời lịch 9 là khả tuần tự view. Tuy nhiên, nếu đồ thị S' chứa chu trình, điều kiện này không kéo theo thời lịch tương ứng không là khả tuần tự view. Bây giờ ta giả sử rằng có n cặp cung tách biệt, *đó là do ta đã áp dụng n lần quy tắc 3c trong sự xây dựng đồ thị trình tự.* Khi đó có 2^n đồ thị khác nhau, mỗi một đồ thị chứa đúng một cung trong mỗi cặp. Nếu một đồ thị nào đó trong các đồ thị này là phi chu trình, khi đó thời lịch tương ứng là khả tuần tự view. Thuật toán này đòi hỏi một phép kiểm thử vét cạn các đồ thị riêng biệt, và như vậy thuộc về lớp bài toán NP-đầy đủ.

3. Điều khiển tương tranh dựa trên khóa

Một phương pháp để đảm bảo tính khả tuần tự là yêu cầu việc truy xuất đến mục dữ liệu được tiến hành theo kiểu loại trừ tương hỗ; có nghĩa là trong khi một giao dịch đang truy xuất một mục dữ liệu, không một giao dịch nào khác có thể sửa đổi mục dữ liệu này. Phương pháp chung nhất được dùng để thực thi yêu cầu này là cho phép một giao dịch truy xuất một mục dữ liệu chỉ nếu nó đang giữ khóa trên mục dữ liệu đó.

Khóa (Lock)

Có nhiều phương thức khóa mục dữ liệu. Ta hạn chế việc nghiên cứu trên hai phương thức:

1. **Shared:** Nếu một giao dịch T_i nhận được một khóa ở phương thức shared (ký hiệu là S) trên mục Q, khi đó T_i có thể đọc, nhưng không được viết Q.
2. **Exclusive.** Nếu một giao dịch T_i nhận được một khóa ở phương thức Exclusive (ký hiệu là X), khi đó T_i có thể cả đọc lẫn viết Q.

Hàm tương thích

http://www.ebook.edu.vn

Mỗi giao dịch đòi hỏi một khóa ở một phương thức thích hợp trên mục dữ liệu Q, phụ thuộc vào kiểu hoạt động mà nó sẽ thực hiện trên Q. Giả sử một giao dịch T_i đòi hỏi một khóa phương thức A trên mục Q mà trên nó giao dịch T_j ($T_j \neq T_i$) hiện đang giữ một khóa phương thức B. Nếu giao dịch T_i có thể được cấp một khóa trên Q ngay, bất chấp sự hiện diện của khóa phương thức B, khi đó ta nói phương thức A tương thích với phương thức B. Hàm tương thích giữa hai phương thức khóa được biểu diễn bởi một ma trận. Quan hệ cho bởi ma trận **comp** sau:

	S	X
S	True	False
X	False	False

Comp(A, B) = True có nghĩa là các phương thức A và B tương thích.

Các thủ tục cấp và thu hồi khóa như sau:

- **lock-S(Q):** yêu cầu một khóa **shared** trên mục dữ liệu Q.
- **lock-X(Q):** yêu cầu một khóa **exclusive** trên mục dữ liệu Q.
- **unlock(Q):** thu hồi khóa trên mục dữ liệu Q

Để truy xuất một mục dữ liệu, giao dịch T_i đầu tiên phải khóa mục này. Nếu mục này đã bị khóa bởi một giao dịch khác ở phương thức không tương thích, bộ điều khiển tương tranh sẽ không cấp khóa cho đến tận khi tất cả các khóa không tương thích bị giữ bởi các giao dịch khác được tháo. Như vậy T_i phải chờ đến tận khi tất cả các khóa không tương thích bị giữ bởi các giao dịch khác được giải phóng.

Giao dịch T_i có thể tháo khóa một mục dữ liệu mà nó đã khóa trước đây. Một giao dịch cần thiết phải giữ một khóa trên một mục dữ liệu chừng nào mà nó còn truy xuất mục này. Hơn nữa, đối với một giao dịch việc tháo khóa ngay sau truy xuất cuối cùng đến mục dữ liệu không luôn luôn là điều mong muốn vì như vậy tính khả tuần tự có thể không được đảm bảo. Để minh hoạ cho tình huống này, ta xét ví dụ sau: A và B là hai tài khoản có thể được truy xuất bởi các giao dịch T_1 và T_2 . Giao dịch T_1 chuyển 50$ từ tài khoản B sang tài khoản A và được xác định như sau:

T_1 : **Lock-X(B);**
Read(B);
B:=B-50;
Write(B);
Unlock(B);
Lock-X(A);
Read(A);

http://www.ebook.edu.vn

A:=A+50;
Write(A);
Unlock(A);

Giao dịch T_2 hiển thị tổng số lượng tiền trong các tài khoản A và B (A + B) và được xác định như sau:

T_2 : **Lock-S(A);**
Read(A);
Unlock(A);
Lock-S(B);
Read(B);
Unlock(B);
Display(A+B);

Giả sử giá trị của tài khoản A và B tương ứng là 100$ và 200$. Nếu hai giao dịch này thực hiện tuần tự, hoặc theo thứ tự T_1, T_2 hoặc theo thứ tự T_2 , T_1 , và khi dó T_2 sẽ hiển thị giá trị 300$. Tuy nhiên nếu các giao dịch này thực hiện tương tranh, giả sử theo thời lịch schedule-1, trong trường hợp như vậy giao dịch T_2 sẽ hiển thị giá trị 250$, một kết quả không đúng vì giao dịch T_1 đã tháo khóa mục B quá sớm và T_2 đã tham khảo một trạng thái không nhất quán.

Thời lịch 12 cho thấy các hoạt động được thực hiện bởi các giao dịch cũng như các thời điểm khi các khóa được cấp bởi bộ quản trị điều khiển tương tranh. Giao dịch đưa ra một yêu cầu khóa không thể thực hiện hành động kế của mình đến tận khi khóa được cấp bởi bộ quản trị điều khiển tương tranh; do đó, khóa phải được cấp trong khoảng thời gian giữa hoạt động yêu cầu khóa và hành động sau của giao dịch. Sau này ta sẽ luôn giả thiết khóa được cấp cho giao dịch ngay trước hành động kế và như vậy ta có thể bỏ qua cột bộ quản trị điều khiển tương tranh trong bảng.

T_1	T_2	Bộ quản trị điều khiển tương tranh
Lock-X(B)		
		Grant-X(B,T_1)
Read(B)		
B:=B-50		
Write(B)		
Unlock(B)		
	Lock-S(A)	
		Grant-S(A,T_2)

http://www.ebook.edu.vn

	Unlock(A)	
	Lock-S(B)	
		Grant-S(B,T_2)
	Read(B)	
	Unlock(B)	
	Display(A+B)	
Lock-X(A)		
		Grant-X(A,T_1)
Read(A)		
A:=A+50		
Write(A)		
Unlock(A)		

Thời lịch 12

Bây giờ giả sử rằng tháo khóa bị làm trễ đến cuối giao dịch. Giao dịch T_3 tương ứng với T_1 với tháo khóa bị làm trễ được định nghĩa như sau:

T_3 : **Lock-X(B);**

Read(B);

B:=B-50;

Write(B);

Lock-X(A);

Read(A);

A:=A+50;

Write(A);

Unlock(B);

Unlock(A);

Giao dịch T_4 tương ứng với T_2 với tháo khóa bị làm trễ được xác định như sau:

T_4 : **Lock-S(A);**

Read(A);

Lock-S(B);

Read(B);

Display(A+B);

Unlock(A);

Unlock(B);

http://www.ebook.edu.vn

Các thời lịch có thể trên T_3 và T_4 không để cho T_4 hiển thị trạng thái không nhất quán.

Tuy nhiên, sử dụng khóa có thể dẫn đến một tình huống không mong đợi. Ta xét thời lịch 13 gồm một phần công việc trên T_3 và T_4 sau:

T_3	T_4
Lock-X(B)	
Read(B)	
B:=B-50	
Write(B)	
	Lock-S(A)
	Read(A)
	Lock-S(B)
Lock-X(A)	

Thời lịch 13

Do T_3 giữ một khóa phương thức Exclusive trên B, nên yêu cầu một khóa phương thức shared của T_4 trên B phải chờ đến khi T_3 tháo khóa. Cũng vậy, T_3 yêu cầu một khóa Exclusive trên A trong khi T_4 đang giữ một khóa shared trên nó và như vậy phải chờ. Ta gặp phải tình huống trong đó T_3 chờ đợi T_4 đồng thời T_4 chờ đợi T_3, một sự chờ đợi vòng tròn, và như vậy không giao dịch nào có thể tiến triển. Tình huống này được gọi là deadlock (khóa chết). Khi tình huống khóa chết xảy ra hệ thống buộc phải cuộn lại một trong các giao dịch. Mỗi khi một giao dịch bị cuộn lại, các mục dữ liệu bị khóa bởi giao dịch phải được tháo khóa và nó trở nên sẵn có cho giao dịch khác, như vậy các giao dịch này có thể tiếp tục được sự thực hiện của nó.

Nếu ta không sử dụng khóa hoặc tháo khóa mục dữ liệu ngay khi có thể sau đọc hoặc viết mục, ta có thể rơi vào trạng thái không nhất quán. Mặt khác, nếu ta không tháo khóa một mục dữ liệu trước khi yêu cầu một khóa trên một mục dữ liệu khác, deadlock có thể xảy ra. Có các phương pháp tránh deadlock trong một số tình huống, tuy nhiên nói chung deadlock là khó tránh khi sử dụng khóa nếu ta muốn tránh trạng thái không nhất quán. Deadlock được ưa thích hơn trạng thái không nhất quán vì chúng có thể điều khiển được bằng cách cuộn lại các giao dịch trong khi đó trạng thái không nhất quán có thể dẫn đến các vấn đề thực tế mà hệ CSDL không thể điều khiển.

Nghi thức khóa (locking protocol)

Xét $\{ T_0 , T_1 , ..., T_n \}$ một tập các giao dịch tham gia vào thời lịch S. Ta nói T_i đi trước T_j trong S, và được viết là $T_i \rightarrow T_j$, nếu tồn tại một mục dữ liệu Q sao cho T_i giữ

http://www.ebook.edu.vn

khóa phương thức A trên Q , T_j giữ khóa phương thức B trên Q muộn hơn và **comp(A,B)** = false. Nếu $T_i \rightarrow T_j$, thì T_i sẽ xuất hiện trước T_j trong bất kỳ thời lịch tuần tự nào.

Ta nói một thời lịch S là hợp lệ dưới một nghi thức khóa nếu S là một thời lịch tuân thủ các quy tắc của nghi thức khóa đó. Ta nói rằng một nghi thức khóa đảm bảo tính khả tuần tự xung đột nếu và chỉ nếu đối với tất cả các thời lịch hợp lệ, quan hệ → kết hợp là phi chu trình.

Cấp khóa

Khi một giao dịch T_i yêu cầu một khóa trên một mục dữ liệu Q ở phương thức M, khóa sẽ được cấp nếu các điều kiện sau được thỏa mãn:

(i) Không có giao dịch khác đang giữ một khóa trên Q ở phương thức xung đột với M.

(ii) Không có một giao dịch nào đang chờ được cấp một khóa trên M và đã đưa ra yêu cầu về khóa trước T_i.

Nghi thức khóa hai kỳ (Two-phase locking protocol)

Nghi thức khóa hai kỳ là một nghi thức đảm bảo tính khả tuần tự. Nghi thức này yêu cầu mỗi một giao dịch phát ra yêu cầu khóa và tháo khóa qua hai giai đoạn:

1. **Giai đoạn tăng trưởng (Growing phase):** một giao dịch có thể nhận được các khóa, nhưng nó không thể tháo bất kỳ khóa nào.
2. **Giai đoạn thu hẹp (Shrinking phase):** một giao dịch có thể tháo các khóa nhưng không thể nhận được một khóa mới nào.

Khởi đầu, một giao dịch ở giai đoạn tăng trưởng. Giao dịch được cấp các khóa cần thiết. Mỗi khi giao dịch tháo một khóa, nó đi vào giai đoạn thu hẹp và nó không thể phát ra bất kỳ một yêu cầu khóa nào nữa. Các giao dịch T_3 và T_4 là hai kỳ. Các giao dịch T_1 và T_2 không là hai kỳ. Nghi thức khóa hai kỳ đảm bảo tính khả tuần tự xung đột, nhưng không đảm bảo tránh được deadlock và việc cuộn lại hàng loạt. Cuộn lại hàng loạt có thể tránh được bởi nghi thức khóa hai kỳ nghiêm ngặt.

Nghi thức khóa hai kỳ nghiêm ngặt (**Strict Two-Phase Locking**) gồm hai luật:

1. Một giao dịch T muốn đọc (sửa) mục dữ liệu Q nó phải yêu cầu cấp khóa Shared (Exclusive) trên Q.
2. Tất cả các khóa do một giao dịch đang nắm giữ được giải phóng khi giao dịch được bàn giao.

Một sự tinh chế nghi thức khóa hai kỳ cơ sở dựa trên việc cho phép chuyển đổi khóa: nâng cấp một khóa shared sang exclusive và hạ cấp một khóa exclusive thành khóa shared. Chuyển đổi khóa không thể cho phép một cách tuỳ tiện, nâng cấp chỉ được phép

http://www.ebook.edu.vn

diễn ra trong giai đoạn tăng trưởng, còn hạ cấp chỉ được diễn ra trong giai đoạn thu hẹp. Một giao dịch thử nâng cấp một khóa trên một mục dữ liệu Q có thể phải chờ. Nghi thức khóa hai kỳ với chuyển đổi khóa cho phép chỉ sinh ra các thời lịch khả tuần tự xung đột. Nếu các khóa exclusive được giữ đến tận khi bàn giao, các thời lịch sẽ là cascadeless.

Các thủ tục liên quan được sử dụng trong nghi thức khóa 2 kỳ nghiêm ngặt:

- **lock-S(Q):** yêu cầu một khóa **shared** trên mục dữ liệu Q.
- **lock-X(Q):** yêu cầu một khóa **exclusive** trên mục dữ liệu Q.
- **unlock(Q):** thu hồi (giải phóng, tháo) khóa trên mục dữ liệu Q.
- **Upgrade(Q)**: chuyển đổi khóa từ phương thức shared (S) sang phương thức exclusive (X) trên mục dữ liệu Q, Upgrade chỉ được phép xảy ra trong giai đoạn tăng trưởng (giai đoạn xin khóa).
- **Downgrade(Q)**: chuyển đổi khóa từ phương thức exclusive sang phương thức shared trên mục dữ liệu Q, Downgrade chỉ được phép xảy ra trong giai đoạn thu hẹp (giai đoạn tháo khóa).

Ta xét một ví dụ: Các giao dịch T_8 và T_9 được nêu trong ví dụ chỉ được trình bày bởi các hoạt động ý nghĩa là **Read** và **Write.**

T_8 : **Read(A_1);**
Read(A_2);
...
Read(A_n);
Write(A_1).

T_9 : **Read(A_1);**
Read(A_2);
Display($A_1 + A_2$).

Nếu ta sử dụng nghi thức khóa hai kỳ, khi đó T_8 phải khóa A_1 ở phương thức exclusive. Bởi vậy, sự thực hiện tương tranh của hai giao dịch trở thành thực hiện tuần tự. Ta thấy rằng T_8 cần một khóa exclusive trên A_1 chỉ ở cuối sự thực hiện của nó, khi nó write(A_1). Như vậy, T_8 có thể khởi động khóa A_1 ở phương thức shared và đổi khóa này sang phương thức exclusive sau này. Như vậy ta có thể nhận được tính tương tranh cao hơn, vì như vậy T_8 và T_9 có thể truy xuất đến A_1 và A_2 đồng thời.

T_8	T_9
Lock-S(A_1)	
	Lock-S(A_2)

http://www.ebook.edu.vn

Lock-S(A_2)	
	Lock-S(A_2)
Lock-S(A_3)	
…	
	Unlock(A_1)
	Unlock(A_2)
UpGrade(A_1)	

Chú ý rằng một giao dịch thử cập nhật một khóa trên một mục dữ liệu Q có thể buộc phải chờ. Việc chờ bắt buộc này xảy ra khi Q đang bị khóa bởi giao dịch khác ở phương thức shared.

Nghi thức khóa hai kỳ với chuyển đổi khóa chỉ sinh ra các thời lịch khả tuần tự xung đột, các giao dịch có thể được tuần tự hoá bởi các điểm khóa của chúng. Hơn nữa, nếu các khóa exclusive được giữ đến tận khi kết thúc giao dịch, thời lịch sẽ là cascadeless.

Quản lý khóa

Trong hệ quản trị CSDL, các khóa của các giao dịch được quản lý bởi bộ phận quản lý khóa (lock manager) thông qua bảng khóa (lock table). Mỗi mục của bảng khóa tương ứng với một đối tượng (trang, mẫu tin, …) chứa các thông tin: số lượng giao dịch đang giữ khóa trên đối tượng này (số lượng giao dịch có thể > 1 nếu đối tượng được khóa theo phương thức Shared), phương thức khóa (Shared hay Exclusive), con trỏ đến hàng đợi yêu cầu khóa trên đối tượng đó.

Khi một giao dịch T_i cần một khóa trên một mục dữ liệu Q, nó gửi một yêu cầu cấp khóa đến bộ quản lý khóa, yêu cầu được xử lý như sau:

1. Nếu yêu cầu một khóa Shared, hàng đợi các yêu cầu của mục Q là rỗng và mục dữ liệu Q không bị khóa theo phương thức Exclusive thì bộ phận quản lý khóa cấp khóa cho giao dịch T_i và cập nhật lại thông tin trên bảng khóa.
2. Nếu yêu cầu một khóa Exclusive và không có giao dịch nào đang giữ khóa trên Q thì bộ phận quản lý khóa cấp khóa cho giao dịch T_i và cập nhật lại thông tin trên bảng khóa.
3. Ngược lại, thêm yêu cầu này vào hàng đợi khóa của mục dữ liệu Q và giao dịch T_i tạm thời ngưng.

Khi giao dịch được bàn giao hoặc bị bỏ dở, nó phải tháo tất cả các khóa nó đang nắm giữ. Khi một khóa được giải phóng, bộ quản lý khóa cập nhật lại bảng khóa và xem xét cấp phát khóa cho các yêu cầu khóa đang ở trong hàng đợi tương ứng.

http://www.ebook.edu.vn

Quản lý deadlock

Một hệ thống ở trạng thái deadlock nếu tồn tại một tập hợp các giao dịch sao cho mỗi giao dịch trong tập hợp đang chờ một giao dịch khác trong tập hợp. Chính xác hơn, tồn tại một tập các giao dịch { T_0 , T_1 , ..., T_n } sao cho T_0 đang chờ một mục dữ liệu được giữ bởi T_1 , T_1 đang chờ một mục dữ liệu đang bị chiếm bởi T_2 , ..., T_{n-1} đang chờ một mục dữ liệu được giữ bởi T_n và T_n đang chờ một mục T_0 đang chiếm. Không một giao dịch nào có thể tiến triển được trong tình huống như vậy. Một cách xử lý tình huống này là cuộn lại một vài giao dịch tham gia vào deadlock.

Có hai phương pháp chính giải quyết vấn đề deadlock: ngăn ngừa deadlock, phát hiện deadlock và khôi phục. Nghi thức ngăn ngừa deadlock đảm bảo rằng hệ thống sẽ không bao giờ đi vào trạng thái deadlock. Sơ đồ phát hiện deadlock và khôi phục (deadlock-detection and deadlock-recovery scheme) cho phép hệ thống đi vào trạng thái deadlock và sau đó cố gắng khôi phục. Cả hai phương pháp đều có thể dẫn đến việc cuộn lại giao dịch. Phòng ngừa deadlock thường được sử dụng nếu xác suất hệ thống đi vào deadlock cao, phát hiện và khôi phục hiệu quả hơn trong các trường hợp còn lại.

Phòng ngừa deadlock (Deadlock prevention)

Một cách phòng ngừa deadlock là sử dụng thứ tự ưu tiên và cuộn lại quá trình. Với thứ tự ưu tiên, một giao dịch T_2 yêu cầu một khóa bị giữ bởi giao dịch T_1 , khóa đã cấp cho T_1 có thể bị lấy lại và cấp cho T_2 , T_1 bị cuộn lại. Để điều khiển ưu tiên, ta gán một nhãn thời gian duy nhất cho mỗi giao dịch. Hệ thống sử dụng các nhãn thời gian này để quyết định một giao dịch phải chờ hay cuộn lại. Khóa vẫn được sử dụng để điều khiển tương tranh. Nếu một giao dịch bị cuộn lại, nó vẫn giữ nhãn thời gian cũ của nó khi tái khởi động. Hai sơ đồ phòng ngừa deadlock sử dụng nhãn thời gian khác nhau được đề nghị:

1. Sơ đồ **Wait-Die** dựa trên kỹ thuật không ưu tiên. Khi giao dịch T_i yêu cầu một mục dữ liệu bị chiếm bởi T_j , T_i được phép chờ chỉ nếu nó có nhãn thời gian nhỏ hơn của T_j nếu không T_i bị cuộn lại (die).
2. Sơ đồ **Wound-Wait** dựa trên kỹ thuật ưu tiên. Khi giao dịch T_i yêu cầu một mục dữ liệu hiện đang bị giữ bởi T_j , T_i được phép chờ chỉ nếu nó có nhãn thời gian lớn hơn của T_j , nếu không T_j bị cuộn lại (Wounded).

Một điều quan trọng là phải đảm bảo rằng, mỗi khi giao dịch bị cuộn lại, nó không bị "chết đói" (starvation) có nghĩa là nó sẽ không bị cuộn lại lần nữa và được phép tiến triển.

Cả hai sơ đồ **Wound-Wait** và **Wait-Die** đều tránh được sự chết đói: tại một thời điểm, có một giao dịch với nhãn thời gian nhỏ nhất. Giao dịch này không thể bị yêu cầu cuộn lại trong cả hai sơ đồ. Do nhãn thời gian luôn tăng và do các giao dịch không được gán nhãn

http://www.ebook.edu.vn

thời gian mới khi chúng bị cuộn lại, một giao dịch bị cuộn lại sẽ có nhãn thời gian nhỏ nhất (vào thời gian sau) và sẽ không bị cuộn lại lần nữa.

Tuy nhiên, có những khác nhau lớn trong cách thức hoạt động của hai sơ đồ:

- Trong sơ đồ **Wait-Die**, một giao dịch già hơn phải chờ một giao dịch trẻ hơn giải phóng mục dữ liệu. Như vậy, giao dịch già hơn có xu hướng bị chờ nhiều hơn. Ngược lại, trong sơ đồ **Wound-Wait,** một giao dịch già hơn không bao giờ phải chờ một giao dịch trẻ hơn.
- Trong sơ đồ **Wait-Die,** nếu một giao dịch T_i chết và bị cuộn lại vì nó đòi hỏi một mục dữ liệu bị giữ bởi giao dịch T_j , khi đó T_i có thể phải tái phát ra cùng dãy các yêu cầu khi nó khởi động lại. Nếu mục dữ liệu vẫn bị chiếm bởi T_j , T_i bị chết lần nữa. Như vậy, T_i có thể bị chết vài lần trước khi nhận được mục dữ liệu cần thiết. Trong sơ đồ **Wound-Wait,** giao dịch T_i bị thương và bị cuộn lại do T_j yêu cầu mục dữ liệu nó chiếm giữ. Khi T_i khởi động lại, và yêu cầu mục dữ liệu, bây giờ, đang bị T_j giữ, T_i chờ. Như vậy, có ít cuộn lại hơn trong sơ đồ **Wound-Wait.**

Một vấn đề nổi trội đối với cả hai sơ đồ là có những cuộn lại không cần thiết vẫn xảy ra.

Sơ đồ dựa trên timeout

Một cách tiếp cận khác để quản lý deadlock được dựa trên lock timeout. Trong cách tiếp cận này, một giao dịch đã yêu cầu một khóa phải chờ nhiều nhất một khoảng thời gian xác định. Nếu khóa không được cấp trong khoảng thời gian này, giao dịch được gọi là mãn kỳ (time out), giao dịch tự cuộn lại và khởi động lại. Nếu có một deadlock, một hoặc một vài giao dịch dính líu đến deadlock sẽ time out và cuộn lại, để các giao dịch khác tiến triển. Sơ đồ này nằm trung gian giữa phòng ngừa deadlock và phát hiện và khôi phục deadlock.

Sơ đồ timeout dễ thực thi và hoạt động tốt nếu giao dịch ngắn và nếu sự chờ đợi lâu là do deadlock. Tuy nhiên, khó quyết định được khoảng thời gian timeout. Sơ đồ này cũng có thể đưa đến sự chết đói.

Phát hiện deadlock và khôi phục

Nếu một hệ thống không dùng nghi thức phòng ngừa deadlock, khi đó sơ đồ phát hiện và khôi phục phải được sử dụng. Một giải thuật kiểm tra trạng thái của hệ thống được gọi theo một chu kỳ để xác định xem deadlock có xảy ra hay không. Nếu có, hệ thống phải khôi phục lại từ deadlock, muốn vậy hệ thống phải:

- Duy trì thông tin về sự cấp phát hiện hành các mục dữ liệu cho các giao dịch cũng như các yêu cầu mục dữ liệu chưa được giải quyết.
- Cung cấp một thuật toán sử dụng các thông tin này để xác định hệ thống đã đi vào

http://www.ebook.edu.vn

trạng thái deadlock chưa.

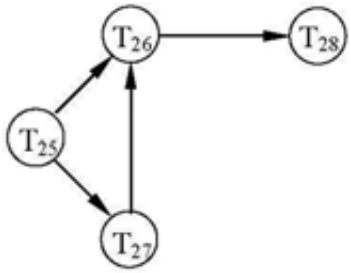
- Phục hồi từ deadlock khi phát hiện được deadlock đã xảy ra.

Phát hiện deadlock

Deadlock có thể mô tả chính xác bằng đồ thị định hướng được gọi là đồ thị chờ (wait for graph). Đồ thị này gồm một cặp G = < V, E >, trong đó V là tập các đỉnh và E là tập các cung. Tập các đỉnh gồm tất cả các giao dịch trong hệ thống. Mỗi phần tử của E là một cặp $T_i \rightarrow T_j$, nó chỉ ra rằng T_i chờ T_j giải phóng một mục dữ liệu nó cần.

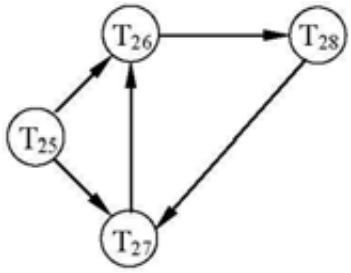
Khi giao dịch T_i yêu cầu một mục dữ liệu đang bị giữ bởi giao dịch T_j khi đó cung $T_i \rightarrow T_j$ được thêm vào đồ thị. Cạnh này bị xoá đi chỉ khi giao dịch T_j không còn giữ mục dữ liệu nào mà T_i cần.

Deadlock tồn tại trong hệ thống nếu và chỉ nếu đồ thị chờ chứa một chu trình. Mỗi giao dịch tham gia vào chu trình này được gọi là bị deadlock. Để phát hiện deadlock, hệ thống phải duy trì đồ thị chờ và gọi theo một chu kỳ thủ tục tìm kiếm chu trình. Ta xét ví dụ sau:



Đồ thị chờ (phi chu trình)

Do đồ thị không có chu trình nên hệ thống không ở trong trạng thái deadlock. Bây giờ, giả sử T_{28} yêu cầu một mục dữ liệu được giữ bởi T_{27} , cung $T_{28} \rightarrow T_{27}$ được thêm vào đồ thị, điều này dẫn đến tồn tại một chu trình $T_{26} \rightarrow T_{28} \rightarrow T_{27} \rightarrow T_{26}$ có nghĩa là hệ thống rơi vào tình trạng deadlock và T_{26} , T_{27} , T_{28} bị deadlock.



Vấn đề đặt ra là khi nào thì chạy thủ tục phát hiện? câu trả lời phụ thuộc hai yêu tố sau:

http://www.ebook.edu.vn

1. Deadlock thường xảy ra hay không?
2. Bao nhiêu giao dịch sẽ bị ảnh hưởng bởi deadlock?

Nếu deadlock thường xảy ra, việc chạy thủ tục phát hiện diễn ra thường xuyên hơn. Các mục dữ liệu được cấp cho các giao dịch bị deadlock sẽ không sẵn dùng cho các giao dịch khác đến khi deadlock bị phá vỡ. Hơn nữa, số chu trình trong đồ thị có thể tăng lên. Trong trường hợp xấu nhất, ta phải gọi thủ tục phát hiện mỗi khi có một yêu cầu cấp phát không được cấp ngay.

Khôi phục từ deadlock

Khi thuật toán phát hiện xác định được sự tồn tại của deadlock, hệ thống phải khôi phục từ deadlock. Giải pháp chung nhất là cuộn lại một vài giao dịch để phá vỡ deadlock. Ba việc cần phải làm là:

1. **Chọn nạn nhân**. Đã cho một tập các giao dịch bị deadlock, ta phải xác định giao dịch nào phải cuộn lại để phá vỡ deadlock. Ta sẽ cuộn lại các giao dịch sao cho giá phải trả là tối thiểu. Nhiều nhân tố xác định giá của cuộn lại:

   a. Giao dịch đã tính toán được bao lâu và bao lâu nữa.

   b. Giao dịch đã sử dụng bao nhiêu mục dữ liệu.

   c. Giao dịch cần bao nhiêu mục dữ liệu nữa để hoàn tất.

   d. Bao nhiêu giao dịch bị cuộn lại.

2. **Cuộn lại (Rollback).** Mỗi khi ta đã quyết định được giao dịch nào phải bị cuộn lại, ta phải xác định giao dịch này bị cuộn lại bao xa. Giải pháp đơn giản nhất là cuộn lại toàn bộ: bỏ dở giao dịch và bắt đầu lại nó. Tuy nhiên, sẽ là hiệu quả hơn nếu chỉ cuộn lại giao dịch đủ xa như cần thiết để phá vỡ deadlock. Nhưng phương pháp này đòi hỏi hệ thống phải duy trì các thông tin bổ sung về trạng thái của tất cả các giao dịch đang chạy.
3. **Sự chết đói (Starvation).** Trong một hệ thống trong đó việc chọn nạn nhân dựa trên các nhân tố giá, có thể xảy ra là một giao dịch luôn là nạn nhân của việc chọn này và kết quả là giao dịch này không bao giờ có thể hoàn thành. Tình huống này được gọi là sự chết đói. Phải đảm bảo việc chọn nạn nhân không đưa đến chết đói. Một giải pháp xem số lần bị cuộn lại của một giao dịch như một nhân tố về giá.

http://www.ebook.edu.vn

Chương 5
LẬP TRÌNH CƠ SỞ DỮ LIỆU

I. Lập trình với ADO.NET

1. Giới thiệu

1.1. DataSet và DataTable:

- DataSet là trung tâm của kiến trúc ADO.NET, mục tiêu là hỗ trợ hiệu quả thao tác trên dữ liệu từ nhiều nguồn, cũng như tương tác dữ liệu trong mô hình ứng dụng nhiều lớp (multiple tier).
- Có thể xem DataSet như là một cấu trúc dữ liệu để lưu trữ dữ liệu trong bộ nhớ chính. DataSet chứa một tập các đối tượng DataTable (có cấu trúc logic tương tự như một bảng trong CSDL), các ràng buộc trên chúng và cả mối quan hệ giữa các bảng này.
- Một đối tượng DataTable (hoặc một view của nó thuộc lớp DataView), có thể được kết buộc với các control như ComboBox, DataList, DataGrid,…
- Các lớp đối tượng DataSet, DataTable, DataView, DataColumn,… nằm trong namespace System.Data.

1.2. Data Provider :

- Trong namespace *System.Data* có 3 namespace tương ứng với 3 loại Data Provider: Data Provider for SQL Server (*System.Data.SqlClient*), Data Provider for ODBC (*System.Data.Odbc*) và Data Provider for OLE DB (*System.Data.OleDb*).
- Ở đây ta sử dụng hệ quản trị SQL Server, nên sẽ sử dụng trực tiếp Data Provider for SQL Server (tất nhiên ta cũng có thể thông qua Provider for ODBC hoặc OLE DB để thao tác với CSDL SQL Server).
- Các lớp đối tượng chính để kết nối và thao tác với CSDL là trong namespace System.Data.SqlClient là: SqlConnection, SqlCommand, SqlDataReader, SqlDataAdapter. (Tương tự đối với hai Provider còn lại).

1.2.1. SqlConnection :

Một đối tượng thuộc lớp này thể hiện một kết nối đến CSDL. Các thông số để kết nối được chỉ định trong Connection String

Ví dụ:

http://www.ebook.edu.vn

//dùng Window Authentication

ConnectionString = "Data Source = .; Initial Catalog = Northwind; Integrated Security = SSPI"

hoặc :

ConnectionString= "Data Source = .; Initial Catalog = Northwind; User ID = ws01; Password="

(dấu "." chỉ local host)

1.2.2. *SqlCommand* : Một đối tượng thuộc lớp này thể hiện một lệnh thực thi trên hệ quản trị CSDL. Có thể thiết lập thuộc tính CommandType của đối tượng Command để chỉ ra lệnh được khai báo ở dạng text hay là tên stored procedure.

1.2.3. *SqlDataReader :* là lớp đối tượng dùng để đọc kết quả truy vấn được từ CSDL. Có thể xem SqlDataReader là một RecordSet chỉ có thể đọc và đọc tuần tự một chiều.

1.2.4. *SqlDataAdapter :* một đối tượng Data Adapter có thể xem như một cầu nối giữa DataSet và CSDL, để chuyển dữ liệu từ CSDL vào DataSet và cập nhật những thay đổi trên DataSet trở lại vào CSDL.

1.3. Trình tự thao tác CSDL với ADO.Net :

- Tạo lập và thiết lập các thông số cho đối tượng Connection (nếu chưa thiết lập trước đó).
- Mở kết nối bằng phương thức Open của đối tượng Connection.
- Thực hiện các công việc đọc/ghi với CSDL vừa kết nối tới.
- Đóng kết nối.

Lưu ý:

- ✓ *Không phải luôn luôn mở và đóng kết nối mỗi khi thực hiện một lệnh, có thể mở kết nối một lần và thực hiện nhiều lệnh trước khi đóng nó.*
- ✓ *Luôn sử dụng try và catch để bắt các lỗi phát sinh từ CSDL khi thực hiện các lệnh mở kết nối hoặc thực thi lệnh trong đối tượng command, nếu có lỗi hiển thị thông điệp lỗi dễ hiểu cho người sử dụng (NSD).*

2. Đọc dữ liệu

Trong phần này chỉ mô tả một số điểm cần lưu ý, sinh viên tự tìm hiểu cách thực hiện cụ thể trong các ví dụ và ebook được cung cấp hoặc MSDN.

2.1. Data Reader

DataReader là cách tốt (tiết kiệm tài nguyên - bộ nhớ) để đọc dữ liệu trong trường hợp chỉ cần lấy dữ liệu để hiển thị, không cần thao tác phức tạp hay thao tác trên nhiều tập

http://www.ebook.edu.vn

dữ liệu (không cần đến datatable hay dataset).

2.2. DataAdapter

- Như đã nói ở trên, ta có thể sử dụng DataAdapter như là một cầu nối để lấy dữ liệu từ CSDL vào Dataset. Ta có thể định nghĩa hoặc không định nghĩa trước bảng (đối tượng DataTable) và cấu trúc của bảng sẽ nhận dữ liệu. Nếu lệnh trả về n tập dữ liệu, n bảng tương ứng sẽ được tạo ra trong dataset để chứa các tập dữ liệu này.
- Để lấy dữ liệu, ta sử dụng phương thức Fill của đối tượng DataAdapter. Phương thức này có nhiều hàm quá tải (overload), hỗ trợ nhiều cách truyền tham số (DataSet, DataTable, DataSet và tên DataTable,…).
- Phương thức Fill có thể được gọi mà không cần mở kết nối trước, trong trường hợp này, kết nối tương ứng sẽ được mở và đóng lại ngay sau khi thực hiện xong việc lấy dữ liệu.

  Lưu ý: *Trong trường hợp ta muốn dữ liệu đưa vào đối tượng DataTable thỏa ràng buộc khóa chính (tự động loại bỏ dữ liệu trùng trên khóa chính nếu có), ta có thể thiết lập thuộc tính MissingSchemaAction của đối tượng DataAdapter là AddWithKey. Tuy nhiên, đặt lựa chọn này sẽ làm chậm đáng kể quá trình đọc dữ liệu, thay vào đó, nếu có thể ta nên định nghĩa trước cấu trúc và khóa chính của đối tượng DataTable này, rồi đọc bình thường, tác dụng cũng sẽ tương tự.*

2.3. ExecuteScalar :

Trong trường hợp câu truy vấn chỉ trả về một giá trị, ta sử dụng phương thức ExecuteScalar của đối tượng command để thực thi truy vấn và nhận giá trị trả về.

Lưu ý: *Trong trường hợp ta gọi thực thi một thủ tục (với CommandType là stored procedure), giá trị của các tham số output mà ta khai báo trong Parameters của command sẽ được cập nhật tương ứng.*

3. Ghi dữ liệu

3.1. ExecuteNonQuery

Cách thông dụng nhất để ghi dữ liệu là đưa trực tiếp các lệnh cập nhật dữ liệu (Insert, Update, Delete, lệnh tạo các đối tượng trong CSDL) (và hầu hết các lệnh T-SQL khác), hoặc tên của thủ tục thường trú thực hiện các công việc này vào một đối tượng Command, sau đó gọi phương thức ExecuteNonQuery của đối tượng Command để thực hiện.

3.2. DataAdapter

- DataAdapter thực hiện cập nhật CSDL theo cách ánh xạ những thay đổi trên

http://www.ebook.edu.vn

DataSet vào CSDL (thành các thao tác insert, update, delete tương ứng).

- Cơ sở của việc ánh xạ này là việc quản lý tình trạng của các dòng (DataRow) trong DataTable, thể hiện qua thuộc tính RowState. Khi một dòng trong DataTable được thêm, xóa, cập nhật, nó sẽ có trạng thái tương ứng là Inserted, Deleted, và Updated, sau khi phương thức AcceptChanges được gọi, nó trở lại trạng thái UnChanged.
- Để thực hiện cập nhật ta gán InsertCommand, DeleteCommand và UpdateCommand cho đối tượng DataAdapter, sau đó gọi phương thức Update của nó. Khi đó, tất cả những thay đổi trên DataSet sẽ được ánh xạ về CSDL theo cách mà ta chỉ định trong các Commands (xem thêm ví dụ).
- Phương thức Update của DataAdapter cũng có nhiều hàm quá tải, cho phép ta lựa chọn Update một dòng, một table hay toàn bộ DataSet.

II. Thiết kế chức năng đọc/ ghi dữ liệu

1. Thiết kế chức năng ghi dữ liệu:

- Giao diện đáp ứng yêu cầu chung: dễ sử dụng (dễ hiểu, gợi nhớ, không "bẫy" người sử dụng), có tính thẩm mỹ, tính tiện dụng (cho phép người sử dụng thao tác nhanh: sắp xếp các mục hợp lý, hỗ trợ di chuyển bằng phím tab, hỗ trợ phím tắt,…).
- Kiểm tra chặt chẽ các ràng buộc toàn vẹn, đảm bảo thao tác thêm/cập nhật sau khi thực hiện xong không gây ra mâu thuẫn trong CSDL.
- Cung cấp cách thức nhập liệu phù hợp nhất với nghiệp vụ thực tế.

  Ví dụ: nếu thực tế NSD nhập liệu cho một tập đối tượng cùng lúc, mỗi đối tượng có ít thuộc tính và xử lý đơn giản thì nên nhập liệu bằng lưới (grid). Nếu đối tượng có nhiều thuộc tính hoặc xử lý phức tạp thì có thể nhập riêng từng đối tượng, nhưng nên hiển thị song song một lưới chứa danh sách các đối tượng đã nhập để NSD có thể kiểm tra lại khi cần.
- Lựa chọn cách xử lý để giảm thiểu thời gian làm việc của NSD (tất nhiên vẫn phải đảm bảo tính an toàn và đúng đắn): thời điểm kiểm tra ràng buộc toàn vẹn (xem mục 2), ghi nhận dữ liệu một lần hay sau mỗi lần NSD nhập xong một đối tượng, thời điểm mở và đóng kết nối với CSDL,…

2. Kiểm tra ràng buộc toàn vẹn

Khi xây dựng chức năng nhập liệu (cũng như cập nhật dữ liệu), phải đảm bảo rằng các ràng buộc toàn vẹn không bị vi phạm. Tuy nhiên, cần phải lưu ý cân nhắc xem kiểm tra

http://www.ebook.edu.vn

ràng buộc toàn vẹn ở mức nào, ở thời điểm nào, và dưới hình thức nào là hợp lý.

Một số nguyên tắc:

- Thiết kế giao diện sao cho có thể hạn chế lỗi của người sử dụng, ví dụ: sử dụng ComboBox, Check box,… để đảm bảo ràng buộc tham chiếu và ràng buộc miền giá trị rời rạc. Tuy nhiên, cũng cần cân nhắc kỹ vì giao diện quá "cứng", quá nghiêm ngặt sẽ cản trở và làm chậm thao tác của NSD.
- Trong các trường hợp có thể, cố gắng xử lý kiểm tra ràng buộc toàn vẹn ở mức trên (tầng giao diện, kế đến là tầng nghiệp vụ), phản hồi ngay và rõ ràng cho NSD nếu có lỗi sai. Nếu để chương trình đẩy dữ liệu xuống CSDL, sau khi nhận báo lỗi từ CSDL mới phản hồi cho NSD thì sẽ mất nhiều thời gian.
  - ✓ Các trường hợp thông thường có thể kiểm tra ràng buộc toàn vẹn ở tầng giao diện hoặc nghiệp vụ: ràng buộc đơn giản như miền giá trị, liên thuộc tính trên một quan hệ,…, ràng buộc phức tạp hơn (liên bộ, liên thuộc tính) nhưng các dữ liệu liên quan cần thiết để kiểm tra nó đã được chương trình đọc sẵn trước đó.
  - ✓ Nếu việc kiểm tra ràng buộc cần các dữ liệu liên quan khác chưa được chương trình đọc sẵn trước đó, kiểm tra ở CSDL trong đa số trường hợp sẽ hiệu quả hơn là đọc các dữ liệu đó lên để kiểm tra ở tầng trên.
- Nếu các ràng buộc của ứng dụng có các tham số đặt trong bảng tham số ở CSDL hoặc trong tập tin, nên đọc các tham số này lên một lần và sử dụng lại cho các lần nhập liệu, thay vì phải đọc lại từ CSDL hay tập tin mỗi khi nhập một đối tượng liên quan.
- Nếu các đối tượng trong CSDL được quản lý bằng mã, chương trình nên có cơ chế tự động phát sinh các mã này, để tránh gây ra các vi phạm trên ràng buộc khóa chính (Trừ những trường hợp nghiệp vụ thực tế đòi hỏi mã cho NSD ghi).

3. Thiết kế chức năng đọc dữ liệu:

Có thể chia thành hai dạng chức năng đọc dữ liệu chính: đọc dữ liệu lên Form và đọc dữ liệu lên báo biểu (Report). Dữ liệu đọc lên Form có thể được thay đổi và cập nhật xuống CSDL, dữ liệu đọc lên báo biểu chỉ nhằm mục đích hiển thị thông tin.

Khi đọc dữ liệu lên Form, ta sử dụng các đối tượng dữ liệu của ngôn ngữ lập trình (DataSet, DataTable, DataReader…) để lưu trữ tạm dữ liệu trong quá trình hiển thị. Ta có thể sao chép dữ liệu từ các đối tượng dữ liệu vào các đối tượng hiển thị như Textbox, DataGridView, Combobox,… trên form, hoặc kết buộc trực tiếp các đối tượng dữ liệu với đối tượng hiển thị. Với cách thứ hai, khi dữ liệu được cập nhật: khi người sử dụng thay đổi dữ liệu trên các đối tượng hiển thị, dữ liệu trong các đối tượng dữ liệu cũng sẽ thay

http://www.ebook.edu.vn

đổi theo.

Khi đọc dữ liệu lên report, ngoại trừ những trường hợp phức tạp mà dữ liệu được lấy từ nhiều nguồn: từ CSDL, từ file, từ form khác,…, thông thường ta đưa dữ liệu trực tiếp lên report mà không thông qua các đối tượng dữ liệu để giảm bớt một khoản chi phí trung gian. Các môi trường thiết kế report (Crystal report,…) thường hỗ trợ riêng phương thức đọc dữ liệu, không phụ thuộc vào các cách thức đọc dữ liệu của môi trường lập trình, dù rằng trong một số trường hợp vẫn hỗ trợ tích hợp (như Crystal report for .NET, trong đó dataset của .NET được Crystal report xem như một nguồn dữ liệu).

Tuy nhiên, cho dù đọc dữ liệu lên form hay lên report, người thiết kế vẫn phải chú ý đến những nguyên tắc cơ bản sau:

- Hạn chế đọc dữ liệu nhiều lần từ CSDL: nếu có thể, đọc các dữ liệu cần thiết một lần thay vì đọc thành nhiều lần từ CSDL để giảm chi phí thiết lập kết nối với dữ liệu. Hơn nữa, ở phía CSDL, việc đọc 1 lần n dòng dữ liệu sẽ nhanh hơn đọc k lần, mỗi lần n/k dòng.
    - ✓ Ví dụ: trên form có một lưới liệt kê danh sách các lớp trong trường theo từng khối. Mỗi lần NSD chọn lại khối trong một combobox, danh sách các lớp trong lưới được lọc lại theo khối đó. Vậy ta nên đọc một lần tất cả các lớp trong trường từ CSDL vào một đối tượng dữ liệu (DataTable chẳng hạn), sau đó tùy yêu cầu mà hiển thị phần dữ liệu phù hợp, hay mỗi lần NSD chọn khối ta lại đọc lại từ CSDL? Câu trả lời cho tình huống này và những tình huống tương tự là: nếu tất cả dữ liệu không quá lớn (vài trăm dòng trở xuống), ta nên đọc tất cả lên ứng dụng một lần.
- Giảm thiểu lượng dữ liệu chuyển từ CSDL lên ứng dụng: đọc đúng dữ liệu cần thiết, không đọc thừa.
- Dữ liệu phải được hiển thị theo định dạng thân thiện với người sử dụng. Trong CSDL, để tối ưu hoá lưu trữ và truy xuất, dữ liệu có thể ở dạng mà người sử dụng không hiểu hoặc không cần (ví dụ những mã đối tượng được phát sinh thêm để phục vụ việc lưu trữ, định dạng ngày giờ, …). Người sử dụng không cần biết dữ liệu bên dưới được lưu trữ như thế nào, cấu trúc ra sao, họ chỉ cần thấy được những thông tin được hiển thị và sắp xếp theo cách quen thuộc và tiện lợi nhất cho nghiệp vụ của họ.

4. Lưu ý chung:

- Nếu xử lý có thể mất nhiều thời gian, nên có phản hồi để NSD biết rằng chương trình vẫn đang làm việc (progress bar, waiting message,…).

http://www.ebook.edu.vn

- Nên sử dụng thủ tục thường trú thay vì viết các lệnh SQL trực tiếp trong mã nguồn chương trình, vì các thủ tục đã được biên dịch trước nên thực hiện nhanh hơn.
- Dùng try…catch để bắt các ngoại lệ (exception) có thể xảy ra, nhất là khi thực hiện các thao tác đóng/mở kết nối và đọc/ghi trên CSDL và thông báo lỗi theo cách mà NSD có thể hiểu được.
- Thiết kế chương trình theo mô hình 3 lớp, thiết kế các module nhỏ gọn, rõ ràng để dễ kiểm tra, bảo trì, và tăng khả năng tái sử dụng.

III. Tạo báo biểu với Crystal Report

1. Giới thiệu

Crystal Report là một phần mềm hỗ trợ lập báo biểu từ đơn giản đến phức tạp. Hiện nay ngoài các phiên bản Crystal Report riêng (Standard, Professional, Developer, Advanced), còn có một phiên bản đặc biệt tích hợp với Visual Studio.NET.

Ngoài việc cung cấp môi trường để thiết kế báo biểu, Crystal Report 9 phiên bản Developer và Advanced có hỗ trợ Report Creation API, cho phép người lập trình ứng dụng có thể tạo lập/ thay đổi cấu trúc/ nội dung report lúc runtime.

Phiên bản Crystal Report tích hợp với .NET có hỗ trợ Run Time Object Model, cho phép thực hiện một số thay đổi trên report khi chương trình thực hiện (truyền tham số, thay đổi các thông số để đăng nhập database, thay đổi kích thước và vị trí các đối tượng), ta cũng có thể gán các đối tượng dữ liệu của .NET (ví dụ DataTable) làm data source cho report. Cửa sổ thiết kế report được tích hợp vào môi trường phát triển ứng dụng .NET.

Crystal Report for .NET không hoàn toàn là một phần của Crystal Report (xét phiên bản 9), nó có một số tính năng hỗ trợ riêng cho .NET, đồng thời Crystal Report 9 cũng có một số tính năng riêng mà Crystal Report for .NET không hỗ trợ. Cửa sổ thiết kế report của Crystal Report 9 cũng đầy đủ và dễ dùng hơn. Ta có thể thiết kế một report bằng Crytal Report 9, sau đó đưa vào ứng dụng .NET.

2. Xây dựng một report cơ bản

2.1. *Các phần (session) của report*:

Một báo biểu (report) gồm có những phần chính như sau :

- Report header: phần thông tin đầu tiên của báo biểu. Một báo biểu thường sẽ gồm nhiều trang, report header là phần xuất hiện chỉ một lần ở trang đầu tiên của toàn báo biểu. Ví dụ như báo cáo có tiêu đề **"**Báo cáo thu chi tháng 12 năm 2008**"** thì tiêu đề sẽ được đặt trong phần report header
- Page header: phần hiển thị thông tin xuất hiện ở đầu mỗi trang của báo biểu. Ví

http://www.ebook.edu.vn

dụ như một báo cáo dạng bảng có nhiều cột như: STT, Tên, Địa chỉ, SĐT… thì đầu mỗi trang cần lặp lại những tên cột để người đọc nhận biết dễ dàng ý nghĩa mỗi cột. Khi đó các tiêu đề cột được đặt vào phần page header.

- Details: phần hiển thị thông tin chi tiết của báo biểu. Một báo biểu thường bao gồm nhiều mục với vai trò như nhau tương ứng với các mẩu tin (record) của database mà báo biểu sử dụng. Phần details sẽ liệt kê những mẩu tin đó. Ví dụ báo cáo thu chi thì các mục thu chi sẽ được liệt kê trong phần details.
- Báo biểu footer: phần hiển thị thông tin xuất hiện chỉ một lần ở cuối báo biểu. Ví dụ: các thông tin như tổng số (grand total), người lập báo cáo là ai, tại đâu, vào ngày nào, sẽ được đặt ở báo biểu footer.
- Page footer: phần thông tin xuất hiện cuối mỗi trang. Ví dụ số trang được đặt ở page footer.

2.2. *Các loại đối tượng trong report*

Các đối tượng trong báo biểu được quản lý qua cửa sổ Field explorer (View → Field Explorer). Các đối tượng này có thể được chọn để đưa vào hiển thị trong báo biểu. Gồm có:

- Database fields: Các trường thuộc dạng CSDL (có thể là table, stored procedure, SQL command). Thông thường các trường trong mục này sẽ được hiển thị trong phần detail của báo biểu.
- Formula fields: các trường tạo thành từ việc thiết lập các công thức. Ta có thể tạo mới một đối tượng formula bằng cách sử dụng Formula Editor hoặc Formula Expert. Formula được viết bằng cú pháp Crystal hoặc Basic. Có hai loại:
  - ✓ Formula Field : Là các trường mà dữ liệu có được nhờ tính toán trên các trường khác.

    Ví dụ: Khi làm một report báo cáo hóa đơn bán hàng, giả sử CSDL chỉ lưu trữ giá và số lượng của mặt hàng mua trong hóa đơn mà không lưu trữ thành tiền, khi đó ta có thể tạo một Formula field "Thành tiền" được tính bằng công thức:

    Thành tiền = Giá * Số lượng.

    Khi đó ta có thể tạo report với cột thành tiền (mặc dù không được lưu trong database).
  - ✓ Selection Formula: Là các công thức để chọn dữ liệu, gồm có chọn bộ (tương tự các điều kiện chọn trong mệnh đề where) hay chọn nhóm (tương tự điều kiện chọn trong mệnh đề having). Selection formula có thể được xây dựng trong cửa

http://www.ebook.edu.vn

sổ Formula Editor/ Formula Expert hoặc Select Expert.

- SQL Expression field: SQL Expression cũng là trường mà dữ liệu của nó được tính toán từ những trường khác (ví dụ count, sum,… hay một công thức tính toán bất kỳ). Tuy nhiên, khác với Formula field, SQL Exppression được gửi về xử lý ở CSDL (và do đó phải được viết bằng cú pháp SQL), kết quả được đưa trả về báo biểu qua SQL Expression field.
- Parameter fields: các trường tham số cho report. Đây có thể là một tham số ta tự khai báo, hoặc Crystal Report sẽ tự động thêm vào khi ta đưa một thủ tục thường trú có tham số vào trong database field. Lưu ý, khi chạy báo biểu trong Crystal, những trường tham số sẽ được hỏi giá trị, ta cần nhập vào ngay trong Crystal để hiển thị tạm thời.
- Group Name Field : Các thuộc dùng để gom nhóm dữ liệu trong report.

  Ví dụ : Ta muốn hiển thị danh sách học sinh theo từng lớp, khi đó thuộc tính Lớp sẽ là một đối tượng trong Group Name Field (Thêm một nhóm bằng cách chọn Insert Group hoặc chọn Group Expert từ menu database).
- Running Total Field : trường chứa giá trị tổng hợp (aggregate) : max, min, sum, count,…
- Special fields: các trường đặt biệt có sẵn của Crystal như số trang, ngày hiện tại… Thông thường những trường này sẽ được hiển thị trong những phần header, footer.

Ngoài ra, ta có thể đưa vào report những kiểu đối tượng khác như text(label), hình ảnh, đường kẻ, biểu đồ,…

2.3. *Xây dựng report:*

a. Thiết kế report : xác định các thông tin cần hiển thị, cách bố trí, tổ chức thông tin.

Một số tiêu chí khi thiết kế report :

- Đáp ứng mục tiêu nghiệp vụ, phù hợp với người sử dụng (thông tin kết xuất phải là thông tin mà người sử dụng có thể hiểu được, khớp với nghiệp vụ thực tế).
- Số lượng vừa đủ, sắp xếp, gom nhóm hợp lý, tránh làm NSD bị rối mắt vì báo biểu dày đặc dữ liệu.
- Trình bày dữ liệu đúng vị trí.
- Trình bày dữ liệu đúng lúc cần (ví dụ hiển thị lại tên các cột dữ liệu khi sang trang mới).

http://www.ebook.edu.vn

b. Xây dựng report theo bản thiết kế: Xem hướng dẫn chi tiết trong ebook và tài liệu kỹ thuật của Crystal Report.

Một số lưu ý khi xây dựng report :

- Nếu dữ liệu được hiển thị trong báo biểu sẽ thay đổi trong các lần báo biểu được hiển thị, bỏ lựa chọn Save Data with Report (File → Report Options)
- Khi sử dụng Formula, các tính toán, chọn dòng/nhóm sẽ được thực hiện bởi Crystal Report sau khi đã đọc toàn bộ dữ liệu liên quan từ CSDL lên. Điều này có thể làm nặng đường truyền và tăng chi phí đọc ghi nếu dữ liệu liên quan lớn. Ngoài ra, một hệ quản trị CSDL luôn có cơ chế điều chỉnh để cải thiện tốc độ xử lý trong trường hợp dữ liệu lớn, vì vậy, việc thực hiện xử lý chọn trên dòng/nhóm hay các tính toán mà dữ liệu kết quả nhỏ hơn dữ liệu trung gian (count/sum, max,…) được thực hiện ở hệ quản trị CSDL sẽ tối ưu hơn. Do đó, ta nên chuyển các công thức thành SQL Command/Expression hay thủ tục thường trú bất cứ khi nào có thể.
- Nên tận dụng stored prcedure vì các ích lợi của nó so với việc thực thi trực tiếp các câu lệnh SQL (hỗ trợ bảo mật, được biên dịch trước, tái sử dụng và dễ bảo trì).

2.4. *Xem trước report:*

Ta có thể xem trước report bằng cách nhấn F5 (Refresh Report Data) và chuyển qua tab preview.

3. Hiển thị report trong một ứng dụng .NET

3.1. *Crystal Report Viewer :*

Để hiển thị Crystal report trên .NET Windows form, ta sử dụng control CrystalReportViewer (có thể kéo thả control này từ toolbox vào form).

3.2. *Kết buộc báo biểu vào Report Viewer :*

Giả sử ta đã sử dụng một control CrystalReportViewer tên là rptViewer. Ta có thể sử dụng một trong những cách sau để đưa một Crystal report vào ứng dụng và kết buộc vào viewer để hiển thị :

a. Kết buộc dạng "Untyped report": Giả sử ta có sẵn một report tên MyReport rên đĩa C.

- ✓ Kết buộc bằng tên: Gán tên và đường dẫn đến báo biểu cho thuộc tính ReportSource của rptviewer.

http://www.ebook.edu.vn

rptViewer.ReportSource = "C:\\ MyReport.rpt"

✓ Kết buộc report object :

- Project → Add reference. Chọn CrystalDecisions.CrystalReports.Engine
- Khai báo đối tượng thuộc lớp ReportDocument, tải báo biểu vào đối tượng này và kết buộc vào rptViewer

```
(using CrystalDecisions.CrystalReports.Engine;)
                    ReportDocument oRpt=null;
oRpt = new ReportDocument();
oRpt.Load("c:\\MyReport.rpt");
rptViewer.ReportSource = oRpt;
```

b. Kết buộc dạng "strongly-typed report":

- Nếu tạo báo biểu ngay trong .NET : Project → Add New Item.
- Nếu đã có sẵn report: Project →Add existing Item. Tìm và chọn report muốn đưa vào.

Với cách này, trong project sẽ xuất hiện một lớp đối tượng mới tương ứng với báo biểu mới thêm vào (lớp đối tượng này kế thừa từ lớp ReportClass, và ReportClass kế thừa ReportDocument). Một đối tượng thuộc lớp này, ngoài các thuộc tính và phương thức kế thừa từ lớp ReportDocument, còn có một số thuộc tính thể hiện các thông tin riêng của từng report: các session, tham số,… Tuy nhiên các thuộc tính này phần lớn là read only. Để thay đổi một số thông tin trên báo biểu (ví dụ truyền tham số), ta vẫn chủ yếu sử dụng các phương thức và thuộc tính của lớp ReportDocument.

Giả sử ta đã thêm một strongly-typed report và project và có được lớp rptMyReport tương ứng, ta kết buộc report vào viewer như sau:

```
rptMyReport report = new rptMyReport();
rptViewer.ReportSource = report;
```

3.3. *Thay đổi thông tin kết nối đến nguồn dữ liệu:*

Giả sử ta có các chuỗi ServerName, DatabaseName, UserID, Password lưu thông tin để kết nối đến data source cho report, ta gán các thông tin này cho report như sau :

```
(using CrystalDesisions.Shared)
```

http://www.ebook.edu.vn

```
TableLogOnInfo Info;
 for (int i=0;i<report.Database.Tables.Count;i++)
{

  Info = report.Database.Tables[i].LogOnInfo;
  Info.ConnectionInfo.ServerName = ServerName;  // ”.” nếu là localhost
  Info.ConnectionInfo.DatabaseName =DatabaseName;
  Info.ConnectionInfo.UserID = UserID;
  report.Database.Tables[i].ApplyLogOnInfo(Info);

  /*sửa lại location của table (có dạng Database.Owner.TenTable/storedProcedure) cho
  khớp với tên database và owner mới */
  string location = report.Database.Tables[i].Location;
  location = location.Substring(location.LastChỉ mụcOf(".")+1);
  report.Database.Tables[i].Location = location;//report sẽ tự điền vào database và owner
  mới
  }
```

Lưu ý :

- Nếu kết nối sử dụng Windows Authentication (Integrated Security), ta để trống UserID.
- Trong đoạn lệnh trên, nên đặt lệnh report.Database.Tables[i].Location = location; trong try…catch để bắt Exception có thể xảy ra nếu các thông tin kết nối được gán không hợp lệ:

```
try
{
        report.Database.Tables[i].Location = location;
}
catch (EngineException ex)
{
    //báo lỗi
    //hiển thị dialog yêu cầu NSD nhập lại các thông số để kết nối
```

http://www.ebook.edu.vn

```
}
```

3.4. *Truyền giá trị cho tham số*

Giả sử trong report có tham số @MaNganh, dùng để đọc danh sách sinh viên của một ngành cụ thể. Trong chương trình ta truyền giá trị cho tham số này như sau :

```
(string MaNganh = "CNTT";)
  ParameterDiscreteValue ParamValue = new ParameterDiscreteValue();
  ParamValue.Value = MaNganh;
  ParameterValues values = new ParameterValues();
  values.Add(ParamValue);
  report.DataDefinition.ParameterFields["@MaNganh"].ApplyCurrentValues(values);
  rptViewer.ReportSource = report;
```

(Lưu ý : Lệnh rptViewer.ReportSource = report; phải được thực hiện **sau cùng**, sau khi đã cấu hình cho đối tượng report (thay đổi thông số kết nối data source, truyền giá trị tham số,…)).

http://www.ebook.edu.vn

BÀI TẬP QUẢN LÝ SINH VIÊN

Cho CSDL như sau:

Khoa : Mỗi khoa có một mã khoa để quản lý, một tên khoa và ghi nhận năm thành lập khoa.

maKhoa	tenKhoa	namThanhLap
varchar(10)	nvarchar(100)	int

KhoaHoc : Mỗi khóa học có một mã để quản lý, năm bắt đầu khoá học và năm kết thúc khóa học.

MaKhoaHoc	namBatDau	namKetThuc
varchar(10)	int	int

SinhVien : Mỗi sinh viên có một mã để quản lý và thuộc về một lớp nào đó (xác định bởi maLop).

MaSV	hoTen	namSinh	danToc	maLop
varchar(10)	nvarchar(100)	int	nvarchar(20)	varchar(10)

ChuongTrinh : Mỗi chương trình có một mã để quản lý và một tên chương trình.

MaCT	tenChuongTrinh
varchar(10)	nvarchar(100)

MonHoc : Mỗi môn học có một mã để quản lý và thuộc về một khoa nào đó (xác định bởi maKhoa)

MaMH	tenMonHoc	maKhoa
varchar(10)	nvarchar(100)	varchar(10)

KetQua : Mỗi kết quả thi ghi nhận điểm của một sinh viên làm bài thi cho 1 môn học nào đó ở một lần thi cụ thể. (1 sinh viên có thể thi 1 môn nào đó trên 1 lần)

MaSV	MaMH	lanThi	diem
varchar(10)	varchar(10)	Int	float

GiangKhoa: Mỗi dòng trong bảng này cho biết một môn học được giảng dạy tại một khoa nào đó trong một chương trình nào đó. Trong chương trình này, môn học đó được quy định số tiết lý thuyết và thực hành cụ thể và tương đương với bao nhiêu tín chỉ. (soTinChi bao gồm cả tín chỉ lý thuyết lẫn thực hành).

maCT	maKhoa	maMH	namHoc	hocKy	soTietLyThuyet	soTietThucHanh	soTinChi
varchar(10)	varchar(10)	varchar(10)	int	int	int	Int	int

Lop : Một lớp có một mã lớp để quản lý, thuộc về một khoa nào đó và mở ra cho một khóa học nhất định, trong một chương trình nhất định. Số thứ tự được đánh tăng dần cho các lớp cùng khoá học, cùng khoa và cùng chương trình.

MaLop	maKhoaHoc	maKhoa	maCT	soThuTu
varchar(10)	varchar(10)	varchar(10)	varchar(10)	int

Yêu cầu:

1. Hãy cài đặt CSDL trên
2. Hãy cài đặt khóa chính và khóa ngoại của CSDL trên.
3. Nhập các bộ dữ liệu sau :

http://www.ebook.edu.vn

Khoa

MaKhoa	tenKhoa	namThanhLap
CNTT	Công nghệ thông tin	1995
VL	Vật Lý	1970

Khóa học:

MaKhoaHoc	namBatDau	namKetThuc
K2002	2002	2006
K2003	2003	2007
K2004	2004	2008

SinhVien

MaSV	hoTen	namSinh	danToc	maLop
0212001	Nguyễn Vĩnh An	1984	Kinh	TH2002/01
0212002	Nguyên Thanh Bình	1985	Kinh	TH2002/01
0212003	Nguyễn Thanh Cường	1984	Kinh	TH2002/02
0212004	Nguyễn Quốc Duy	1983	Kinh	TH2002/02
0311001	Phan Tuấn Anh	1985	Kinh	VL2003/01
0311002	Huỳnh Thanh Sang	1984	Kinh	VL2003/01

ChuongTrinh

MaCT	tenChuongTrinh
CQ	Chính Qui

MonHoc

MaMH	tenMonHoc	maKhoa
THT01	Toán Cao cấp A1	CNTT
VLT01	Toán cao cấp A1	VL
THT02	Toán rời rạc	CNTT
THCS01	Cấu trúc dữ liệu 1	CNTT
THCS02	Hệ điều hành	CNTT

KetQua

maSV	maMH	lanThi	diem
0212001	THT01	1	4
0212001	THT01	2	7
0212002	THT01	1	8
0212003	THT01	1	6
0212004	THT01	1	9
0212001	THT02	1	8
0212002	THT02	1	5.5
0212003	THT02	1	4
0212003	THT02	2	6
0212001	THCS01	1	6.5
0212002	THCS01	1	4
0212003	THCS01	1	7

http://www.ebook.edu.vn

GiangKhoa

maCT	maKhoa	maMH	namHoc	hocKy	soTietLyThuyet	soTietThucHanh	soTinChi
CQ	CNTT	THT01	2003	1	60	30	5
CQ	CNTT	THT02	2003	2	45	30	4
CQ	CNTT	THCS01	2004	1	45	30	4

Lop

MaLop	maKhoaHoc	maKhoa	maCT	soThuTu
TH2002/01	K2002	CNTT	CQ	1
TH2002/02	K2002	CNTT	CQ	2
VL2003/01	K2003	VL	CQ	1

4. Viết các câu truy vấn sau :
   - 4.1. Danh sách các sinh viên khoa "Công nghệ Thông tin" khoá 2002-2006
   - 4.2. Cho biết các sinh viên (MSSV, họ tên ,năm sinh) của các sinh viên học sớm hơn tuổi qui định (theo tuổi qui định thi sinh viên đủ 18 tuổi khi bắt đầu khóa học)
   - 4.3. Cho biết sinh viên khoa CNTT, khoá 2002-2006 chưa học môn cấu trúc dữ liệu 1
   - 4.4. Cho biết sinh viên thi không đậu (Diem <5) môn cấu trúc dữ liệu 1 nhưng chưa thi lại.
   - 4.5. Với mỗi lớp thuộc khoa CNTT, cho biết mã lớp, mã khóa học, tên chương trình và số sinh viên thuộc lớp đó
   - 4.6. Cho biết điểm trung bình của sinh viên có mã số 0212003 (điểm trung bình chỉ tính trên lần thi sau cùng của sinh viên)
5. Hãy viết các function sau :
   - 5.1. Với 1 mã sinh viên và 1 mã khoa, kiểm tra xem sinh viên có thuộc khoa này không (trả về đúng hoặc sai)
   - 5.2. Tính điểm thi sau cùng của một sinh viên trong một môn học cụ thể
   - 5.3. Tính điểm trung bình của một sinh viên (chú ý : điểm trung bình được tính dựa trên lần thi sau cùng), sử dụng function 5.2 đã viết
   - 5.4. Nhập vào 1 sinh viên và 1 môn học, trả về các điểm thi của sinh viên này trong các lần thi của môn học đó.
   - 5.5. Nhập vào 1 sinh viên, trả về danh sách các môn học mà sinh viên này phải học.

6. Hãy viết các Stored Procedure sau:
   - 6.1. In danh sách các sinh viên của 1 lớp học
   - 6.2. Nhập vào 2 sinh viên, 1 môn học, tìm xem sinh viên nào có điểm thi môn học đó lần đầu tiên là cao hơn.
   - 6.3. Nhập vào 1 môn học và 1 mã sv, kiểm tra xem sinh viên có đậu môn này trong lần thi đầu tiên không, nếu đậu thì xuất ra là "Đậu", không thì xuất ra "Không đậu"
   - 6.4. Nhập vào 1 khoa, in danh sách các sinh viên (mã sinh viên, họ tên, ngày sinh) thuộc khoa này.
   - 6.5. Nhập vào 1 sinh viên và 1 môn học, in điểm thi của sinh viên này của các lần thi môn học đó.
     Ví dụ: Lần 1 : 10
     Lần 2: 8
   - 6.6. Nhập vào 1 sinh viên, in ra các môn học mà sinh viên này phải học.
   - 6.7. Nhập vào 1 môn học, in danh sách các sinh viên đậu môn này trong lần thi đầu tiên.
   - 6.8. In điểm các môn học của sinh viên có mã số là maSinhVien được nhập vào.
     (Chú ý: điểm của môn học là điểm thi của lần thi sau cùng)

http://www.ebook.edu.vn

6.8.1. Chỉ in các môn đã có điểm
6.8.2. Các môn chưa có điểm thì ghi điểm là null
6.8.3. Các môn chưa có điểm thì ghi điểm là <chưa có điểm>

Thêm 1 quan hệ

XepLoai

maSV	diemTrungBinh	ketQua	hocLuc

6.9. *Đưa dữ liệu vào bảng xếp loại. Sử dụng function 5.3 đã viết ở trên*
Qui định : ketQua của sinh viên là "Đạt" nếu *diemTrungBinh* (chỉ tính các môn đã có điểm) của sinh viên đó lớn hơn hoặc bằng 5 và không quá 2 môn dưới 4 điểm, ngược lại thì kết quả là không đạt
Đối với những sinh viên có *ketQua* là "Đạt" thì *hocLuc* được xếp loại như sau:
- *diemTrungBinh* >= 8 thì *hocLuc* là "Giỏi"
- 7 < = *diemTrungBinh* < 8 thì *hocLuc* là "Khá"
- Còn lại là "Trung bình"

6.10. Với các sinh viên có tham gia đầy đủ các môn học của khoa, chương trình mà sinh viên đang theo học, hãy in ra điểm trung bình cho các sinh viên này.
(Chú ý: Điểm trung bình được tính dựa trên điểm thi lần sau cùng). Sử dụng function 5.3 đã viết ở trên

7. Hãy cài đặt các ràng buộc toàn vẹn sau (bằng check constraint, unique constraint, rule hoặc trigger):

Miền giá tri

7.1. ChuongTrinh.ma chỉ có thể là 'CQ' hoặc 'CD' hoặc 'TC'
7.2. Chỉ có 2 học kỳ là 'HK1' và 'HK2'
7.3. Số tiết lý thuyết (GiangKhoa.soTietLyThuyet) tối đa là 120
7.4. Số tiết thực hành (GiangKhoa.soTietThucHanh) tối đa là 60
7.5. Số tín chỉ (GiangKhoa.soTinChi) của một môn học tối đa là 6
7.6. Điểm thi (KetQua.diem) được chấm theo thang điểm 10 và chính xác đến 0.5 (làm bằng 2 cách: kiểm tra và báo lỗi nếu không đúng qui định; tự động làm tròn nếu không đúng qui định về độ chính xác)

Liên thuộc tính trên 1 quan hệ

7.7. Năm kết thúc khóa học phải lớn hơn hoặc bằng năm bắt đầu
7.8. Số tiết lý thuyết của mỗi giảng khóa không nhỏ hơn số tiết thực hành

Liên bộ trên 1 quan hệ

7.9. Tên chương trình phải phân biệt.
7.10. Tên khoa phải phân biệt
7.11. Tên môn học phải duy nhất
7.12. Sinh viên chỉ được thi tối đa 2 lần cho một môn học
7.13. Liên thuộc tính trên nhiều quan hệ
7.14. Năm bắt đầu khóa học của một lớp không thể nhỏ hơn năm thành lập của khoa quản lý lớp đó
7.15. Sinh viên chỉ có thể dự thi các môn học có trong chương trình và thuộc về khoa mà sinh viên đó đang theo học

Tổng hợp

7.16. Hãy bổ sung vào quan hệ LOP thuộc tính SISO và kiểm tra sĩ số của một lớp phải bằng số lượng sinh viên đang theo học lớp đó

http://www.ebook.edu.vn

BÀI TẬP LẬP TRÌNH CƠ SỞ DỮ LIỆU

Cho cơ sở dữ liệu sau (có tên tập tin cơ sở dữ liệu **QLThuVien**):

- **NhaXuatBan(MANXB, TenNXB)**: *Mỗi nhà xuất bản có một mã số (MANXB) để phân biệt và tên nhà xuất bản (TenNXB)*
- **TheLoai(MaTL, TenTL):** *Sách được phân loại theo thể loại. Mỗi thể loại có mã thể loại (MaTL), tên thể loại (TenTL).*
- **Sach(MaSach, TuaDe, MANXB, TacGia, SoLuong, NgayNhap, MaTL):** *Mỗi cuốn sách có một mã số để phân biệt (MaSach), tên sách (TuaDe), do một nhà xuất bản xuất bản (MANXB), tác giả (TacGia), số lượng bản hiện có trong thư viện (SoLuong) và ngày nhập sách (NgayNhap) và thuộc về một thể loại.*
- **BanDoc(MaThe , TenBanDoc, DiaChi, SoDT)**: *Mỗi bạn đọc có một số thẻ để phân biệt (MaThe), họ tên bạn đọc (TenBanDoc), địa chỉ (DiaChi) và số điện thoại (SoDT).*
- **MuonSach(MaThe, MaSach , NgayMuon, NgayTra)**: *Một bạn đọc có thể mượn nhiều cuốn sách, với mỗi cuốn sách người ta ghi nhận ngày mượn (NgayMuon) và ngày trả (NgayTra).*

Ghi chú: *các field có gạch dưới là khoá của lược đồ quan hệ tương ứng.*

Dữ liệu mẫu cho các table như sau:

NhaXuatBan

MANXB	TenNXB
N001	Giáo dục
N002	Khoa học kỹ thuật
N003	Thống kê

BanDoc

MaThe	TenBanDoc	DiaChi	SoDT
050001	Trần Xuân	17 Yersin	858936
050002	Lê Nam	5 Hai Bà Trưng	845623
060001	Nguyễn Năm	10 Lý Tự Trọng	823456
060002	Trần Hùng	20 Trần Phú	841256

Sach

MaSach	TuaDe	MANXB	TacGia	SoLuong	NgayNhap	MaTL
TH0001	Sử dụng Corel Draw	N002	Đậu Quang Tuấn	3	08/09/2005	TH
TH0002	Lập trình mạng	N003	Phạm Vĩnh Hưng	2	03/12/2003	TH
TH0003	Thiết kế mạng chuyên nghiệp	N002	Phạm Vĩnh Hưng	5	04/05/2003	TH
TH0004	Thực hành mạng	N003	Trần Quang	3	06/05/2004	TH
TH0005	3D Studio kỹ xảo hoạt hình T1	N001	Trương Bình	2	05/02/2004	TH
TH0006	3D Studio kỹ xảo hoạt hình T2	N001	Trương Bình	3	05/06/2004	TH
TH0007	Giáo trình Access 2000	N001	Thiện Tâm	5	11/12/2005	TH

MuonSach

MaThe	MaSach	NgayMuon	NgayTra
050001	TH0006	12/12/2006	01/03/2007
050001	TH0007	12/12/2006	
050002	TH0001	08/03/2006	15/04/2007
050002	TH0004	04/03/2007	
050002	TH0002	04/03/2007	04/04/2007
050002	TH0003	02/04/2007	15/04/2007
060002	TH0001	08/04/2007	
060002	TH0007	15/03/2007	15/04/2007

TheLoai

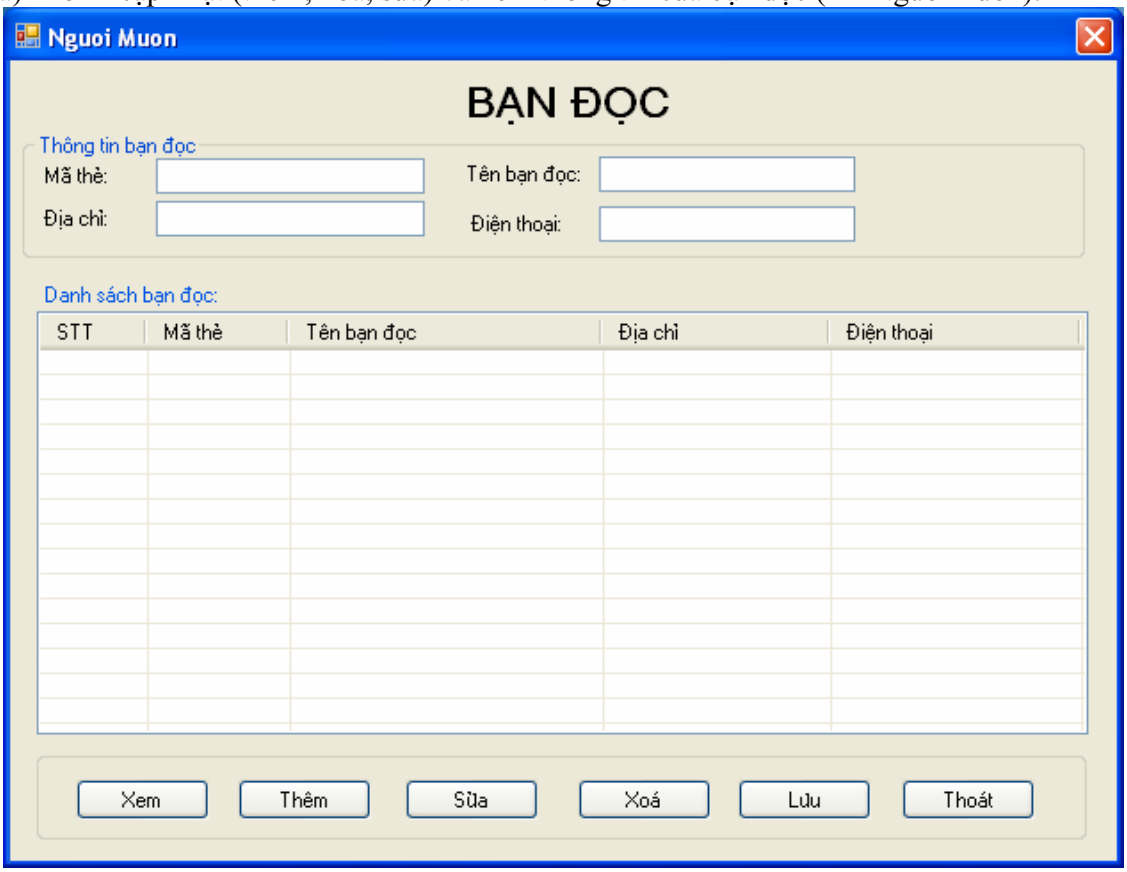
MATL	TENTL
TH	Tin học
HH	Hoá học
KT	Kinh tế
TN	Toán học

http://www.ebook.edu.vn

1) Tạo các table và thiết lập mối quan hệ (relationship) giữa các table. Căn cứ vào dữ liệu mẫu để chọn kiểu dữ liệu cho phù hợp cho các field trong các bảng.
2) Cài đặt các ràng buộc sau:
   RB1. Số lượng sách >=0
   RB2. Mã thẻ gồm 6 ký tự, được tạo theo quy tắc: hai chữ cuối của năm tạo thẻ ghép với số thứ tự của thẻ trong năm đó. (ví dụ: 050001 trong đó 05 là năm 2005, 1 là số thứ tự của thẻ trong năm 2005) (gợi ý: tạo hàm sinh mã thẻ)
   RB3. Mã sách gồm 6 ký tự, được tạo theo quy tắc: mã thể loại ghép với số thứ tự của cuốn sách trong thể loại đó (gợi ý: tạo hàm sinh mã sách).
   RB4. Mỗi đọc giả không được giữ quá ba quyển sách.
   RB5. Đọc giả không được phép mượn lại cuốn sách mà họ đang nợ.
   RB6. Số lượng trong bảng sách sẽ được thay đổi tuỳ theo thao tác cho bạn đọc mượn, nhận sách trả của bạn đọc hay nhập thêm sách.

   Các ràng buộc trên khi bị vi phạm sẽ hiện ra thông báo bằng tiếng Việt.
3) Nhập dữ liệu cho các bảng.
4) Viết Stored Procedure CapNhatSach (X, ThaoTac) thực hiện cập nhật số lượng của cuốn sách có mã số X tăng hay giảm 1 đơn vị tuỳ theo thao tác cho nhận trả sách hay cho mượn sách, trong đó SoLuong luôn luôn thoả điều kiện >=0. Nếu ThaoTac=1 là cho mượn sách, ThaoTac = 2 là nhận sách trả. X và ThaoTac là 2 tham số input.
5) Phân tích và xây dựng các thủ tục thường trú, các hàm cần thiết.
6) Thiết kế, phân tích xử lý và lập trình cho các form sau (*lưu ý đảm bảo các ràng buộc toàn vẹn dữ liệu*).

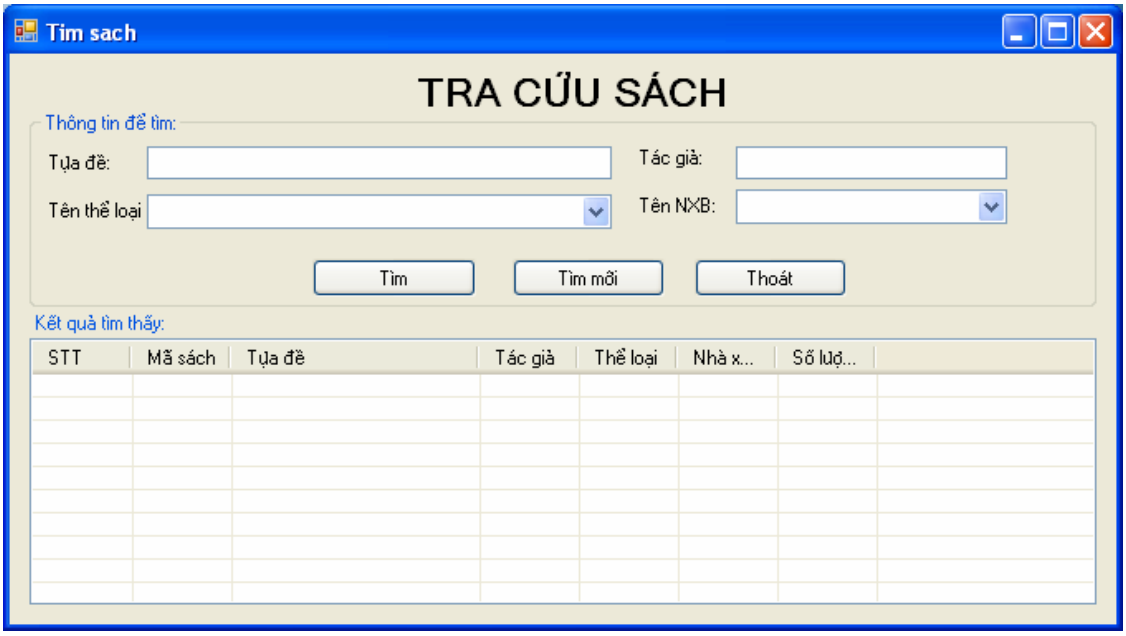
   a) Form cập nhật (thêm, xoá, sửa) và xem thông tin của bạn đọc (frmNguoiMuon):



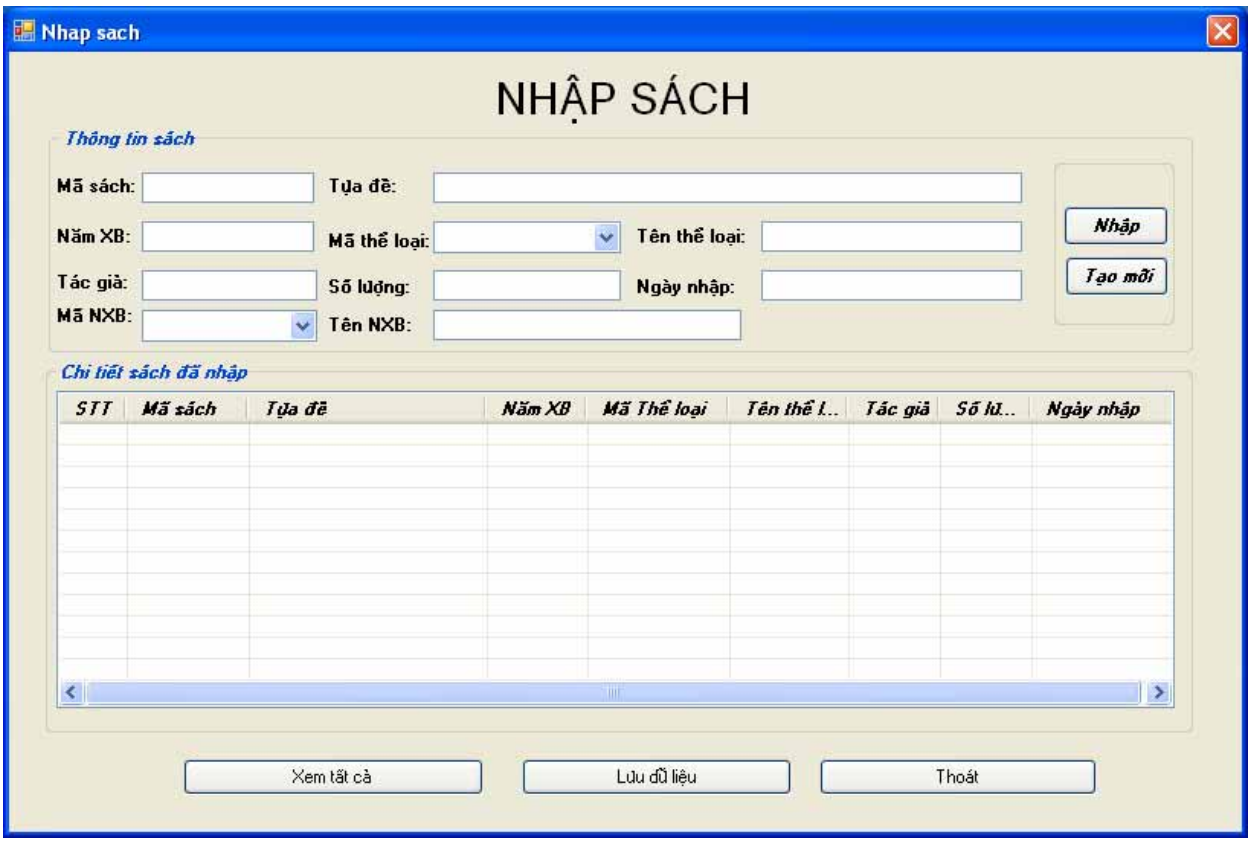
http://www.ebook.edu.vn

b) Tương tự thiết kế các form cập nhật thông tin thể loại (frmTheLoai) và form cập nhật thông tin nhà xuất bản (frmNhaXuatBan).

c) Form tra cứu sách (frmTraCuu):



d) Form nhập thông tin sách (frmSach):



http://www.ebook.edu.vn

e) Form xử lý mượn/trả sách:

f) Thiết kế form chính (frmManHinhChinh) với bố trí menu phù hợp để gọi sử dụng các form trên.

7) Tương tự như câu 6, áp dụng thiết kế chương trình theo mô hình 3 lớp.

http://www.ebook.edu.vn

TÀI LIỆU THAM KHẢO

Tiếng việt:

[1] Phạm Hữu Khang, *Quản trị SQL Server 2000*, NXB Thống kê, 2005.

[2] Dương Quang Thiện, *SQL Server 2000 Lập trình T-SQL*, NXB Văn hoá Sài Gòn, 2007.

[3] Vũ Tuyết Trinh, SQL Server 2008 (Slide bài giảng).

Tiếng Anh:

[4] Bill Hamilton, *ADO.NET Cookbook,* O'Reilly, 2003 (Ebook).

[5] Ramakrishnan, R. and Gehrke, J., *Database Management Systems*, Third Edition, McGraw Hill, 2003.

[6] Ramez Elmasri, Shamkant B. Navathe, *Fundamentals of database systems,* Addison Wesley - 4th edition, 2004.

[7] Sumathi, S. and Esakkirajan, S., *Fundamentals of Relational Database Management Systems,* Springer-Verlag, 2007.

http://www.ebook.edu.vn